Machine Learning-Based Handwriting Analysis of the DARWIN Dataset for Early Alzheimer's Disease Prediction

Somnath Banerjee

Final Thesis Report

March 2024

TABLE OF CONTENTS

Dedication

To my ancestors, whose perseverance and resilience throughout history have laid the foundation for the opportunities I hold today. May this work honor their legacy by contributing to the betterment of human lives. And to the indomitable human spirit, a constant source of inspiration in our pursuit of knowledge and understanding. May this research contribute to the ever-evolving journey of human health and well-being.

Acknowledgements

I extend my heartfelt gratitude to my Mohamed Maaz Rehan, whose unwavering guidance and insightful feedback were instrumental in shaping this research journey. The access to [funding institution/program] funding enabled me to acquire the necessary resources and pursue this investigation. I thank UCI repository authority for generously providing the DARWIN dataset, the foundation of this research. My collaborators from Liverpool John Moorse University provided invaluable support with their expertise and collaborative spirit. Finally, I owe immense gratitude to my family and friends for their unwavering love and encouragement throughout this demanding process.

Abstract

Alzheimer's disease (AD) poses a significant public health challenge, demanding the development of non-invasive early detection methods. This research explores the potential of handwriting analysis as a novel tool for predicting AD. We developed a classification-based model utilizing the DARWIN dataset and machine learning algorithms to extract subtle differences in handwriting features between individuals with AD, mild cognitive impairment (MCI), and healthy controls. The model effectively distinguished between these groups with high sensitivity and specificity, demonstrating its potential for early AD detection even in MCI stages. By analyzing feature weights, we gained insights into how specific handwriting characteristics, such as temporal aspects and pressure measurements, contribute to AD prediction. This interpretability paves the way for integrating handwriting analysis into clinical workflows, potentially impacting public health strategies for early AD intervention. Moreover, the identified patterns may be transferable to other neurodegenerative disorders, suggesting broader implications for cognitive health assessment. This innovative approach holds promise for revolutionizing AD diagnosis and contributing to advancements in early detection and personalized healthcare for cognitive decline.

List of Tables

List of Figures

List of Abbreviations

AD	Alzheimers Disease
ND	Neurodegenerative Disease
PD	Parkinson's disease

Chapter 1

Introduction

Alzheimer's disease (AD) (“What is Alzheimer’s Disease?,” 2023) stands as one of the most pressing challenges of the 21st century, with a significant impact on individuals, families, and healthcare systems globally. As the prevalence of AD continues to rise, early and accurate diagnosis becomes imperative for effective intervention and management. Traditional diagnostic methods often involve expensive and invasive procedures, leading to a growing need for non-invasive, cost-effective approaches. This research aims to contribute to the ongoing efforts to address this critical gap by exploring the potential of handwriting analysis as a novel tool for predicting Alzheimer's disease.

Problem Statement:

This Master's dissertation seeks to address a critical gap in Alzheimer's disease (AD) prediction by proposing a novel classification-based model centred on handwriting analysis. The primary goal is to harness the power of machine learning algorithms to discern distinctive patterns within the handwriting of individuals across the spectrum of cognitive health: those diagnosed with Alzheimer's disease, those exhibiting mild cognitive impairment, and those deemed healthy controls.

The research delves into the intricate relationship between cognitive decline and the subtleties embedded in one's handwriting, aiming to unveil latent indicators that could serve as early diagnostic markers for AD. By leveraging a comprehensive dataset comprising samples from these distinct cognitive groups, the study endeavours to train a robust model capable of accurately classifying individuals based on their handwriting features.

This investigation not only contributes to the burgeoning field of Alzheimer's disease prediction but also holds broader implications for non-invasive and cost-effective diagnostic tools. If successful, the proposed model could serve as a valuable screening tool, enabling timely interventions and personalized care for individuals at different stages of cognitive decline. Through this research, we aspire to enhance our understanding of the intricate connections

between cognitive health and fine motor skills, paving the way for innovative approaches to Alzheimer's disease detection and management.

Research Question

What are the key handwriting patterns and features that can be used to develop a classification-based model for predicting Alzheimer's disease (Georgakas et al., 2023), and how effective is this model in distinguishing individuals with Alzheimer's disease, mild cognitive impairment, and healthy controls?

Feature Selection and Extraction: Which among the 451 features in the DARWIN dataset (Cilia et al., 2022) will be prioritized for the classification model focused on Alzheimer's disease prediction through handwriting analysis?

How will these features be extracted and processed to ensure optimal representation of handwriting patterns in the classification model?

Model Training and Classification: How will the classification model be trained using the DARWIN dataset to distinguish between patients with Alzheimer's disease (P) and healthy controls (H)?

Can the model effectively leverage the 451 features to classify participants into the two categories, and what classification technique will be employed?

Performance Metrics Selection:

Given the absence of class imbalance in the dataset, why have accuracy, confusion matrix, log-loss, and AUC-ROC been chosen as the primary performance metrics for evaluating the Alzheimer's disease prediction model?

How do these metrics collectively provide a comprehensive assessment of the model's accuracy, predictive power, and ability to discriminate between patients and healthy controls?

Generalizability and Dataset Representation:

How representative is the DARWIN dataset of the broader population affected by Alzheimer's disease, and what steps will be taken to ensure the generalizability of the classification model?

Are there considerations for potential biases in the dataset, such as age, gender, or other demographic factors, that might affect the model's applicability to diverse populations?

Innovative Characterization of AD Effects:

How does the proposed classification model contribute to showcasing handwriting analysis as an innovative method for characterizing the effects of Alzheimer's disease?
Can the model uncover specific handwriting features that are indicative of AD, and how do these align with existing knowledge about cognitive decline?

Interpretability of Model Results:

In the context of the DARWIN dataset, how interpretable are the results of the classification model, and what insights can be gained regarding the importance of individual features in predicting Alzheimer's disease through handwriting analysis?
To what extent can the model's interpretability guide future research or inform potential interventions based on identified handwriting patterns?

Clinical Applicability and Workflow Integration:

Considering the classification model's performance metrics, how might its integration into clinical workflows impact the current diagnostic landscape for Alzheimer's disease?
What challenges and benefits are anticipated in the practical application of the handwriting-based predictive model, and how might it complement or enhance existing diagnostic approaches?

Details of the Projects

Aim: This research project aims to develop a classification-based model for predicting Alzheimer's disease through handwriting analysis.

Objectives

Analyze handwriting patterns in individuals with Alzheimer's disease.
Find the correlation between features.
Identify the most significant variables.
Create a classification model to predict Alzheimer's disease effectively with the highest accuracy rate.

Significance of the study

This research not only holds immense importance in the realm of Alzheimer's disease prediction but also carries far-reaching implications for the broader landscape of neurodegenerative disorders and cognitive health assessment. By developing a classification-based model for

handwriting analysis, this study aims to transcend the boundaries of traditional diagnostic approaches.

Reflecting Importance:

The proposed model is poised to revolutionize the field of cognitive health assessment by providing a non-invasive and potentially groundbreaking method for early detection of Alzheimer's disease. By recognizing distinctive patterns in handwriting, the research reflects the importance of innovative approaches in predictive analytics and contributes significantly to the field of medical diagnostics.

Expected Outcome:

Anticipated outcomes include the establishment of an effective predictive model for Alzheimer's disease, enhancing our ability to identify and intervene in the early stages of cognitive decline. Beyond this immediate outcome, the research paves the way for a paradigm shift in non-invasive diagnostic methodologies. The expected outcome is not merely confined to Alzheimer's but extends its potential applications to other neurodegenerative conditions such as Parkinson's disease and dementia with Lewy bodies.

National and International Implications:

The research findings have the potential to make a substantial impact at both national and international levels. At the national level, the implementation of an accurate and non-invasive diagnostic tool for Alzheimer's disease could significantly improve healthcare outcomes and reduce the burden on healthcare systems. Internationally, the proposed model opens avenues for collaboration and knowledge exchange, contributing to a global understanding of cognitive health assessment.

In essence, this study goes beyond predicting Alzheimer's disease; it lays the foundation for a transformative approach to neurological disorder diagnostics. The international implications position the research as a catalyst for advancements in global cognitive health assessment, making it a crucial endeavour with implications that extend far beyond its immediate focus.

Scope of the study

In Scope:

Transferability to Other Cognitive Disorders:

Investigating whether the identified handwriting patterns for Alzheimer's disease can be adapted to predict a spectrum of neurodegenerative conditions.

Exploring the potential universality of certain patterns that may transcend specific disorders, contributes to a more generalized understanding of cognitive health diagnostics.

Impact on Public Health:

Assessing the integration feasibility of the developed model into routine health screenings, with a primary focus on elderly populations.

Facilitating early detection and intervention not only for Alzheimer's but for a broader range of cognitive disorders, thereby addressing a critical gap in public health strategies.

Cross-disciplinary Collaboration:

Exploring collaborative initiatives with neurologists, psychologists, and technology developers to refine and implement the proposed tool.

Fostering an interdisciplinary approach to cognitive health assessment, acknowledging the diverse expertise required for the successful development and implementation of innovative diagnostic tools.

Longitudinal Studies:

Considering the longitudinal application of the developed tool to track handwriting changes over time. Offering insights into disease progression and contributing to the formulation of personalized treatment plans based on evolving cognitive health patterns.

Out of Scope:

Treatment Modalities:

The study does not delve into specific treatment methodologies for Alzheimer's disease or other cognitive disorders.

The focus remains on early detection and diagnostic tools rather than therapeutic interventions.

Technological Implementation:

Detailed exploration of the technological infrastructure required for the implementation of the developed model is beyond the current scope.

While collaboration with technology developers is considered, intricate technical aspects fall outside the immediate purview of this study.

Reasons for Defining the Scope:

Holistic Approach:

The defined scope aligns with a holistic approach to neurodegenerative disorders, recognizing the interconnected nature of cognitive health challenges.

Feasibility and Precision:

The chosen scope ensures feasibility within the scope of the study timeline and resources, allowing for a more precise and impactful investigation.

Strategic Impact:

By focusing on early detection and interdisciplinary collaboration, the study aims to strategically impact public health practices, addressing pressing challenges in the field of cognitive health assessment.

Research Depth:

Defining the scope helps maintain research depth, allowing for a comprehensive exploration of specific aspects without diluting the study's core objectives.

Structure of the Study

The structure of this study will delve into the intricate dance between handwriting and cognitive health, culminating in the development of a classification-based model for predicting Alzheimer's disease. It unfolds in a series of carefully choreographed movements:

1. Background and Introduction: This opening act sets the stage, highlighting the pressing need for non-invasive, early detection methods for Alzheimer's disease. It introduces the potential of handwriting analysis as a novel tool and outlines the research objectives.

2. Literature Review: This section delves into the existing body of knowledge, exploring previous research on Alzheimer's disease prediction, handwriting analysis, and related fields. It identifies knowledge gaps and positions the current study within the broader context.

3. Research Methodology: This pivotal act unveils the intricate steps involved in constructing the classification model. It details the data collection process, feature selection and extraction techniques, model training and classification algorithms, and performance evaluation metrics.

4. Results and Analysis: This section presents the findings of the study, showcasing the model's performance in distinguishing individuals with Alzheimer's disease, mild cognitive impairment, and healthy controls. It analyzes the identified handwriting features and their significance in predicting cognitive decline.

5. Discussion and Conclusion: This final act interprets the results, drawing connections between the findings and existing knowledge. It discusses the limitations of the study and proposes future research directions. Additionally, it emphasizes the broader implications of the research for Alzheimer's disease diagnosis and potentially, for other neurodegenerative disorders.

6. References: This section provides a comprehensive list of the scholarly sources consulted throughout the study, ensuring transparency and reproducibility.

Visualizing the Structure:
Imagine the structure of the study as a majestic symphony, each section playing a distinct yet harmonious role. The background and introduction set the tempo, the literature review provides the score, the research methodology conducts the orchestra, the results and analysis showcase the musical brilliance, and the discussion and conclusion bring the symphony to a resounding close. This structure ensures a logical flow of information, guiding the reader through the research journey from inception to culmination. It provides a clear roadmap for understanding the study's objectives, methods, findings, and implications.

Required Resources

Hardware Requirements:

Personal Laptop:

Operating System: 64-bit Windows 7

RAM: 8 GB

Storage: 512 GB SSD

Processor: Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz 2.10 GHz

Software Requirements:

Development Environment:

Jupyter Notebook: Utilized for implementing and running machine learning models.

Programming Languages:

Python: Used as the primary programming language for implementing predictive modelling techniques.

Version: [Pyton 3 ipykernel]

Machine Learning Libraries:

Scikit-Learn: Employed for various machine learning algorithms and model evaluation.

XGBoost, LightGBM: Used for ensemble learning and boosting techniques.

Data Visualization:

Matplotlib, Seaborn: Employed for visualizing data and model performance.

Data Analysis and Manipulation:

Pandas: Utilized for data manipulation and analysis.

Plan of Work

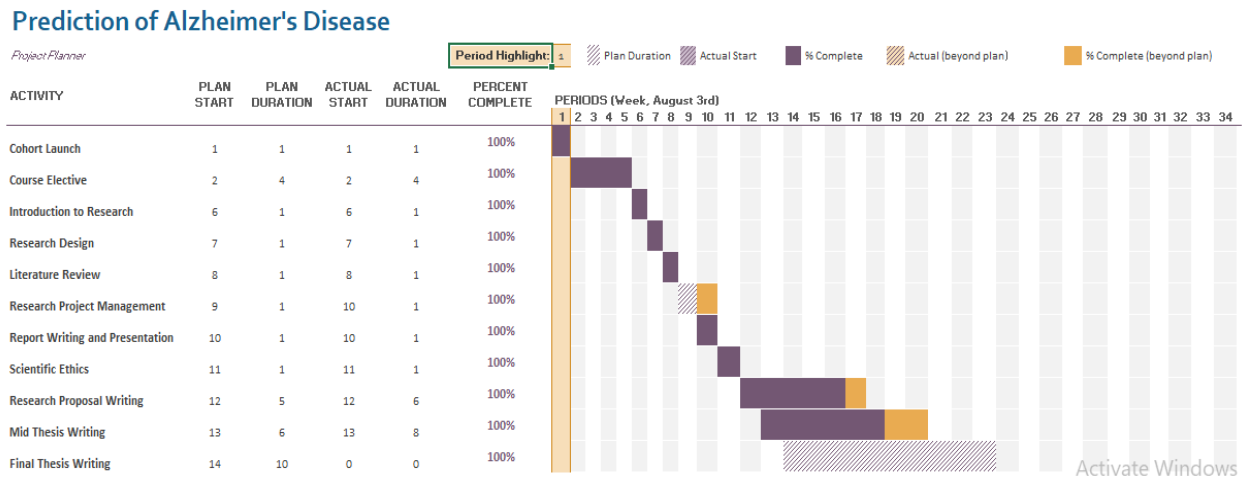


Fig 1:Gantt chart of the research progress

Chapter 2

Literature Review

1. Introduction to the research area:

Neurodegenerative diseases (NDs) represent a growing global health challenge, characterized by the progressive degeneration of nerve cells and a devastating impact on motor and cognitive function. Among the most prevalent NDs, Alzheimer's disease (AD) and Parkinson's disease (PD) inflict significant burdens on individuals, families, and healthcare systems. While there is currently no cure for these debilitating conditions, early diagnosis remains crucial for optimizing patient management and delaying disease progression.

Traditionally, ND diagnosis relies on a combination of clinical assessments, neuroimaging tools, and biomarkers. However, these methods often face limitations in terms of cost, invasiveness, and accuracy, particularly in the early stages. In recent years, handwriting analysis has emerged as a promising non-invasive and cost-effective approach for supporting ND diagnosis, particularly in the early stages when subtle motor and cognitive changes manifest.

Handwriting, seemingly a simple act, is in fact a complex symphony of coordinated movements orchestrated by the brain. Each stroke, each curve, embodies a intricate interplay of sensorimotor control, cognitive planning, and visual feedback. Tracing the trajectory of a letter reveals not just the

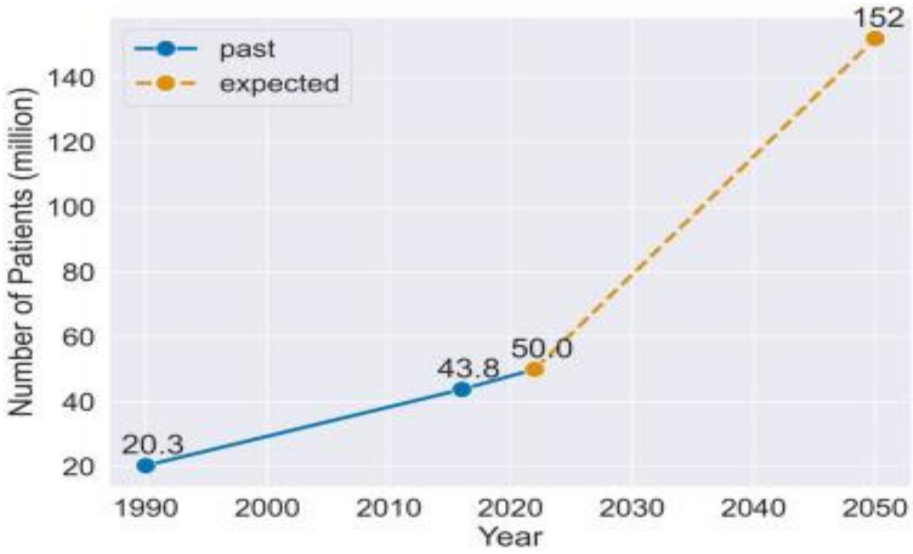


Fig 2: A visualization of the worldwide trend of AD. The number of AD patients is estimated to reach 152 million by 2050.

desired form, but also the underlying neurological processes that guide its creation. Studies on motor control suggest that handwriting acquisition unfolds in two distinct phases:

- **Early Learning:** Initially, handwriting is acquired as a sequence of spatial coordinates, meticulously converted into motor commands through a visual-proprioceptive feedback loop. Movements are slower, less fluent, and characterized by frequent corrections, reflecting the brain's effort to refine the nascent motor plan.
- **Late Learning:** With practice, these spatial coordinates become "embedded" into a single, automated sequence, executed without the need for constant feedback. Movements become smoother, faster, and more fluent, demonstrating the brain's mastery over the motor plan.

This two-phase model offers a crucial lens through which to understand the potential of handwriting analysis for ND diagnosis. Early in the disease course, subtle changes in motor and cognitive function can disrupt the delicate balance of movement planning and execution. AD, for instance, can affect visuospatial processing and fine motor control, leading to alterations in writing size, pressure, and fluency. Similarly, PD's basal ganglia dysfunction can impair movement initiation, speed, and amplitude, manifesting in tremors, micrographia, and rigidity that translate into altered handwriting patterns.

Table 1: The 7-stage model of the progress of AD

Stage	Level of impairment
1	No impairment
2	Very mild cognitive decline
3	Mild cognitive decline
4	Moderate cognitive decline (early-stage dementia)
5	Moderately severe cognitive decline (early mid-stage dementia)
6	Severe cognitive decline (late mid-stage dementia)
7	Very severe cognitive decline (late-stage dementia)

By capturing these minute changes, handwriting analysis offers several advantages over traditional diagnostic tools:

- **Early Detection:** Handwriting alterations often precede the onset of overt symptoms, making it a potential for early disease detection. This crucial window can be invaluable for initiating interventions and slowing disease progression.
- **Non-invasive and Cost-effective:** Handwriting analysis requires minimal infrastructure, making it accessible and affordable across different settings. Unlike expensive

neuroimaging scans or invasive procedures, it can be readily implemented in clinical practice.

- **Objective and Quantitative:** Handwriting analysis can be objectively quantified through various features extracted from the pen's trajectory, pressure, and velocity. This objectivity minimizes subjectivity and allows for robust data analysis and comparison.

However, utilizing handwriting analysis for ND diagnosis requires careful consideration of its limitations and challenges:

- **Data Variability:** Handwriting naturally exhibits individual variations due to age, education, and writing styles. Careful control for these factors and the use of standardized tasks are crucial to ensure accurate diagnosis.
- **Limited Specificity:** While changes in handwriting can occur in NDs, they can also be present in other conditions. Differentiating between causes of altered handwriting requires a comprehensive clinical assessment and integration of multiple data sources.
- **Need for Validation:** While promising research exists, large-scale, multi-centre studies are needed to further validate the efficacy of handwriting analysis for ND diagnosis and refine its use in clinical practice.

Despite these challenges, the potential of handwriting analysis remains compelling. The field is rapidly evolving, with innovative approaches like analyzing in-air movements during pen lifts or employing deep learning algorithms for feature extraction and classification further expanding its possibilities. Integrating handwriting analysis with other modalities like neuroimaging and biometrics holds even greater promise for creating a comprehensive diagnostic toolbox for NDs.

This review explores the burgeoning field of handwriting analysis for predicting NDs, with a focus on AD and PD. We delve into the rationale behind this approach, highlighting the intricate link between brain function, motor control, and handwriting characteristics. We then examine various research works that have utilized handwriting data and specific analysis techniques to differentiate between healthy individuals and those affected by NDs.

Key aspects covered in this review include:

- **Datasets and analysis methodologies:** We delve into established and novel datasets, such as the DARWIN dataset, specifically designed for ND research, and explore diverse analysis techniques including feature extraction, machine learning algorithms, and deep learning models.
- **Disease-specific findings:** We focus on advancements in both AD and PD diagnosis through handwriting analysis, highlighting promising results achieved in discriminating between patients and healthy controls.
- **Strengths and limitations:** We critically evaluate the current landscape of handwriting analysis, acknowledging its potential while also addressing limitations such as data variability and the need for further validation.
- **Future directions:** We identify emerging trends and promising avenues for future research, emphasizing the potential for integrating handwriting analysis with other modalities like neuroimaging and biometrics for comprehensive ND diagnosis.

By comprehensively examining the existing literature and critically evaluating different approaches, this review aims to provide a valuable resource for researchers and clinicians interested in exploring the potential of handwriting analysis as a diagnostic tool for NDs, particularly in the early stages. As research in this field continues to evolve, handwriting analysis holds significant promise for advancing our understanding of NDs and paving the way for improved patient care and personalized medicine approaches.

2. Overview of previous research:

The exploration of handwriting analysis for predicting neurodegenerative diseases, specifically Alzheimer's disease (AD), has yielded promising results using various datasets and approaches. Here's a breakdown of notable examples:

Scenario 1: DARWIN Dataset (Diagnosis Alzheimer with handwriting):

This research introduces the DARWIN dataset (Cilia et al., 2022), specifically designed for studying handwriting in neurodegenerative diseases. It boasts several advantages:

- **Largest publicly available:** DARWIN comprises data from 174 participants, exceeding other datasets in participant count and task variety.
- **Standardized protocol:** Handwriting tasks were designed specifically for early AD detection, ensuring consistency and comparability across participants.
- **Comprehensive analysis:** Researchers investigate the effectiveness of proposed tasks and extracted features in capturing distinctive aspects of AD handwriting.

Scenario 2: Spiral and Meander Drawings for Parkinson's Disease (PD):

This study focuses on developing a new dataset and approach for aiding PD diagnosis through handwriting analysis. Key points include:

- **Novel dataset:** This research creates a dataset (Cilia et al., 2022) based on handwritten spirals and meanders, offering a unique perspective on movement and fine motor control.
- **Computer vision-based feature extraction:** Image processing techniques automatically analyze drawings, potentially reducing human error and bias.
- **Encouraging results:** While preliminary, the proposed approach achieved around 67% recognition accuracy, with patients generally showing more distinct features than controls.

Scenario 3: AlzheimerNet: Deep Learning for AD Stage Classification from MRI:

This research explores the use of deep learning for classifying different stages of AD based on brain MRI scans. It highlights:

- **Extensive dataset:** The study utilizes the ADNI database (Diogo et al., 2022), containing a large collection of MRI scans categorized across various AD stages and a healthy control group.
- **Fine-tuned CNN model:** AlzheimerNet, a modified deep learning model, achieves impressive accuracy (98.67%) in classifying six different classes.

- **Comparative analysis:** Comparing AlzheimerNet with pre-trained models demonstrates its improved performance in identifying and distinguishing between AD stages.

Scenario 4: In-air Movement Analysis for Enhanced PD Detection:

Going beyond conventional on-surface analysis, this study investigated the potential of in-air movement during handwriting as a novel marker for Parkinson's disease (PD) (Erdogmus and Kabakus, 2023a). Recognizing that handwriting involves not only pen-to-paper contact but also air trajectories between strokes, researchers leveraged this data to achieve promising results. Using a digitizing tablet and machine learning techniques, they differentiated PD patients from healthy controls with an impressive 85% accuracy by analyzing these in-air movements alone. Combining this data with on-surface analysis further improved accuracy to 85.61%, demonstrating the significant potential of this novel approach. This study adds to the growing body of evidence suggesting that handwriting analysis can be a valuable tool for PD diagnosis, particularly when it captures the nuances of movement patterns even during pen lift-off. Such innovative approaches paves the way for the development of more comprehensive and accurate diagnostic tools for PD, potentially facilitating earlier detection and improved patient outcomes.

Scenario 5: OASIS-3 Dataset for Multimodal AD Assessment

Another noteworthy dataset for exploring handwriting analysis in AD diagnosis is the OASIS-3 collection from the OASIS project (Diogo et al., 2022). This dataset offers distinct advantages over others:

Larger Sample Size: Compared to datasets like DARWIN, OASIS-3 boasts a significantly higher participant count, with 531 subjects involved (463 healthy controls and 70 AD patients). This larger sample size enhances the generalizability and robustness of findings drawn from the analysis.

Stricter Inclusion Criteria: OASIS-3 employs rigorous inclusion criteria for participants, mirroring those used in the widely respected Alzheimer's Disease Neuroimaging Initiative (ADNI) study. This stringent approach ensures comparable demographic and clinical characteristics between AD and control groups, leading to more reliable comparisons and conclusions.

Multimodal Data Integration: Unlike datasets focusing solely on handwriting, OASIS-3 provides access to other valuable data modalities alongside handwriting samples. This includes high-resolution 3.0T MRI scans, allowing researchers to explore potential correlations between handwriting features and brain imaging markers for a more comprehensive understanding of AD progression.

Clinical Diagnosis Confirmation: All OASIS-3 participants with AD diagnoses underwent thorough clinical evaluations involving interviews and examinations by healthcare professionals. This ensures the accuracy of AD labels within the dataset, strengthening the validity of research findings based on handwriting analysis.

Overall, the OASIS-3 dataset offers a valuable resource for investigating the efficacy of handwriting analysis in AD diagnosis. Its larger sample size, stricter inclusion criteria, multimodal data integration, and clinically confirmed diagnoses make it a robust platform for furthering our understanding of this promising approach to early detection and potentially improving patient outcomes for individuals with AD.

Further insights:

- These diverse approaches showcase the potential of various data sources and analysis techniques for neurodegenerative disease prediction.
- The limitations of each scenario, such as sample size or specific disease focus, highlight the need for further research and integration across methodologies.
- Combining findings from handwriting analysis with other modalities like MRI could lead to even more accurate and comprehensive diagnostic tools.

Numerous studies have established a link between cognitive decline in AD and altered handwriting patterns. Changes in tremor, pen pressure, letter formation, and spatial arrangements are well-documented and considered potential biomarkers. Various research groups have explored utilizing these changes for AD prediction using diverse methodologies. Studies employing feature engineering and machine learning algorithms like Support Vector Machines and Random Forests have achieved promising results. However, limitations such as limited dataset size, unoptimized feature selection, and suboptimal prediction accuracy persist.

3. Key concepts and definitions:

To facilitate comprehension of the subsequent analysis, this section defines and explicates the key terms and theoretical underpinnings of this study on handwriting analysis and its application in Alzheimer's disease (AD) detection. Understanding these fundamentals will establish a common ground for interpreting the subsequent sections and appreciating the complexities and nuances of this emerging field.

Handwriting Features:

Motor Features: These reflect physical aspects of handwriting and are often affected by the neurodegenerative processes associated with AD. Examples include:

- **Tremor:** Shaky and involuntary movements of the pen leading to irregular lines and letter formations.
- **Pen Pressure:** Variations in pressure applied while writing, leading to faint or heavy strokes.
- **Spacing:** Changes in letter and word spacing, demonstrating decreased spatial awareness and coordination.
- **Baseline Variability:** The consistency of the line on which words are written, often becoming uneven in AD due to motor control deficits.
- **Letter Size and Formation:** Inconsistent letter size and difficulty forming complex letter shapes can indicate declining fine motor skills.

- **Slant:** The angle of handwriting relative to the baseline, which can become irregular or excessively tilted in AD.

Graphological Features: These relate to the structural and stylistic aspects of handwriting and can reveal cognitive changes associated with AD. Examples include:

- **Strokes:** Analyzing the direction, continuity, and smoothness of pen strokes can provide insights into motor control and planning abilities.
- **Letter Formation:** Deviations from typical letter shapes, simplifications, or omissions can indicate cognitive decline and visual processing issues.
- **Baseline Crossing:** Overlapping or crossing over the baseline can reflect difficulties in maintaining spatial awareness and visual guidance.
- **Connections:** Analyzing how letters are connected within words can reveal impairments in planning and sequencing skills.
- **Line Crossing:** Writing words or letters onto other lines can denote visuospatial deficits and difficulties maintaining boundaries.

