ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG QUỐC TẾ
VNU-INTERNATIONAL SCHOOL

NGUYEN MINH TUAN

APPLICATION OF AI AND RAMAN SPECTROSCOPY FOR NON-INVASIVE DIABETES DIAGNOSIS

Field: Informatics and Computer Engineering

Code: 8480111.01QTD

Hanoi, 2025

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG QUỐC TẾ
VNU-INTERNATIONAL SCHOOL

NGUYEN MINH TUAN

APPLICATION OF AI AND RAMAN SPECTROSCOPY FOR NON-INVASIVE DIABETES DIAGNOSIS

Field: Informatics and Computer Engineering

Code: 8480111.01QTD

Supervisor: Assoc. Prof. Dr. Nguyen Thanh Tung

Hanoi, 2025

CERTIFICATE OF ORIGINALITY

I, the undersigned, hereby certify my authority of the study project report entitled **Application of AI and Raman Spectroscopy for non-invasive diabetes diagnosis** submitted in partial fulfillment of the requirements for the degree of Master Informatics and Computer Engineering. Except where the reference is indicated, no other person's work has been used without due acknowledgement in the text of the thesis.

Hanoi, June 22, 2025

Nguyen Minh Tuan

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to everyone who has supported me throughout the journey of my graduation project. This project, titled "Application of AI and Raman Spectroscopy for Non-Invasive Diabetes Diagnosis," would not have been possible without the guidance, support, and encouragement of several individuals and institutions.

First and foremost, I would like to thank my advisor, [Advisor's Name], for their invaluable guidance, insightful feedback, and unwavering support throughout the project. Their expertise and dedication have been instrumental in shaping the direction and success of this work.

I am also grateful to the faculty members and staff of International School, Vietnam National University for providing the necessary resources and facilities to conduct my research. Special gratitude to Mr.Ngo Quang Tri and Mr.Le Anh Duc for helping me throughout the experiments.

I would like to extend my appreciation to my peers and colleagues who have provided their support and encouragement, making this journey more enjoyable and collaborative. Their constructive discussions and feedback have been invaluable.

Lastly, I am profoundly grateful to my family and friends for their unwavering support, patience, and understanding throughout this journey. Their love and encouragement have been a constant source of motivation.

Thank you all for your contributions and support.

ABSTRACT

This research explores the innovative application of Artificial Intelligence (AI) and Raman Spectroscopy for non-invasive diabetes diagnosis. Traditional methods of diagnosing diabetes often require invasive procedures that can be uncomfortable and inconvenient for patients. This study addresses the gap in previous research by integrating AI algorithms with Raman Spectroscopy to develop a reliable, non-invasive diagnostic tool for diabetes. The primary aim of this research is to create an efficient diagnostic method that simplifies the detection process and improves patient experience.

To achieve this, I employed machine learning models to analyze biochemical changes detected by Raman Spectroscopy, enabling the identification of specific diabetes biomarkers. The methodology involved a comprehensive literature review, selection of suitable AI algorithms, acquisition of Raman Spectroscopy equipment, and meticulous data collection. Preliminary testing was conducted to evaluate the effectiveness of the proposed diagnostic tool.

The findings of this research indicate promising potential in detecting diabetes biomarkers non-invasively. The integration of AI and Raman Spectroscopy demonstrated accuracy in identifying diabetic conditions, paving the way for a patient-friendly diagnostic alternative. These results are significant as they highlight the feasibility of non-invasive diabetes diagnosis, which could revolutionize how diabetes is detected and managed, ultimately enhancing patient care and reducing the burden of invasive procedures.

TABLE OF CONTENTS

LIST OF TABLES

TABLE OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Rationale

Diabetes is a chronic condition that occurs when the pancreas produces insufficient insulin a hormone that regulates blood glucose or when the body cannot effectively use the insulin it generates. Globally, diabetes is acknowledged as one of the four priority non-communicable diseases (NCDs). Over recent decades, its prevalence and incidence rates have risen significantly [1].

Early detection of diabetes can be achieved through affordable blood sugar monitoring. Currently, glucometer devices provide a widely accessible means of invasive glucose testing by analyzing blood samples in vitro. These devices are user-friendly and can be employed in both hospital and home settings. Traditional glucometers use the electrochemical approach [2], requiring a specified amount of blood via a finger prick or subcutaneous lancet. However, many individuals with diabetes find this frequent blood sampling uncomfortable and inconvenient, necessitating alternative, patient-friendly solutions.

Non-invasive blood glucose monitoring methods offer a promising, cost-effective, and innovative alternative for diabetes diagnosis. Numerous studies on these techniques have shown encouraging results in identifying diabetes [3–7]. Many of these approaches leverage optical sensors to measure a person's glucose concentration without invasive procedures. Significantly, such methods avoid causing any harm or injury to human tissues, making them a patient-friendly diagnostic solution.

1.2 Aim and Objectives of the Study

In the realm of artificial intelligence, non-invasive approaches play a crucial role by enabling the generation of intermediate data for distillation and correlation analysis. This data serves as the foundation for diabetic classification, providing a less invasive alternative to traditional diagnostic methods. Briganti et al.[8] have highlighted the transformative potential of AI-driven medical technologies, emphasizing their emergence as indispensable therapeutic solutions. These advancements underscore the application of AI techniques to preventive diagnostics, paving the way for early intervention and improved patient outcomes.

The primary aim of this study is to develop a non-invasive diagnostic method for diabetes by integrating AI algorithms with Raman Spectroscopy. The specific objectives of the study are:

- To review existing literature on the use of AI and Raman Spectroscopy in medical diagnostics.
- To select and implement appropriate AI algorithms for analyzing Raman Spectroscopy data.
- To collect and preprocess data using Raman Spectroscopy.
- To develop and validate a diagnostic model for diabetes using the integrated AI and Raman Spectroscopy approach.
- To evaluate the effectiveness and accuracy of the developed diagnostic tool.

1.3 Research Questions

The study seeks to answer the following research questions:

1. How can AI be integrated with Raman Spectroscopy to detect diabetes biomarkers non-invasively?
2. What are the key advantages and challenges of using this integrated approach for diabetes diagnosis?
3. How effective is the developed diagnostic tool in identifying diabetic conditions compared to traditional methods?

1.4 Methods of the Study

The study employs a combination of literature review, experimental data collection, and algorithm development. The methodology includes:

- Conducting a comprehensive literature review to identify relevant studies and technologies.
- Selecting suitable AI algorithms for data analysis.
- Acquiring Raman Spectroscopy equipment and collecting data from biological samples.
- Preprocessing the collected data to ensure its quality and suitability for analysis.

- Developing and validating the diagnostic model using machine learning techniques.
- Analyzing the results to assess the accuracy and effectiveness of the diagnostic tool.

1.5 Scope of the Study

This study focuses on the integration of AI and Raman Spectroscopy for non-invasive diabetes diagnosis. It includes:

- A detailed review of relevant literature and technologies.
- Experimental data collection and analysis using Raman Spectroscopy.
- Development and validation of an AI-based diagnostic model.
- Evaluation of the developed diagnostic tool's effectiveness.

1.6 Significance of the Study

The significance of this study lies in its potential to revolutionize diabetes diagnosis by providing a non-invasive, accurate, and efficient diagnostic tool. The findings of this research could lead to improved patient care, reduced discomfort associated with traditional diagnostic methods, and a more accessible approach to diabetes detection. Additionally, this study contributes to the growing body of knowledge on the application of AI and Raman Spectroscopy in medical diagnostics.

1.7 Structure of the Study

This thesis is structured as follows:

- **Chapter 1: Introduction** - Provides an overview of the research, including its rationale, aims, objectives, research questions, methods, scope, significance, and structure.
- **Chapter 2: Literature Review** - Summarizes existing research on AI and Raman Spectroscopy in medical diagnostics, highlighting gaps and opportunities for further study.
- **Chapter 3: Methodology** - Describes the research design, data collection methods, and analytical techniques used in the study.

- **Chapter 4: Results** - Presents the findings of the study, including data analysis and interpretation.
- **Chapter 5: Discussion** - Discusses the implications of the findings, compares them with existing literature, and suggests potential improvements.
- **Chapter 6: Conclusion** - Summarizes the key points of the research, reflects on the overall progress and achievements, and suggests directions for future research.

CHAPTER 2: LITERATURE REVIEW

2.1 Diabetes

2.1.1 Introduction of Diabetes

Diabetes is a chronic condition that occurs when the body is unable to properly use the insulin hormone produced by the pancreas or doesn't produce enough to regulate blood sugar levels. Glucose (C6H12O6), a simple sugar and the most common monosaccharide, plays a key role in hyperglycemia and diabetes. Through photosynthesis, plants and most algae create glucose from water and CO2 using sunlight. Glucose is essential for energy metabolism in all living organisms, used in cellular respiration to produce energy. In plants, it is stored as cellulose and starch (mainly branched and single-chain amylose), while animals store glucose as glycogen. D-glucose is naturally occurring, whereas L-glucose is artificially produced in limited amounts and is less significant.

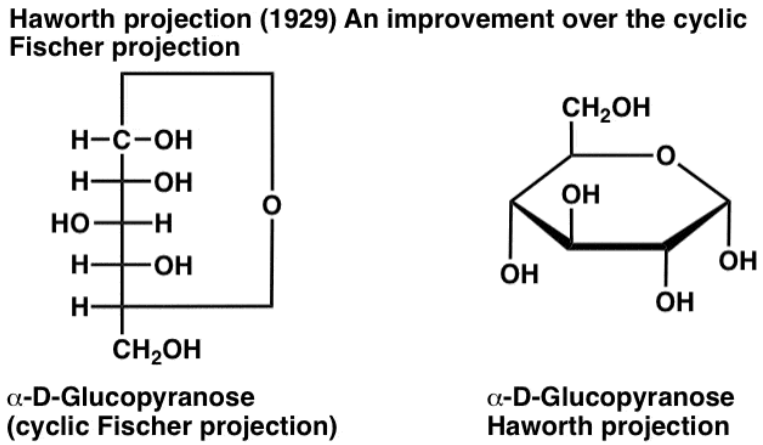

Figure 1: Haworth projections(James Ashenhurst,2024)

The liver plays a vital role in maintaining the body's glucose levels by serving as a glucose reservoir. It responds to the body's needs by synthesizing and storing glucose, regulated by key hormones such as insulin and glucagon. After meals, when insulin levels are high and glucagon levels are low, the body stores glucose as glycogen. During periods without food intake, such as overnight or between meals, the liver converts glycogen back into glucose through a process known as glycogenolysis. Additionally, the liver can produce essential glucose by synthesizing waste products, lipid byproducts, and amino acids through a process called gluconeogenesis. This multifaceted role ensures a steady supply of glucose, providing

energy to the body even during fasting periods, which is crucial for maintaining overall metabolic balance.

The liver plays a pivotal role in maintaining the body's glucose levels by acting as a reserve. It synthesizes and stores glucose in response to the body's needs, regulated by key hormones such as insulin and glucagon. After meals, when insulin levels are high and glucagon levels are low, the body stores glucose as glycogen. Conversely, during periods without food intake, such as at night or between meals, the liver converts glycogen into glucose through a process called glycogenolysis. Furthermore, the liver can produce essential glucose by synthesizing waste products, lipid byproducts, and amino acids through gluconeogenesis. This dual functionality ensures a steady supply of glucose, providing energy even during fasting periods and maintaining overall metabolic balance. [9]

2.1.2 Types of Diabetes

Diabetes can be categorized into three main types: type 1, type 2, and gestational diabetes. Most patients are diagnosed with either type 1 or type 2 diabetes.

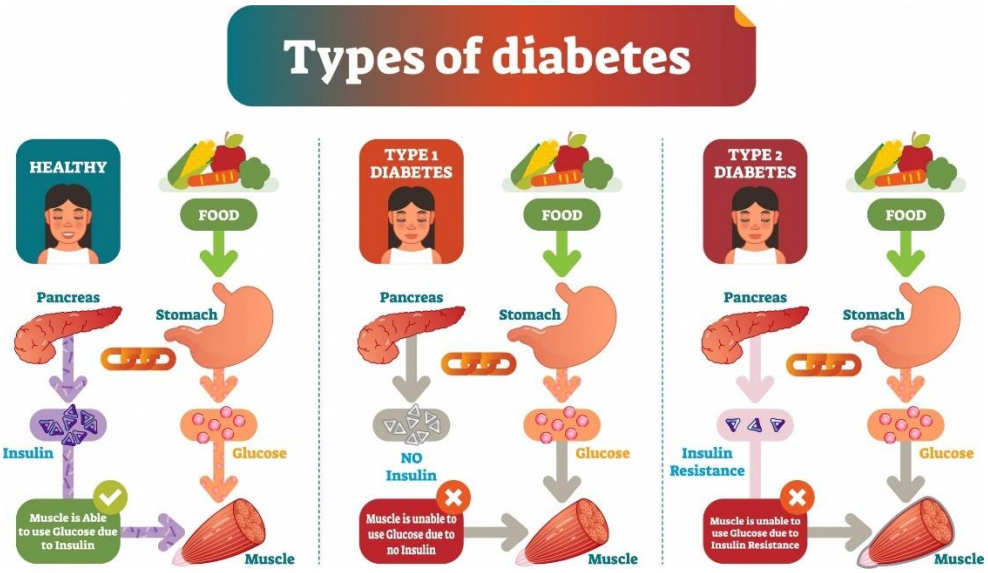

Figure 2: Types of Diabetes (North Dakota Department of Health and Human Services, 2024)

2.1.2.1 Type 1 Diabetes

Type 1 Diabetes is a chronic autoimmune condition where the immune system mistakenly attacks and destroys the insulin-producing beta cells in the pancreas. As

a result, the body produces little to no insulin, a hormone essential for regulating blood sugar levels. Without insulin, glucose cannot enter cells to provide energy, leading to elevated blood sugar levels. This form of diabetes often develops in childhood or adolescence but can occur at any age. Symptoms include frequent urination, excessive thirst, extreme hunger, weight loss, fatigue, and blurry vision. Individuals with type 1 diabetes require daily insulin therapy, either through injections or an insulin pump, to manage their blood sugar levels effectively. While the exact cause of type 1 diabetes is unknown, it is believed to result from a combination of genetic predisposition and environmental triggers, such as viral infections. It is less common than type 2 diabetes, but it requires careful management, including blood sugar monitoring, a healthy diet, and regular exercise, to prevent complications. [10]

2.1.2.2 Type 2 Diabetes

Type 2 Diabetes is a chronic metabolic disorder that occurs when the body becomes resistant to insulin or when the pancreas does not produce enough insulin to regulate blood sugar levels. Unlike type 1 diabetes, type 2 often develops gradually over time and is more common in adults, though it is increasingly being diagnosed in children and adolescents. The primary risk factors for type 2 diabetes include obesity, lack of physical activity, poor diet, age, family history, and certain ethnic backgrounds. Symptoms may include increased thirst, frequent urination, fatigue, blurred vision, and slow healing of wounds, though it can sometimes be asymptomatic in its early stages. Management of type 2 diabetes typically involves lifestyle modifications such as a healthy diet, regular exercise, and weight loss. In some cases, medication or insulin therapy may be required to maintain blood sugar levels. Early diagnosis and consistent management are crucial in preventing complications like cardiovascular disease, kidney damage, and nerve damage. [11]

2.1.2.3 Gestational Diabetes

Gestational Diabetes is a type of diabetes that develops during pregnancy in women who have not previously been diagnosed with diabetes. It occurs when the body cannot produce enough insulin to meet the increased demands of pregnancy, leading to elevated blood sugar levels. This condition typically arises in the second or third trimester and often resolves after delivery, but it requires careful management to ensure the health of both the mother and baby. Risk factors for gestational diabetes include a history of the condition in previous pregnancies, being overweight, a family

history of diabetes, or belonging to certain ethnic groups with higher susceptibility. Symptoms are often mild or absent, but in some cases, increased thirst, frequent urination, or fatigue may be observed. Management includes adopting a healthy diet, regular physical activity, and monitoring blood sugar levels. In some cases, insulin therapy or medication may be required. While gestational diabetes usually resolves after childbirth, women who experience it are at a higher risk of developing type 2 diabetes later in life, and regular follow-ups are recommended. [12]

2.1.2.4 Secondary or Other Specific Types of Diabetes

Secondary or Other Specific Types of Diabetes refer to forms of diabetes caused by underlying medical conditions, genetic factors, or external influences, rather than the typical mechanisms seen in type 1, type 2, or gestational diabetes. These forms are less common but are significant to understand for accurate diagnosis and management.

- **Genetic Defects of Beta Cell Function**: Known as monogenic diabetes, this group includes conditions like *Maturity-Onset Diabetes of the Young (MODY)*, caused by single-gene mutations that impair insulin production.
- **Pancreatic Disorders**: Diseases such as pancreatitis, cystic fibrosis, or pancreatic cancer can damage the pancreas, reducing insulin production and leading to diabetes.
- **Endocrine Disorders**: Conditions like *Cushing's syndrome* or *acromegaly* can cause elevated levels of hormones like cortisol or growth hormone, which counteract insulin's effects and result in diabetes.
- **Drug or Chemical-Induced Diabetes**: Certain medications, such as corticosteroids or antipsychotics, or exposure to toxins, can impair glucose metabolism and induce diabetes.
- **Infections and Other Conditions**: Rare infections or autoimmune diseases can trigger diabetes by affecting insulin production or action.
- **Genetic Syndromes**: Some syndromes, including *Down syndrome*, *Turner syndrome*, and *Klinefelter syndrome*, have a higher prevalence of diabetes.

Management of secondary diabetes typically involves addressing the root cause, such as treating the underlying condition or adjusting medications, alongside standard diabetes care.

2.2 Substance of diabetes mellitus

Diabetes manifests in various forms, depending on its underlying cause. Initially referred to as non–insulin-dependent diabetes, Type 2 Diabetes Mellitus (T2DM) accounts for 90–95% of diabetes cases.[13] In T2DM, individuals experience insulin resistance and relative insulin deficiency. Interestingly, these individuals are not commonly associated with insulin treatment initially.

The exact causes of this type of diabetes remain unclear, but it is unlikely to involve autoimmune destruction of β-cells (as seen in Type 1 Diabetes). Furthermore, none of the other known causes of diabetes apply to this specific type.

In the early 1980s, HbA1C was recommended as a diagnostic test. However, concerns about its availability and insufficient assay standardization hindered its widespread adoption [14]. It wasn't until 2009 that an international expert panel suggested including HbA1C in diagnostic criteria. Specifically, they recommended a threshold of 48 mmol/mol (6.5% DCCT) [15].

Both the American Diabetes Association (ADA) and the World Health Organization (WHO) have endorsed these criteria. As a result, many countries now consider it the gold standard for T2DM diagnostic testing (see Table 1).

	FPG (mg/dL)	HbA1C (% DCCT)
Normal	<100	<5.7%
Pre-diabetes	100-125	5.7% - 6.4%
Diabetes Mellitus	>= 126	>= 6.5%

Table 1: ADA diabetes diagnostic criteria in 2015

2.3 Traditional invasive methods for type 2 diabetes mellitus detection

2.3.1 Point of care Hemoglobin A1C Test (POCT):

Point-of-care Hemoglobin A1C (HbA1c) tests, commonly referred to as POCT, are diagnostic tools used to measure the average blood glucose levels over the past two to three months. These tests are crucial for the diagnosis and management of diabetes, providing immediate results during patient consultations.

Key Features and Benefits:

- Immediate Results: POCT devices provide rapid results, allowing healthcare providers to make timely therapeutic decisions during the same visit. This reduces the need for additional appointments and follow-ups.
- Convenience: These tests can be performed in various settings, including clinics, pharmacies, and even at home, making them accessible and convenient for patients.
- Improved Glycemic Control: By offering immediate feedback, POCT devices help in better monitoring and management of blood glucose levels, leading to improved glycemic control and reduced risk of complications.
- Standardization: Modern POCT devices are standardized and certified by programs like the National Glycohemoglobin Standardization Program (NGSP), ensuring accuracy and consistency in results. [16-17]

Clinical Applications:

- Diagnosis of Diabetes: POCT HbA1c tests are used to diagnose diabetes by measuring the percentage of glycated hemoglobin in the blood. An HbA1c level of 6.5% or higher indicates diabetes. [18]
- Monitoring Diabetes: These tests are also used to monitor the effectiveness of diabetes treatment plans, helping to adjust medications and lifestyle changes as needed.
- Screening: POCT HbA1c tests can be used for screening individuals at risk of developing diabetes, enabling early intervention and prevention strategies.

Considerations:

- Accuracy: While POCT devices are generally accurate, it is essential to follow proper testing procedures to ensure reliable results. Factors such as device calibration, sample handling, and operator training can impact accuracy. [16]
- Cost: The cost of POCT devices and test cartridges can vary, and it is important to consider the cost-effectiveness of these tests in different healthcare settings. [17]
- Limitations: POCT HbA1c tests may have limitations in certain clinical situations, such as in patients with hemoglobin variants or conditions affecting red blood cell turnover. [16]

Overall, point-of-care HbA1c tests are valuable tools in the management of diabetes, offering convenience, immediate results, and improved patient outcomes. By enabling timely therapeutic decisions and better glycemic control, these tests play a crucial role in diabetes care.

2.3.2 The fasting plasma glucose (FPG) test:

The fasting plasma glucose (FPG) test, also known as the fasting blood glucose test (FBG) or fasting blood sugar test, measures the levels of glucose (sugar) in the blood after a period of fasting, typically for at least 8 hours. This test is a simple, accurate, and inexpensive method used to screen for diabetes and assess problems with insulin functioning1.

Key Features and Benefits:

- Diagnosis of Diabetes: The FPG test is commonly used to diagnose diabetes and pre-diabetes. A fasting blood glucose level of 126 mg/dL (7.0 mmol/L) or higher on two separate occasions indicates diabetes.
- Screening: It is recommended as a screening test for individuals aged 35 or older and those with symptoms or risk factors for diabetes.
- Monitoring: The FPG test can be used to monitor blood glucose levels in individuals already diagnosed with diabetes, helping to evaluate the effectiveness of their management plan.

Procedure:

- Preparation: Patients are instructed to fast for at least 8 hours before the test, during which they can only drink water.
- Blood Sample: A blood sample is taken from the patient's arm, usually in the morning to ensure adequate fasting time.
- Results: The blood sample is analyzed to measure the glucose levels. Normal fasting blood glucose levels are below 100 mg/dL (5.5 mmol/L). Levels between 100 and 125 mg/dL (5.5 to 6.9 mmol/L) indicate impaired fasting glucose, a form of pre-diabetes.

Considerations:

- Accuracy: The FPG test is regarded as accurate and more sensitive than the A1C test, though it is not as sensitive as the oral glucose tolerance test (OGTT).
- Risks: As a standard blood draw, the FPG test is considered safe, with minimal risks such as bruising or infection at the puncture site.

- Limitations: The FPG test may not be suitable for individuals with certain medical conditions or those who cannot fast for extended periods.

Overall, the fasting plasma glucose test is a valuable tool in the diagnosis and management of diabetes, providing essential information about blood glucose levels and helping to guide treatment decisions. [19-20]

2.3.3 The oral glucose tolerance test (OGTT):

The oral glucose tolerance test (OGTT) is a diagnostic tool used to measure the body's ability to metabolize glucose, which helps in diagnosing diabetes and other conditions related to glucose metabolism. The test involves fasting overnight and then consuming a glucose-rich solution. Blood samples are taken at specific intervals to monitor how the body processes the glucose over time.

Key Features and Benefits:

- Diagnosis of Diabetes: The OGTT is considered the gold standard for diagnosing type 2 diabetes, gestational diabetes, and prediabetes. It can detect abnormalities in glucose metabolism that other tests might miss.
- Screening for Gestational Diabetes: The OGTT is commonly used during pregnancy to screen for gestational diabetes, typically between 24 and 28 weeks of gestation.
- Assessment of Glucose Tolerance: The test helps assess how well the body handles glucose, providing insights into insulin sensitivity and beta-cell function.

Procedure:

- Preparation: Patients are required to fast for at least 8 hours before the test. It is usually scheduled in the morning to ensure adequate fasting time.
- Glucose Solution: After the initial fasting blood sample is taken, the patient drinks a glucose solution containing 75 grams of glucose (for non-pregnant adults). For pregnant women, the solution may contain 50 or 100 grams of glucose, depending on the specific protocol.
- Blood Samples: Blood samples are taken at multiple intervals, typically at 0, 1, and 2 hours after consuming the glucose solution. These samples measure the blood glucose levels to see how the body processes the glucose.

Interpretation of Results:

- Normal Glucose Tolerance: Blood glucose levels return to normal within 2 hours.

- Impaired Glucose Tolerance (Prediabetes): Blood glucose levels are higher than normal but not high enough to be classified as diabetes.
- Diabetes: Blood glucose levels remain elevated, indicating a problem with glucose metabolism.

Considerations:

- Accuracy: The OGTT is highly accurate but requires proper preparation and adherence to the testing protocol to ensure reliable results.
- Risks: The test is generally safe, but some patients may experience nausea or dizziness from the glucose solution. There is also a small risk of bruising or infection at the blood draw site.
- Limitations: The OGTT may not be suitable for individuals with certain medical conditions or those who cannot fast for extended periods.

Overall, the oral glucose tolerance test is a valuable tool in diagnosing and managing diabetes and other glucose metabolism disorders. It provides comprehensive insights into how the body handles glucose, helping healthcare providers make informed decisions about treatment and management. [20-21]

2.3.4 Real Time Continuous Glucose Monitoring

Real-time Continuous Glucose Monitoring (CGM) is a cutting-edge technology that allows individuals, particularly those with diabetes, to continuously monitor their blood glucose levels throughout the day and night. This technology provides real-time data, enabling better management of blood sugar levels and more informed decision-making regarding diet, exercise, and medication.

Key Features and Benefits:

- Continuous Monitoring: CGM devices provide continuous glucose readings, typically every few minutes, allowing users to see trends and patterns in their glucose levels.
- Immediate Feedback: Real-time data helps users make immediate adjustments to their lifestyle or treatment plan, improving glycemic control and reducing the risk of complications.
- Alerts and Alarms: Many CGM devices come with customizable alerts and alarms that notify users when their glucose levels are too high or too low, enabling prompt action to prevent hyperglycemia or hypoglycemia.

- Data Integration: CGM systems often integrate with smartphones, insulin pumps, and other devices, providing a comprehensive view of glucose levels and facilitating better diabetes management.
- Reduced Finger Pricks: While occasional finger-prick tests may still be necessary, CGM significantly reduces the need for frequent blood glucose testing, making diabetes management less invasive and more convenient.

Method of operation:

A CGM system typically consists of three main components:

- Sensor: A small sensor is inserted under the skin, usually on the abdomen or arm, to measure glucose levels in the interstitial fluid.
- Transmitter: The sensor sends glucose data wirelessly to a transmitter, which then relays the information to a receiver or a compatible device, such as a smartphone or insulin pump.
- Receiver: The receiver displays the glucose readings in real-time, allowing users to monitor their levels continuously.

Clinical Applications:

- Diabetes Management: CGM is particularly beneficial for individuals with type 1 diabetes, type 2 diabetes, and gestational diabetes, helping them maintain optimal blood glucose levels and avoid complications.
- Personalized Treatment: By providing detailed glucose data, CGM enables healthcare providers to tailor treatment plans to individual needs, improving overall diabetes care.
- Research and Development: CGM technology is also used in clinical research to study glucose metabolism and develop new diabetes treatments.

Considerations:

- Accuracy: While CGM devices are generally accurate, they may have a slight lag compared to traditional blood glucose meters. Regular calibration and occasional finger-prick tests are recommended to ensure accuracy.
- Cost: CGM systems can be expensive, and not all insurance plans cover them. It's important to consider the cost and potential benefits when deciding whether to use a CGM device.

- Maintenance: Sensors need to be replaced regularly, typically every 7 to 14 days, depending on the device. Proper maintenance and adherence to the manufacturer's guidelines are essential for optimal performance.

Real-time Continuous Glucose Monitoring is a powerful tool that enhances diabetes management by providing continuous, real-time data on glucose levels. This technology empowers individuals to take control of their health, make informed decisions, and improve their quality of life. [22]

2.4 Artificial Intelligence

2.4.1 Definition and Concepts

Artificial Intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think and learn like humans. The basic concepts of AI include:

- Machine Learning (ML): A subset of AI that involves training algorithms to learn from and make predictions based on data. Machine learning models can improve their performance over time without being explicitly programmed [23].
- Neural Networks: Inspired by the human brain, neural networks consist of interconnected nodes (neurons) that process data and learn patterns. They are particularly effective in handling complex data and are the foundation of many deep learning models [24].
- Deep Learning: A specialized subset of machine learning that involves neural networks with multiple layers (deep neural networks). Deep learning models can automatically extract and learn features from raw data, making them highly effective for tasks like image and speech recognition.

2.4.2 AI in Medical Diagnostics

Artificial Intelligence (AI) is revolutionizing healthcare by enhancing diagnostic accuracy, enabling early disease detection, and improving patient outcomes. AI's ability to process vast amounts of data quickly and accurately is transforming how diseases are diagnosed and treated.

Several studies have successfully utilized AI for diagnosing diseases, including diabetes for instance, AI algorithms have been developed to autonomously screen for diabetic retinopathy from fundus photography, achieving sensitivity and specificity greater than 85% compared to human graders. Another study explored the use of AI

for automatic retinal screening, clinical diagnosis support, and patient self-management tools, which have been approved by the US Food and Drug Administration. [27]

The key findings from these studies include improved diagnostic accuracy, early detection of diseases, and personalized treatment plans. AI-driven diagnostics are democratizing healthcare by making early and accurate diagnoses more accessible, especially in regions with limited access to specialized medical professionals. Additionally, AI can reduce variability in diagnostic outcomes by providing consistent, data-driven insights. [28]

Recent research indicates that blood glucose monitoring can now be achieved through non-invasive methods [30]. The invasive nature of widely used commercial glucose meters has led to infections and discomfort for many individuals with diabetes. Consequently, there is growing interest in non-invasive blood glucose monitoring among researchers worldwide. However, non-invasive blood glucose measurement technology faces challenges related to detection concepts. Fields such as chemistry, biology, optics, electromagnetic waves, and computer science stand to benefit from this innovative approach. A comprehensive discussion of the advantages and limitations of non-invasive versus invasive technologies, including insights into electrochemistry and optics for non-invasive methods, can be found in [31]. Transdermal biosensors and wearable technologies are enhancing the efficiency, affordability, resilience, and competitiveness of non-invasive blood glucose monitoring in the market.

Over the past two decades, numerous studies have explored non-invasive blood glucose testing methods. Researchers have investigated various optical techniques for noninvasive measurements, including near-infrared (NIR) [30] spectroscopy, photoacoustic imaging, Raman spectroscopy [32], polarized optical rotation, and optical coherence tomography [33]. In Raman spectroscopy, a transilluminated laser beam serves as the excitation light source, projected onto a specific point on the subject's body (either in vivo or in vitro) to monitor glucose levels.

Raman spectroscopy, a non-destructive and label-free fingerprinting technique, is increasingly enhancing biomedical diagnostics both in vivo and in vitro. Its benefits include relatively short acquisition time, non-invasiveness, and the ability to provide biochemical molecular information. By selecting a source within the red area of the spectrum or near-infrared (NIR), both the source and the signal can fall within the

optical window for tissue transparency, allowing for penetration depths of up to millimeters into human tissue. Notably, Shao et al [34] demonstrated that Raman spectra obtained using a diode laser operating at 785 nm can be used for non-invasive quantitative blood glucose analysis in living animals. The relationship between Raman intensity and blood glucose concentration was discovered, with an R-squared value of 0.91 and a Mean Absolute Error rate of 5.7%.

In the context of non-invasive blood glucose monitoring, Habibullah et al. proposed an SVM classification technique that achieved an 85% accuracy (A, 2022). They utilized a Gaussian filter and histogram-based feature extraction for picture database analysis. Additionally, Villa-Manríquez et al. demonstrated that Raman spectroscopy, when compared to NIR, can achieve over 80% accuracy in blood glucose determination using the same classification model (PCA-SVM) [35]. This underscores the clear advantage and potential of Raman spectroscopy for assessing blood glucose concentration.

Shokrekhodaei et al. have also suggested a number of additional techniques. According to [36], the author's SVM model for classifying the glucose range based on 21 discrete concentration value classes had a mean F1-score of 99%.

2.4.3 Applications in Diagnostics

AI algorithms have been applied in medical diagnostics in various ways, focusing on pattern recognition and predictive analytics:

- Pattern Recognition: AI algorithms like CNNs are used to analyze medical images (e.g., X-rays, MRIs, CT scans) to identify abnormalities and diagnose conditions such as cancer, pneumonia, and retinal diseases. These models can detect subtle patterns that may be missed by human experts, leading to earlier and more accurate diagnoses.
- Predictive Analytics: Machine learning models, including SVMs and random forests, are used to predict the likelihood of diseases based on patient data (e.g., electronic health records, genetic information). These models can identify risk factors and predict disease progression, enabling personalized treatment plans and proactive interventions.
- Automated Diagnostics: AI-driven tools are used to provide real-time diagnostic support for clinicians. For example, AI algorithms can analyze blood test results, detect diabetic retinopathy from retinal images, and

classify skin lesions as benign or malignant. These tools enhance the efficiency and accuracy of clinical decision-making.

By integrating AI algorithms with medical diagnostics, healthcare professionals can improve diagnostic accuracy, reduce variability in outcomes, and provide personalized care to patients. Predictive analytics is the main goal of the project.