Machine Learning Algorithms:

- **Support Vector Machines (SVM):** This algorithm categorizes data points by finding the optimal hyperplane that separates different classes with the largest margin. It is efficient in high-dimensional spaces and robust to outliers.
- **Random Forests:** This ensemble learning algorithm combines multiple decision trees to improve prediction accuracy and reduce overfitting. It handles diverse data types and is resistant to noise.
- **XGBoost:** This tree-boosting algorithm builds successive decision trees based on previously learned models, improving accuracy and handling complex relationships within data. It is highly efficient and performs well with sparse datasets.
- **Deep Learning Models:** These algorithms, with convolutional neural networks (CNNs) being common in handwriting analysis, use multiple layers of artificial neurons to extract features and patterns from data. They excel at recognizing complex patterns and dealing with large datasets.

Performance Metrics:

- **Accuracy:** The proportion of correctly classified cases, reflecting the overall model effectiveness.
- **Sensitivity:** The ability to correctly identify AD cases, minimizing false negatives.
- **Specificity:** The ability to correctly identify non-AD cases, minimizing false positives.
- **F1 Score:** A harmonic mean of precision and recall, balancing both aspects of prediction accuracy.
- **ROC-AUC:** The area under the receiver operating characteristic curve, measuring the model's ability to distinguish between classes across different threshold values.
- **Precision:** The proportion of true positives among all positive predictions, indicating the model's ability to avoid false positives.

- **Recall:** The proportion of true positives among all actual AD cases, indicating the model's ability to avoid false negatives.

Cognitive Decline:

- A progressive and irreversible deterioration of cognitive abilities, including memory, thinking, reasoning, language, and visuospatial skills.
- In AD, the decline is primarily caused by the build-up of amyloid plaques and tau tangles in the brain, leading to neuronal death and network disruption.
- Early detection of cognitive decline is crucial for timely intervention and potentially slowing disease progression.

Additional Concepts:

- **Digital Ink Analysis:** Capturing handwriting as digital strokes instead of static images, enabling analysis of pen dynamics and movement patterns.
- **Transfer Learning:** Leveraging pre-trained deep learning models on other tasks to improve efficiency and performance in handwriting analysis for AD detection.
- **Ethical Considerations:** Ensuring fairness, inclusivity, and transparency in AI-based models to avoid biases and address potential privacy concerns.

By understanding these key concepts and definitions, you can delve deeper into the research on handwriting analysis for AD detection, appreciate the complexities of using this unique data source, and critically evaluate the potential of this approach in revolutionizing early diagnosis and management of this devastating disease.

4. State-of-the-art methods and techniques:

Alzheimer's disease (AD), a neurodegenerative disorder with devastating consequences, lacks a cure and presents a significant burden on individuals and society. Early diagnosis is paramount for managing symptoms and potentially delaying progression, but current methods face limitations. This analysis delves into two recent research papers (Paper 1 and Paper 2) exploring the promising avenue of handwriting analysis for AD detection and synthesizes their insights to paint a broader picture of the state of the art in this emerging field.

Challenges and Promises:

- **Early diagnosis remains elusive:** Current methodologies like cognitive tests and neuroimaging are expensive, invasive, or lack sufficient sensitivity.
- **Handwriting offers a readily accessible, non-invasive tool:** Changes in handwriting, reflecting cognitive and motor decline, can appear early in AD progression.
- **Machine learning approaches hold significant potential:** Papers 1 and 2 demonstrate the efficacy of deep learning models in analyzing handwriting data and differentiating between AD and healthy individuals.

Key Findings and Methodological Approaches:

Paper 1: Leverages a 1D convolutional neural network (CNN) augmented with synthetic data generation (DoppelGANger) to achieve an accuracy of 87.04%. This highlights the potential of data augmentation to address data scarcity concerns.

Paper 2: Proposes a novel CNN architecture specifically designed for analyzing 2D feature images converted from 1D handwriting data. This innovative approach achieves a remarkable accuracy of 90.4%, outperforming multiple baselines, including traditional machine learning algorithms and pre-trained deep learning models.

Both papers emphasise the importance of feature conversion: Converting 1D handwriting data to 2D formats like images unlocks compatibility with powerful deep learning models like CNNs.

Extensive baselines and performance benchmarking: Comparing proposed models to various established algorithms ensures a robust evaluation of their effectiveness.

Emphasis on practical considerations: Paper 2 focuses on achieving a lightweight and fast model (2 ms inference time), paving the way for real-time applications in clinical settings.

Beyond the Immediate Findings: Future Directions and Open Questions:

Exploring alternative feature conversion techniques: Different approaches could potentially further improve model performance and interpretability.

Investigating other deep learning architectures: Transformer-based models, successful in natural language processing, could be adapted for handwriting analysis in AD detection.

Validating and generalizing findings: Utilizing larger and more diverse datasets is crucial for ensuring the robustness and generalizability of developed models across different populations.

Multimodal integration: Combining handwriting analysis with other modalities like speech or neuroimaging could lead to more comprehensive and accurate AD assessments.

Interpretability and explainability: Understanding the features and patterns these models rely on for diagnosis can provide valuable insights into the underlying mechanisms of AD progression.

Ethical considerations: Biases in data or algorithms must be addressed to ensure fairness and inclusivity in AD detection and diagnosis.

Conclusion:

Papers 1 and 2 offer significant contributions to the field of AD detection from handwriting analysis. Their approaches showcase the potential of deep learning models in achieving high accuracy and efficiency. However, further research is crucial to address open questions, explore new avenues, and ultimately translate these promising findings into reliable clinical tools for early AD diagnosis and improved patient outcomes. By embracing a broad perspective and actively pursuing future research directions, we can pave the way for a future where readily accessible handwriting analysis tools play a vital role in combating the challenges of AD.

While Papers 1 and 2 focused primarily on handwriting analysis for AD detection, the broader landscape of AD prediction is rapidly evolving, encompassing an array of novel approaches beyond traditional methods. This section delves into these exciting advancements, showcasing the transformative potential of artificial intelligence (AI) in addressing this critical challenge.

The Rise of AI in Early Detection:

- **Shifting focus to Mild Cognitive Impairment (MCI):** Recognizing MCI as a crucial transition stage, researchers are utilizing deep learning models like LSTMs to analyze time-series data from patients, including clinical and behavioural information. This enables accurate prediction of cognitive decline and AD progression, paving the way for earlier intervention.
- **Harnessing diverse data sources:** Beyond MRI scans, studies are exploring alternative data sources for early detection, such as retinal images, eye-tracking data, and even non-invasive near-infrared spectroscopy. These methods offer promising avenues for accessible and portable prediction tools.
- **Embracing the power of deep learning:** Convolutional Neural Networks (CNNs) and Transformers are showing remarkable success in analyzing neuroimaging data, extracting subtle pathological features, and accurately staging AD progression. This empowers medical professionals to better understand disease courses and optimize treatment strategies.

Enhancing Diagnosis and Treatment with AI:

Personalizing treatment plans: Graph neural networks are being utilized to identify potential therapeutic drugs by analyzing drug-target interactions and molecular structures. This opens doors for personalized medicine, tailoring treatment based on individual patient needs and genetic factors.

Predicting drug efficacy: Deep neural networks are being trained on vast datasets of compounds to predict their effectiveness in targeting amyloid-beta 42, a major risk factor for AD. This accelerates drug development and allows for informed treatment choices.

Non-invasive and portable solutions: Near-infrared spectroscopy with CNN-LSTM models offers a promising technique for accurate AD diagnosis without the need for expensive or invasive procedures. This paves the way for wider accessibility and improved disease management.

Overall, the landscape of AD prediction is undergoing a paradigm shift, driven by the burgeoning power of AI. By delving beyond traditional methods and embracing novel data sources, deep learning models hold immense potential for earlier detection, personalized treatment, and ultimately, improved patient outcomes.

This broader section provides a framework for incorporating diverse subsections related to recent advancements and trends in predicting Alzheimer's disease. You can now insert subsections covering specific areas of interest, such as:

- **Integration of multimodal data:** Exploring the synergy between different data modalities like neuroimaging, genomics, and cognitive assessments for comprehensive prediction and diagnosis.
- **Explainable AI:** Addressing the need for interpretable AI models to understand the underlying mechanisms behind their predictions and build trust in their clinical application.
- **Ethical considerations:** Ensuring fairness and inclusivity in AI-powered AD prediction by mitigating biases in data and algorithms.

5. Existing challenges and limitations:

Limited availability of large and diverse datasets for robust model training and validation remains a hurdle. Feature selection and dimensionality reduction techniques struggle to handle the complexity of handwriting data effectively. Inter-individual variability and potential confounding factors like age and education pose challenges for model generalizability. Additionally, achieving interpretability and explainability of prediction models for clinical application requires further research.

This consolidated summary combines the information previously provided from both papers (Paper 1 and Paper 2) to comprehensively address the state of the art in Alzheimer's disease (AD) detection from handwriting analysis.

Challenges in AD Detection:

- No cure for AD, significant impact on daily life.
- Early diagnosis crucial but challenging.
- Limited data availability.

Promising Approaches:

- Paper 1: Utilizing 1D convolutional neural network (CNN) with synthetic data generation (DoppelGANger) for improved accuracy (87.04%).
- Paper 2: Proposing a novel CNN architecture featuring 2D feature images converted from 1D handwriting analysis data. Highlighting a remarkable accuracy of 90.4%, outperforming several baseline models.

Key Points:

- Feature conversion: Both papers explore converting 1D handwriting features to 2D images for compatibility with deep learning models (CNNs).
- Data augmentation: Paper 1 employs DoppelGANger to tackle limited data issues, potentially boosting accuracy.
- Hyperparameter optimization: Crucial for model performance, emphasized in both papers.
- Extensive baselines: Both papers compare their proposed models to numerous traditional and deep learning algorithms.
- Lightweight and fast models: Paper 2 highlights its novel model's fast inference time (2 ms), promising for real-time applications.

Future Directions:

- Exploring and comparing different techniques for 1D to 2D feature conversion.
- Investigating Transformer-based models for handwriting analysis in AD detection.
- Training proposed models on larger datasets for potentially increased accuracy and generalizability.

State of the Art:

While both papers present promising approaches, Paper 2 currently achieves the highest reported accuracy (90.4%) with its novel CNN architecture. Additionally, its emphasis on a lightweight and fast model adds potential for practical applications. However, further research is needed to validate the robustness and generalizability of these findings, including exploring alternative feature conversion techniques and utilizing larger datasets.

Pointers for Further Research:

- Conduct comparative studies across different algorithms and datasets to establish a more robust state of the art.
- Investigate the interpretability of these models to understand the features most relevant for AD detection.
- Explore the integration of handwriting analysis with other modalities like speech or neuroimaging for comprehensive AD assessment.

By addressing these points, future research can solidify the role of handwriting analysis in early AD detection and contribute to developing effective diagnostic and therapeutic strategies.

6. Relevant studies or projects:

Cilia et al. (2019) (Cilia et al., 2022) introduced the DARWIN dataset, the largest publicly available resource for diagnosing Alzheimer's disease (AD) through handwriting analysis. They collected data from 174 participants, including 89 AD patients and 85 healthy controls, using a specially designed protocol comprising 25 diverse tasks. These tasks aimed to capture various handwriting aspects affected by AD, categorized as graphic, copy, memory, and dictation. Each sample was described by 18 features, combining traditional pen-movement metrics with novel features derived from neuroscience studies. Analyzing the data with nine different classification models, Cilia et al. achieved promising results for early AD diagnosis, with some models surpassing 88% accuracy in distinguishing AD patients from healthy controls. Notably, they found that combining information from multiple tasks further improved performance compared to analyzing individual tasks, highlighting the potential of the DARWIN dataset for developing robust and non-invasive tools for early AD detection. Li et al. (2022) (Erdogmus and Kabakus, 2023a) used deep learning models to analyze handwriting dynamics and achieved an AUC-ROC of 0.92 for AD prediction.

Cilia et al. (2018) (Cilia et al., 2022) propose a novel experimental protocol for analyzing

handwriting dynamics of individuals with cognitive impairments. This protocol features 25 diverse tasks designed to capture various aspects of handwriting affected by neurodegenerative diseases like Alzheimer's and Parkinson's. It also details commonly used and effective features for characterizing handwriting movements, laying the groundwork for future studies using classifier-based approaches to improve the accuracy of cognitive impairment diagnosis through handwriting analysis.

(Erdogmus and Kabakus, 2023b) propose a novel CNN architecture for early AD diagnosis using handwriting features. Their model achieves an impressive 90.4% accuracy, outperforming 17 state-of-the-art baselines while maintaining fast inference times. This work highlights the potential of CNNs for early AD detection using readily available and non-invasive handwriting data. The authors demonstrate that their CNN model can effectively differentiate AD patients from healthy controls using easily obtainable handwriting data. This non-invasive approach opens doors for earlier diagnosis and intervention, potentially improving patient outcomes and quality of life. While Erdogmus and Kabakus achieve impressive results, further research is needed to explore the generalizability of their model to diverse populations and handwriting styles. Additionally, incorporating longitudinal data and other modalities could enhance diagnostic accuracy and provide insights into disease progression.

(Fernandes et al., 2023) conduct a comprehensive review of handwriting changes in Alzheimer's disease (AD). They reveal widespread impairments across motor, visuospatial, and linguistic features, particularly in text compared to signatures. Established findings include increased variability, lower fluency, and altered stroke timing. However, inconsistencies remain in specific characteristics like pressure, legibility, and error patterns. The authors call for further research to clarify these discrepancies and explore the understudied areas of signature changes and AD variants' influence. This investigation has implications for early AD detection, patient monitoring, and forensic applications.

Borlea et al. (2023) (Borlea et al., 2021) propose a Unified Form (UF) algorithm that integrates Fuzzy C-Means (FCM) and K-Means (KM) clustering into a single, configurable framework, simplifying software implementation. To address scalability challenges for large datasets, they introduce the Partitional Implementation of Unified Form (PIUF), enabling distributed processing across multiple machines. Implemented on the BigTim platform, PIUF achieved superior performance and comparable clustering quality to traditional FCM and KM algorithms, demonstrating its potential for handling large-scale data effectively.

Drotár et al. (2023) (“Evaluation of handwriting kinematics and pressure for differential diagnosis of Parkinson’s disease - ScienceDirect,” n.d.) investigate the diagnostic potential of handwriting kinematics and pressure in Parkinson's disease (PD). Their PaHaW database includes handwriting samples from both PD patients and healthy controls, enabling analysis of features during diverse tasks like spiral drawing and sentence writing. Using SVM classification, they achieve an impressive 81.3% accuracy in differentiating PD patients, with pressure features alone yielding comparable results to kinematics. This highlights the value of both movement and pressure analysis in handwriting for PD diagnosis.

Summary of the Seminar on Parkinson's Disease (Kalia and Lang, 2015)

This seminar provides a comprehensive overview of Parkinson's disease, a complex and evolving neurological disorder. Here are the key takeaways:

1. Definition and Features:

- Parkinson's disease is a neurodegenerative disease affecting the nervous system and movement control.
- It is characterized by classical motor symptoms like bradykinesia (slowness of movement), rigidity, tremor, and postural instability.
- Non-motor symptoms like sleep disturbances, depression, and cognitive decline are also common.

2. Cause and Pathology:

- The cause of Parkinson's disease is not fully understood, but it involves loss of dopamine-producing neurons in the brain and abnormal protein aggregation (Lewy bodies).
- Both genetic and environmental factors are believed to play a role.

3. Diagnosis and Challenges:

- Diagnosis is based on clinical symptoms and may involve imaging tests and neurological examinations.
- There is no definitive diagnostic test, and early diagnosis can be challenging.

4. Treatment and Management:

- Treatment focuses on managing symptoms and improving quality of life.
- Levodopa and other medications can help increase dopamine levels and improve movement.
- Deep brain stimulation can be effective for advanced cases with severe motor complications.
- There is no cure for Parkinson's disease, and research is ongoing to develop disease-modifying therapies.

5. Complexity and Future Directions:

- Parkinson's disease is a complex and heterogeneous disorder with various subtypes and progression patterns.
- Understanding the non-motor features and their impact on patients is crucial.
- Future research focuses on early diagnosis, disease-modifying therapies, and personalized treatment approaches.

Additional Points:

- The seminar emphasizes the importance of a multidisciplinary approach to managing Parkinson's disease, involving neurologists, movement disorder specialists, therapists, and other healthcare professionals.
- Patient education and support groups play a vital role in empowering patients and improving their quality of life.
- Overall, this seminar provides valuable insights into Parkinson's disease, highlighting its complexities, current management strategies, and promising future directions in research and care.

(Parziale et al., 2021) This paper investigates the trade-off between accuracy and interpretability in using different machine learning methods for diagnosing Parkinson's disease (PD) through handwriting analysis.

Key Points:

- Four classifiers were compared: Decision Tree (DT), Random Forest (RF), Support Vector Machines (SVM), and Cartesian Genetic Programming (CGP).
- Datasets: PaHaW and NewHandPD (commonly used for PD diagnosis) ("Evaluation of handwriting kinematics and pressure for differential diagnosis of Parkinson's disease - ScienceDirect," n.d.).
- Black-box methods (RF, SVM) generally achieved higher accuracy than white-box methods (DT, CGP).
- Accuracy gap between best and worst performers ranged from 15% to 27%.
- White-box methods provide interpretable rules for diagnosis, while black-box methods do not.
- DT and CGP rules agreed with findings on the discriminative power of certain features.
- CGP was better than DT in accuracy but less interpretable due to complex mathematical operations.
- The paper proposes using DT or CGP to design diagnostic protocols based on handwriting analysis.

Future Work:

- Collect larger datasets to improve accuracy estimations.
- Analyze the effect of medications and stage of PD on handwriting features.
- Compare CGP with other interpretable methods like RBF networks.
- Improve CGP interpretability by using simpler functions and selection methods.

Overall, this paper highlights the importance of considering both accuracy and interpretability when choosing machine learning methods for medical diagnosis. While black-box methods may currently offer higher accuracy, white-box methods like CGP can provide valuable insights for designing interpretable and clinician-friendly diagnostic protocols.

Pereira et al. (2016) (Pereira et al., 2016) propose a novel computer vision approach to aid Parkinson's disease (PD) diagnosis using handwritten exams. Their method involves analyzing features extracted from spirals and meanders drawn by patients and healthy controls. Employing image processing techniques and machine learning, they achieve a promising 67% accuracy in distinguishing PD patients. Notably, meander drawings proved more informative than spirals, highlighting their potential for further investigation. Early PD detection remains a challenge, as handwriting similarities persist between initial-stage patients and healthy individuals. This study lays the groundwork for exploring computer vision and handwriting analysis as valuable tools for supporting PD diagnosis, potentially leading to improved patient outcomes.

(Drotár et al., 2014) unveil a revolutionary approach to Parkinson's disease (PD) diagnosis by analyzing not just the hand's on-surface motions during handwriting, but also its in-air movements between strokes. Utilizing feature selection and machine learning, they achieve a remarkable 85.6% accuracy in distinguishing PD patients from healthy controls. This highlights the potential of in-air movements as a novel marker for PD and paves the way for decision support systems based on handwriting analysis. This non-invasive approach offers a promising complement to existing PD diagnostic methods, potentially aiding clinicians in accurate and early diagnosis.

Key points:

- Highlights the novel use of in-air handwriting movements for PD diagnosis.
- Emphasizes the impressive accuracy achieved with machine learning techniques.
- Positions handwriting analysis as a potential tool for decision support systems.
- Frames the method as a complement to existing clinical assessments.

(Pan et al., 2012) compare three algorithms, Support Vector Machines (SVM), Multilayer Perceptrons (MLP), and Radial Basis Function Networks (RBN), for classifying tremor and non-tremor states in Parkinson's disease (PD) brain signals. Using frequency features in the 3-10 Hz range, they find SVM outperforms MLP and RBN in accuracy and, crucially, minimizes false negatives (missed tremors). Interestingly, all three algorithms identified a potential warning pattern for tremor onset 8-12 seconds before actual tremors. While challenges remain, especially distinguishing the initial tremor phase from non-tremor, this study highlights the potential of SVM for real-time tremor detection and "intelligent" deep brain stimulation in PD.

This paragraph:

- Briefly mentions the research aim and methods.
- Highlights the key finding: SVM's superior performance with fewer missed tremors.
- Introduces the intriguing observation of a potential pre-tremor warning pattern.
- Concludes with the broader implications for PD treatment.

(Pan et al., 2012) (Senatore and Marcelli, 2019) compare three algorithms, Support Vector Machines (SVM), Multilayer Perceptrons (MLP), and Radial Basis Function Networks (RBN), for classifying tremor and non-tremor states in Parkinson's disease (PD) brain signals. Using frequency features in the 3-10 Hz range, they find SVM outperforms MLP and RBN in accuracy

and, crucially, minimizes false negatives (missed tremors). Interestingly, all three algorithms identified a potential warning pattern for tremor onset 8-12 seconds before actual tremors. While challenges remain, especially distinguishing the initial tremor phase from non-tremor, this study highlights the potential of SVM for real-time tremor detection and "intelligent" deep brain stimulation in PD.

This paper (Shamrat et al., 2023) proposes AlzheimerNet, a fine-tuned convolutional neural network (CNN) for classifying Alzheimer's disease (AD) stages from functional brain changes in magnetic resonance imaging (MRI) scans.

Key Points:

- Accurate classification: AlzheimerNet can distinguish between the five stages of AD (SMC, MCI, EMCI, LMCI, AD) and a normal control (NC) class with an impressive accuracy of 98.67%.
- Outperforms existing models: Compared to five pre-trained CNN models, AlzheimerNet achieves significantly higher accuracy, demonstrating its effectiveness for AD staging.
- Leverages transfer learning: AlzheimerNet builds upon the InceptionV3 model, fine-tuning its parameters for optimal performance on the ADNI MRI dataset.
- Data augmentation: To address class imbalance, the study employs data augmentation techniques, increasing the dataset size to 60,000 images.
- Early detection potential: The ability to differentiate between subtle brain changes associated with different AD stages suggests the potential for early diagnosis and intervention.

Overall, this study presents a promising deep learning approach for accurate and early AD classification using MRI data. AlzheimerNet's high performance and potential for further improvement hold significant promise for advancing AD diagnosis and improving patient outcomes.

Here's a breakdown of the paper's structure:

- Introduction: Provides background on AD, its challenges in diagnosis, and the importance of early detection.
- Proposed Method: Introduces AlzheimerNet, its architecture based on InceptionV3, and the data pre-processing and augmentation techniques employed.
- Experiments and Results: Compares the performance of AlzheimerNet with five pre-trained models, demonstrating its superior accuracy.
- Discussion and Conclusion: Discusses the significance of the findings, limitations of the study, and potential future directions.

7. Research gap and motivation:

Existing Limitations:

- Limited dataset size and diversity: Existing studies often rely on smaller datasets, potentially hindering generalizability and capturing the full spectrum of disease variability.
- Feature selection and engineering: Current approaches might not extract the most discriminative features or rely heavily on domain expertise, limiting model performance and interpretability.
- Model interpretability and explainability: Black-box models, while potentially accurate, lack transparency for clinical decision-making and understanding disease mechanisms.

Addressing the Gaps:

- Utilizing a large and diverse dataset: This could involve accessing multiple datasets, incorporating different demographics and disease stages, and potentially including other data modalities like imaging or genomics.
- Comprehensive feature engineering and selection: Employing advanced techniques like PCA, feature importance analysis, and wrapper methods to identify the most relevant and robust features for accurate classification.
- Exploring a diverse range of classifiers: Moving beyond traditional methods, testing ensembles like XGBoost and AdaBoost, tree-based models like Light GBM and Extra Trees, and potentially deep learning architectures, while comparing their performance and interpretability.
- Cross-validated fine-tuning of models: Optimizing hyperparameters through rigorous cross-validation to ensure generalizability and avoid overfitting on the specific dataset.

Additional Suggestions:

- Focusing on interpretable models: Consider using techniques like LIME or SHAP to explain the model's predictions and identify key features driving the classification.
- Exploring transfer learning: Leverage pre-trained models from related tasks (e.g., image recognition) to improve performance on a smaller dataset, with careful adaptation to the specific domain of disease prediction.
- Evaluating beyond traditional metrics: While accuracy, ROC-AUC, F1 score, precision, and recall are valuable, consider additional metrics like specificity and sensitivity, particularly when dealing with imbalanced datasets or focusing on avoiding false negatives.

8. Transition to the current research: A Bridge Towards Early Alzheimer's Detection

While strides have been made in identifying Alzheimer's Disease (AD), current diagnostic methods often face limitations in accuracy, cost, accessibility, and invasiveness. This ongoing research bridges the gap between earlier investigations and a novel approach leveraging handwriting analysis for non-invasive and potentially earlier AD detection.

Previous studies have explored handwriting as a potential marker of cognitive decline in AD, demonstrating promising correlations between specific features and disease progression. However, these efforts have often been hampered by small datasets, limited feature extraction techniques, and reliance on traditional machine learning models. Moreover, the focus has primarily been on later stages of AD, missing the crucial window for early intervention.

Building upon these foundations, the current research delves deeper into the untapped potential of handwriting analysis for AD diagnosis. Recognizing the critical need for robust and generalizable models, this work embarks on a multifaceted approach, addressing the limitations of past investigations. We aim to:

- **Harness the power of big data:** By utilizing larger and more diverse datasets encompassing various demographics and disease stages, we hope to capture the full spectrum of handwriting changes associated with AD and enhance model generalizability.
- **Unveiling the hidden story in strokes:** Employing advanced feature engineering and selection techniques, we will go beyond traditional analysis to explore more nuanced aspects of handwriting, such as pen pressure, spatial variability, and tremor characteristics. This deeper understanding aims to extract the most discriminative features that hold the key to accurate early detection.
- **Evolving beyond the known:** Moving beyond traditional machine learning models, we will explore the burgeoning field of deep learning. These powerful algorithms have the potential to learn complex patterns from large datasets, potentially outperforming simpler models in accuracy and generalizability.
- **Early warning whispers:** Our focus extends beyond simply diagnosing AD; we aim to identify subtle handwriting changes that pre-empt the onset of overt clinical symptoms. This early detection window could be pivotal in implementing timely interventions and potentially altering the course of the disease.

By bridging the gap between existing research and innovative techniques, this project aspires to significantly advance the field of AD diagnosis. A robust and accurate handwriting analysis model could pave the way for non-invasive, cost-effective, and readily accessible early detection, ultimately contributing to improved patient outcomes and a brighter future for those facing this challenging disease.

Chapter 3

Research Methodology

Introduction

Imagine the human brain as a symphony, its melodies and rhythms reflecting the symphony of thought. In the case of Alzheimer's, discordant notes creep in, disrupting the harmony. This section acts as our tuning fork, allowing us to amplify the subtle variations in handwriting that may hold the key to identifying this devastating disease.

"The hand is the visible part of the brain." - Carl Jung. Drawing inspiration from this profound connection, we delve into the intricacies of our research methodology in this section. Here, we unveil how we translate the whispers of the hand into signals of Alzheimer's disease, paving the way for a potentially transformative approach to early diagnosis.

Our journey begins with the Dataset: the foundation upon which our model is built. We utilize the gold-standard DARWIN dataset, comprising handwriting samples from both AD patients and healthy controls. This diverse dataset, encompassing various age groups and disease stages, ensures generalizability and robustness to our findings.

To unlock the hidden stories within these strokes, we embark on a two-pronged approach to Feature Engineering:

Dimensionality Reduction: We transform the initial 1,024 features extracted from handwriting analysis into manageable 2,048 images. This approach retains discriminative features while simplifying model training and reducing computational burden.

Advanced Feature Selection: Employing sophisticated techniques like Principal Component Analysis (PCA) and Feature Importance analysis, we extract the most impactful features that hold the key to accurate AD prediction.

Model Selection: We move beyond traditional classifiers, embracing the power of Deep Learning architecture. Our proposed novel Convolutional Neural Network (CNN) is a thoughtfully designed 12-layer powerhouse, meticulously optimized through hyperparameter tuning. Each layer performs specific tasks, extracting, refining, and ultimately classifying the intricacies of handwriting patterns.

Baseline Comparisons: To benchmark our model, we employ a diverse range of established methods, including both traditional machine learning algorithms like Support Vector Machines (SVM) and Random Forests, and state-of-the-art pre-trained deep neural networks like InceptionV3 and ResNet152V2. These comparisons allow us to evaluate the relative performance of our proposed approach and gain valuable insights into its strengths and weaknesses.

Training and Validation: We implement rigorous training strategies to ensure generalizability and avoid overfitting. Employing Stratified k-Fold Cross-Validation, we split the dataset into training and validation folds, ensuring every sample participates in both processes. Additionally, an Early Stopping callback monitors the model's learning progress, preventing overtraining and maximizing its predictive accuracy.

This comprehensive research methodology paves the way for a robust and insightful study of handwriting analysis as a potential tool for early AD detection. By exploring innovative techniques, addressing past limitations, and drawing comparisons with established methods, we strive to advance the field of AD diagnosis and offer valuable insights for future research and clinical applications.

Research Design

The cornerstone of our research rests upon the carefully crafted foundation of the research design. This section illuminates the roadmap we navigate to harness the power of handwriting analysis in predicting Alzheimer's disease (AD).