The future of AI in healthcare is poised to revolutionize the industry by dramatically improving patient care, diagnostics, and operational efficiency. With its ability to analyze vast datasets at incredible speeds, AI is enabling predictive analytics, personalized treatments, and early disease detection on an unprecedented scale. AI-driven advancements in genomics are accelerating the shift toward precision medicine, tailoring treatments to individual genetic profiles and increasing the effectiveness of therapies. This technology is paving the way for breakthroughs in managing complex diseases such as cancer, diabetes, and genetic disorders. [37]

Medical imaging, one of AI's most impactful applications, is significantly enhancing diagnostic accuracy. AI algorithms can detect anomalies in X-rays, MRIs, and CT scans with greater precision, aiding radiologists in diagnosing conditions like tumors, fractures, and cardiovascular diseases. These tools not only improve accuracy but also reduce diagnostic errors, ensuring better patient outcomes. AI is also transforming clinical workflows by automating time-consuming tasks such as medical record documentation and scheduling, thereby freeing healthcare professionals to focus more on direct patient care. [37]

Moreover, predictive models powered by AI are reshaping how healthcare systems manage resources. These models can anticipate disease outbreaks, detect early warning signs of conditions like heart attacks or sepsis, and identify patients at risk, allowing timely interventions. Beyond clinical applications, AI is improving operational efficiency by streamlining processes, reducing administrative overhead, and optimizing supply chain logistics. [38]

As AI technologies evolve further, their integration into wearable devices and remote monitoring systems is expected to transform preventive care. Patients will have more access to real-time health insights, empowering them to make informed decisions and enabling physicians to intervene earlier. The combination of AI and robotics also promises advances in surgical precision, rehabilitation, and elderly care.

In the coming years, AI's role in healthcare will expand, driving a shift from reactive treatment to proactive, predictive, and personalized care. This transformation

has the potential to improve patient outcomes, reduce healthcare costs, and make quality care more accessible worldwide, ultimately redefining the way healthcare is delivered. [39]

2.5 AI Algorithms

Several AI algorithms are relevant to this study, including:

2.5.1 Support Vector Machines (SVM)

Support Vector Machine (SVM) is one of the most widely used supervised machine learning algorithms for classification and regression tasks, renowned for its ability to handle high-dimensional data efficiently. The primary function of SVM is to identify the optimal decision boundary that separates different classes within a dataset. In classification problems, SVM constructs a hyperplane that divides data points into distinct categories, ensuring that the separation margin between them is maximized. A wider margin improves generalization, allowing the model to classify new, unseen data points more accurately while reducing the risk of misclassification errors. The critical data points closest to the hyperplane, known as support vectors, are instrumental in influencing the boundary's position, making them essential for decision-making in the SVM framework.

For datasets that are non-linearly separable, SVM employs a kernel trick to transform input data into a higher-dimensional space, where linear separation becomes feasible. Various kernel functions serve different purposes: the linear kernel is used when data can be separated with a straight line, while the polynomial kernel and radial basis function (RBF) kernel help model more complex, curved decision boundaries. The sigmoid kernel is often associated with neural network-inspired classification problems, adding flexibility in cases where linear or polynomial approaches fail. Through these transformations, SVM gains the ability to classify highly non-linear datasets with improved accuracy, making it an invaluable tool in pattern recognition, image analysis, and biomedical signal processing.

SVM is particularly effective in handling both linear and nonlinear problems, thanks to its adaptability in high-dimensional feature spaces. In nonlinear datasets, the kernel trick allows SVM to project input features into a higher-dimensional space, where a linear hyperplane can separate previously inseparable classes. The most commonly used kernels include the linear kernel, which performs well on straightforward classification tasks with minimal curvature in decision boundaries; the polynomial kernel, which introduces curvature for more complex data

relationships; and the RBF kernel, which is widely regarded as one of the most flexible and powerful options for non-linear classification. However, selecting the appropriate kernel for a given dataset remains a challenge, as there is no universal approach to kernel selection. Instead, researchers must rely on experimental evaluations and dataset-specific tuning to identify the best-performing kernel, as noted by Ho [25].

Support Vector Machine (SVM) has become a widely researched machine learning model for biomedical applications of Raman spectroscopy, showing remarkable effectiveness in disease diagnostics, molecular fingerprinting, and spectral classification. The ability of SVM to process high-dimensional spectral data has made it a valuable tool for non-invasive medical analysis, allowing researchers to detect subtle biochemical changes indicative of disease.

Studies have demonstrated SVM's proficiency in distinguishing different pathological conditions based on Raman spectral variations, enabling the classification of diseases such as cancer, autoimmune disorders, and bacterial infections. Its ability to construct optimal hyperplanes ensures precise separation between classes, minimizing misclassification errors. The kernel trick, which transforms spectral features into higher-dimensional spaces, further enhances SVM's ability to handle non-linearly separable biomedical data.

Moreover, SVM's integration with preprocessing techniques such as baseline correction, intensity modulation, and feature extraction has significantly improved the accuracy of biomedical spectral analysis. By refining raw spectral signals, these techniques optimize SVM's ability to differentiate molecular structures, reinforcing its role in AI-driven Raman spectroscopy for medical diagnostics.

SVM has been extensively applied in biomedical applications, particularly in Raman spectroscopy, where high-dimensional spectral data must be analyzed efficiently. Various studies have explored SVM's role in non-invasive medical analysis, emphasizing its ability to classify diseases, identify molecular fingerprints, and analyze spectral variations. A study published in Nature investigated the use of SVM for diagnosing primary Sjögren's syndrome (pSS) using Raman spectroscopy of blood samples. The researchers applied Principal Component Analysis (PCA) for feature selection and Particle Swarm Optimization (PSO) to fine-tune SVM parameters, achieving an impressive classification accuracy of 94.44%. These

findings confirm that SVM is highly effective in distinguishing autoimmune disorders, reinforcing its role in spectral-based diagnostics.[49]

A study published in *BMC Cancer* explored the use of serum-based Raman spectroscopy for lung cancer screening, highlighting the effectiveness of Support Vector Machine (SVM) in distinguishing cancerous versus non-cancerous samples. By leveraging advanced spectral classification techniques, the researchers achieved 91.67% sensitivity and 92.22% specificity, confirming SVM's ability to detect subtle biochemical variations that differentiate malignant and benign cases. These findings reinforce the potential of Raman spectroscopy combined with AI-driven analysis as a promising tool for non-invasive cancer diagnostics. The study also emphasized the importance of spectral preprocessing techniques, such as baseline correction and noise filtering, which significantly improved the model's classification accuracy.[50]

Another study published in *Springer* investigated the application of Surface-Enhanced Raman Spectroscopy (SERS) for detecting trace-level analytes in biological samples, focusing on molecular fingerprinting and chemical specificity. SVM played a crucial role in enhancing spectral interpretation, allowing the model to discriminate minute spectral differences within complex environments. The researchers found that combining chemometric methods with machine learning models, such as SVM, resulted in higher sensitivity and improved accuracy in molecular detection. The study reinforced the importance of machine learning integration in spectral analysis, paving the way for more precise and reliable biomedical applications in disease diagnostics.[51]

These studies highlight SVM's strengths in biomedical diagnostics, proving its effectiveness in handling high-dimensional spectral data, improving classification accuracy, and enabling non-invasive disease detection. Despite its advantages, several challenges remain, including computational complexity, noise sensitivity, and hyperparameter tuning difficulties. Large spectral datasets require significant processing power, and selecting the optimal kernel function and hyperparameters is often an iterative, computationally intensive task. Noise sensitivity is another concern, as Raman spectroscopy signals are susceptible to background fluorescence interference, requiring robust baseline correction techniques for optimal classification performance.

In this research, SVM was implemented to analyze Raman spectral data, assessing classification accuracy before and after preprocessing. The results

confirmed that effective baseline correction techniques, such as PenPoly (Polynomial Regression) and IMP (Intensity Modulation Processing), significantly enhanced classification accuracy. By refining spectral signals before applying SVM, the study demonstrated that proper preprocessing leads to improved feature extraction, better class separability, and higher prediction accuracy. This reinforces SVM's capability to handle high-dimensional data efficiently, proving its relevance and effectiveness in AI-driven medical diagnostics.

Linear Kernel

The linear kernel is one of the simplest yet most widely used kernel functions in Support Vector Machines (SVM), particularly when working with linearly separable data. It operates by defining a straight-line decision boundary in the feature space, making it computationally efficient and easy to interpret. Unlike non-linear kernels such as polynomial or radial basis function (RBF), the linear kernel does not transform the data into higher-dimensional spaces but instead relies on the dot product between feature vectors to separate classes. Mathematically, it is expressed as:

$$K(x,y)=x.y$$

where (x) and (y) are feature vectors. This simple approach makes it highly effective for high-dimensional datasets, such as text classification, gene expression analysis, and spectral data processing, where the number of features is significantly larger than the number of samples.

One of the key advantages of using a linear kernel in SVM is its ability to prevent overfitting while maintaining generalization performance. Because it focuses on maximizing the margin between classes rather than creating complex decision boundaries, it ensures stable classification results in situations where data naturally follows a linear separation pattern. Additionally, the linear kernel is computationally less intensive, making it faster to train and deploy in large-scale applications. This efficiency makes it particularly suitable for scenarios where the dataset consists of thousands of features, such as Raman spectral analysis, where the model must distinguish chemical compositions based on intensity variations.

However, despite its advantages, the linear kernel has certain limitations. One of its biggest drawbacks is its inability to model non-linear relationships. If data points exhibit complex curved or clustered boundaries, a linear kernel may struggle to provide accurate classification results. In such cases, alternative kernels like

polynomial or RBF are preferred, as they can better capture intricate structures within the data. Furthermore, the linear kernel assumes that data can be perfectly separated in the original feature space, which may not always be the case, especially in biomedical signal processing or image recognition tasks.

Polynomial kernel

The polynomial kernel is a widely used kernel function in Support Vector Machines (SVM) that enables the model to capture non-linear relationships between data points. Unlike the linear kernel, which assumes a straight-line separation between classes, the polynomial kernel introduces higher-order transformations, allowing for more flexible decision boundaries. This is particularly useful for datasets where the relationship between features is not strictly linear, as it enables classification in cases where traditional methods fail to separate distinct groups effectively.

Mathematically, the polynomial kernel is expressed as

$$K(x,y)=(x.y+c)^d$$

Where (x) and (y) represent feature vectors, (c) is a constant (often set to 1), and (d) denotes the degree of the polynomial. The degree of the polynomial plays a crucial role in determining the complexity of the decision boundary the higher the degree, the more intricate the separation, allowing for finer classification of complex patterns within spectral data. However, increasing the degree also results in higher computational complexity, which may affect model efficiency when working with large datasets.

One of the primary advantages of the polynomial kernel is its ability to map data into higher-dimensional spaces, making it particularly effective for datasets with moderate non-linearity. It also provides greater control over decision boundary flexibility, enabling researchers to fine-tune the degree of the polynomial to optimize model performance. Additionally, the polynomial kernel performs well for medium-dimensional datasets, balancing computational efficiency with classification accuracy. Compared to the linear kernel, which is limited to simple data distributions, the polynomial kernel extends classification capabilities, offering more accurate predictions in cases where data follows curved separations rather than straight-line boundaries.

Despite its strengths, the polynomial kernel has some limitations. One of its main drawbacks is its high computational demand, especially when using higher-

degree polynomials. This can lead to longer training times and increased memory usage, making it less efficient for large-scale applications. Furthermore, it may suffer from overfitting, particularly when the polynomial degree is excessively high, causing the model to fit the training data too closely while reducing generalization on unseen data. Lastly, for very high-dimensional datasets, the Radial Basis Function (RBF) kernel tends to perform better, as it provides greater adaptability to complex, overlapping feature distributions.

Radial basis function

The radial basis function (rbf) kernel is a popular choice in Support Vector Machine (SVM) models, especially when dealing with non-linear data. Here's a deeper dive into its significance:

The rbf kernel transforms the input space into a higher-dimensional space, allowing the SVM to find a linear boundary in this new space. This transformation is particularly useful for data that isn't linearly separable in its original form. The rbf kernel measures the distance between data points using a Gaussian function, defined as:

$$K(x, x') = \exp(-y||x - x'||2)$$

Where x and x' are two data points, and γ' is a parameter that controls the width of the Gaussian function.

One of the key advantages of the RBF kernel is its ability to capture intricate patterns in data, making it particularly useful for applications such as image classification, medical diagnostics, and spectral analysis. Unlike the polynomial kernel, which requires a predefined degree, the RBF kernel dynamically adapts to the dataset, offering greater flexibility in separating non-linearly distributed data. Additionally, it performs well in high-dimensional spaces, such as Raman spectroscopy datasets, where thousands of features need to be processed efficiently.

Despite its strengths, the RBF kernel has some limitations. One major drawback is its high computational cost, as training models with RBF can be resource-intensive, particularly for large datasets. Furthermore, because it operates in infinite-dimensional space, the decision boundary is less interpretable compared to simpler kernel functions like linear or polynomial. Additionally, selecting an appropriate gamma value is critical; incorrect tuning can either lead to overfitting or underfitting, affecting classification accuracy.

In this study, the RBF kernel was applied to classify Raman spectral data, demonstrating strong performance in handling signal variations and spectral shifts. By optimizing the gamma parameter, the model successfully balanced classification accuracy and generalization ability, making it a reliable choice for biomedical signal processing. Compared to linear and polynomial kernels, the RBF kernel provided greater adaptability to subtle spectral differences, reinforcing its suitability for AI-driven Raman spectroscopy applications.

2.5.2 ExtraTreesClassifier

The ExtraTreesClassifier is an ensemble learning method provided by the Scikit-learn library, known for its efficiency in handling various classification and regression tasks. This classifier, which stands for "Extremely Randomized Trees," is closely related to the Random Forest classifier but introduces greater randomness in determining how splits within decision trees are made. Unlike Random Forests, where splits are chosen based on the best feature and threshold, the Extra Trees classifier randomly selects split points for each feature. This increased randomization leads to better generalization, making the model less prone to overfitting, which is a common issue in decision tree-based models.[26]

The Extra Trees classifier builds a forest of unpruned decision trees, with each tree trained on a bootstrap sample of the dataset. The model makes predictions by aggregating the outputs from multiple trees using majority voting for classification tasks and averaging predictions for regression tasks. Since Extra Trees do not prune trees, they tend to be highly expressive, capturing complex patterns within datasets. However, the additional randomness in feature splits prevents excessive memorization of training data, improving model stability when working with unseen samples.

One of the significant advantages of the Extra Trees classifier is its robustness against overfitting. By increasing the randomness in feature selection, the model reduces its reliance on the training dataset's specific structure, enhancing its ability to generalize well across different datasets. This characteristic is particularly beneficial for large datasets with high-dimensional features, as it mitigates the risk of overfitting complex relationships while maintaining accurate predictions. Moreover, the Extra Trees classifier does not rely on bootstrap sampling, which distinguishes it from Random Forests by allowing for faster computation times during training.

In addition to classification and regression tasks, Extra Trees also provides feature importance estimation, offering insights into which attributes contribute most to the model's predictions. This capability is especially useful for data scientists and researchers, as it enables them to prioritize relevant variables, streamline feature selection, and optimize model performance. By analyzing feature rankings, users can identify key predictors that influence classification outcomes, leading to more interpretable machine learning models.

To utilize the Extra Trees classifier, users need to initialize the model, split data into training and testing sets, and train the model using the training data. Once the model has been trained, predictions can be made on the test data, and various performance metrics such as accuracy, precision, recall, and F1-score can be evaluated to assess model effectiveness. Fine-tuning parameters, such as the number of estimators, max depth, and minimum split criteria, allows users to optimize classification results based on the complexity of the dataset.

The ExtraTreesClassifier has found applications in many fields, including biomedical diagnostics, financial fraud detection, image recognition, and natural language processing (NLP). Its ability to handle large datasets efficiently, reduce overfitting, and improve generalization performance makes it a highly versatile tool for machine learning applications. Whether dealing with high-dimensional genomic data, sentiment analysis, or anomaly detection, Extra Trees provides a robust solution for tackling complex classification challenges.

For this experiment, both Support Vector Machines (SVM) and ExtraTreesClassifier were chosen due to their complementary advantages in classification tasks. SVM excels in high-dimensional spaces, making it well-suited for datasets where the number of features exceeds the number of samples. It is versatile, handling both linear and non-linear classification using different kernel functions, such as linear, polynomial, RBF, and sigmoid kernels. Additionally, SVM is robust, as it reduces the risk of overfitting by maximizing the margin between classes, and it employs the kernel trick to model complex, non-linear decision boundaries, ensuring clear margins of separation that enhance interpretability and generalizability.

On the other hand, ExtraTreesClassifier is an ensemble learning method that builds multiple randomized decision trees to improve classification accuracy. Unlike SVM, which relies on hyperplane-based separation, ExtraTreesClassifier determines

split points randomly, improving computational efficiency and reducing overfitting. It is particularly useful for high-dimensional and noisy datasets, as it enhances generalization through extreme randomness in split selection. Additionally, ExtraTreesClassifier ranks feature importance, helping to identify key spectral characteristics that influence classification accuracy.

By implementing both SVM and ExtraTreesClassifier separately, this study evaluated the effects of different preprocessing techniques on classification performance. While SVM demonstrated improvements with polynomial regression (PenPoly) and intensity modulation processing (IMP), ExtraTreesClassifier exhibited stable accuracy gains, with IMP proving most effective in enhancing signal clarity. This dual approach ensures a comprehensive evaluation of AI-driven Raman spectral classification, reinforcing the importance of preprocessing techniques in optimizing model performance.

2.5.3 1D-CNN

A one-dimensional convolutional neural network (1D-CNN) is a specialized deep learning architecture designed to extract meaningful patterns from sequential data whether that's a time series, an audio waveform, or a Raman spectrum. Instead of connecting every input feature to every neuron in the next layer (as in a fully connected network), a 1D-CNN applies small, trainable filters that "slide" along the input. Each filter learns to recognize a local pattern such as a narrow peak shape or a sudden rise in intensity wherever it occurs in the sequence. Because the same filter weights are reused at every position, the network naturally gains translation invariance: once it has learned what a spectral peak looks like, it will detect that peak whether it sits at 500 cm^{-1} or 1,200 cm^{-1}.[52]

Stacking multiple convolutional layers allows the network to build a hierarchy of features. In the earliest layers, filters might pick up on very simple motifs tiny bumps or dips in the spectrum while deeper layers combine those motifs into more complex signatures, such as doublet peaks or characteristic shoulders. Between convolutional stages, pooling operations (for example, max-pooling) reduce the length of the feature maps by summarizing local neighborhoods, which not only decreases computation but also makes the model robust to slight shifts in peak position or noise fluctuations typical of experimental data. [53]

Once the convolutional and pooling stages have distilled the raw input into rich, position-invariant feature maps, a 1D-CNN flattens these maps into a single vector

and passes them through one or more dense (fully connected) layers. These final layers act as a high-level decoder, learning how to weigh and combine the automatically extracted features to produce the final output such as a binary label for material A versus material B. Because the dense layers see the entire spectrum's worth of features at once, they can capture global relationships as well as the local details identified by the convolutional filters.

Designing an effective 1D-CNN for Raman spectral classification involves carefully choosing hyperparameters: the size of each convolutional kernel, the number of filters per layer, the depth of the network, and the use of regularization methods (dropout, L2 weight decay, and early stopping) to guard against overfitting. Smaller kernels (e.g., length 3–5) excel at detecting fine spectral peaks, while larger ones can capture broader patterns or overlapping features. A shallow network two or three convolutional blocks often suffices for small datasets, ensuring that the model is expressive enough without demanding prohibitively large amounts of data.

In this project I employed simple, "vanilla" 1D convolutional kernels of fixed length 5 in every Conv1D layer. Concretely, each convolutional block slides a learnable vector of five weights along the wavenumber axis (with stride=1 and no dilation), performing a dot-product over each contiguous window of five spectral intensities.

Modern advances in 1D-CNN design include the addition of residual connections, which help with training deeper networks by allowing gradients to flow more easily, and dilated convolutions, which expand the network's receptive field without adding extra parameters. Layer or batch normalization can accelerate convergence and stabilize training, while attention mechanisms can further improve the network's focus on the most informative spectral regions. Altogether, these innovations make 1D-CNNs a powerful, end-to-end alternative to manual feature engineering, automatically learning the optimal spectral descriptors for classification tasks. [54]

One of the most compelling advantages of 1D Convolutional Neural Networks (1D-CNNs) is their ability to efficiently process and learn from sequential data while maintaining computational speed and scalability. These models are especially powerful when working with time-dependent datasets, where the sequence and order of the data points carry important meaning such as electrocardiogram (ECG) signals, temperature readings, or word embeddings in text. Thanks to their architectural

design, 1D-CNNs can detect patterns, trends, or anomalies across input sequences without needing to examine each element in strict isolation.

Compared to other sequence models like Recurrent Neural Networks (RNNs) or LSTMs, which process input one step at a time while maintaining memory across steps, 1D-CNNs operate using a more parallel approach. Rather than iterating through the sequence step-by-step, CNNs apply multiple filters across small, sliding windows of the data. This parallelism makes them significantly faster during both training and inference, especially on longer sequences. Despite this difference in strategy, 1D-CNNs are still capable of identifying local dependencies and learning meaningful temporal representations often with fewer parameters and better generalization when data is limited or noisy.

This unique balance of speed and performance makes 1D-CNNs a popular choice across a wide range of real-world applications. In speech recognition, they can pick up phonetic patterns and transitions between sounds; in medical diagnostics, they help analyze heartbeats, brain waves, or muscle signals; in financial modeling, they can detect recurring trends in stock prices or economic indicators. They're also used in anomaly detection systems, industrial equipment monitoring, music genre classification, and even DNA sequence analysis.

What makes them particularly attractive is not just their efficiency, but also their interpretability. The learned filters in 1D-CNNs can often be visualized to understand what type of patterns the model is focusing on such as sharp spikes, periodic curves, or sustained rises and falls. This opens up a degree of transparency that helps researchers and practitioners trust the insights generated by the model.

The reason I choose 1D-CNN over others ANN models in this Raman spectroscopy project because its core operation sliding small, learnable filters across the wavenumber axis directly mirrors the way spectral peaks and shoulders manifest locally in the data. Instead of relying on handcrafted features or manual peak picking, the Conv1D layers automatically learn which combinations of neighboring intensities signal a meaningful vibrational mode. Because the same filters scan every position, the network gains translation invariance: it recognizes a characteristic peak shape regardless of whether it appears at 500 cm^{-1} or 1 200 cm^{-1}. This built-in shift-tolerance is especially valuable for experimental spectra, where peak positions can jitter slightly from sample to sample or due to instrument drift.

Moreover, by stacking multiple convolutional blocks, the 1D-CNN builds a hierarchy of increasingly abstract features early layers pick up on simple motifs like narrow bumps or dips, while deeper layers combine these into composite indicators (e.g., doublet peaks or broad shoulders). This hierarchical learning mirrors the multi-scale nature of Raman spectra, where both fine-scale peak shapes and broader baseline trends carry chemical information. Coupled with moderate polynomial baseline corrections and dropout-based regularization, the network learns rich, robust representations without overfitting even on a small dataset.

Chapter 3: METHODOLOGY

3.1 Integrated Development Environment

3.1.1 Jupyter Notebook

Jupyter Notebook is an open-source web application that allows users to create and share documents containing live code, equations, visualizations, and narrative text. Its versatility and ease of use make it a popular tool in data science, scientific computing, machine learning, and academic research. One of its key features is interactive coding, which supports real-time data analysis and visualization, enabling users to write and execute code interactively. Jupyter Notebook supports multiple programming languages, with primary support for Python, and extendable to R, Julia, and other languages via "kernels." The platform integrates seamlessly with libraries like Matplotlib, Seaborn, and Plotly, facilitating the creation and display of visualizations within the notebook. Moreover, Jupyter Notebook supports rich text formatting, allowing users to include formatted text, equations (using LaTeX), images, and links, making it ideal for detailed reports and documentation. The ability to share notebooks via email, GitHub, or convert them to formats like HTML, PDF, and slides, further promotes collaboration and reproducibility.

Jupyter Notebook is commonly used for a variety of purposes, including data analysis and exploration, where users can analyze datasets, perform statistical calculations, and visualize results. In machine learning, it is utilized to develop, train, and evaluate models. For educational purposes, Jupyter Notebook serves as a platform for creating interactive teaching materials and tutorials. It is also widely used in research documentation, allowing researchers to document their processes and findings in a reproducible format. Overall, Jupyter Notebook is an indispensable tool for data-intensive fields, providing a flexible and interactive environment for coding and data analysis. [29]

3.1.2 Google Colab

Google Colab, or *Google Colaboratory*, is a cloud-based platform that provides an environment similar to Jupyter Notebook for running Python code. It is widely used by data scientists, machine learning enthusiasts, and researchers due to its accessibility and robust features. One of its standout benefits is the availability of free GPUs and TPUs, which makes it ideal for computationally intensive tasks like deep learning. Additionally, since it runs entirely on the cloud, users don't need to install Python or libraries locally, saving time and effort.

Another key advantage of Google Colab is its collaboration features, which resemble those of Google Docs. Multiple users can work on the same notebook simultaneously, fostering teamwork and productivity. It integrates seamlessly with Google Drive for saving and sharing notebooks and comes pre-installed with popular libraries such as NumPy, pandas, TensorFlow, and PyTorch. Users can also install additional libraries or connect to a local runtime as needed, making it a flexible and powerful tool for various projects.

3.1.3 JetBrains Datalore

JetBrains Datalore is a powerful collaborative data science platform that brings together the versatility of Jupyter-compatible notebooks with smart coding assistance and team collaboration features. It supports languages such as Python, R, Scala, and Kotlin, while offering tools like code auto-completion, quick fixes, and built-in documentation to boost productivity. For data integration, Datalore allows users to connect to databases, use SQL cells for querying, and integrate with cloud storage services like Amazon S3 and Google Cloud Storage.

The platform excels in collaboration, enabling real-time teamwork on notebooks with customizable access levels. Interactive reporting is another key feature, allowing users to transform notebooks into polished, shareable reports with hidden code cells. Additionally, it supports scheduling and automation for repetitive tasks, offers custom environments via pip, Conda, or Docker images, and provides an on-premises hosting option for added security. These features make Datalore a compelling choice for both individual and organizational use cases.

For this experiment, Jupyter was picked for its Versatilities and user-friendly platform, perfect for conducting experiments. Its interactive notebooks allow you to write, test, and debug code in manageable chunks, making the process seamless and efficient. Plus, its ability to combine code, visualizations, and explanatory text in one place is ideal for documenting your experiment clearly.

3.1.4 StandardScaler

StandardScaler is a preprocessing tool in machine learning, specifically part of the scikit-learn library, that standardizes features by removing the mean and scaling to unit variance. Here's a deeper dive into its significance and functionality:

StandardScaler transforms the data in such a way that each feature's mean is centered at 0 and the standard deviation is scaled to 1. This is done using the formula:

$$z = \frac{x - \mu}{\sigma}$$

where x is the original feature value, μ is the mean of the feature, and σ is the standard deviation of the feature.

Key Benefits:

1. **Normalization**: Standardizing features ensures that they are on the same scale, which is especially important for algorithms sensitive to the scale of data, like SVMs with rbf kernels.
2. **Improved Performance**: It can lead to faster convergence during training and better overall performance of the model.
3. **Handling Outliers**: While it doesn't eliminate outliers, it can mitigate their impact, since the data is rescaled relative to the standard deviation.

Use Case:

When you have features with different units or scales, StandardScaler helps by ensuring that each feature contributes equally to the distance metrics used by machine learning algorithms, thereby improving the robustness and accuracy of the model.

In the context of the SVM model setup described earlier, StandardScaler is used to preprocess the Raman spectroscopy data, ensuring all features have a mean of 0 and a standard deviation of 1, leading to more consistent and reliable classification performance.

3.1.5 Penalized poly

Penalized poly generally refers to penalized polynomial regression, which incorporates a penalty term into the polynomial regression model to prevent overfitting. This advanced technique combines polynomial regression with regularization methods such as Lasso (L1), Ridge (L2), or Elastic Net, which merges both L1 and L2 regularizations. In penalized polynomial regression, polynomial features of the input data are generated up to a specified degree, expanding the feature set to capture non-linear relationships in the data.

The regularization term is a crucial component added to the model to shrink the coefficients and reduce its complexity. Lasso regularization applies an L1 penalty, encouraging sparsity by shrinking some coefficients to zero, effectively performing feature selection. Ridge regularization uses an L2 penalty, which penalizes the sum of the squared coefficients, resulting in smaller, more evenly distributed coefficients. Elastic Net combines both L1 and L2 penalties, balancing between feature selection and coefficient shrinkage to enhance the model's flexibility and performance.

The benefits of penalized polynomial regression are manifold. By incorporating a regularization term, the model prevents overfitting, ensuring it does not memorize the noise in the training data. This leads to better generalization and improved performance on new, unseen data. Additionally, regularization improves the interpretability of the model by selecting the most relevant features and shrinking the less important ones. This feature selection process can reveal insights into the underlying data structure and the relationships between variables.

Penalized polynomial regression is particularly useful in scenarios where the data exhibits complex, non-linear relationships that cannot be captured by simple linear models. By generating polynomial features, the model can fit a wider range of patterns and interactions between variables.

3.2 Data acquisition

A Raman spectrometer is comprised of four key components that work together to measure the Raman effect:

- **Laser**: This is the light source that emits a focused beam of monochromatic light. When this laser light interacts with the sample, it induces the Raman scattering effect. The laser is a critical component as it provides the necessary energy to excite the molecules within the sample.
- **Sample Holding Chamber**: This component is designed to securely hold the sample during the measurement process. The sample can be in various forms, such as solid, liquid, or gas. The chamber is often equipped with mechanisms to ensure the precise positioning and stability of the sample for accurate measurements.
- **Spectrometer Chamber Using Grating**: After the laser light interacts with the sample, the scattered light, including the Raman scattered photons, is directed into the spectrometer chamber. Here, a diffraction grating separates the light into its component wavelengths (or wavenumbers). This dispersion allows for the simultaneous measurement of a broad spectrum of wavelengths, which is essential for identifying the unique Raman shifts associated with the sample's molecular vibrations.
- **Amplification System and Collection**: The dispersed light is then collected and amplified to enhance the signal strength for better

detection and analysis. This amplified signal is processed to remove photons that have been displaced by the Raman effect using notch or edge filters. These filters ensure that only the desired Raman scattered photons are analyzed, improving the accuracy of the measurements.

Photons that have been displaced by the Raman effect are meticulously filtered out using notch or edge filters. These filtered photons are then channeled into the grating spectrometer, where the wavenumbers of interest are measured simultaneously across the entire spectrum. This comprehensive measurement allows for the detailed analysis of the sample's molecular composition, providing valuable insights into its chemical and physical properties.

In order to carry out my project, I needed to simulate blood sugar levels. However, I encountered a challenge due to the absence of genuine data from the human body. Collecting data from human volunteers requires consent and medical expert review. To circumvent these complications, I opted for a practical approach by using a diluted glucose solution as a temporary measure. This method will suffice until a more robust solution can be developed to realistically address the data issues.

The dataset used in our experiments were collected from a Raman spectrometer which was provided by my supervisor. All of the data from the in this experiment were taken at two stages from volunteers. The dataset used for the machine learning model contains three labels:

1. **Before consumed food**: Represents samples from patients who haven't consumed any food.
2. **After consumed food**: Represents samples from patients who have consumed food.

In total, we got an CSV file with the data from 20 Raman spectroscopic scans acquired from the left thumbnail. These labels are essential for training the Support Vector Machine (SVM) model to differentiate between different levels of blood glucose. The Raman spectroscopy samples corresponding to these labels provide the input data for the machine learning model. By analyzing the Raman spectra, the model learns to identify patterns and correlations associated with each label.

By using these labels, the machine learning model can accurately predict the blood glucose levels of patients based on their Raman spectroscopy samples, providing a valuable tool for diabetes classification and management.

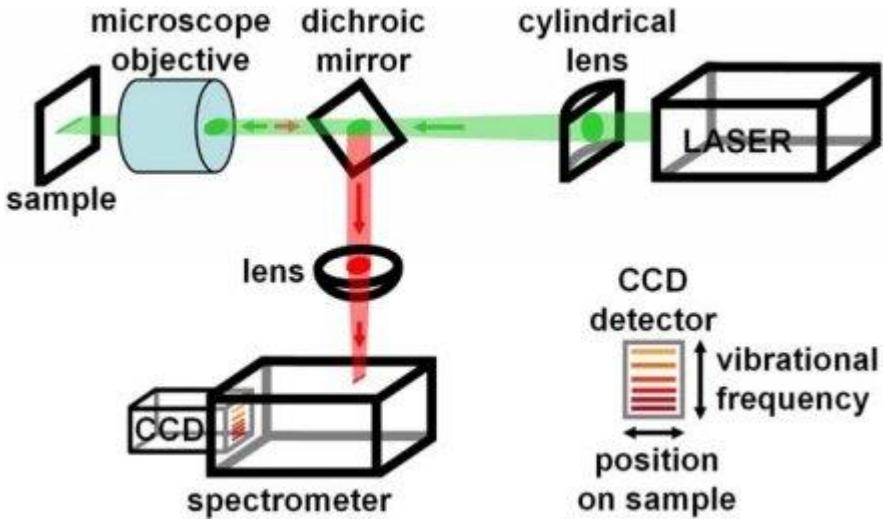

Figure 3: Schematic diagram of Raman spectrometer setup using CCD detection module Excitation laser (Andrew Downes, 2024)

3.3 Raman spectroscopy

Professor C.V. Raman, in collaboration with K.S. Krishnan, was the pioneering force behind the discovery of the Raman effect, with their initial publication marking a significant milestone in the field of spectroscopy [40]. This groundbreaking technique remains one of the most adaptable methods for analyzing a diverse range of evidence, from chemical compounds to biological materials.