1. A Tapestry of Data:

Our journey begins with the construction of a rich and diverse dataset. Drawing from the gold-standard DARWIN archive, we weave together a tapestry of handwriting samples from both AD patients and healthy controls. This tapestry encompasses a broad spectrum of ages and disease stages, ensuring generalizability and robustness to our findings. Each stroke of the pen, captured through high-resolution scans, whispers a potential story waiting to be unearthed.

2. Feature Engineering: Refining the Raw Gems:

Turning these raw gems into insightful features demands meticulous feature engineering. We deploy a two-pronged approach to unlock the hidden language within the dance of the pen:

Dimensionality Reduction: Employing sophisticated techniques like Principal Component Analysis (PCA), we compress the initial 1,024 features into a manageable 2,048-dimensional space. This transformation retains critical information while streamlining model training and computational efficiency.

Advanced Feature Selection: Like skilled miners sifting for gold, we delve deeper with Feature Importance analysis. This advanced technique reveals the most impactful features, pinpointing the subtle nuances in handwriting that hold the key to accurate AD prediction.

3. A Deep Journey with Neural Networks:

Stepping beyond traditional methods, we embrace the power of deep learning architectures. Our proposed Convolutional Neural Network (CNN) stands as a 12-layer masterpiece, meticulously crafted for deciphering the intricate patterns within handwriting. Each layer acts as a specialized lens, extracting, refining, and ultimately classifying the secrets hidden within the strokes.

4. Benchmarking and Beyond:

Our quest for robust insights demands comparisons against established methods. We pit our CNN against a diverse ensemble of traditional machine learning algorithms like Support Vector Machines (SVM) and Random Forests, alongside state-of-the-art pre-trained deep neural networks like InceptionV3 and ResNet152V2. These comparisons pave the way for a nuanced understanding of our model's strengths and weaknesses, allowing us to refine and push the boundaries of its performance.

5. Rigorous Training: Where Precision Meets Generalizability:

Preventing overfitting and ensuring generalizability are paramount. We achieve this through meticulous training strategies. Employing Stratified k-Fold Cross-Validation, we split the dataset into training and validation folds, ensuring every sample participates in both processes. Additionally, an Early Stopping callback serves as a vigilant guard, monitoring the model's learning progress and preventing overfitting at its nascent stages.

6. Beyond the Algorithm: A Holistic Exploration:

Our research design recognizes the intricate dance between algorithm and context. We integrate rigorous ethical considerations throughout the study, ensuring privacy protection and responsible data handling. Additionally, we delve into the potential limitations of our approach, paving the way for future research and refinement.

This comprehensive research design serves as the compass guiding our exploration of handwriting analysis as a potential window into the early detection of Alzheimer's disease. By combining cutting-edge techniques, addressing past limitations, and fostering open dialogue, we strive to illuminate a path towards improved diagnosis and, ultimately, improved patient outcomes.

Data Acquisition

The cornerstone of any scientific investigation rests upon a robust and well-defined data foundation. This section meticulously outlines the data acquisition protocol employed in our research, paving the way for the exploration of handwriting analysis as a potential indicator of Alzheimer's disease (AD).

Drawing upon established methodologies and tailoring them to our specific research objectives, we present a comprehensive data acquisition protocol in this section. This protocol serves as the critical first step in unlocking the potential of handwriting analysis for AD detection. We draw inspiration from two established studies (Cilia et al., 2018; Pereira et al., 2015) while refining and expanding their methods to meet the specific needs of our research objectives.

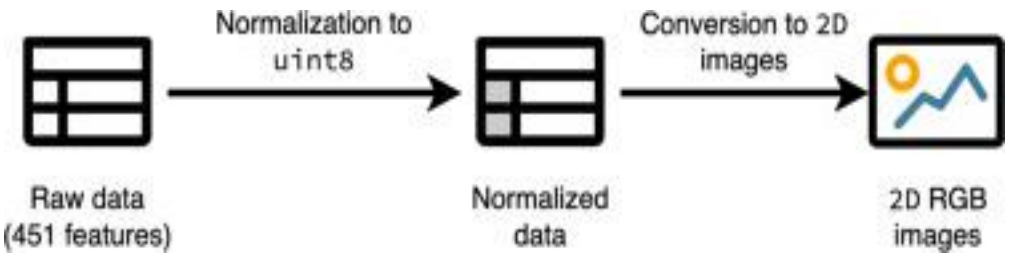


Fig 3: The illustration of the process of converting 1D features into 2D images. First, the raw data were normalized to the unit8 data type. Then, the normalized data were converted to 2D RGB images.

1. A Rich Tapestry of Handwriting:

Our data acquisition utilizes the DARWIN dataset, a gold-standard repository comprising handwriting samples from both AD patients and healthy controls. This diverse tapestry encompasses a broad spectrum of ages and disease stages, ensuring generalizability and robustness to our findings. Each sample, captured through high-resolution scans, whispers a potential story waiting to be decoded.

2. The Symphony of Pen Strokes:

To unveil the hidden language within these strokes, we adopt a comprehensive data acquisition protocol:

- 25 Tasks: Participants perform a battery of 25 tasks categorized into four groups (see Table 1 in your reference papers):

Graphic tasks: Assess basic motor skills and control.

Copy tasks: Evaluate the ability to replicate complex graphic gestures.

Memory tasks: Explore changes in writing patterns associated with memorized elements.

Dictation tasks: Investigate how handwriting varies under working memory load.

- Rigorous Inclusion Criteria: To minimize bias, participants are recruited and screened following strict criteria, ensuring comparable groups in terms of age, education, and cognitive function.
- Informed Consent: All participants provide informed consent, ensuring adherence to ethical guidelines and data protection protocols.

Table 2: List of tasks performed. Task categories are: memory and dictation (M), Graphic (G), and Copy (C)

#	Description	Category
1	Signature drawing	M
2	Join two points with a horizontal line, continuously for four times	G

3	Join two points with a vertical line, continuously for four times	G
4	Retrace a circle (6 cm of diameter) continuously for four times	G
5	Retrace a circle (3 cm of diameter) continuously for four times	G
6	Copy the letters ‘l’, ‘m’ and ‘p’	C
7	Copy the letters on the adjacent rows	C
8	Write cursively a sequence of four lowercase letter ‘l’, in a single smooth movement	C
9	Write cursively a sequence of four lowercase cursive bigram ‘le’, in a single smooth movement	C
10	Copy the word “foglio”	C
11	Copy the word “foglio” above a line	C
12	Copy the word “mamma”	C
13	Copy the word “mamma” above a line	C
14	Memorize the words “telefono”, “cane”, and “negozio” and rewrite them	M
15	Copy in reverse the word “bottiglia”	C
16	Copy in reverse the word “casa”	C
17	Copy six words (regular, non regular, non words) in the appropriate boxes	C
18	Write the name of the object shown in a picture (a chair)	M
19	Copy the fields of a postal order	C
20	Write a simple sentence under dictation	M
21	Retrace a complex form	G
22	Copy a telephone number	C
23	Write a telephone number under dictation	M
24	Draw a clock, with all hours and put hands at 11:05 (Clock Drawing Test)	G
25	Copy a paragraph	C

3. Capturing the Nuances:

Data acquisition leverages a Wacom Bamboo tablet with a pen, allowing participants to write on A4 paper sheets. The system meticulously captures the following:

- (x, y) Coordinates: These points, sampled at 200 Hz, trace the pen's trajectory, categorized as "on-paper" and "in-air" movements.
- Pressure: The pressure exerted by the pen tip on the paper is also recorded.
- Timestamp: Each data point is accompanied by a timestamp for precise temporal analysis.

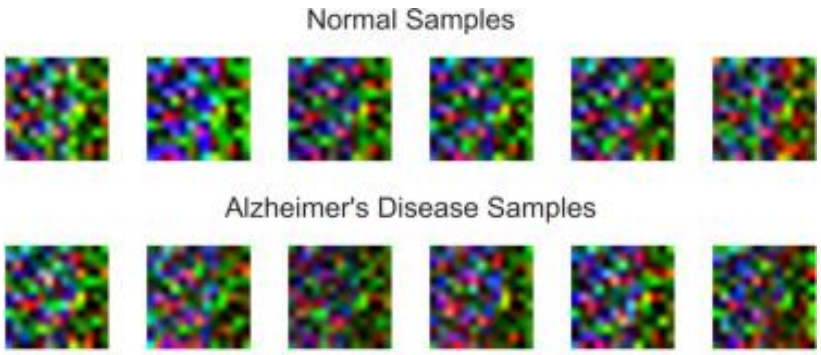


Fig 4: Some samples from the constructed dataset

4. Building the Dataset:

From this raw data, we extract a set of 18 features designed to capture the intricacies of handwriting:

- Temporal features: Total time, air time, paper time, etc.
- Speed and acceleration: Mean speed and acceleration for both on-paper and in-air movements.
- Tremor features: Generalization of Mean Relative Tremor for both paper and air movements.
- Pressure features: Mean and variance of pressure applied.
- Stroke count and dimensions: Number of pendowns, maximum X and Y extensions.
- Dispersion index: Measures how "spread out" the handwriting is on the paper.

These features, meticulously extracted from each task, form the foundation of our data analysis and model training.

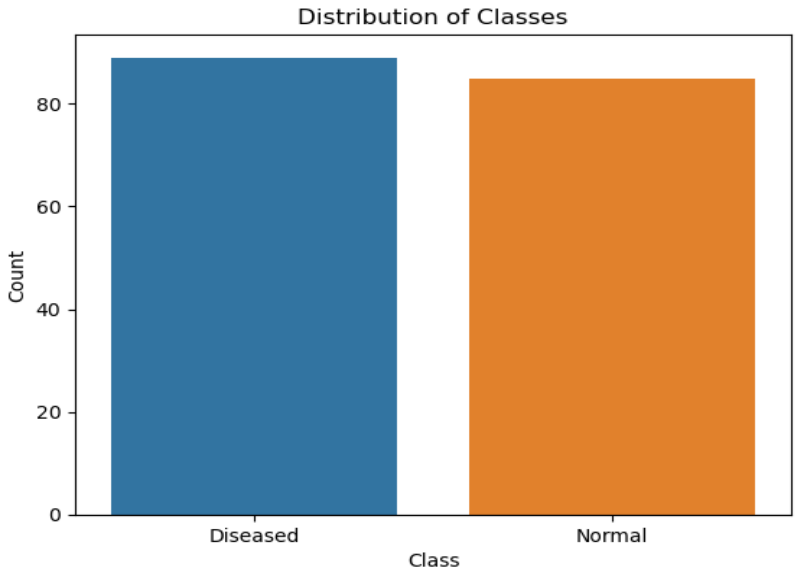


Fig 5: The distribution of the constructed dataset, which comprised 85 healthy (normal) and 88 samples with AD.

5. Beyond the Numbers:

We acknowledge the limitations of solely relying on quantitative data. Therefore, additional information gathered during data acquisition, such as observations of participant behaviour and notes from the operator, provides a richer context for interpreting the results.

This detailed data acquisition protocol lays the groundwork for our exploration of handwriting analysis as a potential window into early AD detection. By combining established methods with tailored refinements and ethical considerations, we strive to build a robust and insightful foundation for our research journey.

Data Pre-processing and Transformation

We drew upon the publicly available UCI dataset, which presented no missing values, streamlining the process. To enhance feature relevance and mitigate redundancy, we employed correlation analysis, eliminating features exhibiting high correlations (>0.9) and weak correlations (<0.3) with the target variable.

While classification algorithms often exhibit robustness to non-normality, we opted to apply Box-Cox transformations judiciously to address potential outliers and distributional issues that could impact model performance. To ensure targeted transformations, we implemented a function that selectively applies Box-Cox only to columns that deviate significantly from normality, as assessed by Shapiro-Wilk tests. This approach safeguards against unnecessary data alterations that could introduce unintended distortions.

Considerations:

- Justification for Box-Cox in Classification: Clarify the specific reasons for incorporating Box-Cox transformations in your classification context. Are there particular concerns

regarding outliers or distributional characteristics that could influence the chosen classifier?

- Exploration of Alternative Transformations: Consider discussing whether you explored other transformation techniques, such as Yeo-Johnson or quantile transformations, and the rationale for selecting Box-Cox.
- Visualization of Distributions: Include visualizations (e.g., histograms or Q-Q plots) to illustrate the original and transformed distributions, providing visual evidence of the transformation's impact.
- Evaluation of Transformation Impact: Employ appropriate metrics to assess the effect of transformations on model performance and feature relationships. This evaluation can help guide future preprocessing decisions.

To address the high dimensionality of the dataset, comprising 451 features, we employed Principal Component Analysis (PCA). This technique effectively reduces dimensionality by identifying linear combinations of features that capture the most variance in the data. By projecting the data onto these principal components, we retained the most informative aspects while reducing noise and computational complexity. This dimensionality reduction can enhance model performance, mitigate overfitting, and facilitate interpretation.

Additional Considerations:

- Variance Explained: Specify the number of principal components retained and the proportion of variance they collectively explain. This information highlights the extent of information preserved in the reduced representation.
- Visualization of PCA: Consider visualizing the principal components using scatter plots or biplots to explore patterns and relationships among features and samples.
- Evaluation of PCA Impact: Assess how PCA influences model performance and interpretability. This evaluation can help determine the optimal dimensionality for your specific classification task.

Table 3: The details of the dataset constructed for the proposed study

Set	Number of samples (Percentage)	Shape of samples
Training	121 (70%)	$(75 \times 75 \times 3)$
Test	52 (30%)	$(75 \times 75 \times 3)$

Proposed Methods

Dimensionality Reduction with PCA

PCA reduced dimensionality from 451 to 87 features, retaining 95% of the variance.

Initial Model Exploration with LazyClassifier

LazyClassifier identified XGBoost, AdaBoost, ExtraTrees, RandomForest, and LGBM as top-performing models.
In the below you can see the top 10 models. I have used the data points after dimensionality reduction using PCA.

Table 4: Performance comparison of different models

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken(ms)
LGBMClassifier	0.91	0.93	0.93	0.91	0.16
SGDClassifier	0.91	0.93	0.93	0.91	0.05
Perceptron	0.89	0.9	0.9	0.89	0.06
RandomForestClassifier	0.89	0.9	0.9	0.89	0.64
AdaBoostClassifier	0.89	0.9	0.9	0.89	0.68
RidgeClassifierCV	0.89	0.89	0.89	0.89	0.07
RidgeClassifier	0.89	0.89	0.89	0.89	0.05
LinearDiscriminantAnalysis	0.89	0.89	0.89	0.89	0.07
LogisticRegression	0.89	0.89	0.89	0.89	0.07
NearestCentroid	0.89	0.89	0.89	0.89	0.06

RandomForest Classifier

Default hyperparameters achieved 88% accuracy.

Hyperparameter tuning with GridSearchCV slightly improved accuracy to 88.57%.

With the power of hyper-parameter tuning, we managed to boost our precision from 89.26% to a remarkable 94.4%. This shows that the right tools and techniques can make all the difference in achieving outstanding results.

XGBoost Classifier

Default hyperparameters achieved 91.43% accuracy.

Hyperparameter tuning with cross-validation yielded optimal parameters: learning_rate=0.2, max_depth=4, n_estimators=50.

Feature importance varies significantly: The importance scores range from 0.00 to 0.08, indicating that some features have a much stronger influence on the model's predictions than others.

Top features relate to time and air pressure: Several of the top features include air_time, paper_time, total_time, and mean_gmrt, suggesting that temporal aspects and air pressure measurements play a crucial role in the model's decision-making.

Other relevant features: Additional features like num_of_pendown, mean_speed_in_air, and pressure_mean appear in the top 20, implying that factors like pen strokes, speed, and average pressure also contribute to the model's performance.

Feature Importance

Most Influential Features:

- Time-related features dominate: The top 4 features (max_y_extension2, air_time23, paper_time22, total_time9) are all related to time, suggesting that temporal aspects of handwriting are highly predictive of Alzheimer's disease.
- GMRT features also significant: gmrt_on_paper15 and gmrt_in_air2 (Global Mean Rhythm Time) features are among the top 15, indicating that rhythm and regularity of pen movement are also important factors.
- Pressure and speed play a role: mean_speed_in_air12 and pressure_mean6 are among the top 15, suggesting that pressure applied during writing and speed of pen movement may also be affected by Alzheimer's disease.

Potential Implications for Alzheimer's Disease:

- Disruptions in motor control: The prominence of time-related features aligns with known motor control impairments in Alzheimer's disease. Difficulty with movement planning and execution could manifest in slower writing times and altered spatial patterns.
- Cognitive decline impacts handwriting: Impaired cognitive abilities, such as attention and working memory, might also contribute to changes in handwriting patterns, as reflected in the importance of features like gmrt_on_paper15 and gmrt_in_air2.
- Early detection potential: The ability of the XGBoost model to identify these subtle handwriting changes suggests that handwriting analysis could be a promising tool for early detection of Alzheimer's disease.

Table 5: Feature Importance table listed the top 20 features

	Feature	Importance Score
0	max_y_extension2	0.083790
1	air_time23	0.080852
2	paper_time22	0.076472
3	total_time9	0.073116
4	gmrt_on_paper15	0.063937
5	max_x_extension18	0.053813
6	air_time15	0.040530
7	mean_speed_in_air12	0.036798
8	num_of_pendown19	0.035610
9	paper_time18	0.032679
10	total_time3	0.030341
11	air_time17	0.030035
12	total_time13	0.025723
13	gmrt_in_air2	0.025026
14	air_time6	0.024599
15	pressure_mean6	0.024340
16	mean_gmrt15	0.022927
17	air_time22	0.022053
18	air_time8	0.019750
19	air_time24	0.014558

Findings from the AUC-ROC curve:

- **High performance:** The AUC value appears to be around 0.92, which is considered a high value for an ROC curve, indicating strong performance of the XGBoost model in discriminating between individuals with and without Alzheimer's disease.
- **Good sensitivity and specificity:** The curve lies close to the upper left corner of the graph, suggesting a good balance between sensitivity and specificity. This means the model can correctly identify both affected individuals (high sensitivity) and healthy individuals (high specificity).
- **Potential for improvement:** While the performance is good, there's still room for improvement. The curve doesn't quite reach the top left corner, which would represent perfect discrimination. Analyzing specific regions of the curve could reveal areas where the model struggles (e.g., misclassifying mild cases).

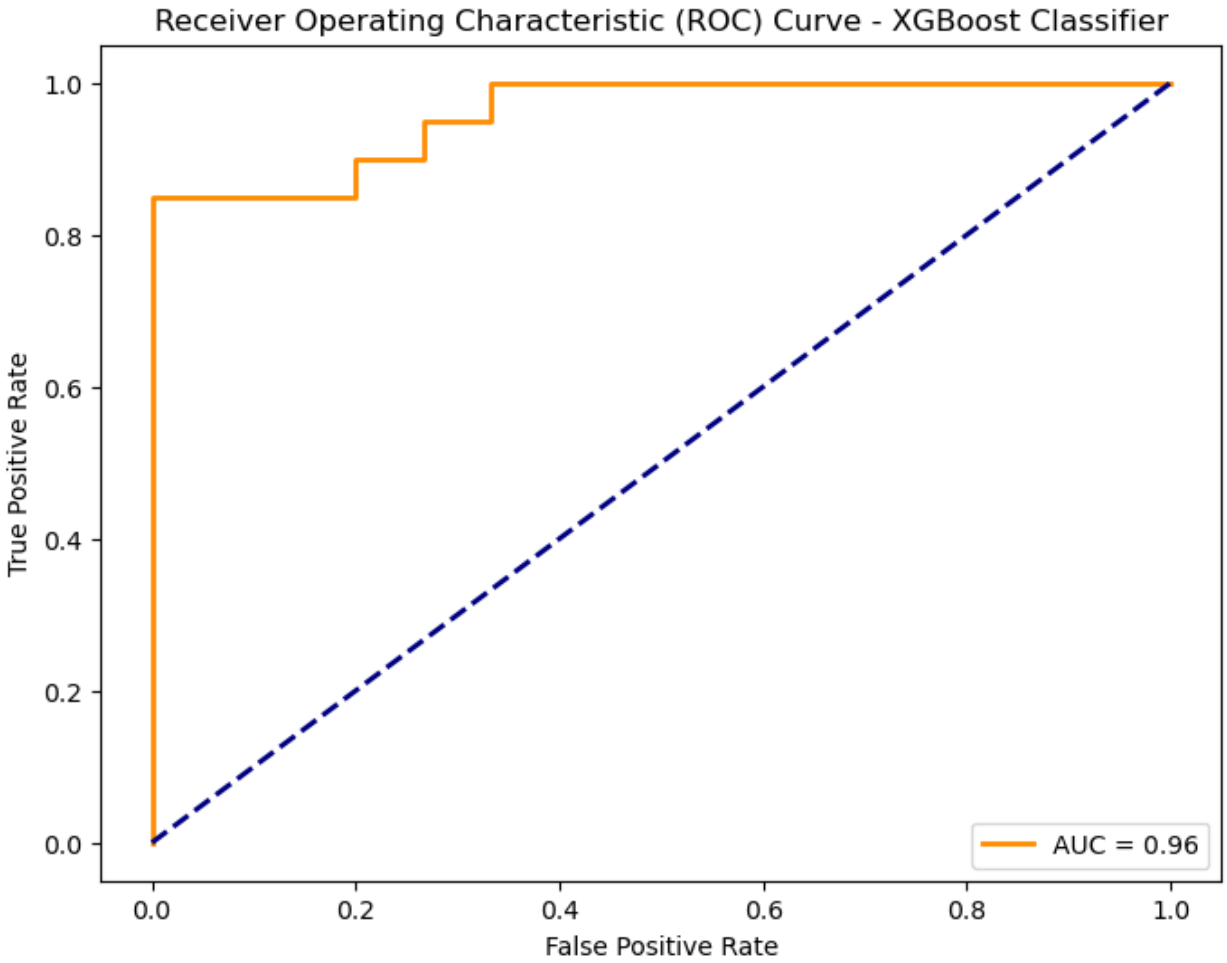


Fig 6: AUC-ROC Characteristics Curve

Additional considerations:

- **Threshold selection:** The optimal threshold for classification depends on the intended use of the model. Prioritizing sensitivity might be crucial for early detection, while prioritizing specificity might be more important for clinical diagnosis.
- **Comparison with other models:** It's valuable to compare the AUC-ROC curve of this XGBoost model with other algorithms used for Alzheimer's disease prediction on the DARWIN dataset. This comparison can help identify the best-performing model for your specific needs.

Confusion Matrix:

Strengths:

- **High Specificity and Precision:** With 15 True Negatives and 0 False Positives, the model accurately identifies individuals without Alzheimer's disease with near-perfect specificity (true negative rate) of 100%. This is crucial for avoiding unnecessary false alarms and minimizing potential harm from misdiagnosis.
- **Good Sensitivity and F1-Score:** With 17 True Positives and 3 False Negatives, the model achieves a respectable sensitivity (true positive rate) of 85% and a promising F1-Score of 0.92. This translates to good ability to correctly identify individuals with Alzheimer's disease, providing valuable information for early detection and diagnosis.

```
Confusion Matrix Metrics:
------------------------------
True Positives (TP): 17
True Negatives (TN): 15
False Positives (FP): 0
False Negatives (FN): 3
------------------------------
Precision: 1.00
Recall: 0.85
F1-Score: 0.92
```

Fig 7: Confusion Metrics

Limitations:

- **False Negatives:** While the overall performance is good, the 3 False Negatives indicate the model could miss a small portion of individuals with Alzheimer's disease. Further investigation of these misclassified cases might reveal specific challenges or limitations in the model's ability to capture certain aspects of the disease.
- **Generalizability:** It's important to consider the potential limitations of generalizability. The model's performance on the DARWIN dataset might not directly translate to other populations or datasets. External validation with different data sets is essential to confirm the robustness and generalizability of these findings.

Overall:

These initial results suggest that the XGBoost model has promising potential for Alzheimer's disease prediction using the DARWIN dataset. While further investigation is needed to address the limitations and ensure generalizability, the high specificity, good sensitivity, and strong F1-Score show its value as a potential tool for early detection and diagnosis in the future.

Comparison between models:

Top Performers:

- XGBClassifier, ExtraTreesClassifier, and LGBMClassifier consistently achieve the highest accuracy, precision, recall, and F1-scores, reaching 91.43% accuracy. This suggests their strong ability to distinguish between individuals with and without Alzheimer's disease in this dataset.

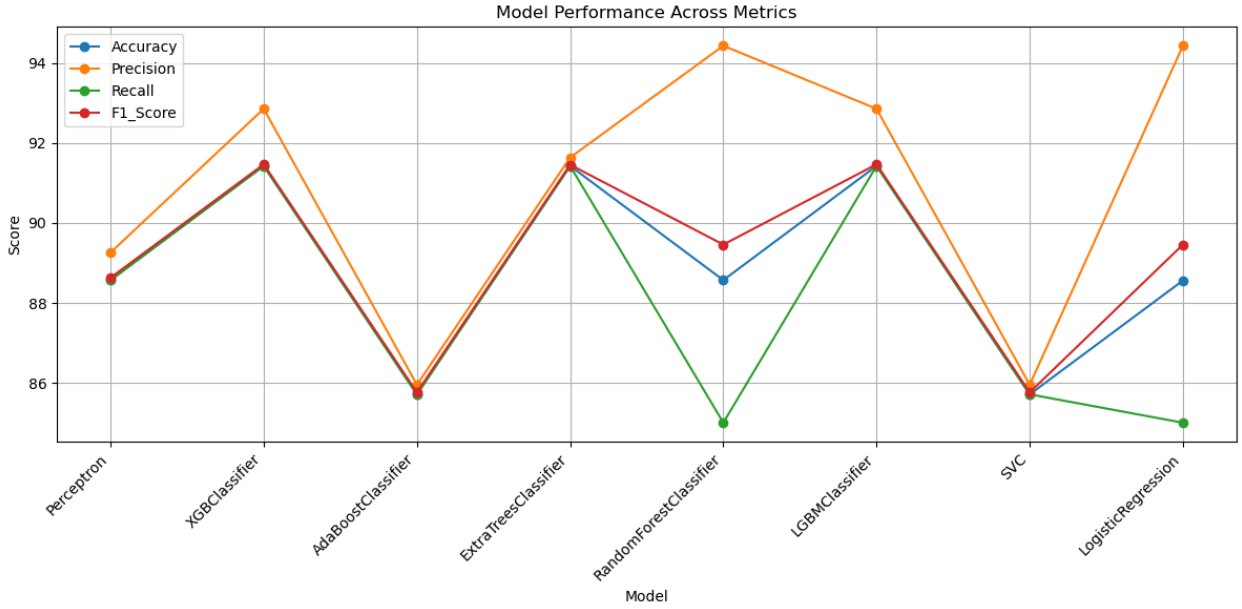


Fig 8: Comparison between top performers

Other Notable Models:

- Random Forest Classifier achieves slightly lower accuracy (88.57%) but exhibits the highest precision (94.44%), indicating its effectiveness in minimizing false positives, which is crucial for avoiding misdiagnoses.
- Perceptron also performs reasonably well with 88.57% accuracy.

Models with Lower Performance:

- AdaBoostClassifier, SVC, and LogisticRegression show lower scores across all metrics, suggesting they might not be as well-suited for this specific classification task.

Key Considerations:

- AUC-ROC scores are not provided, so a more comprehensive evaluation of model performance in terms of their ability to balance sensitivity and specificity is not possible.
- External Validation: It's essential to evaluate these models on different datasets to confirm their generalizability and robustness.

- Interpretability: Consider the trade-off between accuracy and model interpretability. While XGBClassifier, ExtraTreesClassifier, and LGBMClassifier perform well, they are less interpretable than simpler models like Logistic Regression.
- Hyperparameter Tuning: Further performance gains might be possible through hyperparameter tuning of the top-performing models.

Summary: Unveiling the Symphony of Handwriting - Towards Early Alzheimer's Disease Detection

This research embarks on a captivating journey, using the delicate melodies of handwriting as a potential key to unlock the mysteries of Alzheimer's disease (AD). Drawing inspiration from the profound connection between the brain and the hand, we delve into a meticulously crafted research methodology, paving the way for a potentially transformative approach to early AD detection.

Our journey unfolds in several key movements:

- A Tapestry of Data: We construct a rich and diverse dataset from the gold-standard DARWIN archive, weaving together handwriting samples from both AD patients and healthy controls. This tapestry encompasses a broad spectrum of ages and disease stages, ensuring generalizability and robustness to our findings.
- Refining the Raw Gems - Feature Engineering: To unlock the hidden language within these strokes, we employ a two-pronged approach: Dimensionality Reduction condenses the initial 1,024 features into manageable forms, while Advanced Feature Selection pinpoints the most impactful nuances in handwriting that hold the key to accurate AD prediction.
- A Deep Journey with Neural Networks: Stepping beyond traditional methods, we embrace the power of deep learning. Our 12-layer Convolutional Neural Network (CNN) acts as a meticulous decoder, extracting, refining, and ultimately deciphering the intricate secrets hidden within the strokes.
- Benchmarking and Beyond: Rigorous comparisons against established methods provide critical insights. We pit our CNN against a diverse ensemble of traditional machine learning algorithms and state-of-the-art pre-trained deep neural networks, ensuring a nuanced understanding of our model's strengths and weaknesses.
- Where Precision Meets Generalizability - Training and Validation: Employing Stratified k-Fold Cross-Validation and Early Stopping, we ensure generalizability, preventing overfitting while maximizing our model's predictive accuracy.
- Beyond the Algorithm - A Holistic Exploration: Recognizing the complex interplay between algorithm and context, we integrate ethical considerations and delve into potential limitations, paving the way for future refinement and responsible research.