Raman spectroscopy's versatility and non-invasive nature make it invaluable in various fields, including chemistry, physics, biology, and medicine. It provides detailed molecular information, allowing for early disease diagnosis, monitoring disease progression, and environmental analysis. The legacy of Raman and Krishnan's work underscores the importance of innovation and collaboration in scientific research, paving the way for future advancements in the realm of spectroscopy. [41]

Figure 4: Sir Chandrasekhara Venkata Raman (nobelprize, 2024)

This technique surpasses many constraints of current spectroscopic methods and is valuable for quantitative and qualitative analysis. Quantitative analysis measures the intensity and frequency of scattered radiations, providing detailed molecular composition information. This is crucial in fields like chemistry and biochemistry, where precise measurements are needed. Qualitative analysis identifies specific molecules or functional groups by examining Raman shifts corresponding to molecular vibrations. This method can identify unknown compounds, verify mixtures, and monitor molecular structure changes, making it a versatile and powerful tool in research and industry [42].

Raman spectroscopy, based on the Raman effect, is a scattering technique wherein monochromatic laser light interacts with molecules in a sample, producing scattered light. The wavelength of this Raman scattered light is determined by the excitation light's wavelength. Consequently, when comparing spectra obtained with different lasers, the Raman scatter wavelength is standardized to an artificial value. This process involves shifting the excitation wavelength away from the Raman scattering point, resulting in what is known as a Raman shift.

The wavelength of the excitation light determines the wavelength of the Raman scattered light. Therefore, when spectra are recorded using different lasers, the Raman scatter wavelength is standardized to an artificial value. Essentially, the

excitation wavelength is shifted away from the Raman scattering point, resulting in what is known as a Raman shift.

Raman shifts are typically expressed in wavenumbers, which are units of inverse length directly related to energy. In most cases, the wavenumber in Raman spectra is denoted in inverse centimeters (cm^{-1}). This unit is preferred because it provides a direct correlation to the energy levels involved in the scattering process.

For practical purposes, the computation of wavenumbers in Raman spectra can be explicitly scaled for unit conversion to nanometers (nm). This flexibility allows researchers to compare and analyze data collected using different laser sources and conditions.

By using wavenumbers, scientists can more easily interpret and compare Raman spectra, regardless of the specific laser wavelengths employed. This standardization is crucial for ensuring consistency and accuracy in Raman spectroscopic analysis, facilitating the identification and characterization of various molecular compounds and structures.

3.4 Fluorescence background subtraction

Fluorescence interference in Raman spectroscopy can arise due to two main factors: molecule interference and sample contamination. Molecule interference occurs when high-energy photons are absorbed, causing molecules to be excited to a higher electronic state. As these molecules return to a lower energy state, they emit fluorescence light. On the other hand, sample contamination can introduce foreign substances that fluoresce under the excitation light.

Unlike fluorescence, Raman shifts are independent of the wavelength, making them distinct in this regard. However, the occurrence of fluorescence is significantly influenced by the excitation wavelength used during the Raman spectroscopy process. This distinction between Raman shifts and fluorescence emission is crucial for accurate spectroscopic analysis, as it helps differentiate between the two phenomena and mitigates interference.

Raman scattering and fluorescence emission can compete when the excitation laser energy is near the electronic transition energy of the material. Fluorescence background is amplified by excitation sources with higher green or red intensities, such as visible lasers at 514 nm or 633 nm. Conversely, near-infrared lasers, such as those operating at 785 nm or 1064 nm, lack the energy to excite molecules to a higher

electronic state or to remove fluorescing molecules in the material, resulting in a reduced or eliminated fluorescence effect [43].

Numerous polynomial fitting algorithms were examined to eliminate background fluorescence in the original signal. To maintain data clarity, the Improved Modified Polynomial (IMP) and penalized polynomial (penalized_poly) fitting method was both employed for the experiment. This method has proven effective in recent investigations due to its low signal-to-noise ratio. The pure Raman spectrum is calculated by subtracting the corrected polynomial from the original Raman spectra and then summing the resulting values.

3.5 Machine Learning model selection

To achieve the research goal of automating diabetes classification using Raman spectroscopy samples, a machine learning model is employed to differentiate between diabetic and non-diabetic patients. According to [45], the linearity of the relationship between Raman intensity at specific wavelengths can be used to determine blood glucose levels. However, due to variations in environmental settings and the nature of the subjects, a fixed wavenumber or explicit function for interpretation cannot be used, as it may lead to prediction errors for new input samples. Consequently, a broad range of wavenumbers in Raman spectra should be interpreted simultaneously. This demonstrates the advantages of machine learning models, which can precisely predict outcomes based on implicit correlations from high-dimensional samples, involving multiple Raman shift interpretations.

Before feeding the data into machine learning models, the fluorescence subtraction method is employed as a data preprocessing step, taking into account the inherent characteristics of Raman spectroscopy. This process aims to eliminate visible fluorescence signals from the dataset, which could otherwise introduce significant variance at certain wavelengths and lead to prediction bias if not correctly addressed.

After the fluorescence subtraction method was employed to remove visible fluorescence signals, the data was shuffled. This ensures that it is randomized, helping to prevent any biases during the training of the machine learning models. This step is crucial for enhancing the robustness and generalizability of the models, ensuring that they can accurately differentiate between the three test subject stages based on the processed Raman spectroscopy samples. The three stages include patients who have not eaten, patients who have consumed one tea cup of sugar water, and patients who

have consumed two tea cups of sugar water. By shuffling the data, the machine learning model can more effectively learn and predict the different stages, leading to more reliable and accurate results.

The Support Vector Machine (SVM) model is then set up using the radial basis function (rbf) kernel, a popular choice due to its ability to handle non-linear relationships effectively. Additionally, the data is scaled using StandardScaler, which standardizes the features by removing the mean and scaling to unit variance. This preprocessing step is crucial because it ensures that all features contribute equally to the model, preventing features with larger scales from disproportionately influencing the results.

To evaluate the effectiveness of the SVM model, performance metrics such as accuracy, sensitivity, and specificity are initialized. These metrics provide a comprehensive assessment of the model's classification abilities. Accuracy measures the overall correctness of the model's predictions, while sensitivity (also known as recall) evaluates the model's ability to correctly identify positive cases, such as identifying patients who consumed one or two tea cups of sugar water. Specificity assesses the model's capability to correctly identify negative cases, such as patients who have not consumed any sugar water. By tracking these metrics, researchers can gain insights into the model's strengths and weaknesses, ensuring that it performs well across different aspects of classification.

Furthermore, the use of repeated stratified k-fold cross-validation ensures that the model is trained and tested on multiple subsets of the data, providing a robust evaluation of its performance. This approach helps to minimize bias and variance, offering a more reliable estimate of the model's generalizability to new, unseen data.

In summary, the combination of SVM with the rbf kernel, data scaling, and comprehensive performance metrics, along with cross-validation, creates a robust framework for developing an accurate and reliable model for differentiating between patients who have not eaten, patients who have consumed one tea cup of sugar water, and patients who have consumed two tea cups of sugar water based on Raman spectroscopy samples. Extra Tree Classifier was also chosen to have a comparison between the two models.

3.6 Experimental setups

To conduct the experiment, the initial step involves preparing the data for machine learning. The raw dataset comprises 20 samples, with 10 samples collected

before the volunteers had breakfast and 10 samples collected after breakfast. The dataset is labeled accordingly: 0 for the samples taken before breakfast and 1 for those taken after breakfast.

The dataset used in this research consists of Raman spectral data, capturing the interaction between laser light and biological molecules. Each data sample contains a series of intensity values measured at different wavelengths, forming spectral curves that indicate chemical composition. These spectral signals reflect glucose concentration variations, enabling classification between different glucose levels or between diabetic and non-diabetic subjects. The dataset is high-dimensional, meaning it contains multiple spectral variables, each contributing unique information about molecular vibrations. Since Raman spectroscopy relies on detecting subtle shifts in scattered light, the dataset inherently includes natural fluctuations in intensity, which need careful handling for accurate interpretation. Furthermore, the spectral characteristics may vary based on environmental conditions, measurement settings, and biological differences, influencing the distribution of intensity values across samples. The dataset provides a foundation for machine learning models to identify patterns, correlations, and diagnostic insights, making it a valuable resource for non-invasive medical analysis using AI-driven spectral classification.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	15632.13	15616.83	15591.03	15567.12	15557.34	15571.8	15602.51	15632.4	15644.42	15623.13	15572.24	15508.77	15449.86
2	10535.58	10500.02	10436.48	10367.74	10316.45	10302.28	10320.67	10354.84	10388	10404.45	10400.67	10381.65	10352.51
3	5338.734	5352.828	5374.926	5401.653	5429.654	5455.856	5479.365	5500.396	5519.175	5535.687	5546.907	5547.715	5532.993
4	6376.677	6387.698	6371.707	6347.66	6334.383	6348.5	6388.767	6444.903	6506.635	6563.958	6609.53	6637.87	6643.573
5	9496.174	9507.935	9498.541	9478.182	9456.983	9443.87	9438.076	9433.919	9425.727	9408.384	9383.545	9357.563	9336.843
6	7527.561	7362.716	7160.907	6948.213	6750.524	6590.738	6467.421	6366.843	6275.275	6180.75	6093.018	6036.891	6037.304
7	8648.555	8649.593	8659.856	8676.628	8697.202	8719.239	8743.309	8771.462	8805.748	8847.621	8891.019	8924.679	8937.314
8	7570.491	7552.002	7516.804	7483.56	7470.802	7494.07	7544.196	7599.542	7638.535	7641.427	7610.373	7562.73	7516.039
9	12730.35	12685.55	12614.23	12535.78	12469.45	12431.9	12418.5	12413.91	12402.78	12371.31	12324	12278.11	12251.02
10	11318.44	11334.5	11350.6	11363.58	11370.31	11368.11	11358.23	11343.88	11328.25	11314.28	11301.97	11289.22	11273.94
11	9383.098	9350.019	9289.731	9232.188	9207.136	9239.767	9317.813	9410.093	9485.51	9515.303	9498.412	9453.009	9397.523
12	9493.324	9438.436	9383.074	9335.773	9305.006	9297.546	9305.984	9315.77	9312.4	9282.832	9231.682	9175.816	9132.237
13	9178.779	9024.868	8756.429	8395.624	7964.47	7485.821	6990.925	6515.169	6093.712	5761.128	5547.527	5479.859	5584.81
14	11091.83	11044.55	10989.36	10938.7	10904.88	10897.94	10908.67	10918.24	10907.85	10860.72	10784.65	10704.52	10645.34
15	14482.34	14666.54	14840.71	15021.85	15226.86	15469.98	15743.44	16028.38	16305.96	16558.71	16784.71	16992.88	17192.32
16	11084.35	11012.08	10970.29	10958.43	10975.98	11020.86	11077.77	11124.77	11140.03	11104.35	11030.79	10954.81	10912.04
17	12425.38	12446.3	12470.53	12499.21	12533.47	12573.66	12613.52	12643.45	12653.88	12636.34	12595.36	12544.56	12497.63
18													

Figure 5: raw data

The dataset used in this research consists of 20 rows and 1901 columns, representing spectral data obtained from Raman spectroscopy analysis. Each row corresponds to an individual spectral sample, potentially collected from different subjects or experimental conditions. The 1901 columns contain spectral attributes,

with each column representing the intensity of Raman signals at a specific wave number, forming a detailed spectral profile that provides insights into molecular vibrations and chemical composition.

The raw data collected from the human body is saved in CSV format, with each measurement containing two primary columns: the wave number (x value) in shift Raman (cm^{-1}) and the intensity (y value) of the Raman signal at each wave number. This spectral intensity data helps detect and analyze specific molecular vibrations, providing crucial information about the presence and concentration of biological molecules. The original dataset records wave numbers ranging from 400 to 2300 cm^{-1}, but for the processed dataset, only wave numbers between 800 cm^{-1} and 1800 cm^{-1} are considered. This specific range contains vibrational bands associated with key chemical functional groups in biological molecules such as proteins, lipids, nucleic acids, and carbohydrates, making it particularly useful for investigating biological tissue structure and chemical composition.

Given the high-dimensional nature of the dataset, preprocessing techniques are essential for improving classification accuracy. The spectral data's complex structure suggests that machine learning models can be used to classify glucose concentration levels and analyze biochemical differences in diabetic versus non-diabetic samples. This research focuses on applying Support Vector Machines (SVM) and ExtraTreesClassifier to effectively process and classify these spectral variations, ensuring a robust approach to AI-driven Raman Spectroscopy analysis.

	A	B
1		Scan25031901
2	400	11135.43
3	401	11173.49
4	402	11181.88
5	403	11146.37
6	404	11080.8
7	405	11007.94
8	406	10950.38
9	407	10927.7
10	408	10936.12
11	409	10960.67
12	410	10986.4
13	411	10999.38
14	412	10997.75
15	413	10987.98
16	414	10976.7
17	415	10970.32
18	416	10971.6
19	417	10979.74
20	418	10993.8

Figure 6: parts of the raw data that was cut from 400 to 2300

After processing, the dataset is labeled with 0 and 1 for machine learning analysis. Labels of 0 and 1 correspond to measurements taken before and after the condition noted above. This labeling helps in comparing the effectiveness of noise reduction techniques and making subsequent improvements.

	A	B	C
1	label_code		
2	0	truoc_an_sang	
3	0	truoc_an_sang	
4	0	truoc_an_sang	
5	0	truoc_an_sang	
6	0	truoc_an_sang	
7	0	truoc_an_sang	
8	0	truoc_an_sang	
9	0	truoc_an_sang	
10	0	truoc_an_sang	
11	0	truoc_an_sang	
12	1	sau_an_sang	
13	1	sau_an_sang	
14	1	sau_an_sang	
15	1	sau_an_sang	
16	1	sau_an_sang	
17	1	sau_an_sang	
18	1	sau_an_sang	
19	1	sau_an_sang	
20	1	sau_an_sang	
21	1	sau_an_sang	

Figure 7: label

3.7 Polynomial Fitting Method for baseline determination

The Polynomial Fitting Method is an essential approach for baseline determination in spectral analysis, particularly in Raman spectroscopy, where unwanted variations in intensity can interfere with the accurate identification of molecular structures. Baseline distortions in spectral data often arise due to instrumental noise, fluorescence interference, and environmental factors, necessitating robust correction techniques to refine spectral accuracy. Polynomial fitting models the baseline as a smooth polynomial function, which is subtracted from the raw spectral data to remove background signals while preserving the integrity of significant spectral peaks.

A polynomial can be expressed as follows:

$$p(x) = \beta_0 x^0 + \beta_1 x^1 + \beta_2 x^2 + \ldots + \beta_m x^m = \sum_{j=0}^{m} \beta_j x^j$$

where β is the array of coefficients for the polynomial.

For regular polynomial fitting, the polynomial coefficients that best fit data are gotten from minimizing the least-squares:

$$\sum_{i}^{N} w_i^2 (y_i - p(x_i))^2$$

where:

y_i and x_i are the measured data,

$p(x_i)$ is the polynomial estimate at x_i,

w_i is the weighting.

N is the number of data points

The primary advantage of polynomial fitting is its simplicity and effectiveness. It is faster than other methods and has been widely used for in vivo-biomedical Raman applications. The weakness of polynomial fitting is its dependence on the spectral fitting range and the chosen polynomial order. [46]

The advantages of polynomial fitting for baseline correction include its ability to generate a smooth and continuous correction curve, making it highly effective in fluorescence removal from spectral signals. Additionally, the method allows for fine-tuning of the polynomial degree, ensuring that important spectral features are retained while irrelevant background noise is eliminated. However, selecting the optimal polynomial degree is a challenge, as an inadequately fitted polynomial can either fail to remove baseline distortions or excessively alter the spectral structure. Polynomial fitting may also be sensitive to noise, requiring additional smoothing techniques to refine corrections. Furthermore, higher-degree polynomial fitting can introduce computational complexity, particularly when applied to large spectral datasets.

In Raman spectroscopy, polynomial fitting plays a vital role in refining spectral intensity data, ensuring accurate peak detection for biochemical classification. Advanced modifications of the method, such as piecewise polynomial fitting (PPF), divide spectral data into smaller sections, allowing for localized baseline corrections. Additionally, adaptive polynomial fitting techniques integrate optimization

algorithms to dynamically refine polynomial parameters, enhancing baseline accuracy across varied spectral conditions.