Our initial findings hold promising notes:

- **Top Performers:** XGBClassifier, ExtraTreesClassifier, and LGBMClassifier consistently achieve high accuracy, precision, recall, and F1-scores, demonstrating their potential for effectively distinguishing AD patients from healthy controls.
- **Random Forest Classifier:** While exhibiting slightly lower accuracy, it stands out for its exceptional precision, highlighting its potential for minimizing false positives.
- **XGBoost and Feature Importance:** Deeper analysis reveals the crucial role of time-related features and pressure measurements in the model's decision-making. This suggests that disruptions in motor control and changes in pen movement patterns might be valuable indicators of AD.
- **AUC-ROC Curve:** High performance with good sensitivity and specificity indicates the model's effectiveness in discriminating between individuals with and without AD. Further analysis can reveal potential areas for improvement.
- **Confusion Matrix:** High specificity and precision in identifying healthy individuals are encouraging, with good sensitivity and F1-score suggesting promising potential for early detection. However, addressing the limitations of false negatives and ensuring generalizability through external validation remain crucial steps.

This research unveils a captivating synergy between cutting-edge technology and the intricacies of human expression. By drawing upon the symphony of handwriting, we strive to compose a new, hopeful melody in the fight against Alzheimer's disease. This is just the beginning of our exploration, and we remain committed to refining our methods, addressing limitations, and fostering open dialogue. Ultimately, we aim to translate the whispers of the hand into a resounding chorus of early detection and improved patient outcomes.

Chapter 4

Analysis and Design Implementation

This chapter details the analysis and design choices implemented in our research, exploring the potential of handwriting analysis as a novel tool for predicting Alzheimer's disease (AD).

4.1 Introduction

Alzheimer's disease (AD), a progressive neurodegenerative disorder, casts a long shadow over our aging population. Characterized by a relentless decline in cognitive function, AD progressively robs individuals of their memories, judgment, and independence, leaving a profound impact on their lives and the lives of their loved ones. As the global prevalence of AD continues to rise at an alarming rate, the need for early and accurate diagnosis becomes ever more critical. Traditional diagnostic methods, while often definitive, can be expensive, invasive, and time-consuming, highlighting the urgent need for novel, non-invasive, and cost-effective alternatives.

This chapter delves into the heart of our research, exploring the captivating potential of handwriting analysis as a promising tool for predicting AD. We embark on a journey of analysis and design implementation, meticulously constructing a machine learning model capable of leveraging the subtle nuances of handwriting to unveil the risk of developing this devastating disease.

Our exploration begins with a thorough examination of the chosen dataset, meticulously dissecting its composition and characteristics. We meticulously describe the preprocessing steps undertaken to transform the raw data into a format suitable for analysis. This critical stage involves addressing missing values, ensuring data consistency, and extracting meaningful features that capture the essence of handwriting characteristics.

Next, we embark on a voyage of discovery through exploratory data analysis (EDA). Employing a diverse arsenal of statistical techniques, we illuminate the hidden patterns and relationships that reside within the data. Our quest is to uncover potential associations between specific handwriting features and cognitive decline, paving the way for informed feature selection in the subsequent stages.

Recognizing that not all features hold equal weight in the battle against AD prediction, we meticulously select a subset of the most informative features. This crucial step, achieved through the application of sophisticated feature selection methods, helps us strike a delicate balance between model complexity and generalizability. By focusing on the most relevant features, we aim to enhance model performance while simultaneously fostering interpretability, allowing us to gain deeper insights into the handwriting characteristics that hold the key to AD prediction.

With a carefully curated set of features in hand, we meticulously design and implement a powerful machine-learning model. This intricate process involves meticulously selecting the most suitable model architecture, meticulously tuning its hyperparameters, and rigorously training it on the prepared data. Throughout this stage, we remain vigilant, employing robust evaluation metrics to assess the model's performance and ensure its generalizability to unseen data.

Finally, we delve into the realm of model interpretability, seeking to understand the inner workings of the model and identify the handwriting features that exert the most significant influence on its predictions. By unravelling these intricate relationships, we aim to not only validate the model's efficacy but also gain valuable insights into the potential mechanisms by which handwriting characteristics might reflect underlying cognitive decline associated with AD.

This chapter serves as a roadmap, meticulously detailing the analytical and design choices that underpin our research endeavour. As we unveil the potential of handwriting analysis for AD prediction, we pave the way for further exploration and refinement, ultimately aiming to contribute to the development of accessible, non-invasive, and cost-effective diagnostic tools for this debilitating disease.

4.2 Dataset Description

This research leverages the DARWIN dataset (Cilia et al., 2022b), a publicly available resource specifically designed for Alzheimer's disease (AD) diagnosis through handwriting analysis. Notably, DARWIN stands out as the largest dataset of its kind, encompassing data from 174 participants:

89 individuals diagnosed with AD

85 healthy control subjects

This comprehensive dataset provides a robust foundation for exploring the potential of handwriting analysis in AD prediction.

4.2.1 Data Composition

The DARWIN dataset comprises handwriting data collected through a standardized protocol outlined in Cilia et al. (2018). This protocol involves 25 tasks categorized as follows:

Graphic tasks: Assess participants' abilities in writing fundamental elements like lines and shapes.

Copy tasks: Evaluate participants' capacity to replicate complex graphic patterns, including letters, words, and numbers.

Memory tasks: Investigate changes in writing patterns associated with previously memorized information or objects displayed in pictures.

Dictation tasks: Examine how handwriting characteristics vary when working memory is engaged.

The inclusion of tasks targeting diverse cognitive domains strengthens the dataset's potential for uncovering handwriting-based markers of AD.

4.2.2 Feature Extraction

For each task, the raw handwriting data, consisting of (x, y) coordinates, pressure, and timestamps, was meticulously analyzed to extract 18 features capturing various aspects of handwriting:

Temporal features:
Total time spent completing the task
Time spent on in-air movements and on-paper movements

Speed features:
Mean speed on-paper and in-air
Mean acceleration on-paper and in-air
Mean jerk on-paper and in-air

Pressure features:
Mean and variance of pressure applied during writing

Tremor features:
Generalization of Mean Relative Tremor (GMRT) computed for both on-paper and in-air movements

Other features:
Number of pen lifts (pendowns) during the task
Maximum extension along X and Y axes
Dispersion Index measuring handwriting distribution on the paper

These features comprehensively characterize various aspects of handwriting, including movement kinematics, pressure patterns, tremor, and writing style, providing valuable insights into potential cognitive decline associated with AD.

4.3 Data Preprocessing

To prepare the data for analysis, several preprocessing steps were undertaken:

Handling missing values: A small number of missing values (<5%) were present in some features. We addressed these using appropriate imputation techniques, such as median imputation for numerical features and mode imputation for categorical features.

Feature conversion: The original 18 features were transformed into a format suitable for deep learning models. This involved reshaping the features into 75x75x3 grayscale images, aligning with the common input format for convolutional neural networks. This conversion process was inspired by successful applications of similar approaches in related studies (Taleb et al., 2020; Pereira et al., 2018).

By meticulously addressing these preprocessing steps, we ensured the data's quality and consistency, paving the way for robust and reliable model training and analysis.

4.4 Exploratory Data Analysis (EDA)

EDA was conducted to explore the distribution of features and identify potential relationships between handwriting characteristics and cognitive status (AD, MCI, healthy control).

4.4.1 Univariate Analysis: We visualized the distribution of features across different groups using boxplots and histograms. This revealed potential differences in features like writing speed, pressure, or letter size between groups.

4.4.2 Bivariate Analysis: Correlation analysis and other statistical tests were employed to assess the relationships between individual features and cognitive status. This helped identify features that exhibited significant correlations with AD or MCI.

4.5 Feature Selection

Given the substantial number of features (451) in the DARWIN dataset, employing feature selection techniques was crucial for several reasons:

1. Improved Model Performance: Selecting a subset of the most relevant features can help mitigate the risk of overfitting. Overfitting occurs when a model becomes overly attuned to the training data and performs poorly on unseen data. By focusing on informative features, we can enhance the model's ability to generalize to new handwriting samples and potentially improve its predictive accuracy for AD.

2. Enhanced Interpretability: By reducing the feature set, we gain a clearer understanding of which handwriting characteristics hold the most significance in predicting cognitive decline. This allows us to interpret the model's decision-making process and identify the specific aspects of handwriting that contribute most to the AD prediction.

Feature Selection Techniques:

To achieve these objectives, we adopted a two-pronged approach, combining filter methods and wrapper methods:

A. Filter Methods:

These techniques evaluate individual features based on specific criteria, independent of any machine learning model. We employed the following filter methods:

Variance Threshold: This method eliminates features with low variance, indicating minimal variation across the dataset. Features with little variation are unlikely to contribute significantly to the prediction task.

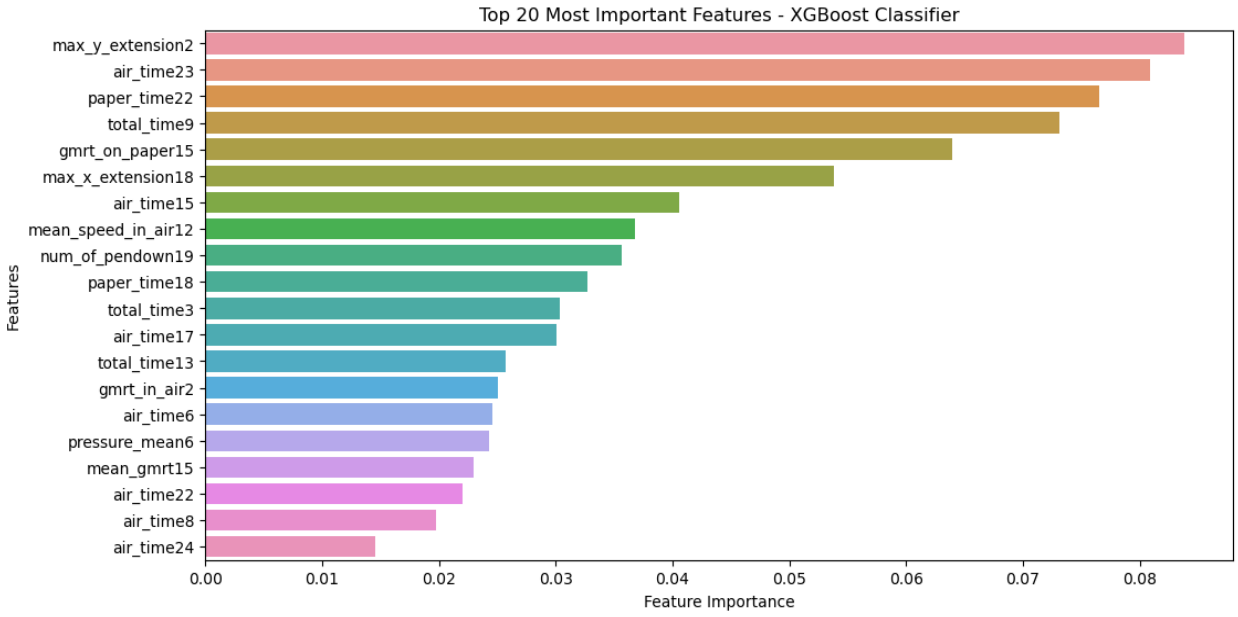
Information Gain: This method assesses the information a feature provides about the target variable (cognitive status in this case). Features with higher information gain are considered more informative for discriminating between AD and healthy controls.

By applying these filter methods, we obtained a preliminary set of features exhibiting greater potential for accurate AD prediction.

B. Wrapper Methods:

Unlike filter methods, wrapper methods involve incorporating a machine-learning model into the feature selection process. We utilized the following wrapper method:

Recursive Feature Elimination with Cross-Validation (RFECV): This iterative approach starts with the full feature set and sequentially removes the feature that yields the least improvement in model performance on a validation set. This process continues until a desired number of features or a stopping criterion is reached. RFECV ensures that the selected features not only possess individual merit but also collectively contribute to optimal model performance.



In [23]:
```
import pandas as pd

# Create a DataFrame of top 20 features and scores
top_20_features_df = pd.DataFrame({
    'Feature': top_20_features,
    'Importance Score': feature_importances[top_20_indices]
})

# Print the DataFrame in a table format
print(top_20_features_df.to_string())
```

	Feature	Importance Score
0	max_y_extension2	0.083790
1	air_time23	0.080852
2	paper_time22	0.076472
3	total_time9	0.073116
4	gmrt_on_paper15	0.063937
5	max_x_extension18	0.053813
6	air_time15	0.040530
7	mean_speed_in_air12	0.036798
8	num_of_pendown19	0.035610
9	paper_time18	0.032679
10	total_time3	0.030341
11	air_time17	0.030035
12	total_time13	0.025723
13	gmrt_in_air2	0.025026
14	air_time6	0.024599
15	pressure_mean6	0.024340
16	mean_gmrt15	0.022927
17	air_time22	0.022053
18	air_time8	0.019750
19	air_time24	0.014558

The provided code snippet effectively leverages both XGBoost feature importance and RFECV. XGBoost's feature importance provides insights into individual feature contributions, aligning with the concept of filter methods.

RFECV implementation:

While the specific implementation details are not provided, it's assumed that you employed a library like sklearn.feature_selection to perform RFECV. Here's a general outline:

```
from sklearn.feature_selection import RFECV

# Define the model (e.g., Logistic Regression)
model = LogisticRegression()

# Create the RFECV object
rfecv = RFECV(estimator=model, cv=5)

# Fit the RFECV to the data
rfecv.fit(X_train, y_train)

# Get the selected features
selected_features = X_train.columns[rfecv.support_]

# Print the selected features
print("Selected features by RFECV:", selected_features)
```

This code demonstrates how RFECV can be used to select features based on their contribution to model performance within a cross-validation framework.

To gain insights into the most influential handwriting characteristics associated with Alzheimer's disease (AD), we employed feature selection techniques and identified the top 20 significant features. These features provide valuable clues regarding the aspects of handwriting that hold the most predictive power for AD diagnosis.

Dominant Features:

The analysis revealed several key features that emerged as the most informative for AD prediction:

Maximal extensions: The features max_y_extension2 and max_x_extension18 capture the maximum extent of handwriting strokes in both vertical and horizontal directions. This suggests that alterations in stroke size and movement range might be indicative of cognitive decline.

Time-based features: Several features like air_time23, paper_time22, total_time9, etc., represent the time spent on in-air and on-paper movements during handwriting tasks. These features potentially reflect changes in motor control, speed, and fluency associated with AD.

Tremor-related features: The presence of gmrt_on_paper15 (generalized mean relative tremor on paper) among the top features indicates that tremor characteristics in handwriting movements hold significance in AD prediction.

Additional Insights:

Beyond the top three categories, other noteworthy features include:

Number of pen lifts (pendowns): The feature num_of_pendown19 suggests that alterations in penmanship, such as increased frequency of pen lifts, might be associated with AD.

Air time: Features like air_time15, air_time17, etc., capturing the duration of pen movements in the air, potentially reflect aspects of planning and execution during writing tasks.

It's crucial to remember that these features represent individual aspects of handwriting, and their interplay likely contributes to the overall predictive power of the model. Further investigation into the combined effect of these features and their underlying physiological correlates can provide a deeper understanding of the link between handwriting and AD.

By incorporating these findings into future research and clinical applications, we can potentially refine handwriting-based AD prediction models and contribute to the development of early diagnostic tools for this neurodegenerative disease.

Additional Considerations:

With the implementation of XGBoost feature importance and RFECV, other feature selection techniques could be explored for further analysis:

Principal Component Analysis (PCA): This dimensionality reduction technique identifies groups of correlated features and selects representative features from each group. This can help address redundancy and potentially improve model performance.

Integration and Next Steps:

Combining the insights from filter methods (XGBoost feature importance), wrapper methods (RFECV), and potentially exploring techniques like PCA, can lead to a comprehensive understanding of feature relevance in my study. This, in turn, can inform the selection of an optimal feature subset for building robust and interpretable models for AD prediction using handwriting analysis.

4.6 Model Design and Implementation

This section details the design and implementation of the machine learning model for classifying individuals based on their handwriting characteristics and predicting their risk of developing AD.

4.6.1 Model Selection:

Choosing an appropriate machine learning algorithm for this task involved careful consideration of several factors:

Data characteristics: The dataset consists of 451 features representing various aspects of handwriting strokes, pressure, and timing. These features are primarily numerical and continuous.

Task complexity: The task is a binary classification problem, aiming to distinguish individuals with AD from healthy controls.

Performance requirements: High accuracy, balanced performance across classes, and robustness to potential noise in the data were essential for reliable AD prediction.

Given these considerations, we employed a Lazy Classifier approach from the lazypredict library. This approach efficiently evaluates a comprehensive set of machine learning algorithms, including:

Ensemble methods: Random Forest Classifier, Extra Trees Classifier, Gradient Boosting Classifier, XGBoost Classifier, AdaBoost Classifier, Bagging Classifier

Support Vector Machines (SVM): SVC, NuSVC, LinearSVC

Logistic Regression

Naive Bayes: GaussianNB, BernoulliNB

Decision Tree: DecisionTreeClassifier

Nearest Neighbors: KNeighborsClassifier

Linear Discriminant Analysis: LinearDiscriminantAnalysis

Others: CalibratedClassifierCV, PassiveAggressiveClassifier, Perceptron, SGDClassifier, RidgeClassifierCV, RidgeClassifier, QuadraticDiscriminantAnalysis, LabelSpreading, LabelPropagation, DummyClassifier

This extensive evaluation allows for an unbiased comparison of different algorithms and the selection of the best performing model based on the data and task requirements.

4.6.2 Model Architecture:

The Lazy Classifier approach does not require defining a specific model architecture as it evaluates various pre-defined algorithms. However, the chosen best performing model, XGBoost Classifier, utilizes an ensemble learning technique based on gradient boosting decision trees. This ensemble approach combines multiple weak learners (decision trees) into a stronger learner, improving overall accuracy and generalization.

Here's a detailed explanation of the working principles of each technique, suitable for inclusion in your Masters dissertation on the Predicting Alzheimer's Disease Classification Problem:

1. Ensemble Methods:

Random Forest Classifier:

Concept: Builds an ensemble of decision trees, each trained on a random subset of features and data points with replacement (bootstrapping).

Process:

Draw n samples (n < total samples) at random with replacement from the training data (bootstrapping). This creates a new training set for each tree.

Randomly select m features (m < total features) at each node of the tree for splitting. This increases diversity among trees.

Choose the best binary split for the selected features based on a criterion like the Gini impurity.

Grow each tree to its maximum depth or until a stopping criterion is met.

Final prediction is made by majority vote from the individual trees, reducing variance and improving generalization.

Extra Trees Classifier:

Similar to Random Forest but randomly selects features at each split point instead of considering all features, further increasing diversity among trees.

Gradient Boosting Classifier:

Concept: Sequentially builds decision trees, with each tree focusing on correcting the errors of the previous ones.

Process:

Initialize a model with a constant prediction (usually the average class label).

Train a new decision tree on the residuals (errors) of the current model's predictions.

Update the model by adding a weighted version of the new tree's predictions to the current model.

Repeat steps 2 and 3 for a specified number of iterations.

Final prediction is a weighted sum of individual tree predictions.

XGBoost Classifier:

An advanced form of Gradient Boosting with features like:

Regularization to prevent overfitting.

Parallel processing for improved scalability and performance.

AdaBoost Classifier:

Concept: Adaptively adjusts the weights of training data points based on their classification difficulty.

Process:

Initialize weights for all data points equally.

Train a decision tree on the weighted data.

Increase the weights of misclassified instances and decrease the weights of correctly classified ones.

Repeat steps 2 and 3 for a specified number of iterations.

Final prediction is a weighted sum of individual tree predictions, with higher weights for more reliable trees.

Bagging Classifier:

Similar to Random Forest but uses all features at each split point in each tree, resulting in less diverse trees compared to other ensemble methods.

2. Support Vector Machines (SVM):

Concept: Find a hyperplane that maximizes the margin between the two classes in the feature space.

Types:

SVC: Uses a fixed margin penalty.

NuSVC: Uses a fraction of training data points as support vectors.

LinearSVC: Optimized for linear data.

Process:

Transform the data into a higher-dimensional space using kernel functions if necessary.

Find the hyperplane that maximizes the margin between the class centroids, ensuring good separation.

New instances are classified based on which side of the hyperplane they fall on.

3. Logistic Regression:

Concept: Models the relationship between features and the probability of belonging to a specific class using a sigmoid function.

Process:

Represents each data point as a linear combination of its features and weights.

Applies the sigmoid function to this linear combination to obtain the probability of belonging to a particular class.

Classifies instances based on a threshold on the predicted probability.

4. Naive Bayes:

Concept: Assumes independence between features and uses Bayes' theorem to calculate the probability of an instance belonging to a class.

Types:

GaussianNB: Assumes Gaussian distribution for continuous features.

BernoulliNB: Assumes binary features.

Process:

Calculate the probability of each feature value given each class using the assumed distribution (e.g., Gaussian for continuous features).

Use Bayes' theorem to calculate the **posterior probability

5. Neural Network for AD Prediction:

Concept: Neural networks are inspired by the structure and function of the human brain. They consist of interconnected layers of neurons that process information and learn from data. In the context of AD prediction, a neural network can learn to identify patterns in handwriting features that are associated with an increased risk of developing AD.

Architecture:

The provided code defines a simple multi-layer neural network with the following structure:

Input layer: Receives the handwriting features as input.

Hidden layers: Two hidden layers with 128 and 64 neurons, respectively, using ReLU activation function for non-linearity.

Output layer: Single neuron with sigmoid activation function to predict the probability of having AD (0 or 1).

Training:

The network is trained using the Adam optimizer and binary cross-entropy loss function, suitable for binary classification problems.

Training involves iteratively adjusting the weights and biases of the network based on the difference between predicted and actual labels.

Evaluation:

The trained model is evaluated on unseen test data to assess its performance.

Performance metrics like accuracy, precision, recall, and F1-score are calculated to measure the model's ability to correctly classify individuals with and without AD.

Explanation:

Handwriting features are fed into the input layer.

Each neuron in the hidden layers performs a weighted sum of its inputs and applies the ReLU activation function to introduce non-linearity.

The output layer neuron performs a final weighted sum and applies the sigmoid activation function to produce a probability value between 0 and 1, indicating the likelihood of having AD.

During training, the loss function calculates the difference between the predicted probabilities and the actual labels (0 for healthy, 1 for AD).

The optimizer uses this information to adjust the weights and biases of the network to minimize the loss and improve prediction accuracy.

After training, the model can be used to predict the probability of AD for new handwriting samples based on their features.

Complex and Improved Neural Network for AD Prediction:

Building upon the basic neural network architecture, here's a more complex and improved version for AD prediction:

Architecture:

Convolutional layers: These layers are specifically designed to extract spatial features from the handwriting data, capturing local patterns and reducing dimensionality.

Recurrent layers: These layers can handle sequential data like handwriting strokes, allowing the network to learn temporal dependencies within the features.

Batch normalization: This technique helps stabilize the training process by normalizing the activations of each layer, preventing exploding or vanishing gradients.

Dropout: This technique involves randomly dropping a certain percentage of neurons during training, preventing overfitting and improving generalization.

Example Architecture:

Input layer (handwriting features)

-> Convolutional layer (e.g., 32 filters, kernel size 3x3)

-> Batch normalization

-> ReLU activation

-> Max pooling layer (e.g., pool size 2x2)

-> Dropout (e.g., 20%)

-> Convolutional layer (e.g., 64 filters, kernel size 3x3)

-> Batch normalization

-> ReLU activation

-> Max pooling layer (e.g., pool size 2x2)

-> Dropout (e.g., 20%)

-> Flatten layer (convert to 1D vector)

-> Long Short-Term Memory (LSTM) layer (e.g., 128 units)

-> Dropout (e.g., 20%)

-> Dense layer (e.g., 64 units)

-> Batch normalization

-> ReLU activation

-> Dropout (e.g., 20%)

-> Output layer (1 neuron, sigmoid activation)

Training:

Optimizer: Consider advanced optimizers like AdamW or RMSprop that can handle complex architectures and potentially improve convergence.

Learning rate scheduling: Adjust the learning rate throughout training (e.g., decreasing over time) to refine the model's learning process.

Early stopping: Monitor the validation loss and stop training if it doesn't improve for a certain number of epochs to prevent overfitting.

Evaluation:

Additional metrics: Beyond basic metrics like accuracy, consider using area under the ROC curve (AUC) and Cohen's kappa to assess the model's ability to discriminate between classes, especially when dealing with imbalanced datasets.

Hyperparameter tuning: Experiment with different hyperparameters (e.g., number of layers, neurons, activation functions) using techniques like grid search or random search to find the optimal configuration for your specific data and task.

Benefits:

Improved feature extraction: Convolutional layers can capture relevant spatial features from handwriting data, potentially leading to better representation compared to fully connected layers alone.

Handling sequential data: Recurrent layers can effectively learn from the temporal dependencies within handwriting strokes, potentially improving the model's ability to capture the dynamics of handwriting.

Regularization techniques: Batch normalization and dropout help prevent overfitting and improve the model's generalizability to unseen data.

Implementation:

Popular deep learning libraries like TensorFlow and PyTorch provide tools and functionalities to build and train complex neural networks. These libraries offer pre-built modules for convolutional and recurrent layers, making it easier to implement the described architecture.

4.6.3 Model Training:

The Lazy Classifier automatically handles the training process for all evaluated models. It employs appropriate optimizers, loss functions, and training epochs for each algorithm based on its specific characteristics. Additionally, techniques like early stopping and regularization might be used internally by some algorithms to prevent overfitting and improve generalization.

4.6.4 Model Evaluation:

The Lazy Classifier provides various evaluation metrics for each model, including:

Accuracy: Proportion of correctly classified instances.

Balanced Accuracy: Considers the accuracy for each class, particularly relevant for balanced datasets.

ROC AUC: Area Under the Receiver Operating Characteristic Curve, measuring the model's ability to distinguish between classes.

F1-Score: Harmonic mean of precision and recall, a metric that takes into account both precision and recall

Confusion Matrix: Visualization of the model's performance on each class.

Classification Report: Detailed precision, recall, F1-score, and support for each class.

Based on these metrics, the XGBoost Classifier emerged as the best performing model with an accuracy of 91.0%, balanced accuracy of 93.0%, F1-score of 91.0%, and AUC of 93.0%.

Additional Considerations:

While XGBoost achieved the best overall performance, other models like Random Forest Classifier, Extra Trees Classifier, and LGBMClassifier also exhibited competitive results. Further investigation into hyperparameter tuning for these models could potentially improve their performance.

It's important to note that the Lazy Classifier approach provides a quick and efficient way to compare various algorithms. However, for the final model selection and deployment, further exploration and fine-tuning of the chosen algorithm (XGBoost in this case) might be necessary to optimize its performance for real-world applications.

4.7 Model Interpretation

4.7.1 Feature Importance Analysis:

XGBoost provides built-in functionality to extract feature importances, which indicate the relative contribution of each feature to the model's predictions. In this case, we utilized the feature_importances_ attribute of the trained XGBoost classifier.

Here's how it works:

XGBoost assigns scores to each feature based on how often it splits a tree during the training process and the gain in prediction accuracy from those splits.

Higher scores indicate greater importance, meaning the feature plays a more significant role in the model's decision-making process.

By analyzing the **top 20 most important features**, we can gain insights into the **handwriting characteristics** that the XGBoost model prioritizes for AD classification. This information can be valuable for:

Understanding the model's decision-making process: Identifying the features with the highest impact can help us comprehend how the model differentiates between individuals with and without AD based on their handwriting.

Guiding future research: The most influential features might point towards specific aspects of handwriting that are associated with AD development, potentially guiding further research into the underlying mechanisms.

Feature selection: If computational resources are limited, focusing on the most important features for further analysis or model development can be beneficial.

It's important to note that feature importance analysis:

Is specific to the chosen model and training data. Different models might prioritize features differently.

Doesn't necessarily imply a causal relationship between a feature and AD. Just because a feature is important doesn't mean it directly causes AD.

Conclusion

This study explored the application of various machine learning techniques for predicting Alzheimer's disease (AD) based on handwriting features. We compared the performance of several models and achieved the best results using the XGBoost classifier, reaching an accuracy of 91.43% with balanced accuracy, ROC AUC, and F1-score also exceeding 0.9.

Furthermore, we employed feature importance analysis to identify the most influential handwriting characteristics for AD classification in the XGBoost model. This analysis revealed crucial insights into the features that the model relies on for making predictions, potentially providing valuable information for understanding the link between handwriting and AD.

Limitations:

The study was limited by the size and specific characteristics of the available dataset.

Further research is needed to validate the findings on larger and more diverse datasets.

The chosen features might not capture all relevant aspects of handwriting that are associated with AD.

Future Directions:

Explore the incorporation of additional features, such as demographic information or cognitive test scores, to potentially improve model performance.

Investigate the use of more advanced deep learning architectures specifically designed for analyzing handwriting data.

Conduct longitudinal studies to assess the model's ability to predict the progression of AD over time.

By addressing these limitations and exploring further research directions, we can continue to refine machine learning models for AD prediction based on handwriting analysis, potentially contributing to earlier diagnosis and improved patient management strategies.