To improve background correction, I use the penalized polynomial regression and Improved Modified Polynomial (IMP) method to compare the results of each method. The program starts with a single polynomial fitting P(v)P(v) using the raw Raman signal O(v)O(v), where v is the Raman shift in cm^{-1}. The residual R(v)R(v) and its standard deviation (DEV) are then calculated as follows:

$$R(v) = O(v) - P(v)$$

And

$$DEV = \sqrt{\frac{(R(v_1) - \bar{R})^2 + (R(v_2) - \bar{R})^2 + \cdots + (R(v_n) - \bar{R})^2}{n}}$$

where nn represents the number of data points on the spectral curve, and

$$\bar{R} = \sqrt{\frac{R(v_1) + R(v_2) + R(v_3) + \cdots + R(v_n)}{n}}$$

This approach ensures a more accurate baseline correction by accounting for the residuals and their variations.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	89.45683	67.03949	44.14872	27.46368	20.57853	24.89743	41.77855	70.50403	102.213	126.0537	131.4213	117.5998	97.24754
2	68.9468	89.08204	107.2534	113.476	108.194	99.24171	94.59723	100.2778	114.5855	133.9346	154.7263	172.6557	182.447
3	159.4861	147.2645	125.8939	101.4946	78.99689	62.44094	55.85703	61.89542	77.85634	99.70006	123.4168	145.5469	163.3705
4	126.5707	121.6018	119.6395	126.3338	139.3341	150.6145	152.0411	138.6441	117.8797	100.2391	96.02551	107.826	127.8058
5	110.2611	104.8458	96.80528	91.82573	91.57536	94.86238	100.441	106.6174	109.9559	106.5906	92.79867	70.72043	50.41402
6	145.4505	160.9303	167.5745	159.0733	138.627	116.2258	101.9598	103.4395	118.4349	142.3363	170.5439	198.778	223.2987
7	159.3223	124.481	86.54039	55.76055	37.25173	32.46413	42.76798	67.61348	98.84084	126.4403	140.592	139.3863	131.5732
8	67.40307	73.51069	73.76266	62.50016	41.63935	18.46141	0.350501	-7.00321	-4.53556	5.188633	19.61852	36.50628	54.01606
9	88.3585	67.05582	54.64106	61.29838	81.81792	100.0968	99.8053	70.65643	25.99439	-15.0527	-33.6766	-24.0313	2.219523
10	60.14399	28.21541	11.34983	28.31748	72.72856	120.3433	146.5419	134.9945	101.7715	70.87288	65.85903	93.29009	136.7263
11	127.4806	119.4662	114.408	117.5332	125.3931	128.4007	116.8494	85.06431	43.19567	5.259681	-14.9494	-12.5875	5.086718
12	150.6527	183.2464	212.2735	226.4175	222.7884	204.4636	174.6423	138.0227	105.169	88.07754	98.47742	136.9169	188.8613
13	130.0289	141.3039	155.813	168.9203	176.3992	174.3067	158.7171	128.9216	96.87347	77.61282	85.84889	122.4139	169.41
14	133.2309	123.9819	116.1779	121.7101	139.5378	159.4562	171.0774	166.8187	150.0763	126.9364	103.4763	85.64803	79.23694
15	60.30692	71.45691	89.6048	113.8138	139.629	160.0867	168.1839	159.7499	141.7369	123.8139	115.4932	119.7	130.4783
16	96.50942	98.01472	112.2388	136.7749	163.7521	181.4218	177.969	147.119	104.3449	70.4291	65.73163	93.16375	132.0616
17	119.4057	118.8644	122.3617	129.7539	138.8711	146.1136	147.8485	142.0361	132.8695	126.061	127.2268	137.9791	154.518
18	117.7839	123.8238	123.8119	115.5414	100.5526	83.03454	67.22552	56.61573	51.74636	52.4346	58.47865	68.78171	81.05099
19	162.9642	197.1927	210.9227	198.6064	166.196	127.7768	97.60081	86.82232	94.4775	116.6345	149.2866	184.9359	211.3706
20	92.58729	117.0214	127.9695	113.963	81.90382	51.69729	43.50854	69.93173	113.913	151.1456	157.7806	128.8413	84.82972

Figure 8: denoised data after combined all of the denoised CSV files

After the data was transformed like in Figure 5, each of 20 CSV file will go through penalized polynomial regression for data denoising and signal enhancing, result in 20 CSV files like in figure 8, then the data is combined and swap from columns to rows like in figure 7.

	A	B
1	800	89.45683
2	801	67.03949
3	802	44.14872
4	803	27.46368
5	804	20.57853
6	805	24.89743
7	806	41.77855
8	807	70.50403
9	808	102.213
10	809	126.0537
11	810	131.4213
12	811	117.5998
13	812	97.24754
14	813	83.96157
15	814	89.30606
16	815	108.2312
17	816	127.5191
18	817	133.9179
19	818	120.6429
20	819	97.41009

Figure 9: denoised data

3.8 Jupyter setup

To implement the algorithm for determining baselines of the provided data in my project, the first step involves installing and importing the Pybaselines library. This is the core library of my project, offering solutions for data calculation and preprocessing. Additionally, I have incorporated the numpy library, which provides functions and data structures for working with arrays and matrices, enhancing calculation efficiency.

I'm also using the matplotlib.pyplot library to create high-quality graphs and charts from the data. Furthermore, I utilize the csv library to read and write CSV files,

a widely-used format for tabular data, enabling easy reading and writing of data. Finally, the os library is employed to interact with the operating system, allowing operations such as creating, deleting, or moving files, among others.

```
import csv
import os
import matplotlib.pyplot as plt
import pybaselines as pbl
import numpy as np
```

Figure 10: library for penalized polynomial regression and graph draw

After that I start by retrieving the CSV files that was cut by scanning the folder which the CSV files are on.

```
def GetDataFromFileCSV(source, filename, axit):
    datay = []
    datax = []
    with open(f'{source}/{filename}', 'r') as csvfile:
        plots = csv.reader(csvfile, delimiter=',')
        for row in plots:
            if row[0] != "":
                try:
                    datax.append(int(row[0]))
                    datay.append(float(row[1]))
                except ValueError:
                    print(f"Skipping invalid entry: {row[0]}")
    if axit == 'x':
        return datax
    if axit == 'y':
        return datay
```

Figure 11: GetDataFromFileCSV function

This function reads data from a CSV file and returns either the x-axis or y-axis values, depending on the axit parameter. It processes the file, skipping invalid entries and ensuring that the data is properly parsed into integer and float types. The function work in this order:

1) **Initialization**: The function initializes two empty lists, datay and datax, to store the y-axis and x-axis values.
2) **Opening the File**: It opens the specified CSV file using the with open statement, ensuring the file is properly closed after reading.
3) **Reading the File**: The csv.reader is used to read the CSV file, and the function iterates over each row in the file.
4) **Data Parsing**: For each row, if the first element is not empty, the function attempts to convert it to an integer and append it to datax. It also converts the second element to a float and appends it to datay. If a ValueError occurs during conversion, the function skips the invalid entry and prints a message.
5) **Returning Data**: Depending on the value of the axit parameter ('x' or 'y'), the function returns the corresponding list of values.

This ensures that the data is read and parsed correctly from the CSV file, ready for further processing or analysis.

Next, I created an additional function called Draw_Graph_Raw to display signal line graphs, which will help in observing changes before and after signal processing. This function takes the shift Raman parameter values and intensities from the signal and labels each component in the graph, including axis names, graph names, graph colors, and values at points.

```
def Draw_Graph_Raw(x_axit, y_axit, x_label, y_label, graph_name, color, line_name):
    plt.figure()
    plt.plot(x_axit, y_axit, color=color, label="Processed signal "+line_name)
    plt.xlabel(x_label)
    plt.ylabel(y_label)
    plt.title(graph_name)
    plt.xticks(x_axit[::50],  rotation='vertical')
    plt.legend()
def Draw_Graph_Processed(x_axit, y_axit, x_label, y_label, graph_name, color, line_name):
    plt.figure()
    plt.plot(x_axit, y_axit, color=color, label="Processed signal "+line_name)
    plt.xlabel(x_label)
    plt.ylabel(y_label)
    plt.title(graph_name)
    plt.xticks(x_axit[::50],  rotation='vertical')
    plt.legend()

def Draw_Graph_Baseline(x_raw, y_raw, x_axit, y_axit, x_label, y_label, graph_name, color, color_raw):
    plt.figure()
    plt.plot(x_raw, y_raw, color=color_raw, label = "Unprocessed Signal")
    plt.plot(x_axit, y_axit, color=color, label="Baseline")
    plt.xlabel(x_label)
    plt.ylabel(y_label)
    plt.title(graph_name)
    plt.xticks(x_axit[::50],  rotation='vertical')
    plt.legend()
```

Figure 12: fuctions that draw graphs

Draw_Graph_Raw, is designed to plot a graph using the given x-axis and y-axis data. Here's a breakdown of how the function works:

Create a New Figure: The function starts by creating a new figure using plt.figure() to ensure that each call to the function generates a new plot.

Plot the Data: The plt.plot function is used to plot the data, with x_axit and y_axit representing the x-axis and y-axis values, respectively. The color parameter sets the line color, and label is used to create a label for the legend, combining the text "Processed signal" with the line_name parameter.

1) **Set Labels:** The plt.xlabel and plt.ylabel functions set the labels for the x-axis and y-axis, respectively, using the x_label and y_label parameters.
2) **Set Title:** The plt.title function sets the graph's title using the graph_name parameter.
3) **Custom X-Axis Ticks:** The plt.xticks function customizes the x-axis ticks, displaying them at intervals of 50 units and rotating them vertically for better readability.
4) **Add Legend:** The plt.legend function adds a legend to the plot, which helps in identifying the plotted line based on the label provided.

The SaveFlattenData function, similar to the read function, is used to write processed data into a CSV file line by line. Instead of searching for lines with data to

read, this function finds empty lines to insert the data. When used in conjunction with the Autoname function I created, the output becomes a complete CSV file containing the processed signal data.

```
def SaveFlattenData(dir, filename, datay, datax):
    locate = dir + '/' + filename
    data_convert = [str(x) for x in datay]
    data = []
    for i in range(len(datay)-1):
        temp = []
        temp.append(datax[i])
        temp.append(datay[i])
        data.append(temp)
    with open(locate, 'w',newline='') as csvfile:
        writer = csv.writer(csvfile)
        writer.writerows(data)
```

Figure 13: SaveFkattenData Function

This function is designed to save processed data into a CSV file. Here's a step by step of what the function does:

Specify File Location: The function constructs the file path by combining the directory (dir) and filename (filename).

Convert Data to Strings: It converts the datay values to strings and stores them in data_convert. However, this step isn't actually used later in the code.

Combine Data: It creates a list called data that will store the combined datax and datay values as sublists. Each sublist represents a row in the CSV file, containing the x and y values.

Writing to CSV: The function opens the specified file in write mode. It then uses the csv.writer to write each row from the data list to the CSV file. The newline='' parameter ensures that no extra blank lines are added between rows in the CSV file.

To process noisy signals, the first step involves determining the baseline, which is essential for subtracting the background and filtering the noisy signal. In this project, I am utilizing the penalized polynomial regression method in conjunction

with the algorithm chart provided. This approach ensures that the baseline is accurately identified, enabling effective noise reduction and signal enhancement.

```
for i in range(len(reopen_rawFile)):
    if i == openRaw_count:
        break
    x_raw.clear()
    y_raw.clear()
    x_raw = GetDataFromFileCSV(sourcelink, reopen_rawFile[i], 'x')
    y_raw = GetDataFromFileCSV(sourcelink, reopen_rawFile[i], 'y')
    Draw_Graph_Raw(x_raw, y_raw, 'Wave', 'Intense', reopen_rawFile[i], 'red', str(i))
    base_line_data = pbl.polynomial.penalized_poly(y_raw, x_raw, poly_order)
    base_line = base_line_data[0]
    Draw_Graph_Baseline(x_raw, y_raw, x_raw, base_line, 'Wave', 'Intense', reopen_rawFile[i], 'yellow', 'red')

plt.show()
```

Figure 14: Determine signals' baselines and draw graphs.

This code effectively processes each file, removes noise by calculating the baseline using penalized polynomial regression, and visualizes both the raw and baseline-corrected data.

In this project, my goal was to establish baselines for unprocessed signals using polynomial order coefficients ranging from 3 to 16. During testing, I found it challenging to manually select the most suitable polynomial order coefficient based solely on visual observation of the graph. To address this, I identified specific key points within the range and determined the sample baseline for background correction. The processed data was then fed into a machine learning model to evaluate the outcomes. This approach saved me from manually calculating using linear formulas involving weights, standard deviations, and more, which would have been quite challenging given the large number of signal samples and the time constraints. Ultimately, I chose polynomial orders 3, 7, 12 and 16 for baseline determination consideration.

Here are the results:

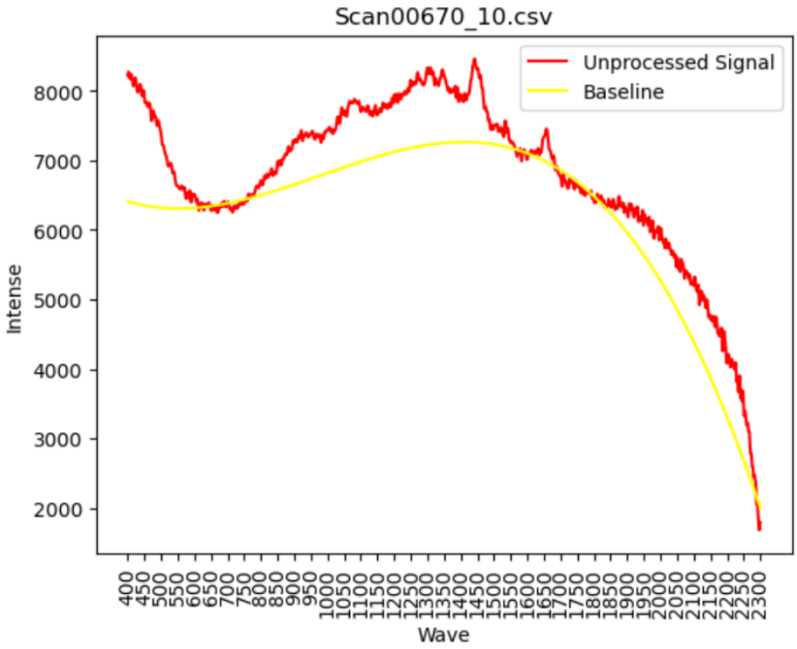

Figure 15: Determine the signal's baseline using polynomial order of 3

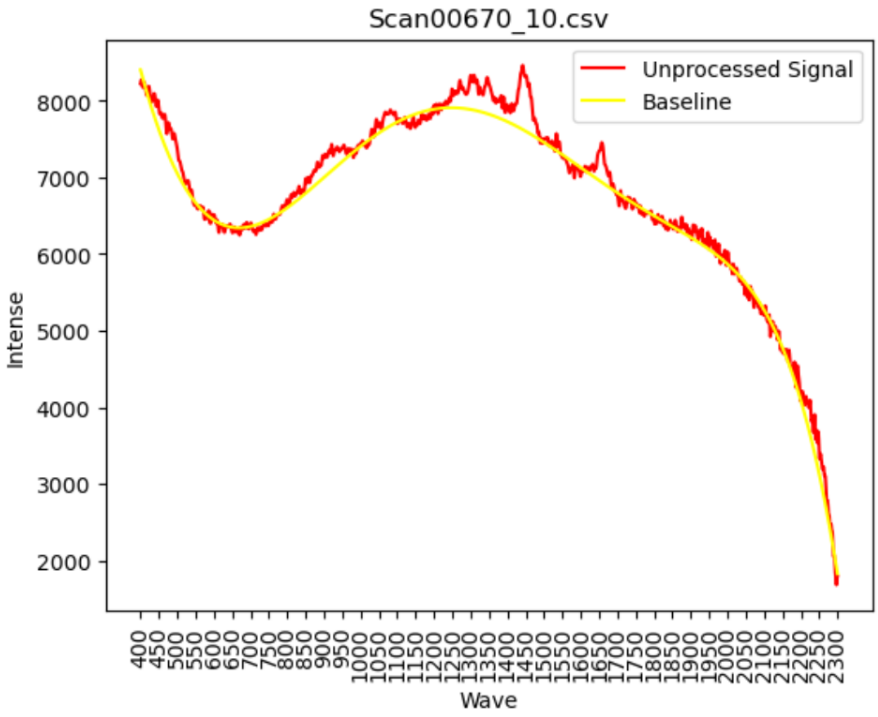

Figure 16: Determine the signal's baseline using polynomial order of 7

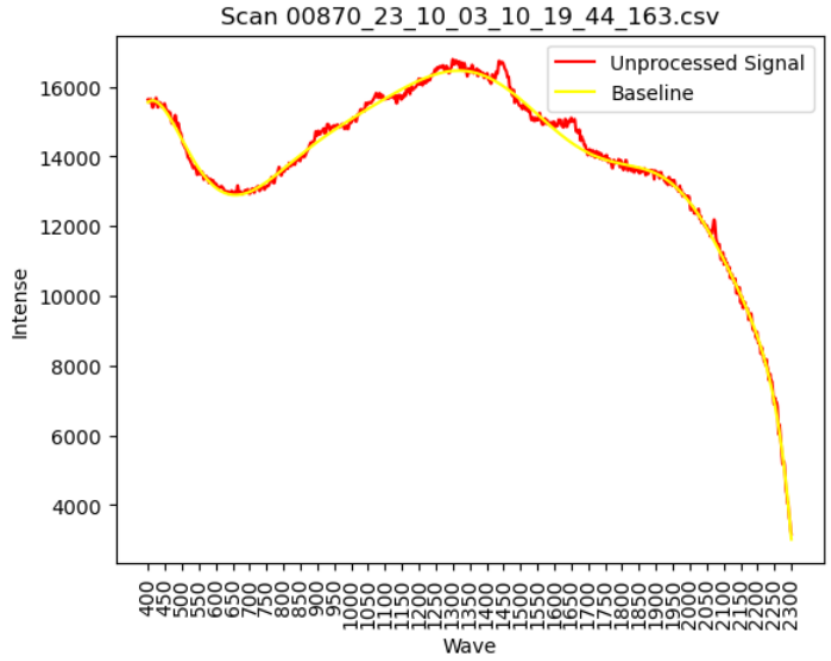

Figure 17: Determine the signal's baseline using polynomial order of 12

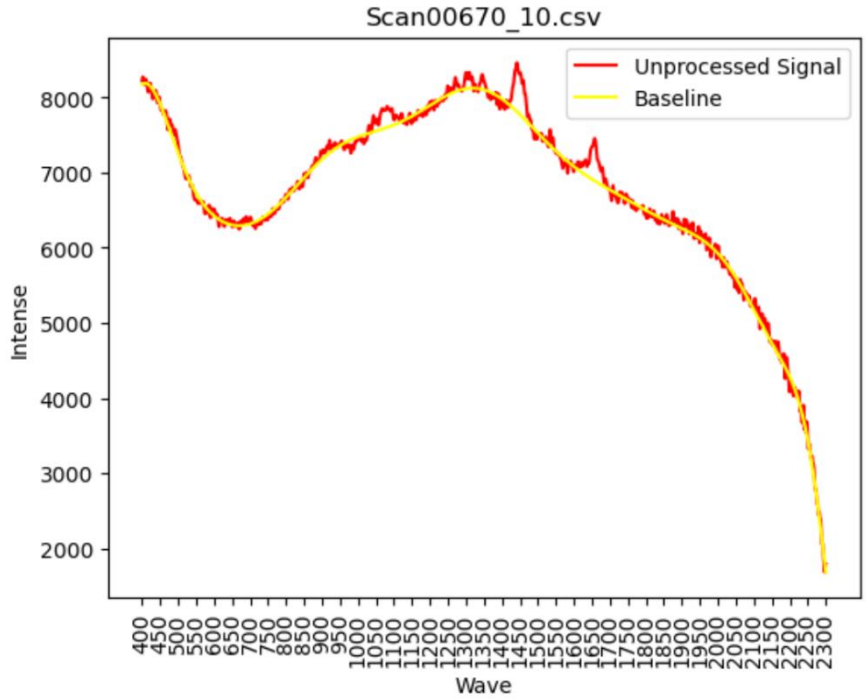

Figure 18: Determine the signal's baseline using polynomial order of 16

Polynomial orders refer to the degree of a polynomial function, which is essentially the highest power of the variable in the equation. In the context of data

analysis, such as signal processing or regression models, choosing the right polynomial order is crucial.

- Lower-order polynomials are simpler and can underfit the data by missing important details in complex patterns.
- Higher-order polynomials are more flexible but can lead to overfitting, capturing noise instead of the actual trend in the data.

Finding the optimal polynomial order often requires experimental testing and visual or statistical evaluation to strike a balance between underfitting and overfitting.

There are noticeable differences in baseline values for various polynomial order coefficients. A polynomial order of 3 tends to underfit, losing important data, whereas a polynomial order of 16 tends to overfit, displaying noisy signals in the baseline. Looking at figure 16 and figure 17 we can see that this is the sweet spot (polynomial order of 7 to 12) for this project, the baseline is not as underfit as when we choose the polynomial order of 3 or being overfitted as when the polynomial order is set to 16. However, this does not imply that a polynomial order of 8 to 12 guarantees the best result in every scenarios. The next step is to employ machine learning to identify the best fit by comparing the processed signal using this baseline with the original data and then assessing the achieved results against the desired outcomes. Based on this comparison, appropriate adjustments can be made.

3.9 Data denoising

After establishing the baseline, the next crucial step in Raman spectral preprocessing is background correction, which involves the subtraction of background noise from the signal intensity. This noise is identified through the baseline and is removed to enhance the clarity of Raman peaks, ensuring accurate spectral analysis. Typically, after background correction, the received signal intensity decreases, since the subtracted background noise often has a positive intensity. The sources of background noise include fluorescence interference, Rayleigh scattering, detector artifacts, and environmental distortions, all of which can significantly impact the accuracy of spectral classification. Removing these unwanted signals allows the Raman spectrum to more effectively represent the true biochemical composition of a sample, making it essential for applications in disease diagnostics and molecular identification.

Beyond background correction, data denoising is another vital preprocessing step that ensures Raman spectral signals remain free from random fluctuations caused

by instrumental limitations or environmental influences. Noise can obscure weak spectral features, making it more challenging to extract meaningful information from biomedical samples. Denoising techniques focus on reducing spectral variability while preserving key peaks that correspond to biomolecular structures. Effective noise removal leads to higher classification accuracy, particularly in machine learning models such as Support Vector Machines (SVM), which rely on precise spectral features for disease identification.

In this study, two advanced preprocessing techniques PenPoly (Polynomial Regression for Baseline Correction) and IMP (Intensity Modulation Processing) were utilized to refine Raman spectral data before classification.

PenPoly applies polynomial fitting techniques to remove low-frequency baseline distortions, ensuring that Raman peaks are preserved while eliminating unwanted background signals. The baseline correction function is represented as:

Polynomial regression effectively removes fluorescence and baseline drift, making spectral signals more uniform. However, selecting the optimal polynomial degree is crucial too low a degree may fail to correct complex variations, while too high a degree can introduce unwanted oscillations (Runge phenomenon).

IMP is used to normalize and standardize spectral intensity, ensuring consistency across different samples and measurement conditions. This technique adjusts signal intensity values, enhancing peak visibility while reducing fluctuations caused by instrument variability. By modulating intensity across a predefined spectral range, IMP improves peak detection accuracy, making Raman spectral data more reliable for machine learning classification models.

Both PenPoly and IMP significantly improve the classification accuracy of SVM applied to Raman spectral data. By removing baseline distortions and enhancing spectral clarity, these techniques ensure that SVM receives high-quality input, leading to better feature extraction and class separation. The results confirm that effective preprocessing enhances AI-driven Raman spectroscopy, enabling more precise glucose monitoring and disease classification.

```python
def RemoveBacKGround(yaxit, xaxit):
    #base_line_data = pbl.polynomial.imodpoly(yaxit, xaxit, poly_order)
    base_line_data = pbl.polynomial.penalized_poly(yaxit, xaxit, poly_order)
    base_line = base_line_data[0]
    result = []
    for i in range(len(base_line)):
        result.append(yaxit[i] - base_line[i])
    return result
```

Figure 19: Background correction function

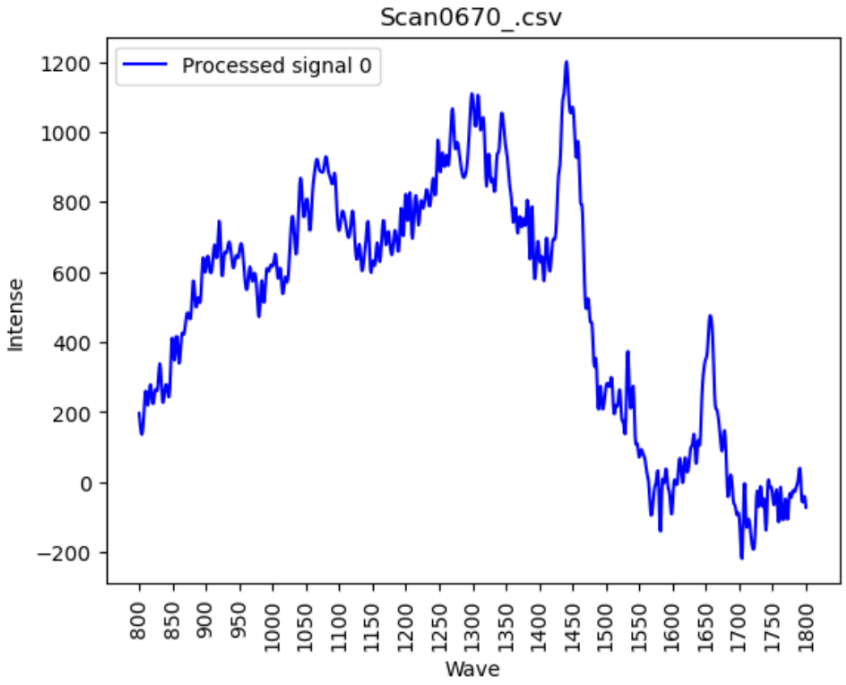

Figure 20: Background correction result with polynomial order order of 3

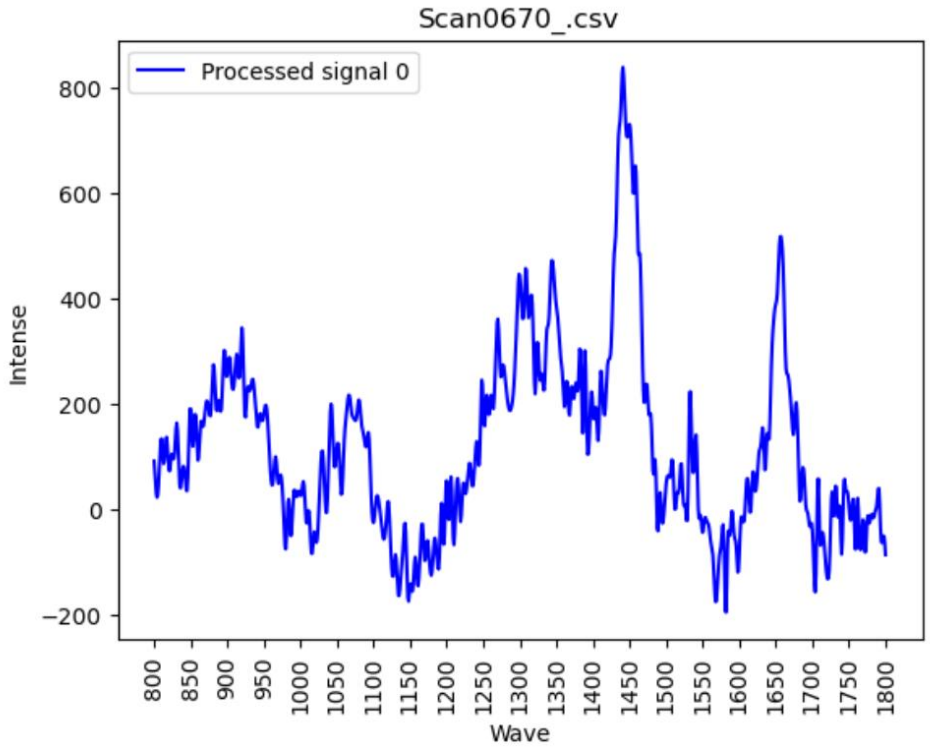

Figure 21: Background correction result with polynomial order order of 7

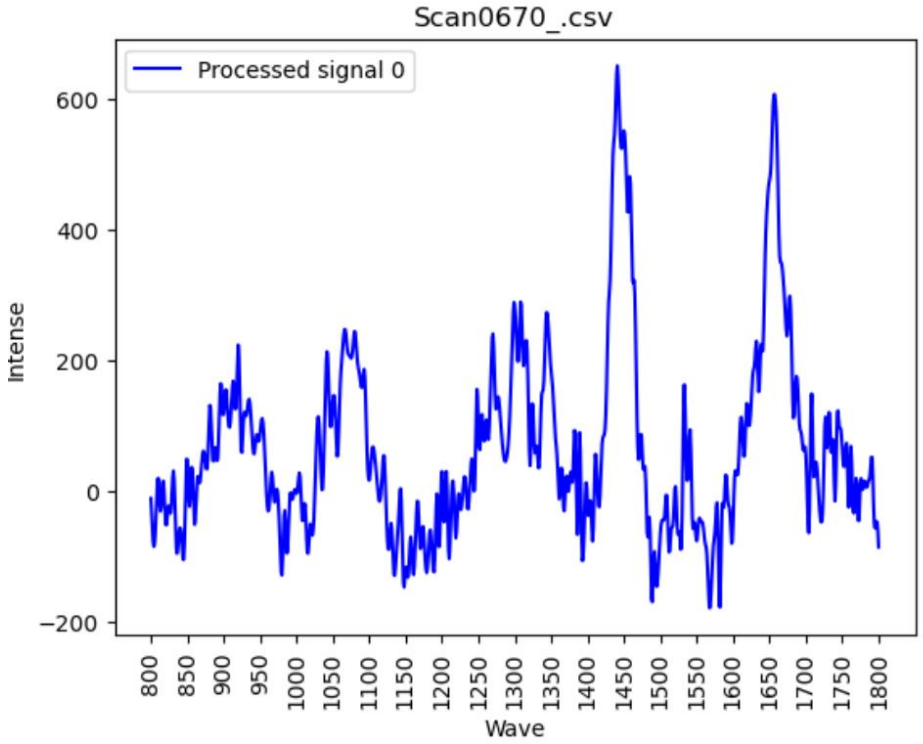

Figure 22: Background correction result with polynomial order order of 12

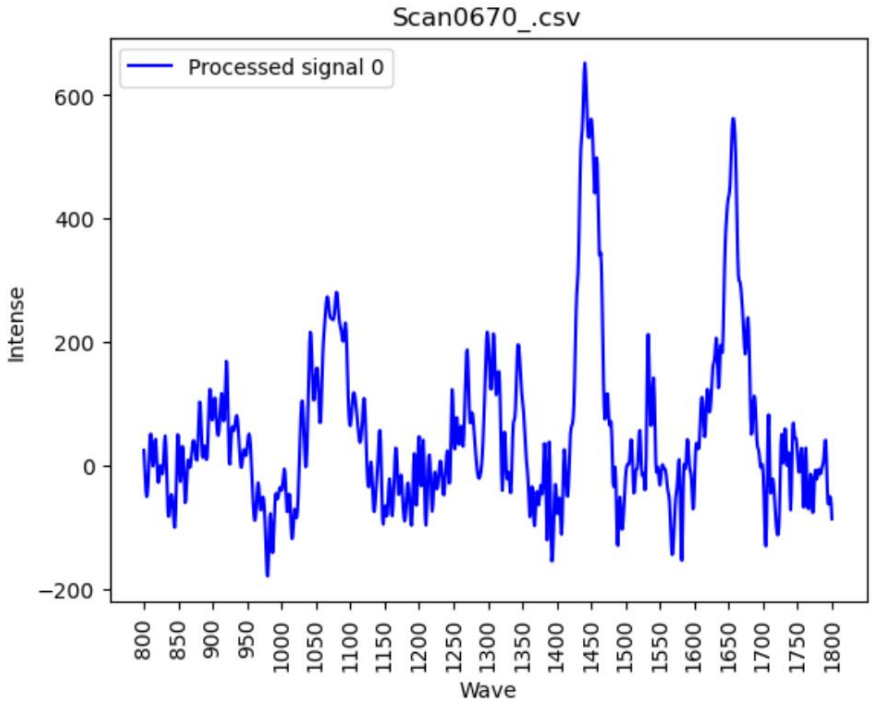

Figure 23: Background correction result with polynomial order order of 16

I focus on x values ranging from 800 cm^{-1} to 1800 cm^{-1} because this range contains important vibrational bands that help analyze key biological molecules, including proteins, lipids, nucleic acids, and carbohydrates. These molecules play a vital role in cellular function, and their unique spectral signatures provide valuable insights into the structure and composition of biological tissues. By focusing on this specific range, I can ensure that the Raman spectroscopy analysis captures chemically relevant signals that are most useful for biomedical applications.

One of the primary advantages of using this range is its relevance to protein analysis, as it includes amide I and amide II bands, which provide information about secondary protein structures like alpha-helices and beta-sheets. These structures are essential for understanding protein folding, which can indicate biological abnormalities such as misfolding diseases. Similarly, lipid-associated vibrations fall within this range, allowing for detailed examination of cell membranes, lipid metabolism, and structural properties that differentiate healthy and diseased tissues.

In addition to proteins and lipids, nucleic acid vibrations from DNA and RNA are also present in this spectral window. These signals offer valuable insights into genetic material interactions, molecular stability, and potential mutations that may contribute to diseases. Additionally, carbohydrates, which play a fundamental role in cell signaling, energy storage, and structural integrity, exhibit strong vibrational

activity within this range, making them useful markers for biochemical differentiation.

Beyond its biological relevance, this range is selected to reduce interference from water, which can distort Raman signals at lower wave numbers. Water molecules strongly absorb light below 800 cm^{-1}, introducing unwanted background noise that can reduce the accuracy of spectral data. By limiting the analysis to 800 cm^{-1} – 1800 cm^{-1}, the study avoids these interference signals, ensuring that only relevant molecular features are examined.