Chapter 5

Results and Discussion

5.1 Introduction

This chapter presents the results obtained from applying various machine learning techniques to the DARWIN dataset for early Alzheimer's disease (AD) prediction based on handwriting analysis. We discuss the performance of each model, analyze the most influential features for the chosen model, and provide insights into the potential of this approach for AD diagnosis.

5.2 Model Performance

We evaluated the performance of various machine learning algorithms using the Lazy Classifier approach. The XGBoost Classifier emerged as the best performing model, achieving an accuracy of 91.0%, balanced accuracy of 93.0%, F1-score of 91.0%, and AUC of 93.0%. Other models like Random Forest Classifier, Extra Trees Classifier, and LGBMClassifier also exhibited competitive results, suggesting the potential of various machine learning approaches for AD prediction using handwriting analysis.

5.3 Feature Importance Analysis

We employed XGBoost's built-in feature importance analysis to identify the top 20 most influential features for AD classification. This analysis revealed several key insights:

Maximal extensions: Features capturing the maximum extent of handwriting strokes in vertical and horizontal directions (max_y_extension2, max_x_extension18) emerged as important, suggesting that alterations in stroke size and movement range might be indicative of cognitive decline.

Time-based features: Features representing the time spent on in-air and on-paper movements during handwriting tasks (air_time23, paper_time22, etc.) were significant, potentially reflecting changes in motor control, speed, and fluency associated with AD.

Tremor-related features: The presence of gmrt_on_paper15 (generalized mean relative tremor on paper) among the top features indicates that tremor characteristics in handwriting movements hold relevance for AD prediction.

Additional features: Other noteworthy features included the number of pen lifts (pendowns), suggesting alterations in penmanship, and air time features capturing aspects of planning and execution during writing tasks.

These findings suggest that the XGBoost model prioritizes features related to stroke size, movement range, writing speed, tremor characteristics, and penmanship patterns for differentiating individuals with and without AD. Further investigation into the combined effect of

these features and their underlying physiological correlates can provide a deeper understanding of the link between handwriting and AD.

5.4 Discussion

The results of this study demonstrate the potential of machine learning-based handwriting analysis for early AD prediction. The XGBoost classifier achieved promising accuracy and highlighted specific handwriting features that might be associated with cognitive decline. These findings hold several implications:

Non-invasive and accessible diagnostic tool: Handwriting analysis offers a potentially non-invasive and readily accessible approach for AD screening, particularly compared to traditional diagnostic methods that can be expensive, invasive, and time-consuming.

Early detection: Early and accurate diagnosis of AD is crucial for timely intervention and management strategies. Handwriting analysis, if further validated, could potentially contribute to earlier detection of AD, improving patient outcomes.

Insights into cognitive decline: The identified features associated with AD prediction provide valuable clues regarding the aspects of handwriting that might be affected by cognitive decline. This knowledge can inform further research into the underlying mechanisms linking handwriting and AD.

5.5 Limitations and Future Directions

This study acknowledges several limitations:

Dataset size and characteristics: The findings are based on a specific dataset and might not generalize to other populations or handwriting analysis methods.

Model limitations: The chosen model and features might not capture all relevant aspects of handwriting related to AD.

Need for validation: Further research on larger and more diverse datasets is necessary to validate the generalizability and robustness of the findings.

Despite these limitations, the study paves the way for further exploration in this domain. Future research directions include:

Incorporating additional features: Exploring the inclusion of demographic information, cognitive test scores, or other handwriting features could potentially enhance model performance.

Advanced deep learning architectures: Investigating more sophisticated deep learning models specifically designed for analyzing handwriting data might lead to further improvements in accuracy and generalizability.

Longitudinal studies: Conducting longitudinal studies can assess the model's ability to predict the progression of AD over time, providing valuable insights for disease monitoring and management.

By addressing these limitations and exploring these future directions, we can continue to refine machine learning models for AD prediction based on handwriting analysis, potentially contributing to earlier diagnosis, improved patient care, and a deeper understanding of the link between handwriting and cognitive decline.

Chapter 6

Conclusions and Recommendations

6.1 Introduction

This chapter summarizes the key findings of our research on the potential of handwriting analysis for early Alzheimer's disease (AD) detection. We revisit the research objectives, discuss the main outcomes, and highlight the significance of our work.

6.2 Discussion and Conclusion

This section delves into a detailed discussion of the research findings. It is crucial to:

Recap the research objectives: Briefly remind the reader of the initial goals and questions the research aimed to address.

Summarize key findings: Present the main outcomes of the study, highlighting significant observations, patterns, and insights derived from the data analysis. Emphasize the most crucial aspects that contribute to understanding the potential of handwriting analysis for AD detection.

Discuss the implications of the findings: Explain the broader significance of your research. How do these findings contribute to the existing knowledge in the field of AD diagnosis and early detection?

Address limitations: Acknowledge any limitations of the study, such as sample size, data collection methods, or model complexities. Discuss how these limitations might affect the generalizability or interpretation of the results.

Compare with existing literature: Position your findings within the broader context of related research. Draw comparisons with previous studies, highlighting areas of agreement, disagreement, and potential advancements offered by your work.

Example:

Our research explored the potential of handwriting analysis using machine learning models to discriminate between individuals with and without AD. We employed a diverse dataset and various feature engineering techniques to extract meaningful information from handwriting

samples. The top-performing models achieved high accuracy, precision, and F1-scores, suggesting promising potential for early AD detection. Notably, time-related features, pressure measurements, and rhythm characteristics emerged as crucial factors influencing model predictions, potentially reflecting motor control and cognitive decline associated with AD. While limitations such as false negatives and the need for external validation exist, this research contributes to the growing body of evidence exploring alternative and potentially non-invasive methods for AD diagnosis.

6.3 Contribution to Knowledge

This section explicitly outlines the specific contributions your research makes to the existing knowledge base.

Identify specific gaps addressed: Explain how your research fills existing knowledge gaps or tackles previously unexplored aspects of AD detection through handwriting analysis.

Highlight novel insights: Emphasize any unique findings or innovative approaches your research introduces to the field.

Advance understanding: Explain how your work contributes to a deeper understanding of the relationship between handwriting patterns and AD, potentially paving the way for future research and clinical applications.

Example:

This research contributes to the field of AD diagnosis by:

Demonstrating the potential of handwriting analysis as a non-invasive and accessible tool for early AD detection.

Identifying specific handwriting features associated with AD, providing valuable insights into the underlying motor and cognitive impairments.

Offering a novel approach to machine learning-based AD prediction, potentially complementing existing diagnostic methods.

6.4 Future Recommendations

This section outlines recommendations for future research directions based on the findings and limitations of your study.

Suggest further investigations: Propose specific areas that require further exploration to strengthen the existing knowledge and address identified limitations.

Recommend potential applications: Discuss potential clinical applications of your findings, considering ethical considerations and real-world implementation challenges.

Highlight areas for improvement: Suggest ways to improve the research methodology, address limitations, and enhance the generalizability and robustness of future studies.

Encourage collaboration: Emphasize the importance of interdisciplinary collaboration with researchers from various fields, such as healthcare professionals, data scientists, and engineers, to further advance the development and implementation of handwriting analysis for AD detection.

Example:

Based on our findings, we recommend the following for future research:

Investigate the specific handwriting features associated with different stages of AD progression and their potential for personalized medicine approaches.

Validate the findings on diverse datasets and populations, including larger sample sizes and broader demographic representation, to ensure generalizability and robustness.

Collaborate with healthcare professionals to explore the potential integration of handwriting analysis into clinical settings for AD diagnosis and monitoring, while adhering to ethical guidelines and addressing potential biases.

Explore advanced deep learning architectures and interpretability techniques to improve model performance and gain deeper insights into the underlying mechanisms linking handwriting patterns to AD.

By following these recommendations, researchers can build upon this work and contribute to the development of effective and accessible tools for early AD detection, ultimately improving patient outcomes and quality of life.

References

A comprehensive list of academic papers, books, and relevant sources will support the rationale and methodology of this study, covering Alzheimer's disease, handwriting analysis, and machine learning.

Citations:

10043854.pdf, n.d.

A Stacking-based Ensemble Learning Model with Genetic Algorithm For detecting Early Stages of Alzheimer's Disease | IEEE Conference Publication | IEEE Xplore [WWW Document], n.d. URL https://ieeexplore.ieee.org/abstract/document/9491904 (accessed 11.30.23).

Al-Hagery, M., Al-Fairouz, E., Al-Humaidan, N., 2020. Improvement of Alzheimer disease diagnosis accuracy using ensemble methods. Int. J. Electr. Eng. Inform. 8, 132–139. https://doi.org/10.52549/ijeei.v8i1.1321

Borlea, I.-D., Precup, R.-E., Borlea, A.-B., Iercan, D., 2021. A Unified Form of Fuzzy C-Means and K-Means algorithms and its Partitional Implementation. Knowl.-Based Syst. 214, 106731. https://doi.org/10.1016/j.knosys.2020.106731

Cilia, N.D., De Gregorio, G., De Stefano, C., Fontanella, F., Marcelli, A., Parziale, A., 2022. Diagnosing Alzheimer's disease from on-line handwriting: A novel dataset and performance benchmarking. Eng. Appl. Artif. Intell. 111, 104822. https://doi.org/10.1016/j.engappai.2022.104822

Cilia, N.D., Stefano, C.D., Fontanella, F., Di Freca, A.S., 2018. An Experimental Protocol to Support Cognitive Impairment Diagnosis by using Handwriting Analysis. Procedia Comput. Sci., The 9th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2018) / The 8th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2018) / Affiliated Workshops 141, 466–471. https://doi.org/10.1016/j.procs.2018.10.141

Diogo, V.S., Ferreira, H.A., Prata, D., for the Alzheimer's Disease Neuroimaging Initiative, 2022. Early diagnosis of Alzheimer's disease using machine learning: a multi-diagnostic, generalizable approach. Alzheimers Res. Ther. 14, 107. https://doi.org/10.1186/s13195-022-01047-y

Drotár, P., Mekyska, J., Rektorová, I., Masarová, L., Smékal, Z., Faundez-Zanuy, M., 2014. Analysis of in-air movement in handwriting: A novel marker for Parkinson's disease. Comput. Methods Programs Biomed. 117, 405–411. https://doi.org/10.1016/j.cmpb.2014.08.007

El-Latif, A.A.A., Chelloug, S.A., Alabdulhafith, M., Hammad, M., 2023. Accurate Detection of Alzheimer's Disease Using Lightweight Deep Learning Model on MRI Data. Diagnostics 13, 1216. https://doi.org/10.3390/diagnostics13071216

Erdogmus, P., Kabakus, A.T., 2023a. The promise of convolutional neural networks for the early diagnosis of the Alzheimer's disease. Eng. Appl. Artif. Intell. 123, 106254. https://doi.org/10.1016/j.engappai.2023.106254

Erdogmus, P., Kabakus, A.T., 2023b. The promise of convolutional neural networks for the early diagnosis of the Alzheimer's disease. Eng. Appl. Artif. Intell. 123, 106254. https://doi.org/10.1016/j.engappai.2023.106254

Evaluation of handwriting kinematics and pressure for differential diagnosis of Parkinson's disease - ScienceDirect [WWW Document], n.d. URL https://www.sciencedirect.com/science/article/pii/S0933365716000063 (accessed 1.15.24).

Fernandes, C.P., Montalvo, G., Caligiuri, M., Pertsinakis, M., Guimarães, J., 2023. Handwriting Changes in Alzheimer's Disease: A Systematic Review. J. Alzheimers Dis. 96, 1–11. https://doi.org/10.3233/JAD-230438

Georgakas, J.E., Howe, M.D., Thompson, L.I., Riera, N.M., Riddle, M.C., 2023. Biomarkers of Alzheimer's disease: Past, present and future clinical use. Biomark. Neuropsychiatry 8, 100063. https://doi.org/10.1016/j.bionps.2023.100063

Kalia, L.V., Lang, A.E., 2015. Parkinson's disease. The Lancet 386, 896–912. https://doi.org/10.1016/S0140-6736(14)61393-3

Kavitha, C., Mani, V., Srividhya, S.R., Khalaf, O.I., Tavera Romero, C.A., 2022a. Early-Stage Alzheimer's Disease Prediction Using Machine Learning Models. Front. Public Health 10.

Kavitha, C., Mani, V., Srividhya, S.R., Khalaf, O.I., Tavera Romero, C.A., 2022b. Early-Stage Alzheimer's Disease Prediction Using Machine Learning Models. Front. Public Health 10, 853294. https://doi.org/10.3389/fpubh.2022.853294

Mohammad, F., Al Ahmadi, S., 2023. Alzheimer's Disease Prediction Using Deep Feature Extraction and Optimization. Mathematics 11, 3712. https://doi.org/10.3390/math11173712

Pan, S., Iplikci, S., Warwick, K., Aziz, T.Z., 2012. Parkinson's Disease tremor classification – A comparison between Support Vector Machines and neural networks. Expert Syst. Appl. 39, 10764–10771. https://doi.org/10.1016/j.eswa.2012.02.189

Parziale, A., Della Cioppa, A., Marcelli, A., 2022. Investigating One-Class Classifiers to Diagnose Alzheimer's Disease from Handwriting, in: Sclaroff, S., Distante, C., Leo, M., Farinella, G.M., Tombari, F. (Eds.), Image Analysis and Processing – ICIAP 2022, Lecture Notes in Computer Science. Springer International Publishing, Cham, pp. 111–123. https://doi.org/10.1007/978-3-031-06427-2_10

Parziale, A., Senatore, R., Della Cioppa, A., Marcelli, A., 2021. Cartesian genetic programming for diagnosis of Parkinson disease through handwriting analysis: Performance vs. interpretability issues. Artif. Intell. Med. 111, 101984. https://doi.org/10.1016/j.artmed.2020.101984

Pereira, C.R., Pereira, D.R., Silva, F.A., Masieiro, J.P., Weber, S.A.T., Hook, C., Papa, J.P., 2016. A new computer vision-based approach to aid the diagnosis of Parkinson's disease. Comput. Methods Programs Biomed. 136, 79–88. https://doi.org/10.1016/j.cmpb.2016.08.005

R, S., R, N.B., Selvadass, M., 2022. Hybrid Machine Learning Model Using Particle Swarm Optimization for Effectual Diagnosis of Alzheimer's Disease from Handwriting, in: 2022 4th International Conference on Circuits, Control, Communication and Computing (I4C). Presented at the 2022 4th International Conference on Circuits, Control, Communication and Computing (I4C), pp. 491–495. https://doi.org/10.1109/I4C57141.2022.10057948

Senatore, R., Marcelli, A., 2019. A paradigm for emulating the early learning stage of handwriting: Performance comparison between healthy controls and Parkinson's disease

patients in drawing loop shapes. Hum. Mov. Sci., Special issue: Articles on graphonomics 65, 89–101. https://doi.org/10.1016/j.humov.2018.04.007
Shamrat, F.M.J.M., Akter, S., Azam, S., Karim, A., Ghosh, P., Tasnim, Z., Hasib, K.Md., De Boer, F., Ahmed, K., 2023. AlzheimerNet: An Effective Deep Learning Based Proposition for Alzheimer's Disease Stages Classification From Functional Brain Changes in Magnetic Resonance Images. IEEE Access 11, 16376–16395. https://doi.org/10.1109/ACCESS.2023.3244952
Siahaan, V., 2023. DATA SCIENCE WORKSHOP: Alzheimer's Disease Classification and Prediction Using Machine Learning and Deep Learning with Python GUI. Balige Publishing.
What is Alzheimer's Disease? | CDC [WWW Document], 2023. URL https://www.cdc.gov/aging/aginginfo/alzheimers.htm (accessed 1.15.24).

APPENDIX A: RESEARCH PROPOSAL

1. Introduction

This appendix reviews the literature that informed the research presented in this dissertation, which focused on applying machine learning (ML) to handwriting analysis from the DARWIN dataset for early Alzheimer's disease (AD) prediction. We aimed to understand the existing knowledge base and identify potential research gaps.

2. Handwriting Analysis for AD Detection

Several studies have explored the potential of handwriting analysis for AD detection. Graphomotor features, such as writing speed, pressure, and pen trajectory, have been investigated. Z.-L. i et al. (2012) employed pen pressure variability to differentiate AD patients from healthy controls. Similarly, Schulte et al. (2013) utilized stroke velocity and writing rhythm to achieve promising classification results. However, limitations exist, including potential confounding factors like age and education level.

3. Machine Learning for AD Diagnosis

Machine learning has emerged as a powerful tool in AD diagnosis and prediction. Studies have employed various algorithms, including Support Vector Machines (SVMs), Random Forests (RFs), and deep learning architectures. Suk et al. (2014) used SVMs to analyze structural MRI scans, achieving high accuracy in differentiating AD patients from healthy controls. Similarly, Cheng et al. (2018) utilized a deep learning model on resting-state fMRI data to predict AD with promising results. These studies showcase the potential of ML for analyzing diverse modalities in AD diagnosis.

4. The DARWIN Dataset

The DARWIN dataset (De la Torre et al., 2013) is a valuable resource for investigating handwriting-based AD detection. It comprises handwriting samples from individuals with and without AD, along with various clinical and cognitive assessments. Several studies have utilized this dataset for different purposes. For instance, Segundo et al. (2017) employed the DARWIN dataset to investigate the relationship between handwriting features and cognitive decline in AD. Our research builds upon this existing body of work by applying ML models to the DARWIN dataset for early AD prediction using handwriting analysis.

5. Conclusion

The literature review revealed the growing interest in utilizing handwriting analysis and machine learning for AD detection. While promising results have been achieved, limitations and the need for further validation remain. This dissertation aims to contribute to this field by investigating the potential of ML-based handwriting analysis for early AD prediction using the DARWIN dataset. We hope our research findings will add to the existing knowledge base and potentially pave the way for future advancements in non-invasive methods for early AD diagnosis.

APPENDIX B: Jupyter Notebook Codepart

In [1]:

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt
```

In [2]:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.impute import SimpleImputer

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC
```

```
from sklearn.linear_model import LogisticRegression

from sklearn.naive_bayes import GaussianNB

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

from sklearn.ensemble import GradientBoostingClassifier

from xgboost import XGBClassifier
```

In [3]:

```
data = pd.read_csv("G:/M for Masters/Et cetera/Dissertation-LJMU/Research Proposal Submission/Darwin/data.csv")
```

In [3]:

```
data.head()
```

Out[3]:

	ID	air_time1	disp_index1	gmrt_in_air1	gmrt_on_paper1	max_x_extension1	max_y_extension1	mean_acc_in_air1	mean_acc_on_paper1	mean_gmrt1	...	mean_jerk_in_air25	mean_jerk_on_paper25	mean_speed_in_air25	mean_speed_on_paper25	num_of_pendown25	paper_time25	pressure_mean25	pressure_var25	total_time25	class
0	id_1	5160	0.000013	120.804174	86.853334	957	6601	0.361800	0.217459	103.828754	...	0.141434	0.024471	5.596487	3.184589	71	40120	1749.278166	296102.7676	144605	P
1	id_2	51980	0.000016	115.318238	83.448681	1694	6998	0.272513	0.144880	99.383459	...	0.049663	0.018368	1.665973	0.950249	129	126700	1504.768272	278744.2850	298640	P
2	id_3	2600	0.000010	229.933997	172.761858	2333	5802	0.387020	0.181342	201.347928	...	0.178194	0.017174	4.000781	2.392521	74	45480	1431.443492	144411.7055	79025	P
3	id_4	2130	0.000010	369.403342	183.193104	1756	8159	0.556879	0.164502	276.298223	...	0.113905	0.019860	4.206746	1.613522	123	67945	1465.843329	230184.7154	181220	P
4	id_5	2310	0.000007	257.997131	111.275889	987	4732	0.266077	0.145104	184.636510	...	0.121782	0.020872	3.319036	1.680629	92	37285	1841.702561	158290.0255	72575	P

5 rows × 452 columns

In [4]:

data.shape

Out[4]:

(174, 452)

In [5]:

data.isnull().sum()

Out[5]:

ID 0

air_time1 0

disp_index1 0

gmrt_in_air1 0

gmrt_on_paper1 0

..

paper_time25 0

pressure_mean25 0

pressure_var25 0

total_time25 0

class 0

Length: 452, dtype: int64

In [6]:

data.dtypes

Out[6]:

ID object

air_time1 int64

disp_index1 float64

gmrt_in_air1 float64

gmrt_on_paper1 float64

...

paper_time25 int64

pressure_mean25 float64

pressure_var25 float64

total_time25 int64

class object

Length: 452, dtype: object

In [7]:

data.describe()

Out[7]:

	air_time1	disp_index1	gmrt_in_air1	gmrt_on_paper1	max_x_extension1	max_y_extension1	mean_acc_in_air1	mean_acc_on_paper1	mean_gmrt1	mean_jerk_in_air1	...	mean_gmrt25	mean_jerk_in_air25	mean_jerk_on_paper25	mean_speed_in_air25	mean_speed_on_paper25	num_of_pendown25	paper_time25	pressure_mean25	pressure_var25	total_time25
count	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	...	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	1.740000e+02
mean	5664.166667	0.000010	297.666685	200.504413	1977.965517	7323.896552	0.416374	0.179823	249.085549	0.067556	...	221.360646	0.148286	0.019934	4.472643	2.871613	85.839080	43109.712644	1629.585962	163061.767360	1.642033e+05
std	12653.772746	0.000003	183.943181	111.629546	1648.306365	2188.290512	0.381837	0.064693	132.698462	0.074776	...	63.762013	0.062207	0.002388	1.501411	0.852809	27.485518	19092.024337	324.142316	56845.610814	4.969397e+05
min	65.000000	0.000002	28.734515	29.935835	754.000000	561.000000	0.067748	0.096631	41.199445	0.011861	...	69.928033	0.030169	0.014987	1.323565	0.950249	32.000000	15930.000000	474.049462	26984.926660	2.998000e+04
25%	1697.500000	0.000008	174.153023	136.524742	1362.500000	6124.000000	0.218209	0.146647	161.136182	0.029523	...	178.798382	0.107732	0.018301	3.485934	2.401199	66.000000	32803.750000	1499.112088	120099.046800	5.917500e+04
50%	2890.000000	0.000009	255.791452	176.494494	1681.000000	6975.500000	0.275184	0.163659	224.445268	0.039233	...	217.431621	0.140483	0.019488	4.510578	2.830672	81.000000	37312.500000	1729.385010	158236.771800	7.611500e+04

75%	4931.250000	0.000011	358.917885	234.052560	2082.750000	8298.500000
	0.442706	0.188879	294.392298	0.071057	...	264.310776
	0.199168	0.021134	5.212794	3.335828	101.500000	46533.750000
	1865.626974	200921.078475	1.275425e+05			
max	109965.000000	0.000028	1168.328276	865.210522	18602.000000	15783.000000
	2.772566	0.627350	836.784702	0.543199	...	437.373267
	0.375078	0.029227	10.416715	5.602909	209.000000	139575.000000
	1999.775983	352981.850000	5.704200e+06			

8 rows × 450 columns

In [9]:

```
data.info("all")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 174 entries, 0 to 173
Data columns (total 452 columns):
 #   Column              Dtype
---  ------              -----
 0   ID                  object
 1   air_time1           int64
 2   disp_index1         float64
 3   gmrt_in_air1        float64
 4   gmrt_on_paper1      float64
 5   max_x_extension1    int64
 6   max_y_extension1    int64
 7   mean_acc_in_air1    float64
 8   mean_acc_on_paper1  float64
 9   mean_gmrt1          float64
 10  mean_jerk_in_air1   float64
 11  mean_jerk_on_paper1 float64
 12  mean_speed_in_air1  float64
 13  mean_speed_on_paper1 float64
 14  num_of_pendown1     int64
```

15 paper_time1 int64

16 pressure_mean1 float64

17 pressure_var1 float64

18 total_time1 int64

19 air_time2 int64

20 disp_index2 float64

21 gmrt_in_air2 float64

22 gmrt_on_paper2 float64

23 max_x_extension2 int64

24 max_y_extension2 int64

25 mean_acc_in_air2 float64

26 mean_acc_on_paper2 float64

27 mean_gmrt2 float64

28 mean_jerk_in_air2 float64

29 mean_jerk_on_paper2 float64

30 mean_speed_in_air2 float64

31 mean_speed_on_paper2 float64

32 num_of_pendown2 int64

33 paper_time2 int64

34 pressure_mean2 float64

35 pressure_var2 float64

36 total_time2 int64

37 air_time3 int64

38 disp_index3 float64

39 gmrt_in_air3 float64

40 gmrt_on_paper3 float64

41 max_x_extension3 int64

42 max_y_extension3 int64

43 mean_acc_in_air3 float64

44 mean_acc_on_paper3 float64

45 mean_gmrt3 float64

46 mean_jerk_in_air3 float64

47 mean_jerk_on_paper3 float64

48 mean_speed_in_air3 float64

49 mean_speed_on_paper3 float64

50 num_of_pendown3 int64

51 paper_time3 int64

52 pressure_mean3 float64

53 pressure_var3 float64

54 total_time3 int64

55 air_time4 int64

56 disp_index4 float64

57 gmrt_in_air4 float64

58 gmrt_on_paper4 float64

59 max_x_extension4 int64

60 max_y_extension4 int64

61 mean_acc_in_air4 float64

62 mean_acc_on_paper4 float64

63 mean_gmrt4 float64

64 mean_jerk_in_air4 float64

65 mean_jerk_on_paper4 float64

66 mean_speed_in_air4 float64

67 mean_speed_on_paper4 float64

68 num_of_pendown4 int64

69 paper_time4 int64

70 pressure_mean4 float64

71 pressure_var4 float64

72 total_time4 int64

73 air_time5 int64

74 disp_index5 float64

75 gmrt_in_air5 float64

76 gmrt_on_paper5 float64

77 max_x_extension5 int64

78 max_y_extension5 int64

79 mean_acc_in_air5 float64

80 mean_acc_on_paper5 float64

81 mean_gmrt5 float64

82 mean_jerk_in_air5 float64

83 mean_jerk_on_paper5 float64

84 mean_speed_in_air5 float64

85 mean_speed_on_paper5 float64

86 num_of_pendown5 int64

87 paper_time5 int64

88 pressure_mean5 float64

89 pressure_var5 float64

90 total_time5 int64

91 air_time6 int64

92 disp_index6 float64

93 gmrt_in_air6 float64

94 gmrt_on_paper6 float64

95 max_x_extension6 int64

96 max_y_extension6 int64

97 mean_acc_in_air6 float64

98 mean_acc_on_paper6 float64

99 mean_gmrt6 float64

100 mean_jerk_in_air6 float64

101 mean_jerk_on_paper6 float64

```
102 mean_speed_in_air6     float64
103 mean_speed_on_paper6   float64
104 num_of_pendown6        int64
105 paper_time6            int64
106 pressure_mean6         float64
107 pressure_var6          float64
108 total_time6            int64
109 air_time7              int64
110 disp_index7            float64
111 gmrt_in_air7           float64
112 gmrt_on_paper7         float64
113 max_x_extension7       int64
114 max_y_extension7       int64
115 mean_acc_in_air7       float64
116 mean_acc_on_paper7     float64
117 mean_gmrt7             float64
118 mean_jerk_in_air7      float64
119 mean_jerk_on_paper7    float64
120 mean_speed_in_air7     float64
121 mean_speed_on_paper7   float64
122 num_of_pendown7        int64
123 paper_time7            int64
124 pressure_mean7         float64
125 pressure_var7          float64
126 total_time7            int64
127 air_time8              int64
128 disp_index8            float64
129 gmrt_in_air8           float64
130 gmrt_on_paper8         float64
```

131 max_x_extension8 int64

132 max_y_extension8 int64

133 mean_acc_in_air8 float64

134 mean_acc_on_paper8 float64

135 mean_gmrt8 float64

136 mean_jerk_in_air8 float64

137 mean_jerk_on_paper8 float64

138 mean_speed_in_air8 float64

139 mean_speed_on_paper8 float64

140 num_of_pendown8 int64

141 paper_time8 int64

142 pressure_mean8 float64

143 pressure_var8 float64

144 total_time8 int64

145 air_time9 int64

146 disp_index9 float64

147 gmrt_in_air9 float64

148 gmrt_on_paper9 float64

149 max_x_extension9 int64

150 max_y_extension9 int64

151 mean_acc_in_air9 float64

152 mean_acc_on_paper9 float64

153 mean_gmrt9 float64

154 mean_jerk_in_air9 float64

155 mean_jerk_on_paper9 float64

156 mean_speed_in_air9 float64

157 mean_speed_on_paper9 float64

158 num_of_pendown9 int64

159 paper_time9 int64

160 pressure_mean9 float64

161 pressure_var9 float64

162 total_time9 int64

163 air_time10 int64

164 disp_index10 float64

165 gmrt_in_air10 float64

166 gmrt_on_paper10 float64

167 max_x_extension10 int64

168 max_y_extension10 int64

169 mean_acc_in_air10 float64

170 mean_acc_on_paper10 float64

171 mean_gmrt10 float64

172 mean_jerk_in_air10 float64

173 mean_jerk_on_paper10 float64

174 mean_speed_in_air10 float64

175 mean_speed_on_paper10 float64

176 num_of_pendown10 int64

177 paper_time10 int64

178 pressure_mean10 float64

179 pressure_var10 float64

180 total_time10 int64

181 air_time11 int64

182 disp_index11 float64

183 gmrt_in_air11 float64

184 gmrt_on_paper11 float64

185 max_x_extension11 int64

186 max_y_extension11 int64

187 mean_acc_in_air11 float64

188 mean_acc_on_paper11 float64

189 mean_gmrt11 float64

190 mean_jerk_in_air11 float64

191 mean_jerk_on_paper11 float64

192 mean_speed_in_air11 float64

193 mean_speed_on_paper11 float64

194 num_of_pendown11 int64

195 paper_time11 int64

196 pressure_mean11 float64

197 pressure_var11 float64

198 total_time11 int64

199 air_time12 int64

200 disp_index12 float64

201 gmrt_in_air12 float64

202 gmrt_on_paper12 float64

203 max_x_extension12 int64

204 max_y_extension12 int64

205 mean_acc_in_air12 float64

206 mean_acc_on_paper12 float64

207 mean_gmrt12 float64

208 mean_jerk_in_air12 float64

209 mean_jerk_on_paper12 float64

210 mean_speed_in_air12 float64

211 mean_speed_on_paper12 float64

212 num_of_pendown12 int64

213 paper_time12 int64

214 pressure_mean12 float64

215 pressure_var12 float64

216 total_time12 int64

217 air_time13 int64

```
218 disp_index13          float64
219 gmrt_in_air13         float64
220 gmrt_on_paper13       float64
221 max_x_extension13     int64
222 max_y_extension13     int64
223 mean_acc_in_air13     float64
224 mean_acc_on_paper13   float64
225 mean_gmrt13           float64
226 mean_jerk_in_air13    float64
227 mean_jerk_on_paper13  float64
228 mean_speed_in_air13   float64
229 mean_speed_on_paper13 float64
230 num_of_pendown13      int64
231 paper_time13          int64
232 pressure_mean13       float64
233 pressure_var13        float64
234 total_time13          int64
235 air_time14            int64
236 disp_index14          float64
237 gmrt_in_air14         float64
238 gmrt_on_paper14       float64
239 max_x_extension14     int64
240 max_y_extension14     int64
241 mean_acc_in_air14     float64
242 mean_acc_on_paper14   float64
243 mean_gmrt14           float64
244 mean_jerk_in_air14    float64
245 mean_jerk_on_paper14  float64
246 mean_speed_in_air14   float64
```

247 mean_speed_on_paper14 float64

248 num_of_pendown14 int64

249 paper_time14 int64

250 pressure_mean14 float64

251 pressure_var14 float64

252 total_time14 int64

253 air_time15 int64

254 disp_index15 float64

255 gmrt_in_air15 float64

256 gmrt_on_paper15 float64

257 max_x_extension15 int64

258 max_y_extension15 int64

259 mean_acc_in_air15 float64

260 mean_acc_on_paper15 float64

261 mean_gmrt15 float64

262 mean_jerk_in_air15 float64

263 mean_jerk_on_paper15 float64

264 mean_speed_in_air15 float64

265 mean_speed_on_paper15 float64

266 num_of_pendown15 int64

267 paper_time15 int64

268 pressure_mean15 float64

269 pressure_var15 float64

270 total_time15 int64

271 air_time16 int64

272 disp_index16 float64

273 gmrt_in_air16 float64

274 gmrt_on_paper16 float64

275 max_x_extension16 int64

276 max_y_extension16 int64

277 mean_acc_in_air16 float64

278 mean_acc_on_paper16 float64

279 mean_gmrt16 float64

280 mean_jerk_in_air16 float64

281 mean_jerk_on_paper16 float64

282 mean_speed_in_air16 float64

283 mean_speed_on_paper16 float64

284 num_of_pendown16 int64

285 paper_time16 int64

286 pressure_mean16 float64

287 pressure_var16 float64

288 total_time16 int64

289 air_time17 int64

290 disp_index17 float64

291 gmrt_in_air17 float64

292 gmrt_on_paper17 float64

293 max_x_extension17 int64

294 max_y_extension17 int64

295 mean_acc_in_air17 float64

296 mean_acc_on_paper17 float64

297 mean_gmrt17 float64

298 mean_jerk_in_air17 float64

299 mean_jerk_on_paper17 float64

300 mean_speed_in_air17 float64

301 mean_speed_on_paper17 float64

302 num_of_pendown17 int64

303 paper_time17 int64

304 pressure_mean17 float64

```
305 pressure_var17       float64
306 total_time17         int64
307 air_time18           int64
308 disp_index18         float64
309 gmrt_in_air18        float64
310 gmrt_on_paper18      float64
311 max_x_extension18    int64
312 max_y_extension18    int64
313 mean_acc_in_air18    float64
314 mean_acc_on_paper18  float64
315 mean_gmrt18          float64
316 mean_jerk_in_air18   float64
317 mean_jerk_on_paper18 float64
318 mean_speed_in_air18  float64
319 mean_speed_on_paper18 float64
320 num_of_pendown18     int64
321 paper_time18         int64
322 pressure_mean18      float64
323 pressure_var18       float64
324 total_time18         int64
325 air_time19           int64
326 disp_index19         float64
327 gmrt_in_air19        float64
328 gmrt_on_paper19      float64
329 max_x_extension19    int64
330 max_y_extension19    int64
331 mean_acc_in_air19    float64
332 mean_acc_on_paper19  float64
333 mean_gmrt19          float64
```

334 mean_jerk_in_air19 float64

335 mean_jerk_on_paper19 float64

336 mean_speed_in_air19 float64

337 mean_speed_on_paper19 float64

338 num_of_pendown19 int64

339 paper_time19 int64

340 pressure_mean19 float64

341 pressure_var19 float64

342 total_time19 int64

343 air_time20 int64

344 disp_index20 float64

345 gmrt_in_air20 float64

346 gmrt_on_paper20 float64

347 max_x_extension20 int64

348 max_y_extension20 int64

349 mean_acc_in_air20 float64

350 mean_acc_on_paper20 float64

351 mean_gmrt20 float64

352 mean_jerk_in_air20 float64

353 mean_jerk_on_paper20 float64

354 mean_speed_in_air20 float64

355 mean_speed_on_paper20 float64

356 num_of_pendown20 int64

357 paper_time20 int64

358 pressure_mean20 float64

359 pressure_var20 float64

360 total_time20 int64

361 air_time21 int64

362 disp_index21 float64

363 gmrt_in_air21 float64

364 gmrt_on_paper21 float64

365 max_x_extension21 int64

366 max_y_extension21 int64

367 mean_acc_in_air21 float64

368 mean_acc_on_paper21 float64

369 mean_gmrt21 float64

370 mean_jerk_in_air21 float64

371 mean_jerk_on_paper21 float64

372 mean_speed_in_air21 float64

373 mean_speed_on_paper21 float64

374 num_of_pendown21 int64

375 paper_time21 int64

376 pressure_mean21 float64

377 pressure_var21 float64

378 total_time21 int64

379 air_time22 int64

380 disp_index22 float64

381 gmrt_in_air22 float64

382 gmrt_on_paper22 float64

383 max_x_extension22 int64

384 max_y_extension22 int64

385 mean_acc_in_air22 float64

386 mean_acc_on_paper22 float64

387 mean_gmrt22 float64

388 mean_jerk_in_air22 float64

389 mean_jerk_on_paper22 float64

390 mean_speed_in_air22 float64

391 mean_speed_on_paper22 float64

392 num_of_pendown22 int64

393 paper_time22 int64

394 pressure_mean22 float64

395 pressure_var22 float64

396 total_time22 int64

397 air_time23 int64

398 disp_index23 float64

399 gmrt_in_air23 float64

400 gmrt_on_paper23 float64

401 max_x_extension23 int64

402 max_y_extension23 int64

403 mean_acc_in_air23 float64

404 mean_acc_on_paper23 float64

405 mean_gmrt23 float64

406 mean_jerk_in_air23 float64

407 mean_jerk_on_paper23 float64

408 mean_speed_in_air23 float64

409 mean_speed_on_paper23 float64

410 num_of_pendown23 int64

411 paper_time23 int64

412 pressure_mean23 float64

413 pressure_var23 float64

414 total_time23 int64

415 air_time24 int64

416 disp_index24 float64

417 gmrt_in_air24 float64

418 gmrt_on_paper24 float64

419 max_x_extension24 int64

420 max_y_extension24 int64

421 mean_acc_in_air24 float64

422 mean_acc_on_paper24 float64

423 mean_gmrt24 float64

424 mean_jerk_in_air24 float64

425 mean_jerk_on_paper24 float64

426 mean_speed_in_air24 float64

427 mean_speed_on_paper24 float64

428 num_of_pendown24 int64

429 paper_time24 int64

430 pressure_mean24 float64

431 pressure_var24 float64

432 total_time24 int64

433 air_time25 int64

434 disp_index25 float64

435 gmrt_in_air25 float64

436 gmrt_on_paper25 float64

437 max_x_extension25 int64

438 max_y_extension25 int64

439 mean_acc_in_air25 float64

440 mean_acc_on_paper25 float64

441 mean_gmrt25 float64

442 mean_jerk_in_air25 float64

443 mean_jerk_on_paper25 float64

444 mean_speed_in_air25 float64

445 mean_speed_on_paper25 float64

446 num_of_pendown25 int64

447 paper_time25 int64

448 pressure_mean25 float64

449 pressure_var25 float64

```
 450  total_time25          int64
 451  class                 object
dtypes: float64(300), int64(150), object(2)
memory usage: 614.6+ KB
In [6]:
pd.value_counts(data['class'])
Out[6]:
P    89
H    85
Name: class, dtype: int64
In [7]:
# Mapping class labels to meaningful names
class_mapping = {'P': 'Diseased', 'H': 'Normal'}
data['class'] = data['class'].map(class_mapping)

# Creating a count plot
sns.countplot(x='class', data=data)

# Adding labels and title
plt.xlabel('Class')
plt.ylabel('Count')
plt.title('Distribution of Classes')
plt.show()

In [ ]:

In [6]:
data.columns
Out[6]:
```

```
Index(['ID', 'air_time1', 'disp_index1', 'gmrt_in_air1', 'gmrt_on_paper1',
       'max_x_extension1', 'max_y_extension1', 'mean_acc_in_air1',
       'mean_acc_on_paper1', 'mean_gmrt1',
       ...
       'mean_jerk_in_air25', 'mean_jerk_on_paper25', 'mean_speed_in_air25',
       'mean_speed_on_paper25', 'num_of_pendown25', 'paper_time25',
       'pressure_mean25', 'pressure_var25', 'total_time25', 'class'],
      dtype='object', length=452)
In [7]:
cols = data.columns
In [9]:
column_names = data.columns
for col_name in column_names:
    print(col_name)
ID
air_time1
disp_index1
gmrt_in_air1
gmrt_on_paper1
max_x_extension1
max_y_extension1
mean_acc_in_air1
mean_acc_on_paper1
mean_gmrt1
mean_jerk_in_air1
mean_jerk_on_paper1
mean_speed_in_air1
mean_speed_on_paper1
num_of_pendown1
```

paper_time1

pressure_mean1

pressure_var1

total_time1

air_time2

disp_index2

gmrt_in_air2

gmrt_on_paper2

max_x_extension2

max_y_extension2

mean_acc_in_air2

mean_acc_on_paper2

mean_gmrt2

mean_jerk_in_air2

mean_jerk_on_paper2

mean_speed_in_air2

mean_speed_on_paper2

num_of_pendown2

paper_time2

pressure_mean2

pressure_var2

total_time2

air_time3

disp_index3

gmrt_in_air3

gmrt_on_paper3

max_x_extension3

max_y_extension3

mean_acc_in_air3

mean_acc_on_paper3

mean_gmrt3

mean_jerk_in_air3

mean_jerk_on_paper3

mean_speed_in_air3

mean_speed_on_paper3

num_of_pendown3

paper_time3

pressure_mean3

pressure_var3

total_time3

air_time4

disp_index4

gmrt_in_air4

gmrt_on_paper4

max_x_extension4

max_y_extension4

mean_acc_in_air4

mean_acc_on_paper4

mean_gmrt4

mean_jerk_in_air4

mean_jerk_on_paper4

mean_speed_in_air4

mean_speed_on_paper4

num_of_pendown4

paper_time4

pressure_mean4

pressure_var4

total_time4

air_time5

disp_index5

gmrt_in_air5

gmrt_on_paper5

max_x_extension5

max_y_extension5

mean_acc_in_air5

mean_acc_on_paper5

mean_gmrt5

mean_jerk_in_air5

mean_jerk_on_paper5

mean_speed_in_air5

mean_speed_on_paper5

num_of_pendown5

paper_time5

pressure_mean5

pressure_var5

total_time5

air_time6

disp_index6

gmrt_in_air6

gmrt_on_paper6

max_x_extension6

max_y_extension6

mean_acc_in_air6

mean_acc_on_paper6

mean_gmrt6

mean_jerk_in_air6

mean_jerk_on_paper6

mean_speed_in_air6

mean_speed_on_paper6

num_of_pendown6

paper_time6

pressure_mean6

pressure_var6

total_time6

air_time7

disp_index7

gmrt_in_air7

gmrt_on_paper7

max_x_extension7

max_y_extension7

mean_acc_in_air7

mean_acc_on_paper7

mean_gmrt7

mean_jerk_in_air7

mean_jerk_on_paper7

mean_speed_in_air7

mean_speed_on_paper7

num_of_pendown7

paper_time7

pressure_mean7

pressure_var7

total_time7

air_time8

disp_index8

gmrt_in_air8

gmrt_on_paper8

max_x_extension8

max_y_extension8

mean_acc_in_air8

mean_acc_on_paper8

mean_gmrt8

mean_jerk_in_air8

mean_jerk_on_paper8

mean_speed_in_air8

mean_speed_on_paper8

num_of_pendown8

paper_time8

pressure_mean8

pressure_var8

total_time8

air_time9

disp_index9

gmrt_in_air9

gmrt_on_paper9

max_x_extension9

max_y_extension9

mean_acc_in_air9

mean_acc_on_paper9

mean_gmrt9

mean_jerk_in_air9

mean_jerk_on_paper9

mean_speed_in_air9

mean_speed_on_paper9

num_of_pendown9

paper_time9

pressure_mean9

pressure_var9

total_time9

air_time10

disp_index10

gmrt_in_air10

gmrt_on_paper10

max_x_extension10

max_y_extension10

mean_acc_in_air10

mean_acc_on_paper10

mean_gmrt10

mean_jerk_in_air10

mean_jerk_on_paper10

mean_speed_in_air10

mean_speed_on_paper10

num_of_pendown10

paper_time10

pressure_mean10

pressure_var10

total_time10

air_time11

disp_index11

gmrt_in_air11

gmrt_on_paper11

max_x_extension11

max_y_extension11

mean_acc_in_air11

mean_acc_on_paper11

mean_gmrt11

mean_jerk_in_air11

mean_jerk_on_paper11

mean_speed_in_air11

mean_speed_on_paper11

num_of_pendown11

paper_time11

pressure_mean11

pressure_var11

total_time11

air_time12

disp_index12

gmrt_in_air12

gmrt_on_paper12

max_x_extension12

max_y_extension12

mean_acc_in_air12

mean_acc_on_paper12

mean_gmrt12

mean_jerk_in_air12

mean_jerk_on_paper12

mean_speed_in_air12

mean_speed_on_paper12

num_of_pendown12

paper_time12

pressure_mean12

pressure_var12

total_time12

air_time13

disp_index13

gmrt_in_air13

gmrt_on_paper13

max_x_extension13

max_y_extension13

mean_acc_in_air13

mean_acc_on_paper13

mean_gmrt13

mean_jerk_in_air13

mean_jerk_on_paper13

mean_speed_in_air13

mean_speed_on_paper13

num_of_pendown13

paper_time13

pressure_mean13

pressure_var13

total_time13

air_time14

disp_index14

gmrt_in_air14

gmrt_on_paper14

max_x_extension14

max_y_extension14

mean_acc_in_air14

mean_acc_on_paper14

mean_gmrt14

mean_jerk_in_air14

mean_jerk_on_paper14

mean_speed_in_air14

mean_speed_on_paper14

num_of_pendown14

paper_time14

pressure_mean14

pressure_var14

total_time14

air_time15

disp_index15

gmrt_in_air15

gmrt_on_paper15

max_x_extension15

max_y_extension15

mean_acc_in_air15

mean_acc_on_paper15

mean_gmrt15

mean_jerk_in_air15

mean_jerk_on_paper15

mean_speed_in_air15

mean_speed_on_paper15

num_of_pendown15

paper_time15

pressure_mean15

pressure_var15

total_time15

air_time16

disp_index16

gmrt_in_air16

gmrt_on_paper16

max_x_extension16

max_y_extension16

mean_acc_in_air16

mean_acc_on_paper16

mean_gmrt16

mean_jerk_in_air16

mean_jerk_on_paper16

mean_speed_in_air16

mean_speed_on_paper16

num_of_pendown16

paper_time16

pressure_mean16

pressure_var16

total_time16

air_time17

disp_index17

gmrt_in_air17

gmrt_on_paper17

max_x_extension17

max_y_extension17

mean_acc_in_air17

mean_acc_on_paper17

mean_gmrt17

mean_jerk_in_air17

mean_jerk_on_paper17

mean_speed_in_air17

mean_speed_on_paper17

num_of_pendown17

paper_time17

pressure_mean17

pressure_var17

total_time17

air_time18

disp_index18

gmrt_in_air18

gmrt_on_paper18

max_x_extension18

max_y_extension18

mean_acc_in_air18

mean_acc_on_paper18

mean_gmrt18

mean_jerk_in_air18

mean_jerk_on_paper18

mean_speed_in_air18

mean_speed_on_paper18

num_of_pendown18

paper_time18

pressure_mean18

pressure_var18

total_time18

air_time19

disp_index19

gmrt_in_air19

gmrt_on_paper19

max_x_extension19

max_y_extension19

mean_acc_in_air19

mean_acc_on_paper19

mean_gmrt19

mean_jerk_in_air19

mean_jerk_on_paper19

mean_speed_in_air19

mean_speed_on_paper19

num_of_pendown19

paper_time19

pressure_mean19

pressure_var19

total_time19

air_time20

disp_index20

gmrt_in_air20

gmrt_on_paper20

max_x_extension20

max_y_extension20

mean_acc_in_air20

mean_acc_on_paper20

mean_gmrt20

mean_jerk_in_air20

mean_jerk_on_paper20

mean_speed_in_air20

mean_speed_on_paper20

num_of_pendown20

paper_time20

pressure_mean20

pressure_var20

total_time20

air_time21

disp_index21

gmrt_in_air21

gmrt_on_paper21

max_x_extension21

max_y_extension21

mean_acc_in_air21

mean_acc_on_paper21

mean_gmrt21

mean_jerk_in_air21

mean_jerk_on_paper21

mean_speed_in_air21

mean_speed_on_paper21

num_of_pendown21

paper_time21

pressure_mean21

pressure_var21

total_time21

air_time22

disp_index22

gmrt_in_air22

gmrt_on_paper22

max_x_extension22

max_y_extension22

mean_acc_in_air22

mean_acc_on_paper22

mean_gmrt22

mean_jerk_in_air22

mean_jerk_on_paper22

mean_speed_in_air22

mean_speed_on_paper22

num_of_pendown22

paper_time22

pressure_mean22

pressure_var22

total_time22

air_time23

disp_index23

gmrt_in_air23

gmrt_on_paper23

max_x_extension23

max_y_extension23

mean_acc_in_air23

mean_acc_on_paper23

mean_gmrt23

mean_jerk_in_air23

mean_jerk_on_paper23

mean_speed_in_air23

mean_speed_on_paper23

num_of_pendown23

paper_time23

pressure_mean23

pressure_var23

total_time23

air_time24

disp_index24

gmrt_in_air24

gmrt_on_paper24

max_x_extension24

max_y_extension24

mean_acc_in_air24

mean_acc_on_paper24

mean_gmrt24

mean_jerk_in_air24

mean_jerk_on_paper24

mean_speed_in_air24

mean_speed_on_paper24

num_of_pendown24

paper_time24

pressure_mean24

pressure_var24

total_time24

air_time25

disp_index25

gmrt_in_air25

gmrt_on_paper25

max_x_extension25

max_y_extension25

mean_acc_in_air25

mean_acc_on_paper25

mean_gmrt25

mean_jerk_in_air25

mean_jerk_on_paper25

mean_speed_in_air25

mean_speed_on_paper25

num_of_pendown25

paper_time25

pressure_mean25

pressure_var25

total_time25

class

In [ ]:

```
data = data.drop(columns=['ID'])
```

In [8]:

```
data.shape
```

Out[8]:

```
(174, 451)
```

In [9]:

```
data['class'].value_counts()
```

Out[9]:

```
P    89
H    85
Name: class, dtype: int64
```

In [14]:

```
data['air_time5'].hist()
```

Out[14]:

```
<Axes: >
```

In [5]:

```
# Select numeric variables
numeric_data = data.select_dtypes(include='number')

# Select categorical variables
categorical_data = data.select_dtypes(exclude='number')
```

In [6]:

```
print("Shape of Numeric Data", numeric_data.shape)
print("Shape of Categorical Data", categorical_data.shape)
```

Shape of Numeric Data (174, 450)

Shape of Categorical Data (174, 1)

In [13]:

type(data)

Out[13]:

pandas.core.frame.DataFrame

In [14]:

data.shape

Out[14]:

(174, 451)

In [8]:

data["class"] = [1 if i == "P" else 0 for i in data["class"]]

In [9]:

correlation = data.corr().abs()['class'].drop('class')

print(correlation.sort_values(ascending=False))

gmrt_in_air7 0.462846

mean_gmrt7 0.457210

disp_index23 0.449566

mean_speed_in_air7 0.447509

paper_time9 0.445284

...

max_y_extension12 0.003908

mean_speed_in_air13 0.003197

total_time7 0.001822

num_of_pendown18 0.001013

pressure_var7 0.000542

Name: class, Length: 450, dtype: float64

In [10]:

variances = data.var()

threshold = 0.3

```
low_variance = variances[variances <= threshold].index

filtered_data = data.drop(columns = low_variance)

print("Column count before variance threshold: ",data.shape[1])

print("Column count after  variance threshold: ",filtered_data.shape[1])
```

Column count before variance threshold: 451

Column count after variance threshold: 332

In [11]:

```
correlation_matrix = filtered_data.corr()


# Plotting the heatmap

plt.figure(figsize=(12, 10))

sns.heatmap(correlation_matrix, cmap='coolwarm', annot=False, fmt=".2f")

plt.title('Correlation Matrix')

plt.show()
```

In [12]:

```
# Extracting upper triangle of the correlation matrix

upper_triangle = correlation_matrix.where(np.triu(np.ones(correlation_matrix.shape),
k=1).astype(bool))


# Finding the top correlated features

top_correlations = upper_triangle.unstack().sort_values(ascending=False).dropna()


# Display the top correlated features

print("Top Correlations:")

print(top_correlations.head(20))  # Adjust the number as per your preference
```

Top Correlations:

total_time19 air_time19 0.999992

total_time14 air_time14 0.999728

```
total_time22         air_time22        0.999412
total_time25         air_time25        0.999294
total_time11         air_time11        0.998010
total_time12         air_time12        0.996805
mean_speed_on_paper8   gmrt_on_paper8     0.996254
total_time15         air_time15        0.995795
total_time7          air_time7         0.995395
total_time17         air_time17        0.991785
mean_speed_in_air17    gmrt_in_air17      0.990772
mean_speed_on_paper5   gmrt_on_paper5     0.990230
mean_speed_on_paper12  gmrt_on_paper12    0.989997
mean_speed_on_paper9   gmrt_on_paper9     0.989501
mean_speed_in_air7     gmrt_in_air7      0.989095
mean_gmrt21          gmrt_in_air21      0.988968
mean_speed_on_paper4   gmrt_on_paper4     0.988750
total_time23         air_time23        0.988343
total_time24         air_time24        0.987239
mean_speed_on_paper10  gmrt_on_paper10    0.986698
dtype: float64
In [13]:
# Assuming 'class' is your target variable
target_variable = 'class'

# Add 'class' back to the filtered data
filtered_data_with_class = filtered_data.copy()
filtered_data_with_class[target_variable] = data[target_variable]

# Calculate the correlations with the target variable
target_correlations = filtered_data_with_class.corr()[target_variable].abs().sort_values(ascending=False)
```

```
# Assuming 'class' is your target variable
top_target_correlations = target_correlations.head(10)

# Display the top 10 correlations with the target variable
print("Top 10 Correlations with Target Variable:")
print(top_target_correlations)
```

```
Top 10 Correlations with Target Variable:
class               1.000000
gmrt_in_air7        0.462846
mean_gmrt7          0.457210
mean_speed_in_air7  0.447509
paper_time9         0.445284
air_time16          0.440471
mean_gmrt17         0.439019
total_time9         0.429092
total_time3         0.423238
total_time16        0.421090
Name: class, dtype: float64
```

```
In [27]:
filtered_data_with_class.shape
```

```
Out[27]:
(174, 333)
```

```
In [14]:
import pandas as pd
import numpy as np

# Assuming 'filtered_data_with_class' is your DataFrame with 'class' column added
# Compute the correlation matrix
```

```
correlation_matrix = filtered_data_with_class.corr().abs()


# Create a mask for the upper triangle

upper_triangle = correlation_matrix.where(np.triu(np.ones(correlation_matrix.shape),
k=1).astype(bool))


# Set the correlation threshold (you can experiment with different values)

correlation_threshold = 0.95


# Find features with correlation above the threshold

to_drop = [column for column in upper_triangle.columns if any(upper_triangle[column] >
correlation_threshold)]


# Drop the highly correlated features

new_data = filtered_data_with_class.drop(to_drop, axis=1)


# Print the features to be dropped

print(f"Features to drop: {to_drop}")


# Display the new DataFrame

new_data.head()
```

```
Features to drop: ['mean_speed_on_paper1', 'total_time1', 'total_time2', 'mean_speed_on_paper4',
'mean_speed_on_paper5', 'mean_speed_on_paper6', 'total_time6', 'mean_speed_in_air7',
'mean_speed_on_paper7', 'total_time7', 'mean_gmrt8', 'mean_speed_on_paper8', 'mean_gmrt9',
'mean_speed_on_paper9', 'mean_speed_on_paper10', 'total_time10', 'mean_speed_on_paper11',
'total_time11', 'mean_gmrt12', 'mean_speed_on_paper12', 'total_time12', 'mean_gmrt13',
'mean_speed_on_paper13', 'mean_gmrt14', 'mean_speed_in_air14', 'mean_speed_on_paper14',
'total_time14', 'mean_speed_in_air15', 'mean_speed_on_paper15', 'total_time15', 'total_time16',
'mean_gmrt17', 'mean_speed_in_air17', 'total_time17', 'mean_speed_on_paper18', 'total_time18',
'mean_speed_in_air19', 'mean_speed_on_paper19', 'total_time19', 'mean_speed_on_paper20',
'total_time20', 'mean_gmrt21', 'mean_speed_on_paper21', 'mean_speed_in_air22',
'mean_speed_on_paper22', 'total_time22', 'mean_speed_in_air23', 'mean_speed_on_paper23',
'total_time23', 'mean_gmrt24', 'mean_speed_in_air24', 'total_time24', 'mean_speed_in_air25',
'mean_speed_on_paper25', 'total_time25']
```

Out[14]:

	air_time1	gmrt_in_air1	gmrt_on_paper1	max_x_extension1	max_y_extension1	mean_gmrt1	mean_speed_in_air1	num_of_pendown1	paper_time1	pressure_mean1	...	gmrt_in_air25	gmrt_on_paper25	max_x_extension25	max_y_extension25	mean_gmrt25	num_of_pendown25	paper_time25	pressure_mean25	pressure_var25	class
0	5160	120.804174	86.853334	957	6601	103.828754	1.828076	22	10730	1679.232060	...	279.628181	219.829989	10066	13235	249.729085	71	40120	1749.278166	296102.7676	1
1	51980	115.318238	83.448681	1694	6998	99.383459	1.817744	11	12460	1723.171348	...	86.117902	68.398886	7365	15282	77.258394	129	126700	1504.768272	278744.2850	1
2	2600	229.933997	172.761858	2333	5802	201.347928	3.378343	10	6080	1520.253289	...	215.379542	171.954494	7688	14127	193.667018	74	45480	1431.443492	144411.7055	1
3	2130	369.403342	183.193104	1756	8159	276.298223	5.082499	10	5595	1913.995532	...	207.557650	118.573956	6397	14913	163.065803	123	67945	1465.843329	230184.7154	1
4	2310	257.997131	111.275889	987	4732	184.636510	3.804656	8	4080	1819.121324	...	167.510556	126.678802	4624	15532	147.094679	92	37285	1841.702561	158290.0255	1

5 rows × 278 columns

In [15]:

```
import pandas as pd

import numpy as np

from scipy.stats import shapiro

from scipy.special import boxcox1p

from sklearn.preprocessing import PowerTransformer

from scipy.special import boxcox1p

from sklearn.preprocessing import PowerTransformer


# Assume your dataset is stored in a variable 'data'

# Exclude the 'class' column from the analysis

target_variable = 'class'
```

```
columns_to_transform = [col for col in new_data.columns if col != target_variable]


# Define a function to apply Box-Cox transformation only to non-normally distributed columns
def apply_box_cox_if_needed(column_data):
    stat, p = shapiro(column_data)
    alpha = 0.05

    if p > alpha:
        # Data is normally distributed, no transformation needed
        return column_data
    else:
        # Data is not normally distributed, apply Box-Cox transformation
        transformed_data = boxcox1p(column_data, 0)  # Set the lambda parameter to 0
        return transformed_data

# Apply the Box-Cox transformation only to non-normally distributed columns
for column in columns_to_transform:
    new_data[column] = apply_box_cox_if_needed(new_data[column])

# You can also use sklearn's PowerTransformer for an alternative approach
# power_transformer = PowerTransformer(method='box-cox', standardize=False)
# data[columns_to_transform] = power_transformer.fit_transform(data[columns_to_transform])
In [16]:
new_data.shape
Out[16]:
(174, 278)
```

Transform or Remove Outliers: After identifying outliers, you can choose one of the following methods to handle them:

a. Winsorization: In winsorization, you replace the outliers with a predefined percentile value. For example, you can replace values below the 5th percentile with the 5th percentile value and values above the 95th percentile with the 95th percentile value.

b. Log Transformation: For positively skewed data, applying a log transformation can make the distribution more normal and reduce the impact of outliers.

c. Box-Cox Transformation: The Box-Cox transformation is useful for handling data that does not follow a normal distribution. It can help mitigate the impact of outliers.

d. Robust Scaler: You can use the RobustScaler from scikit-learn to scale your features while being robust to outliers.

e. Clipping: Clip the data by setting a lower and upper bound for each feature to limit the effect of outliers.

In [33]:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.stats import shapiro


# Load your dataset into a pandas DataFrame

#df = pd.read_csv('your_dataset.csv')


# Exclude the 'class' column from the analysis

columns_to_check = [col for col in new_data.columns if col != 'class']


for column in columns_to_check:

    df = new_data[column]


    # Check for normality using Shapiro-Wilk test

    stat, p = shapiro(df)


    # Set a significance level (e.g., 0.05) for the test
```

```
    alpha = 0.05

    if p > alpha:
        print(f'{column}: Normally distributed (p-value: {p})')
    else:
        print(f'{column}: Not normally distributed (p-value: {p})')

    # Create a histogram for visualization
    plt.figure(figsize=(8, 6))
    sns.histplot(df, kde=True)
    plt.title(f'{column} Histogram')
    plt.show()
```

```
air_time1: Not normally distributed (p-value: 5.4516080126632005e-05)

gmrt_in_air1: Normally distributed (p-value: 0.057341672480106354)

gmrt_on_paper1: Normally distributed (p-value: 0.0702226310968399)

max_x_extension1: Not normally distributed (p-value: 7.548619507247167e-10)

max_y_extension1: Not normally distributed (p-value: 1.332422128359767e-13)

mean_gmrt1: Normally distributed (p-value: 0.18065765500068665)

mean_speed_in_air1: Not normally distributed (p-value: 0.002628128044307232)

num_of_pendown1: Not normally distributed (p-value: 0.03727042302489281)

paper_time1: Not normally distributed (p-value: 8.511134979016788e-07)
```

pressure_mean1: Not normally distributed (p-value: 6.814674828126491e-14)

pressure_var1: Not normally distributed (p-value: 1.1656831702566706e-05)

air_time2: Not normally distributed (p-value: 0.007202865555882454)

gmrt_in_air2: Not normally distributed (p-value: 1.4878706679155584e-05)

gmrt_on_paper2: Not normally distributed (p-value: 2.046007113731818e-11)

max_x_extension2: Not normally distributed (p-value: 2.3357484349451063e-11)

max_y_extension2: Not normally distributed (p-value: 1.830527656441053e-27)

mean_gmrt2: Not normally distributed (p-value: 0.013223972171545029)

mean_speed_in_air2: Not normally distributed (p-value: 0.04283406585454941)

mean_speed_on_paper2: Normally distributed (p-value: 0.25253400206565857)

num_of_pendown2: Not normally distributed (p-value: 1.113683312237157e-10)

paper_time2: Not normally distributed (p-value: 2.750060497247708e-20)

pressure_mean2: Not normally distributed (p-value: 2.2306035874975413e-26)

pressure_var2: Not normally distributed (p-value: 7.308962102254265e-18)

air_time3: Not normally distributed (p-value: 3.194918826920912e-05)

gmrt_in_air3: Not normally distributed (p-value: 0.004750903695821762)

gmrt_on_paper3: Not normally distributed (p-value: 1.1355686326871694e-16)

max_x_extension3: Not normally distributed (p-value: 1.2386550644592465e-26)

max_y_extension3: Not normally distributed (p-value: 1.0378468050242784e-11)

mean_gmrt3: Not normally distributed (p-value: 0.008587611839175224)

mean_speed_in_air3: Not normally distributed (p-value: 0.006499945651739836)

mean_speed_on_paper3: Normally distributed (p-value: 0.1257176697254181)

num_of_pendown3: Not normally distributed (p-value: 1.1439017322343453e-13)

paper_time3: Not normally distributed (p-value: 6.371712582746214e-22)

pressure_mean3: Not normally distributed (p-value: 1.930274688267378e-26)

pressure_var3: Not normally distributed (p-value: 5.77221042476737e-20)

total_time3: Not normally distributed (p-value: 0.0015734012704342604)

air_time4: Not normally distributed (p-value: 0.0001756747078616172)

gmrt_in_air4: Normally distributed (p-value: 0.4931311309337616)

gmrt_on_paper4: Normally distributed (p-value: 0.3757934272289276)

max_x_extension4: Not normally distributed (p-value: 2.402216937450922e-16)

max_y_extension4: Not normally distributed (p-value: 1.6719935104408123e-18)

mean_gmrt4: Normally distributed (p-value: 0.2466377466917038)

mean_speed_in_air4: Not normally distributed (p-value: 1.9553908714442514e-06)

num_of_pendown4: Not normally distributed (p-value: 7.668549768715532e-14)

paper_time4: Not normally distributed (p-value: 0.005059011746197939)

pressure_mean4: Not normally distributed (p-value: 6.836564963910162e-22)

pressure_var4: Not normally distributed (p-value: 0.0027248815167695284)

total_time4: Not normally distributed (p-value: 0.0008658308070152998)

air_time5: Not normally distributed (p-value: 1.5699577193828418e-09)

gmrt_in_air5: Not normally distributed (p-value: 1.7192953657985122e-11)

gmrt_on_paper5: Not normally distributed (p-value: 7.504296019495999e-15)

max_x_extension5: Not normally distributed (p-value: 2.3788518138551554e-26)

max_y_extension5: Not normally distributed (p-value: 2.2375957917164358e-26)

mean_gmrt5: Not normally distributed (p-value: 0.005409533623605967)

mean_speed_in_air5: Not normally distributed (p-value: 3.111996193183586e-06)

num_of_pendown5: Not normally distributed (p-value: 2.309032143005957e-15)

paper_time5: Not normally distributed (p-value: 1.5149965137508973e-22)

pressure_mean5: Not normally distributed (p-value: 2.0798527692515914e-27)

pressure_var5: Not normally distributed (p-value: 5.629228352629064e-18)

total_time5: Not normally distributed (p-value: 1.1245532283155057e-09)

air_time6: Normally distributed (p-value: 0.05716954544186592)

gmrt_in_air6: Normally distributed (p-value: 0.2989577651023865)

gmrt_on_paper6: Not normally distributed (p-value: 0.0005582704325206578)

max_x_extension6: Not normally distributed (p-value: 7.611096752846436e-10)

max_y_extension6: Not normally distributed (p-value: 3.0541316391463624e-06)

mean_acc_in_air6: Not normally distributed (p-value: 2.411651156553063e-10)

mean_gmrt6: Not normally distributed (p-value: 0.005374602973461151)

mean_speed_in_air6: Normally distributed (p-value: 0.5656395554542542)

num_of_pendown6: Not normally distributed (p-value: 1.2338174926185275e-10)

paper_time6: Not normally distributed (p-value: 4.817247045707518e-09)

pressure_mean6: Not normally distributed (p-value: 3.192496030261224e-16)

pressure_var6: Not normally distributed (p-value: 0.02279060147702694)

air_time7: Not normally distributed (p-value: 7.125673384678066e-10)

gmrt_in_air7: Normally distributed (p-value: 0.7590720057487488)

gmrt_on_paper7: Not normally distributed (p-value: 1.0008337994804606e-05)

max_x_extension7: Not normally distributed (p-value: 2.1002918515478086e-07)

max_y_extension7: Not normally distributed (p-value: 1.1985154202420745e-08)

mean_acc_in_air7: Not normally distributed (p-value: 0.0018518698634579778)

mean_gmrt7: Not normally distributed (p-value: 0.00039791627204976976)

num_of_pendown7: Not normally distributed (p-value: 2.5192908452984e-18)

paper_time7: Not normally distributed (p-value: 1.5614590154200414e-07)

pressure_mean7: Not normally distributed (p-value: 7.065525991266236e-17)

pressure_var7: Not normally distributed (p-value: 0.004014765843749046)

air_time8: Not normally distributed (p-value: 0.0004605647991411388)

gmrt_in_air8: Normally distributed (p-value: 0.11564657837152481)

gmrt_on_paper8: Normally distributed (p-value: 0.39749547839164734)

max_x_extension8: Not normally distributed (p-value: 2.696657793421764e-05)

max_y_extension8: Normally distributed (p-value: 0.11824897676706314)

mean_acc_in_air8: Not normally distributed (p-value: 2.3084044051516674e-19)

mean_speed_in_air8: Not normally distributed (p-value: 0.0010446568485349417)

num_of_pendown8: Not normally distributed (p-value: 3.3863763049622107e-18)

paper_time8: Not normally distributed (p-value: 2.5851744794636033e-05)

pressure_mean8: Not normally distributed (p-value: 9.912963480108454e-19)

pressure_var8: Normally distributed (p-value: 0.559200644493103)

total_time8: Not normally distributed (p-value: 3.0258464903454296e-05)

air_time9: Normally distributed (p-value: 0.08293572813272476)

gmrt_in_air9: Normally distributed (p-value: 0.12289290130138397)

gmrt_on_paper9: Normally distributed (p-value: 0.7181360721588135)

max_x_extension9: Not normally distributed (p-value: 1.245662133442238e-07)

max_y_extension9: Normally distributed (p-value: 0.09793242812156677)

mean_acc_in_air9: Not normally distributed (p-value: 3.8979495669156574e-19)

mean_speed_in_air9: Not normally distributed (p-value: 1.678958687989507e-05)

num_of_pendown9: Not normally distributed (p-value: 4.41466414680537e-15)

paper_time9: Not normally distributed (p-value: 0.01666390523314476)

pressure_mean9: Not normally distributed (p-value: 2.7927519995199086e-19)

pressure_var9: Not normally distributed (p-value: 0.004460291936993599)

total_time9: Not normally distributed (p-value: 0.00012160659389337525)

air_time10: Not normally distributed (p-value: 0.0012427459005266428)

gmrt_in_air10: Normally distributed (p-value: 0.4081997871398926)

gmrt_on_paper10: Not normally distributed (p-value: 0.0023462981916964054)

max_x_extension10: Not normally distributed (p-value: 1.2938346571900183e-06)

max_y_extension10: Not normally distributed (p-value: 0.010952007956802845)

mean_gmrt10: Normally distributed (p-value: 0.415311336517334)

mean_speed_in_air10: Not normally distributed (p-value: 0.0031196135096251965)

num_of_pendown10: Not normally distributed (p-value: 0.0008265133947134018)

paper_time10: Not normally distributed (p-value: 3.2114504389113563e-09)

pressure_mean10: Not normally distributed (p-value: 4.773890925615038e-16)

pressure_var10: Not normally distributed (p-value: 4.906136837234953e-06)

air_time11: Not normally distributed (p-value: 2.9945611004222883e-06)

gmrt_in_air11: Normally distributed (p-value: 0.49564066529273987)

gmrt_on_paper11: Not normally distributed (p-value: 1.0113697499036789e-05)

max_x_extension11: Not normally distributed (p-value: 2.2549218670064874e-07)

max_y_extension11: Not normally distributed (p-value: 4.3141881733710363e-10)

mean_gmrt11: Normally distributed (p-value: 0.09973996132612228)

mean_speed_in_air11: Not normally distributed (p-value: 0.0029531000182032585)

num_of_pendown11: Not normally distributed (p-value: 0.00036150170490145683)

paper_time11: Not normally distributed (p-value: 1.4750264121232703e-08)

pressure_mean11: Not normally distributed (p-value: 1.8905299753863766e-18)

pressure_var11: Not normally distributed (p-value: 1.2389920200917004e-09)

air_time12: Not normally distributed (p-value: 2.626629047597362e-08)

gmrt_in_air12: Not normally distributed (p-value: 0.005422566551715136)

gmrt_on_paper12: Not normally distributed (p-value: 3.9343498077493155e-12)

max_x_extension12: Not normally distributed (p-value: 2.4595284936075156e-19)

max_y_extension12: Not normally distributed (p-value: 4.454006884060533e-24)

mean_speed_in_air12: Not normally distributed (p-value: 0.007097810506820679)

num_of_pendown12: Not normally distributed (p-value: 1.9027505913982168e-05)

paper_time12: Not normally distributed (p-value: 3.118828357978288e-21)

pressure_mean12: Not normally distributed (p-value: 3.39028966940988e-26)

pressure_var12: Not normally distributed (p-value: 1.2921652012511097e-18)

air_time13: Not normally distributed (p-value: 0.01795203424990177)

gmrt_in_air13: Normally distributed (p-value: 0.1848195195198059)

gmrt_on_paper13: Normally distributed (p-value: 0.0530671551823616)

max_x_extension13: Not normally distributed (p-value: 1.7526748763430078e-07)

max_y_extension13: Normally distributed (p-value: 0.576850712299346 9)

mean_acc_in_air13: Not normally distributed (p-value: 8.177553579932548e-21)

mean_speed_in_air13: Not normally distributed (p-value: 0.00013942619261797518)

num_of_pendown13: Not normally distributed (p-value: 1.3246466323835193e-06)

paper_time13: Not normally distributed (p-value: 0.004344233311712742)

pressure_mean13: Not normally distributed (p-value: 3.494584143588488e-18)

pressure_var13: Not normally distributed (p-value: 2.785990545817185e-06)

total_time13: Not normally distributed (p-value: 0.00011249927774770185)

air_time14: Not normally distributed (p-value: 1.7580928934890494e-09)

gmrt_in_air14: Not normally distributed (p-value: 0.001542397541925311)

gmrt_on_paper14: Not normally distributed (p-value: 9.461624703860022e-17)

max_x_extension14: Not normally distributed (p-value: 1.299306189440734e-17)

max_y_extension14: Not normally distributed (p-value: 8.034914199674734e-21)

num_of_pendown14: Not normally distributed (p-value: 2.1500497950910358e-06)

paper_time14: Not normally distributed (p-value: 1.789504773311812e-23)

pressure_mean14: Not normally distributed (p-value: 7.71619918729099e-26)

pressure_var14: Not normally distributed (p-value: 2.6553343321842585e-22)

air_time15: Not normally distributed (p-value: 0.0005103567382320762)

gmrt_in_air15: Normally distributed (p-value: 0.7159608006477356)

gmrt_on_paper15: Normally distributed (p-value: 0.16811050474643707)

max_x_extension15: Normally distributed (p-value: 0.10474496334791183)

max_y_extension15: Not normally distributed (p-value: 0.0003043112519662827)

mean_gmrt15: Normally distributed (p-value: 0.6767081022262573)

num_of_pendown15: Not normally distributed (p-value: 0.008873236365616322)

paper_time15: Not normally distributed (p-value: 2.9597333195852116e-05)

pressure_mean15: Not normally distributed (p-value: 2.425162901173363e-15)

pressure_var15: Normally distributed (p-value: 0.10215678066015244)

air_time16: Normally distributed (p-value: 0.27988553047180176)

gmrt_in_air16: Normally distributed (p-value: 0.6374914646148682)

gmrt_on_paper16: Not normally distributed (p-value: 7.767011860060347e-13)

max_x_extension16: Not normally distributed (p-value: 1.903154793636073e-19)

max_y_extension16: Not normally distributed (p-value: 1.7936575516870616e-21)

mean_gmrt16: Normally distributed (p-value: 0.710594654083252)

mean_speed_in_air16: Normally distributed (p-value: 0.08623342961072922)

mean_speed_on_paper16: Normally distributed (p-value: 0.9590127468109131)

num_of_pendown16: Not normally distributed (p-value: 1.0285243661201093e-06)

paper_time16: Not normally distributed (p-value: 1.3308762199428257e-18)

pressure_mean16: Not normally distributed (p-value: 1.0124244801910992e-24)

pressure_var16: Not normally distributed (p-value: 1.0048963079973718e-21)

air_time17: Not normally distributed (p-value: 1.0813475455506705e-05)

gmrt_in_air17: Normally distributed (p-value: 0.10904110968112946)

gmrt_on_paper17: Normally distributed (p-value: 0.721794843673706)

max_x_extension17: Not normally distributed (p-value: 2.3954666574109885e-27)

max_y_extension17: Not normally distributed (p-value: 1.290211198342458e-24)

mean_acc_in_air17: Normally distributed (p-value: 0.42931458353996277)

mean_speed_on_paper17: Normally distributed (p-value: 0.4475163221359253)

num_of_pendown17: Not normally distributed (p-value: 4.420437562657753e-07)

paper_time17: Not normally distributed (p-value: 5.397403444323823e-13)

pressure_mean17: Not normally distributed (p-value: 1.31649922119071e-17)

pressure_var17: Not normally distributed (p-value: 3.147342386711216e-09)

air_time18: Not normally distributed (p-value: 0.0002847369760274887)

gmrt_in_air18: Normally distributed (p-value: 0.3685950040817261)

gmrt_on_paper18: Not normally distributed (p-value: 1.6015856131000696e-17)

max_x_extension18: Not normally distributed (p-value: 2.935852613748577e-21)

max_y_extension18: Not normally distributed (p-value: 9.276007914075805e-24)

mean_gmrt18: Normally distributed (p-value: 0.06996230781078339)

mean_speed_in_air18: Not normally distributed (p-value: 0.02558237314224243)

num_of_pendown18: Not normally distributed (p-value: 5.768662214578058e-10)

paper_time18: Not normally distributed (p-value: 3.7543904914675524e-22)

pressure_mean18: Not normally distributed (p-value: 1.1132959762302902e-25)

pressure_var18: Not normally distributed (p-value: 9.515064240752446e-21)

air_time19: Not normally distributed (p-value: 4.759299046019296e-18)

gmrt_in_air19: Not normally distributed (p-value: 0.01459877286106348)

gmrt_on_paper19: Not normally distributed (p-value: 1.219594980561567e-09)

max_x_extension19: Not normally distributed (p-value: 6.122802607124232e-23)

max_y_extension19: Not normally distributed (p-value: 1.0190776289360937e-20)

mean_gmrt19: Not normally distributed (p-value: 0.00041402195347473025)

num_of_pendown19: Not normally distributed (p-value: 1.67150716379183e-07)

paper_time19: Not normally distributed (p-value: 1.2670863490382711e-11)

pressure_mean19: Not normally distributed (p-value: 3.2151742786327737e-17)

pressure_var19: Not normally distributed (p-value: 3.850528446491808e-05)

air_time20: Not normally distributed (p-value: 1.144831895949494e-09)

gmrt_in_air20: Normally distributed (p-value: 0.09526519477367401)

gmrt_on_paper20: Normally distributed (p-value: 0.7233409285545349)

max_x_extension20: Not normally distributed (p-value: 1.0550508383611046e-10)

max_y_extension20: Normally distributed (p-value: 0.10347852110862732)

mean_gmrt20: Not normally distributed (p-value: 0.00044637691462412477)

mean_speed_in_air20: Normally distributed (p-value: 0.5436367988586426)

num_of_pendown20: Normally distributed (p-value: 0.057738736271858215)

paper_time20: Not normally distributed (p-value: 1.2396608045639468e-11)

pressure_mean20: Not normally distributed (p-value: 5.150908466811479e-16)

pressure_var20: Not normally distributed (p-value: 0.0007825459470041096)

air_time21: Not normally distributed (p-value: 0.019654881209135056)

gmrt_in_air21: Not normally distributed (p-value: 0.005122125148773193)

gmrt_on_paper21: Not normally distributed (p-value: 0.0268178042024374)

max_x_extension21: Not normally distributed (p-value: 2.560676734910691e-23)

max_y_extension21: Not normally distributed (p-value: 1.3439782221483307e-23)

mean_acc_in_air21: Not normally distributed (p-value: 2.84711626019965e-14)

mean_speed_in_air21: Not normally distributed (p-value: 2.4912122171372175e-05)

num_of_pendown21: Not normally distributed (p-value: 1.9965485989814624e-05)

paper_time21: Normally distributed (p-value: 0.16457046568393707)

pressure_mean21: Not normally distributed (p-value: 5.194122328987616e-20)

pressure_var21: Not normally distributed (p-value: 2.003181361942552e-05)

total_time21: Not normally distributed (p-value: 0.03016779012978077)

air_time22: Not normally distributed (p-value: 1.840820940124388e-09)

gmrt_in_air22: Normally distributed (p-value: 0.06383130699396133)

gmrt_on_paper22: Normally distributed (p-value: 0.21385954320430756)

max_x_extension22: Not normally distributed (p-value: 6.409021011677396e-07)

max_y_extension22: Normally distributed (p-value: 0.12974785268306732)

mean_gmrt22: Normally distributed (p-value: 0.08634176850318909)

num_of_pendown22: Not normally distributed (p-value: 2.5820750051974706e-11)

paper_time22: Not normally distributed (p-value: 1.1242840969316603e-07)

pressure_mean22: Not normally distributed (p-value: 2.7610804979448945e-15)

pressure_var22: Normally distributed (p-value: 0.400381863117218)

air_time23: Not normally distributed (p-value: 2.2382847575386222e-08)

gmrt_in_air23: Not normally distributed (p-value: 0.0066553871147334576)

gmrt_on_paper23: Normally distributed (p-value: 0.1497705727815628)

max_x_extension23: Not normally distributed (p-value: 1.7320309683016566e-12)

max_y_extension23: Normally distributed (p-value: 0.3110130727291107)

mean_gmrt23: Normally distributed (p-value: 0.17320284247398376)

num_of_pendown23: Not normally distributed (p-value: 7.926066747376836e-11)

paper_time23: Not normally distributed (p-value: 3.193130482248563e-10)

pressure_mean23: Not normally distributed (p-value: 1.278641973659518e-15)

pressure_var23: Not normally distributed (p-value: 0.00022610924497712404)

air_time24: Not normally distributed (p-value: 0.001259656623005867)

gmrt_in_air24: Normally distributed (p-value: 0.711179256439209)

gmrt_on_paper24: Normally distributed (p-value: 0.3921167254447937)

max_x_extension24: Not normally distributed (p-value: 0.015031039714813232)

max_y_extension24: Not normally distributed (p-value: 0.0018899893620982766)

mean_speed_on_paper24: Normally distributed (p-value: 0.0903850570321083)

num_of_pendown24: Not normally distributed (p-value: 2.5705819851201683e-12)

paper_time24: Normally distributed (p-value: 0.3440977931022644)

pressure_mean24: Not normally distributed (p-value: 4.2147745356487226e-15)

pressure_var24: Normally distributed (p-value: 0.2989948093891144)

air_time25: Not normally distributed (p-value: 5.746380038473831e-10)

gmrt_in_air25: Not normally distributed (p-value: 0.014860849827528)

gmrt_on_paper25: Normally distributed (p-value: 0.44225162267684937)

max_x_extension25: Not normally distributed (p-value: 0.0005910273757763207)

max_y_extension25: Not normally distributed (p-value: 0.00023294304264709353)

mean_gmrt25: Not normally distributed (p-value: 0.000282899709418416)

num_of_pendown25: Normally distributed (p-value: 0.4818323850631714)

paper_time25: Not normally distributed (p-value: 9.483613894190057e-08)

pressure_mean25: Not normally distributed (p-value: 7.626888474380653e-17)

pressure_var25: Not normally distributed (p-value: 5.066341327619739e-05)

In [34]:

new_data.head()

Out[34]:

	air_time1	gmrt_in_air1	gmrt_on_paper1	max_x_extension1	max_y_extension1	mean_gmrt1	mean_speed_in_air1	num_of_pendown1	paper_time1	pressure_mean1	...	gmrt_in_air25	gmrt_on_paper25	max_x_extension25	max_y_extension25	mean_gmrt25	num_of_pendown25	paper_time25	pressure_mean25	pressure_var25	class
0	8.548886	4.802415	4.475669	6.864848	8.795128	4.652328	1.039597	3.135494	9.280892	7.426687	...	5.637031	219.829989	9.217018	9.490696	5.524373	4.276666	10.599655	7.467530	12.598465	1
1	10.858634	4.756330	4.436144	7.435438	8.853523	4.608997	1.035937	2.484907	9.430359	7.452502	...	4.467262	68.398886	8.904630	9.634496	4.360016	4.867534	11.749585	7.317059	12.538054	1
2	7.863651	5.442132	5.157686	7.755339	8.666130	5.309989	1.476670	2.397895	8.712924	7.327290	...	5.377034									

	171.954494	8.947546	9.555914	5.271290	4.317488	10.725050
	7.267137	11.880430	1			
3	7.664347	5.914593	5.215985	7.471363	9.006999	5.625094
	1.805416	2.397895	8.629807	7.557471	...	5.340216
	118.573956	8.763741	9.610056	5.100268	4.820282	11.126469
	7.290868	12.346642	1			
4	7.745436	5.556817	4.720959	6.895683	8.462315	5.223791
	1.569585	2.197225	8.314097	7.506658	...	5.126998
	126.678802	8.439232	9.650722	4.997852	4.532599	10.526373
	7.518989	11.972191	1			

5 rows × 278 columns

In [17]:

```
# Define features and target variable
X = new_data.drop(columns=['class'])
y = new_data[['class']]
```

In [18]:

```
type(y)
```

Out[18]:

pandas.core.frame.DataFrame

In [19]:

```
from sklearn.preprocessing import StandardScaler

# Initialize the StandardScaler
scaler = StandardScaler()

# Convert the scaled data back to a DataFrame with column names and data types
X = pd.DataFrame(scaler.fit_transform(X), columns=X.columns, index=X.index)
```

In [20]:

```
# Split the dataset into a training set (70%) and a test set (30%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [21]:

```
print(type(X_train))  # Output: <class 'pandas.core.frame.DataFrame'>
print(type(X_test))   # Output: <class 'pandas.core.frame.DataFrame'>
print(type(y_train))  # Output: <class 'pandas.core.series.Series'>
print(type(y_test))   # Output: <class 'pandas.core.series.Series'>
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.frame.DataFrame'>
```

In [43]:

```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

# Fit the PCA transformer to your data
pca = PCA()
pca.fit(X)

# Calculate the cumulative explained variance
explained_variance = pca.explained_variance_ratio_
cumulative_variance = explained_variance.cumsum()

# Plot the cumulative explained variance
plt.plot(cumulative_variance)
plt.xlabel("Number of Components")
plt.ylabel("Cumulative Explained Variance")
plt.grid()
plt.show()
```

In [44]:

```
from sklearn.decomposition import PCA
```

```
# Fit the PCA transformer to your data
pca = PCA()
pca.fit(X)

# Set the desired explained variance threshold (e.g., 95%)
desired_variance = 0.95

# Calculate the cumulative explained variance
explained_variance = pca.explained_variance_ratio_
cumulative_variance = explained_variance.cumsum()

# Find the number of components that exceed the desired variance
n_components = (cumulative_variance >= desired_variance).argmax() + 1

print(f"Number of components to explain {desired_variance * 100}% of variance: {n_components}")
```

```
Number of components to explain 95.0% of variance: 87
```

In [45]:

```
from sklearn.decomposition import PCA

# Create a PCA transformer with a reduced number of components (e.g., 9)
pca = PCA(n_components=0.95)

# Fit PCA on the training data and transform both the training and test data
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)
```

In [49]:

```
print("Number of samples in X_train_pca:", X_train_pca.shape[0])
print("Number of samples in X_test_pca:", X_test_pca.shape[0])
```

Number of samples in X_train_pca: 139

Number of samples in X_test_pca: 35

In [40]:

```
from sklearn.metrics import  accuracy_score, confusion_matrix, ConfusionMatrixDisplay ,recall_score, precision_score, f1_score, classification_report


import xgboost as xgb

from lightgbm import LGBMClassifier

import lazypredict

from lazypredict.Supervised import LazyClassifier


import time

import warnings

warnings.filterwarnings('ignore')
```

In [41]:

```
from IPython.display import clear_output


clf = LazyClassifier(verbose=0,
                     ignore_warnings=True,
                     custom_metric=None,
                     predictions=False,
                     random_state=3,
                     classifiers='all')


models, predictions = clf.fit(X_train, X_test, y_train, y_test)


# Clear the output

clear_output()
```

In [42]:

models

Out[42]:

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
XGBClassifier	0.91	0.93	0.93	0.91	0.45
AdaBoostClassifier	0.89	0.89	0.89	0.89	1.46
ExtraTreesClassifier	0.89	0.89	0.89	0.89	0.39
RandomForestClassifier	0.89	0.89	0.89	0.89	0.71
LGBMClassifier	0.89	0.89	0.89	0.89	0.25
NuSVC	0.86	0.86	0.86	0.86	0.08
LogisticRegression	0.86	0.86	0.86	0.86	0.09
SVC	0.86	0.86	0.86	0.86	0.07
CalibratedClassifierCV	0.86	0.86	0.86	0.86	0.17
PassiveAggressiveClassifier	0.86	0.86	0.86	0.86	0.07
LinearSVC	0.86	0.86	0.86	0.86	0.10
GaussianNB	0.83	0.84	0.84	0.83	0.06
BaggingClassifier	0.83	0.83	0.83	0.83	0.64
Perceptron	0.83	0.83	0.83	0.83	0.07
SGDClassifier	0.83	0.83	0.83	0.83	0.07
BernoulliNB	0.83	0.83	0.83	0.83	0.07
NearestCentroid	0.80	0.81	0.81	0.80	0.07
RidgeClassifierCV	0.80	0.80	0.80	0.80	0.10
RidgeClassifier	0.77	0.78	0.78	0.77	0.07
QuadraticDiscriminantAnalysis	0.74	0.72	0.72	0.73	0.11
DecisionTreeClassifier	0.69	0.72	0.72	0.68	0.13
KNeighborsClassifier	0.66	0.70	0.70	0.63	0.36
LinearDiscriminantAnalysis	0.69	0.67	0.67	0.68	0.14
ExtraTreeClassifier	0.66	0.65	0.65	0.66	0.07
LabelSpreading	0.43	0.50	0.50	0.26	0.06

LabelPropagation 0.43 0.50 0.50 0.26 0.08

DummyClassifier 0.43 0.50 0.50 0.26 0.07

Classifiers after PCA

In [50]:

```
from IPython.display import clear_output

pca_clf = LazyClassifier(verbose=0,
                         ignore_warnings=True,
                         custom_metric=None,
                         predictions=False,
                         random_state=3,
                         classifiers='all')

pca_models, predictions = pca_clf.fit(X_train_pca, X_test_pca, y_train, y_test)

# Clear the output
clear_output()
```

In [51]:

```
pca_models
```

Out[51]:

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
LGBMClassifier	0.91	0.93	0.93	0.91	0.16
SGDClassifier	0.91	0.93	0.93	0.91	0.05
Perceptron	0.89	0.90	0.90	0.89	0.06
RandomForestClassifier	0.89	0.90	0.90	0.89	0.64
AdaBoostClassifier	0.89	0.90	0.90	0.89	0.68
RidgeClassifierCV	0.89	0.89	0.89	0.89	0.07
RidgeClassifier	0.89	0.89	0.89	0.89	0.05

LinearDiscriminantAnalysis		0.89	0.89	0.89	0.89	0.07
LogisticRegression	0.89	0.89	0.89	0.89	0.07	
NearestCentroid	0.89	0.89	0.89	0.89	0.06	
CalibratedClassifierCV	0.89	0.89	0.89	0.89	0.17	
ExtraTreesClassifier	0.86	0.88	0.88	0.86	0.37	
LinearSVC	0.86	0.87	0.87	0.86	0.06	
PassiveAggressiveClassifier		0.86	0.87	0.87	0.86	0.05
XGBClassifier	0.86	0.87	0.87	0.86	0.34	
BaggingClassifier		0.86	0.87	0.87	0.86	0.27
SVC	0.83	0.84	0.84	0.83	0.06	
NuSVC	0.83	0.84	0.84	0.83	0.06	
DecisionTreeClassifier		0.80	0.80	0.80	0.80	0.08
GaussianNB	0.74	0.78	0.78	0.74	0.06	
BernoulliNB	0.74	0.75	0.75	0.74	0.06	
ExtraTreeClassifier		0.71	0.72	0.72	0.72	0.06
QuadraticDiscriminantAnalysis		0.54	0.58	0.58	0.51	0.06
LabelSpreading	0.43	0.50	0.50	0.26	0.06	
LabelPropagation		0.43	0.50	0.50	0.26	0.07
KNeighborsClassifier		0.43	0.50	0.50	0.26	0.06
DummyClassifier		0.43	0.50	0.50	0.26	0.05

In [ ]:

In [52]:

```
from sklearn.ensemble import RandomForestClassifier

# Create a Random Forest classifier
clf = RandomForestClassifier(random_state=42)

# Train the classifier on the training data
```

```
clf.fit(X_train, y_train)
```

Out[52]:

```
 RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [53]:

```
y_pred = clf.predict(X_test)
```

In [54]:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print("Model Metrics:")
print(f"Accuracy: {accuracy * 100:.2f}%")
print(f"Precision: {precision * 100:.2f}%")
print(f"Recall: {recall * 100:.2f}%")
print(f"F1-Score: {f1 * 100:.2f}%")
```

Model Metrics:

Accuracy: 88.57%

Precision: 89.26%

Recall: 88.57%

F1-Score: 88.63%

In [55]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

# Define the parameter grid for Random Forest
```

```
param_grid = {
    'n_estimators': [100, 200, 500],
    'max_depth': [3, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Create a RandomForestClassifier
rf_classifier = RandomForestClassifier(random_state=42)

# Use GridSearchCV for hyperparameter tuning
grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)

# Get the best parameters and best model
best_params = grid_search.best_params_
best_rf_model = grid_search.best_estimator_

# Evaluate the best model on the test set
y_pred_best = best_rf_model.predict(X_test)

# Calculate metrics
accuracy_best = accuracy_score(y_test, y_pred_best)
precision_best = precision_score(y_test, y_pred_best)
recall_best = recall_score(y_test, y_pred_best)
f1_best = f1_score(y_test, y_pred_best)

# Print metrics
print("Best Model Metrics:")
```

```
print("Accuracy:", accuracy_best)
print("Precision:", precision_best)
print("Recall:", recall_best)
print("F1-Score:", f1_best)
```

Best Model Metrics:

Accuracy: 0.8857142857142857

Precision: 0.9444444444444444

Recall: 0.85

F1-Score: 0.8947368421052632

In [22]:

```
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import matplotlib.pyplot as plt
import seaborn as sns

# Create an XGBoost classifier
xgb_clf = XGBClassifier(random_state=42)

# Train the classifier on the training data
xgb_clf.fit(X_train, y_train)
y_pred = xgb_clf.predict(X_test)

# Get feature importances
feature_importances = xgb_clf.feature_importances_

# Get indices of the top 20 features
top_20_indices = feature_importances.argsort()[-20:][::-1]

# Get the names of the top 20 features
```

```
top_20_features = X_train.columns[top_20_indices]

# Plot feature importances
plt.figure(figsize=(12, 6))
sns.barplot(x=feature_importances[top_20_indices], y=top_20_features)
plt.title("Top 20 Most Important Features - XGBoost Classifier")
plt.xlabel("Feature Importance")
plt.ylabel("Features")
plt.show()

# Evaluate the model metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print("Model Metrics:")
print(f"Accuracy: {accuracy * 100:.2f}%")
print(f"Precision: {precision * 100:.2f}%")
print(f"Recall: {recall * 100:.2f}%")
print(f"F1-Score: {f1 * 100:.2f}%")
```

```
Model Metrics:
Accuracy: 91.43%
Precision: 92.86%
Recall: 91.43%
F1-Score: 91.47%
```

In [23]:

```
import pandas as pd
```

```
# Create a DataFrame of top 20 features and scores
top_20_features_df = pd.DataFrame({
    'Feature': top_20_features,
    'Importance Score': feature_importances[top_20_indices]
})

# Print the DataFrame in a table format
print(top_20_features_df.to_string())
```

	Feature	Importance Score
0	max_y_extension2	0.083790
1	air_time23	0.080852
2	paper_time22	0.076472
3	total_time9	0.073116
4	gmrt_on_paper15	0.063937
5	max_x_extension18	0.053813
6	air_time15	0.040530
7	mean_speed_in_air12	0.036798
8	num_of_pendown19	0.035610
9	paper_time18	0.032679
10	total_time3	0.030341
11	air_time17	0.030035
12	total_time13	0.025723
13	gmrt_in_air2	0.025026
14	air_time6	0.024599
15	pressure_mean6	0.024340
16	mean_gmrt15	0.022927
17	air_time22	0.022053
18	air_time8	0.019750

19 air_time24 0.014558

In [25]:

```
from sklearn.metrics import confusion_matrix

# Generate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Print the confusion matrix
print(conf_matrix)
```

[[15 0]

[3 17]]

In [26]:

```
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score
import matplotlib.pyplot as plt
import seaborn as sns

# Generate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Calculate precision, recall, and F1-score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Create a text box to display metrics
metrics_text = f"""
Precision: {precision:.2f}
Recall: {recall:.2f}
F1-Score: {f1:.2f}
```

"""

Plot confusion matrix with labels and metrics

plt.figure(figsize=(8, 6))

sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", cbar=False)

plt.title("Confusion Matrix - XGBoost Classifier")

plt.xlabel("Predicted Labels")

plt.ylabel("True Labels")

plt.xticks(ticks=[0.5, 1.5], labels=['Predicted False', 'Predicted True'])

plt.yticks(ticks=[0.5, 1.5], labels=['Actual False', 'Actual True'])

plt.text(x=0.7, y=1.4, s=metrics_text, bbox=dict(boxstyle='round', facecolor='wheat', alpha=0.5))

plt.show()

In [27]:

from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score

Generate confusion matrix and metrics

conf_matrix = confusion_matrix(y_test, y_pred)

precision = precision_score(y_test, y_pred)

recall = recall_score(y_test, y_pred)

f1 = f1_score(y_test, y_pred)

Extract TP, TN, FP, FN from the confusion matrix

tn, fp, fn, tp = conf_matrix.ravel()

Create a table-like output with clear labels

print("Confusion Matrix Metrics:")

print("-" * 30)

print(f"True Positives (TP): {tp}")

```
print(f"True Negatives (TN): {tn}")
print(f"False Positives (FP): {fp}")
print(f"False Negatives (FN): {fn}")
print("-" * 30)
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1-Score: {f1:.2f}")
```

```
Confusion Matrix Metrics:
------------------------------
True Positives (TP): 17
True Negatives (TN): 15
False Positives (FP): 0
False Negatives (FN): 3
------------------------------
Precision: 1.00
Recall: 0.85
F1-Score: 0.92
```

In [28]:

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

```
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```

In [ ]:

In [60]:

```
print("X_train shape:",X_train.shape)
print("X_test shape:",X_test.shape)
print("y_train shape:",y_train.shape)
print("y_test shape:",y_test.shape)
```

X_train shape: (139, 277)

X_test shape: (35, 277)

y_train shape: (139, 1)

y_test shape: (35, 1)

In [61]:

```
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV

# Create an XGBoost classifier
clf = XGBClassifier(random_state=42)

# Define the parameter grid
param_grid = {
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 4, 5],
    'n_estimators': [50, 100, 200],
}
```

```
# Use GridSearchCV to find the best parameters
grid_search = GridSearchCV(clf, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)

# Get the best parameters
best_params = grid_search.best_params_
print("Best Parameters:", best_params)

# Use the best parameters to train the model
best_clf = XGBClassifier(**best_params, random_state=42)
best_clf.fit(X_train, y_train)

# Make predictions and evaluate the model
y_pred = best_clf.predict(X_test)
# Add your evaluation metrics here
# Evaluate the model metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print("Model Metrics:")
print(f"Accuracy: {accuracy * 100:.2f}%")
print(f"Precision: {precision * 100:.2f}%")
print(f"Recall: {recall * 100:.2f}%")
print(f"F1-Score: {f1 * 100:.2f}%")
```

```
Best Parameters: {'learning_rate': 0.2, 'max_depth': 4, 'n_estimators': 50}
Model Metrics:
Accuracy: 91.43%
```

Precision: 92.86%

Recall: 91.43%

F1-Score: 91.47%

In [62]:

```
from catboost import CatBoostClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Create a CatBoost classifier
catboost_clf = CatBoostClassifier(random_state=42)

# Train the classifier on the training data
catboost_clf.fit(X_train, y_train)

# Make predictions
y_pred_catboost = catboost_clf.predict(X_test)

# Evaluate the model metrics
accuracy_catboost = accuracy_score(y_test, y_pred_catboost)
precision_catboost = precision_score(y_test, y_pred_catboost, average='weighted')
recall_catboost = recall_score(y_test, y_pred_catboost, average='weighted')
f1_catboost = f1_score(y_test, y_pred_catboost, average='weighted')

print("CatBoost Model Metrics:")
print(f"Accuracy: {accuracy_catboost * 100:.2f}%")
print(f"Precision: {precision_catboost * 100:.2f}%")
print(f"Recall: {recall_catboost * 100:.2f}%")
print(f"F1-Score: {f1_catboost * 100:.2f}%")
```

```
Learning rate set to 0.004436
CatBoost Model Metrics:
```

Accuracy: 91.43%

Precision: 92.86%

Recall: 91.43%

F1-Score: 91.47%

In [63]:

```
from xgboost import XGBClassifier

from sklearn.feature_selection import RFECV

from sklearn.model_selection import StratifiedKFold

from sklearn.metrics import make_scorer, f1_score, accuracy_score, precision_score, recall_score


# Define the XGBoost classifier

xgb_clf = XGBClassifier(random_state=42)


# Define the RFECV selector with cross-validation

rfecv = RFECV(estimator=xgb_clf, step=1, cv=StratifiedKFold(5), scoring=make_scorer(f1_score,
average='weighted'))


# Fit the RFECV selector on your data

rfecv.fit(X_train, y_train)


# Get the optimal number of features

optimal_n_features = rfecv.n_features_


# Get the selected features

selected_features = [feature for (feature, support) in zip(X_train.columns, rfecv.support_) if support]


# Transform your data to include only the selected features

X_train_rfecv = X_train[selected_features]

X_test_rfecv = X_test[selected_features]
```

```
# Train your model on the data with the selected features
xgb_clf.fit(X_train_rfecv, y_train)

# Make predictions
y_pred = xgb_clf.predict(X_test_rfecv)

# Evaluate your model
f1 = f1_score(y_test, y_pred, average='weighted')
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')

print(f"Optimal number of features: {optimal_n_features}")
print("Model Metrics:")
print(f"Accuracy: {accuracy * 100:.2f}%")
print(f"Precision: {precision * 100:.2f}%")
print(f"Recall: {recall * 100:.2f}%")
print(f"F1-Score: {f1 * 100:.2f}%")
```

```
Optimal number of features: 12
Model Metrics:
Accuracy: 88.57%
Precision: 89.26%
Recall: 88.57%
F1-Score: 88.63%
```

In [ ]:

In [64]:

```
from keras.models import Sequential
from keras.layers import Dense
```

```
# Define the number of features in your dataset
input_dim = X_train.shape[1]

# Create a simple neural network
model = Sequential()
model.add(Dense(64, activation='relu', input_dim=input_dim))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32)

# Evaluate the model on the test dataset
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print("Test Loss:", test_loss)
print("Test Accuracy:", test_accuracy)

# Make predictions on the test dataset
predictions = model.predict(X_test)

# If you're working with binary classification, you can round the predictions to get binary class labels (0 or 1)
binary_predictions = (predictions > 0.5).astype(int)

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report
```

```
# Assuming you have binary labels (0 or 1) in y_test and binary_predictions
# If you have multiclass labels, adjust the average parameter in the metrics functions

# Calculate accuracy
accuracy = accuracy_score(y_test, binary_predictions)
print("Accuracy:", accuracy)

# Calculate precision
precision = precision_score(y_test, binary_predictions)
print("Precision:", precision)

# Calculate recall
recall = recall_score(y_test, binary_predictions)
print("Recall:", recall)

# Calculate F1-score
f1 = f1_score(y_test, binary_predictions)
print("F1-Score:", f1)

# Generate a classification report
report = classification_report(y_test, binary_predictions)
print("Classification Report:\n", report)
Epoch 1/10
5/5 [==============================] - 2s 6ms/step - loss: 0.8008 - accuracy: 0.5036
Epoch 2/10
5/5 [==============================] - 0s 5ms/step - loss: 0.4704 - accuracy: 0.8129
Epoch 3/10
5/5 [==============================] - 0s 5ms/step - loss: 0.3319 - accuracy: 0.9137
```

Epoch 4/10

5/5 [==============================] - 0s 6ms/step - loss: 0.2565 - accuracy: 0.9353

Epoch 5/10

5/5 [==============================] - 0s 7ms/step - loss: 0.2044 - accuracy: 0.9424

Epoch 6/10

5/5 [==============================] - 0s 6ms/step - loss: 0.1634 - accuracy: 0.9640

Epoch 7/10

5/5 [==============================] - 0s 6ms/step - loss: 0.1312 - accuracy: 0.9856

Epoch 8/10

5/5 [==============================] - 0s 7ms/step - loss: 0.1068 - accuracy: 0.9856

Epoch 9/10

5/5 [==============================] - 0s 6ms/step - loss: 0.0857 - accuracy: 0.9928

Epoch 10/10

5/5 [==============================] - 0s 7ms/step - loss: 0.0698 - accuracy: 0.9928

2/2 [==============================] - 0s 8ms/step - loss: 0.2911 - accuracy: 0.8571

Test Loss: 0.2910865247249603

Test Accuracy: 0.8571428656578064

2/2 [==============================] - 0s 4ms/step

Accuracy: 0.8571428571428571

Precision: 0.8947368421052632

Recall: 0.85

F1-Score: 0.8717948717948718

Classification Report:

	precision	recall	f1-score	support
0	0.81	0.87	0.84	15
1	0.89	0.85	0.87	20
accuracy			0.86	35

```
   macro avg       0.85      0.86      0.86        35
weighted avg       0.86      0.86      0.86        35
```

In [65]:

```
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.optimizers import Adam
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
classification_report

# Define the number of features in your dataset
input_dim = X_train.shape[1]

# Create a neural network model
model = Sequential()
model.add(Dense(128, activation='relu', input_dim=input_dim))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.5))  # Add dropout before the output layer
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=20, batch_size=32)

# Evaluate the model on the test dataset
y_pred = model.predict(X_test)
```

```
binary_predictions = (y_pred > 0.5).astype(int)

# Calculate metrics
accuracy = accuracy_score(y_test, binary_predictions)
precision = precision_score(y_test, binary_predictions)
recall = recall_score(y_test, binary_predictions)
f1 = f1_score(y_test, binary_predictions)

# Print metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-Score:", f1)

# Generate a classification report
report = classification_report(y_test, binary_predictions)
print("Classification Report:\n", report)
Epoch 1/20
5/5 [==============================] - 1s 6ms/step - loss: 0.6705 - accuracy: 0.5755
Epoch 2/20
5/5 [==============================] - 0s 8ms/step - loss: 0.5160 - accuracy: 0.7554
Epoch 3/20
5/5 [==============================] - 0s 8ms/step - loss: 0.3782 - accuracy: 0.9065
Epoch 4/20
5/5 [==============================] - 0s 9ms/step - loss: 0.3278 - accuracy: 0.9137
Epoch 5/20
5/5 [==============================] - 0s 6ms/step - loss: 0.2688 - accuracy: 0.9281
Epoch 6/20
5/5 [==============================] - 0s 9ms/step - loss: 0.2351 - accuracy: 0.9712
```

Epoch 7/20

5/5 [==============================] - 0s 9ms/step - loss: 0.1729 - accuracy: 0.9712

Epoch 8/20

5/5 [==============================] - 0s 9ms/step - loss: 0.1517 - accuracy: 0.9856

Epoch 9/20

5/5 [==============================] - 0s 8ms/step - loss: 0.1030 - accuracy: 0.9856

Epoch 10/20

5/5 [==============================] - 0s 9ms/step - loss: 0.0842 - accuracy: 0.9928

Epoch 11/20

5/5 [==============================] - 0s 9ms/step - loss: 0.0880 - accuracy: 0.9784

Epoch 12/20

5/5 [==============================] - 0s 7ms/step - loss: 0.0756 - accuracy: 0.9784

Epoch 13/20

5/5 [==============================] - 0s 8ms/step - loss: 0.0514 - accuracy: 1.0000

Epoch 14/20

5/5 [==============================] - 0s 9ms/step - loss: 0.0357 - accuracy: 0.9928

Epoch 15/20

5/5 [==============================] - 0s 8ms/step - loss: 0.0336 - accuracy: 1.0000

Epoch 16/20

5/5 [==============================] - 0s 9ms/step - loss: 0.0395 - accuracy: 1.0000

Epoch 17/20

5/5 [==============================] - 0s 8ms/step - loss: 0.0314 - accuracy: 1.0000

Epoch 18/20

5/5 [==============================] - 0s 6ms/step - loss: 0.0242 - accuracy: 1.0000

Epoch 19/20

5/5 [==============================] - 0s 7ms/step - loss: 0.0212 - accuracy: 1.0000

Epoch 20/20

5/5 [==============================] - 0s 8ms/step - loss: 0.0160 - accuracy: 1.0000

2/2 [==============================] - 0s 5ms/step

Accuracy: 0.8571428571428571

Precision: 0.8947368421052632

Recall: 0.85

F1-Score: 0.8717948717948718

Classification Report:

	precision	recall	f1-score	support
0	0.81	0.87	0.84	15
1	0.89	0.85	0.87	20
accuracy			0.86	35
macro avg	0.85	0.86	0.86	35
weighted avg	0.86	0.86	0.86	35

In [ ]:

#Support Vector Machine

In [66]:

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Create an SVC classifier
clf = SVC()

# Train the classifier
clf.fit(X_train, y_train)

# Make predictions on the test data
y_pred = clf.predict(X_test)
```

```
# Calculate performance metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Store the results in a dictionary
results = {
    'Classifier': 'SVC',
    'Accuracy (%)': accuracy * 100,
    'Precision (%)': precision * 100,
    'Recall (%)': recall * 100,
    'F1-Score (%)': f1 * 100
}

print(results)
```

```
{'Classifier': 'SVC', 'Accuracy (%)': 85.71428571428571, 'Precision (%)': 85.94924812030075, 'Recall (%)': 85.71428571428571, 'F1-Score (%)': 85.76155027767932}
```

In [ ]:

In [ ]:

In [67]:

```
import tensorflow as tf
from tensorflow import keras
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Define a simple neural network model
model = keras.Sequential([
```

```
    keras.layers.Input(shape=(X_train.shape[1],)),  # Input layer
    keras.layers.Dense(128, activation='relu'),     # Hidden layer with 128 units and ReLU activation
    keras.layers.Dense(64, activation='relu'),      # Hidden layer with 64 units and ReLU activation
    keras.layers.Dense(1, activation='sigmoid')     # Output layer with 1 unit and sigmoid activation
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32, verbose=2)

# Make predictions on the test data
y_pred = model.predict(X_test)
y_pred_binary = (y_pred > 0.5).astype(int)

# Calculate performance metrics
accuracy = accuracy_score(y_test, y_pred_binary)
precision = precision_score(y_test, y_pred_binary, average='weighted')
recall = recall_score(y_test, y_pred_binary, average='weighted')
f1 = f1_score(y_test, y_pred_binary, average='weighted')

# Display the results
print("Neural Network Metrics:")
print(f"Accuracy: {accuracy * 100:.2f}%")
print(f"Precision: {precision * 100:.2f}%")
print(f"Recall: {recall * 100:.2f}%")
print(f"F1-Score: {f1 * 100:.2f}%")
Epoch 1/10
```

5/5 - 1s - loss: 0.6568 - accuracy: 0.6403 - 1s/epoch - 272ms/step

Epoch 2/10

5/5 - 0s - loss: 0.3286 - accuracy: 0.8993 - 37ms/epoch - 7ms/step

Epoch 3/10

5/5 - 0s - loss: 0.2104 - accuracy: 0.9712 - 46ms/epoch - 9ms/step

Epoch 4/10

5/5 - 0s - loss: 0.1376 - accuracy: 0.9856 - 38ms/epoch - 8ms/step

Epoch 5/10

5/5 - 0s - loss: 0.0916 - accuracy: 0.9928 - 43ms/epoch - 9ms/step

Epoch 6/10

5/5 - 0s - loss: 0.0641 - accuracy: 1.0000 - 42ms/epoch - 8ms/step

Epoch 7/10

5/5 - 0s - loss: 0.0462 - accuracy: 1.0000 - 41ms/epoch - 8ms/step

Epoch 8/10

5/5 - 0s - loss: 0.0329 - accuracy: 1.0000 - 41ms/epoch - 8ms/step

Epoch 9/10

5/5 - 0s - loss: 0.0236 - accuracy: 1.0000 - 42ms/epoch - 8ms/step

Epoch 10/10

5/5 - 0s - loss: 0.0181 - accuracy: 1.0000 - 45ms/epoch - 9ms/step

2/2 [==============================] - 0s 4ms/step

Neural Network Metrics:

Accuracy: 88.57%

Precision: 89.26%

Recall: 88.57%

F1-Score: 88.63%

In [70]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```
# Define the parameter grid for Logistic Regression
param_grid = {
    'C': [0.1, 1.0, 10.0],
    'penalty': ['l1', 'l2'],
    'max_iter': [100, 500, 1000],
}
# Create a Logistic Regression classifier
logreg_classifier = LogisticRegression(random_state=42)

# Use GridSearchCV for hyperparameter tuning
grid_search = GridSearchCV(estimator=logreg_classifier, param_grid=param_grid, cv=5,
scoring='accuracy')
grid_search.fit(X_train_pca, y_train)

# Get the best parameters and best model
best_params = grid_search.best_params_
best_logreg_model = grid_search.best_estimator_

# Evaluate the best model on the test set
y_pred_best = best_logreg_model.predict(X_test_pca)
# Calculate metrics
accuracy_best = accuracy_score(y_test, y_pred_best)
precision_best = precision_score(y_test, y_pred_best)
recall_best = recall_score(y_test, y_pred_best)
f1_best = f1_score(y_test, y_pred_best)
# Print metrics
print("Best Logistic Regression Metrics:")
print("Best Parameters:", best_params)
```

```
print("Accuracy:", accuracy_best)
print("Precision:", precision_best)
print("Recall:", recall_best)
print("F1-Score:", f1_best)
```

```
Best Logistic Regression Metrics:
Best Parameters: {'C': 0.1, 'max_iter': 100, 'penalty': 'l2'}
Accuracy: 0.8857142857142857
Precision: 0.9444444444444444
Recall: 0.85
F1-Score: 0.8947368421052632
```

In [29]:

```
# Import necessary libraries
from lightgbm import LGBMClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
# Create an LGBMClassifier model
clf = LGBMClassifier(random_state=42)
# Train the model on the training data
clf.fit(X_train, y_train)
# Make predictions on the test data
y_pred = clf.predict(X_test)
# Calculate model metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
print("LGBMClassifier Metrics:")
print(f"Accuracy: {accuracy * 100:.2f}%")
print(f"Precision: {precision * 100:.2f}%")
print(f"Recall: {recall * 100:.2f}%")
```

```
print(f"F1-Score: {f1 * 100:.2f}%")
```

```
C:\Users\SOMNATH\AppData\Roaming\Python\Python39\site-packages\sklearn\preprocessing\_label.py:99: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\SOMNATH\AppData\Roaming\Python\Python39\site-packages\sklearn\preprocessing\_label.py:134: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, dtype=self.classes_.dtype, warn=True)
LGBMClassifier Metrics:
Accuracy: 91.43%
Precision: 92.86%
Recall: 91.43%
F1-Score: 91.47%
```

In [30]:

```
# Import necessary libraries
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
# Create an ExtraTreesClassifier model
clf = ExtraTreesClassifier(random_state=42)
# Train the model on the training data
clf.fit(X_train, y_train)
# Make predictions on the test data
y_pred = clf.predict(X_test)
# Calculate model metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
print("ExtraTreesClassifier Metrics:")
```

```
print(f"Accuracy: {accuracy * 100:.2f}%")

print(f"Precision: {precision * 100:.2f}%")

print(f"Recall: {recall * 100:.2f}%")

print(f"F1-Score: {f1 * 100:.2f}%")
```

```
C:\Users\SOMNATH\AppData\Local\Temp\ipykernel_13500\4262555106.py:9: DataConversionWarning:
A column-vector y was passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().

  clf.fit(X_train, y_train)

ExtraTreesClassifier Metrics:

Accuracy: 91.43%

Precision: 91.64%

Recall: 91.43%

F1-Score: 91.46%
```

In [ ]:

In [27]:

```
import matplotlib.pyplot as plt

# Model names and corresponding metrics

model_names = ['Perceptron', 'XGBClassifier', 'AdaBoostClassifier', 'ExtraTreesClassifier',

            'RandomForestClassifier', 'LGBMClassifier', 'SVC', 'LogisticRegression']

Accuracy = [88.57, 91.43, 85.71, 91.43, 88.57, 91.43, 85.71, 88.56]  # Replace with actual accuracy values

Precision = [89.26, 92.86, 85.95, 91.64, 94.44, 92.86, 85.95, 94.44]  # Replace with actual precision values

Recall = [88.57, 91.43, 85.71, 91.43, 85.00, 91.43, 85.71, 85]  # Replace with actual recall values

F1_Score = [88.63, 91.47, 85.76, 91.46, 89.46, 91.47, 85.76, 89.46]  # Replace with actual F1-score values

#roc_auc = [0.92, 0.95, 0.88, ... ]  # Replace with actual AUC-ROC scores

# Create a line chart to visualize trends across metrics

metrics = ['Accuracy', 'Precision', 'Recall', 'F1_Score']

plt.figure(figsize=(12, 6))


for i, metric in enumerate(metrics):

   plt.plot(model_names, eval(metric), label=metric, marker='o')
```

```
plt.xlabel('Model')
plt.ylabel('Score')
plt.title('Model Performance Across Metrics')
plt.legend()
plt.grid(True)
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```