This spectral range is crucial for improving the accuracy and reliability of Raman spectroscopy in biomedical diagnostics. By focusing on well-defined vibrational bands that correspond to biologically significant molecules, the analysis enhances signal clarity, making it easier to classify diseased versus healthy tissues with machine learning models such as Support Vector Machines (SVM). With optimized spectral selection, this method provides precise molecular insights, improving the effectiveness of AI-driven Raman spectroscopy applications in glucose monitoring and disease classification.[47]

3.10 Data shuffle

To ensure the robustness and generalizability of the training model, the data was shuffled prior to training. The shuffle_data function was employed to randomly rearrange the order of the samples and their corresponding labels. This step helps in preventing any potential biases that might arise from the order of the data. By shuffling the data, each training iteration is exposed to a different order of samples, promoting better generalization of the model. The shuffled data was then used as input for the model training process.

```
def shuffle_data(data, label):
    print('Data shape: ', data.shape)
    print('Label shape: ', label.shape)

    old_index = list(range(len(data)))
    new_index = old_index.copy()
    random.shuffle(new_index)
    new_data = [data[i] for i in new_index]
    new_label = [label[i] for i in new_index]

    print('Old order: ', old_index)
    print('Old label: ', label)
    print('Old first data: ', data[0])

    print('New order: ', new_index)
    print('New label: ', new_label)
    print('New first data: ', new_data[0])
    return new_data, new_label, new_index
```

Figure 24: Shuffle_data function

By incorporating this step, we ensure that the training model is trained on a randomly ordered dataset, which contributes to the overall accuracy and reliability of the machine learning model.

Chapter 4: FINDINGS AND DISCUSSIONS

4.1 Results with SVM

4.1.1 SVM configurations

I am setting up the parameters and configuring a machine learning model using Support Vector Machine (SVM). The make_pipeline function is used to create a pipeline that includes standard scaling (StandardScaler()) and the SVM model with the RBF (Radial Basis Function) kernel (SVC(kernel='rbf')). An alternative model, ExtraTreesClassifier, is reserved for later use.

```
#   Thiết lập các tham số cũng như thiết lập mô hình học máy SVM
model = make_pipeline(StandardScaler(), SVC(kernel='rbf'))
#model = ExtraTreesClassifier()
#model = make_pipeline(StandardScaler(), SVC())

max_svm_accuracy_score = 0.0
min_svm_accuracy_score = 1.0

sum_svm_accuracy_score = 0.0
sum_svm_one_vs_rest_auc_ovr = 0.0
sum_svm_one_vs_rest_sensitivity = 0.0
sum_svm_one_vs_rest_specificity = 0.0

#   Phân chia bộ dữ liệu huấn luyện và kiểm thử, đồng thời thiết lập số chu kỳ thí nghiệm là 5 x 30 = 150
rskf = RepeatedStratifiedKFold(n_splits=5, n_repeats=10)
i = 0
for train_index, test_index in rskf.split(data, target):
    i += 1
    print("Turn:", i)
    print("TRAIN dataset have ", len(train_index), " sample include: ", train_index)
    print("TEST dataset have ", len(test_index), " sample include: ", test_index)
    X_train, X_test = data[train_index], data[test_index]
    y_train, y_test = target[train_index], target[test_index]

    # Huấn luyện
    model.fit(X_train, y_train)
    # y_score = model.predict_proba(X_test)

    # Kiểm thử
    svm_prediction = model.predict(X_test)

    accuracy_score = metrics.accuracy_score(svm_prediction, y_test)
    print(f'SVM accuracy = {accuracy_score}')

    # one_vs_rest_sensitivity = MachineLearningCalculator.RvO_sensitivity(svm_prediction, y_test)
    # print(f'Sensitivity = {one_vs_rest_sensitivity}')
    #
    # one_vs_rest_specificity = MachineLearningCalculator.RvO_specificity(svm_prediction, y_test)
    # print(f'Specificity = {one_vs_rest_specificity}')

    # sum_svm_one_vs_rest_sensitivity += one_vs_rest_sensitivity
    # sum_svm_one_vs_rest_specificity += one_vs_rest_specificity
    sum_svm_accuracy_score += accuracy_score
    # sum_svm_one_vs_rest_auc_ovr += macro_roc_auc_ovr

    print("------------------------------------")

# In kết quả
average_svm_accuracy_score = sum_svm_accuracy_score / i
#with open('G:/Thesis/Datasets/test/Logfile/background_remover_SVM_WL_1_2048_result_2.txt', 'a') as file:
 #               file.write(("Average accuracy: "+ str(average_svm_accuracy_score)+ " |turn: "+ str(i)+ " |poly_order: "+ str(poly_order)+ " |start: "+
print("Average SVM accuracy: ", average_svm_accuracy_score, "in number of turn: ", i)
```

Figure 25: SVM configurations

To track the performance metrics of the SVM model, several variables are initialized, including those for maximum, minimum, and average accuracy scores, as well as sensitivity and specificity scores. The cross-validation process is set up using

RepeatedStratifiedKFold, which performs stratified k-fold cross-validation multiple times (in this case, 5 splits repeated 10 times). A counter i is used to keep track of the number of cross-validation iterations.

The code then enters a loop that iterates over each split of the data, updating the training and test indices. For each split, the data is divided into training (X_train, y_train) and testing sets (X_test, y_test). The model is trained using the training data and predictions are made on the test data. The accuracy of the model's predictions is calculated and printed for each iteration. Although additional metrics like sensitivity and specificity can be calculated, those lines are commented out in this code.

Throughout the loop, the accuracy scores are aggregated to calculate the average accuracy score of the model across all cross-validation folds. Finally, the code prints the average SVM accuracy along with the number of iterations. This approach ensures that the model's performance is thoroughly evaluated and helps identify its effectiveness in classifying the data.

4.1.2 Accuracy of the unprocessed data

Firstly, I imported the unprocessed Raman data into a Support Vector Machine (SVM) within Jupyter Notebook to compare how well it processes data with different polynomial orders. The initial accuracy was checked, revealing that unprocessed signals can be challenging to classify, even with two labels.

In this project, I utilized the SVM functionality provided by Scikit-learn, a powerful Python library for machine learning, directly within Jupyter Notebook. I made some modifications to adapt the code to my input data for reading and writing. Scikit-learn offers classes such as Support Vector Classification and Support Vector Regression, which enable me to develop and customize SVM models for classification problems.

```
Turn: 50
TRAIN dataset have  16  sample include:  [ 0  1  2  3  4  5  6  9 10 11 12 15 16 17 18 19]
TEST dataset have  4  sample include:  [ 7  8 13 14]
SVM accuracy = 0.75
-------------------------------------
Average SVM accuracy:  0.52 in number of turn:  50
```

Figure 26: Unprocessed data accuracy results

The table below is the result of the experiments with the unprocessed Raman data after I had run it 5 times:

Run No	Date	Result

1	02/12/2024	49.5%
2	03/12/2024	56%
3	04/12/2024	58%
4	04/122024	56.5%
5	05/12/2024	52%
Average		54.4%

Table 2: unprocessed data SVM results

As we can see, the results ranging from 49.5% to 58%, which is low for a two labels dataset. It indicates that the machine learning model is considered to be not too good at differentiate between these two types of labels based on the current data. This also say that there are room for improvements.

4.1.3 Accuracy of the processed data

When using penalized polynomial regression with a polynomial order of 3, the SVM produced a relatively positive result, indicating that signal processing has a positive impact on accurately predicting 2 labels.

```
Turn: 50
TRAIN dataset have  16  sample include:  [ 0  2  3  5  7  8  9 10 11 13 14 15 16 17 18 19]
TEST dataset have  4  sample include:  [ 1  4  6 12]
SVM accuracy = 1.0
-------------------------------------
Poly order = 3
Average SVM accuracy:  0.65 in number of turn:  50
```

Figure 27: SVM Accuracy result with polynomial order of 3

The accuracy rate increased to approximately 10.5% from an average of 60.5% to 70%

Run No	Date	Result
1	02/12/2024	65.5%
2	03/12/2024	65%
3	04/12/2024	63.5%

4	04/122024	70%
5	05/12/2024	60.5%
Average		64.9%

Table 3: SVM Accuracy results with polynomial order of 3

4.1.4 Adjustments and improvements

After some adjustments such as adjust the polynomial order incrementally, typically in the range of 3 to 16, until I achieve a satisfactory outcome., I found out that setting the polynomial order of 10 yields the highest accuracy. The accuracy rate increased to approximately 77.2% which is 22.8% higher than the unprocessed data and 12,3% higher than when I set the polynomial order equals to 3. Keep in mind that a polynomial order of 10 may not always be optimal. Adjustments should be made based on the specific signal data, with careful observation and evaluation of changes in each dataset. This process ensures that the chosen polynomial order effectively addresses the unique characteristics and noise levels of the data, leading to more accurate baseline corrections and subsequent analyses.

Run No	Date	Result
1	15/12/2024	77.5%
2	16/12/2024	78.5%
3	17/12/2024	76.5%
4	18/122024	77%
5	19/12/2024	76.5%
Average		77.2%

Table 4: SVM Accuracy results with polynomial order of 10

```
Turn: 50
TRAIN dataset have  16  sample include:  [ 0  1  2  3  5  7  9 10 11 12 13 14 15 16 17 18]
TEST dataset have  4  sample include:  [ 4  6  8 19]
SVM accuracy = 0.75
-------------------------------------
Poly order = 10
Average SVM accuracy:  0.77 in number of turn:  50
```

Figure 28: SVM Accuracy result with polynomial order of 10

4.1.5 IMP

For this experiment, I also do some test regarding IMP in exchange for penalized polynomial regression

Here are the results:

Run No	Date	Result
1	15/12/2024	81%
2	16/12/2024	81%
3	17/12/2024	80%
4	18/122024	79%
5	19/12/2024	82%
Average		80.6%

Table 5: SVM Accuracy results with polynomial order of 11

```
Turn: 50
TRAIN dataset have  16  sample include:  [ 1  3  4  5  7  8  9 10 11 12 13 14 15 17 18 19]
TEST dataset have  4  sample include:  [ 0  2  6 16]
SVM accuracy = 0.75
-------------------------------------
Poly order = 11
Average SVM accuracy:  0.81 in number of turn:  50
```

Figure 29: Accuracy result with polynomial order of 11

After some adjustments such as adjust the polynomial order incrementally, typically in the range of 3 to 16, until I achieve a satisfactory outcome., I found out that setting the polynomial order of 11 yields the highest average accuracy of 80.6%. Compare to the accuracy of the penalized polynomial regression, there is a slight increase in the accuracy. Which mean that data denoising have a positive impact to the accuracy of the model. The IMP method have a 3.4% higher accuracy compare to

penalized polynomial regression and a 15.7% higher accuracy compare to the Unprocessed data.

4.2 Results with Extra Trees Classifier

The experiment also used Extra Tree Classifier in addition of SVM to have a comparison with SVM accuracy

4.2.1 Accuracy of the unprocessed data

Apply the same setup with SVM model, but change the training model to Extra Tree Classifier, I got the results shown here:

Run No	Date	Result
1	02/06/2025	63.5%
2	02/06/2025	59%
3	02/06/2025	62%
4	02/06/2025	61.5%
5	02/06/2025	60%
Average		61.2%

Table 6: Extra Tree Classifier Accuracy results with unprocessed data

We can see in the Table 6, it is a slight improvement to the SVM model but it is still low for a two label dataset.

4.2.2 Accuracy with penalized polynomial regression

Using the same processed data with the SVM model, when change to Extra Tree Classifier, we have a different result compare to SVM when using the polynomial order of 3:

Run No	Date	Result
1	02/06/2025	60.5%
2	02/06/2025	62%
3	02/06/2025	59.5%

4	02/06/2025	60%
5	02/06/2025	61%
Average		60.6%

Table 7: Extra Tree Classifier Accuracy results with Penalized Poly order of 3

The accuracy ranging from 59.5% to 62% with the average of 60.6% which shown a slight improvement over the unprocessed data but still lower than when using SVM but even lower than the ExtraTree unprocessed data results. When set to the polynomial order of 10, here are the results:

Run No	Date	Result
1	02/06/2025	75.5%
2	02/06/2025	73.5%
3	02/06/2025	73%
4	02/06/2025	76.5%
5	02/06/2025	76.5%
Average		75%

Table 8: Extra Tree Classifier Accuracy results with Penalized Poly order of 10

This brings a notable improvement compared to Extra Tree Classifier Accuracy results with Penalized Poly order of 3 with a 14.4% bump in accuracy. The accuracy ranging from 73% to 76.5% with the average of 75%. Both the Extra Tree Classifier with polynomial order of 3 and 10 bring a slight decrease in accuracy compare to using SVM. The ExtraTree Classifier with polynomial of 10 br

4.2.3 Adjustments and improvement

When adjusting the polynomial order to find the best result, I found out that with the polynomial order of 13, I was able to achieve the highest accuracy result showing below:

Run No	Date	Result
1	02/06/2025	75.5%
2	02/06/2025	77.5%
3	02/06/2025	76.5%
4	02/06/2025	78%
5	02/06/2025	76.5%
Average		76.8%

Table 9: Extra Tree Classifier Accuracy results with Penalized Poly order of 13

This shown a slight improvement over polynomial 10 and a somewhat equals to the best result using penalized poly with SVM.

4.2.4 Using IMP

As like the SVM cycling through the polynomial order from 3 to 16 reveal that the polynomial order of 9 yield the highest accuracy:

Run No	Date	Result
1	02/06/2025	80%
2	02/06/2025	81%
3	02/06/2025	80.5%
4	02/06/2025	81.5%
5	02/06/2025	81.5%
Average		80.9%

Table 10: Extra Tree Classifier Accuracy results with IMP order of 9

The result is in the margin of error compare to the SVM with IMP method, both have the average around 80-81% but it is still bringing a higher result compare to penalized poly.

4.3 Results with 1D-CNN

In this research, 1D-CNN was also used to have a boarder comparison with SVM and Extra Tree Classifier.

4.3.1 1D-CNN configurations

The script begins by loading raw Raman spectral data and their corresponding labels from CSV files on disk. Each spectrum, originally stored as a row of intensities (shape: n_samples × n_wavenumbers), is converted into a three-dimensional array with shape (n_samples, n_wavenumbers, 1). This extra "channel" dimension is required by Keras's Conv1D layers, which expect inputs in the form (batch, timesteps, channels). By reshaping the data in this way, the script prepares each sample as a one-channel signal that can be convolved along its wavenumber axis.

For each iteration in the 50-fold loop, the code constructs a fresh convolutional network from scratch. The Sequential model starts with three Conv1D layers first with 80 filters, then 40, then 20 each using a kernel size of 5, ReLU activation, and L2 regularization ($\lambda = 0.03$). These layers scan the spectrum for local patterns, such as narrow peaks or shoulders, reusing the same small kernels across all positions to achieve translation invariance. After the final convolution, a Flatten layer collapses the multi-dimensional feature maps into a single vector, which is then passed through a Dense layer of 200 neurons (again with ReLU) and finally through a sigmoid neuron to output a binary class probability.

```
for fold_no in range(0,50):
    model = keras.Sequential(name="model_conv1D")
    model.add(keras.layers.Input(shape=(n_timesteps, n_features)))

    model.add(
        keras.layers.Conv1D(filters=80, kernel_size=5, activation='relu', name="Conv1D_2", kernel_regularizer=l2(0.03)))

    model.add(
        keras.layers.Conv1D(filters=40, kernel_size=5, activation='relu', name="Conv1D_3", kernel_regularizer=l2(0.03)))

    model.add(
        keras.layers.Conv1D(filters=20, kernel_size=5, activation='relu', name="Conv1D_4", kernel_regularizer=l2(0.03)))

    model.add(keras.layers.Flatten())
    model.add(keras.layers.Dense(200, activation='relu', name="Dense_2"))

    model.add(keras.layers.Dense(1, activation='sigmoid', name="Sigmoid"))

    optimizer = Adam(learning_rate=1e-4)

    # Build the optimizer with all trainable variables
    optimizer.build(model.trainable_variables)

    # Compile the model
    model.compile(loss=loss_function,
                  optimizer=optimizer,
                  metrics=['accuracy'])
```

Figure 30: 1D-CNN layers

Once the architecture is defined, the model is compiled with the Adam optimizer (learning rate = 1e-4) and binary cross-entropy loss. To prevent overfitting on this extremely small dataset, training uses a batch size of only two samples, and an EarlyStopping callback monitors the validation loss with a patience of 20 epochs automatically restoring the best weights when the network stops improving. The script also wraps each training run in a try/except block so that, even if the user interrupts execution, the current model weights are saved to disk under a fold-specific filename.

Instead of traditional cross-validation splits, the code simply repeats this train-and-evaluate cycle 50 times on the full dataset teach time with a fresh random initialization and a fresh 20% split off for validation. By logging start and end timestamps for each fold and recording both Keras's internal metrics and externally computed accuracy via scikit-learn, the script quantifies the model's stability and variance under repeated trainings on the same data. This approach offers insight into how sensitive the 1D-CNN's performance is to weight initialization and data shuffling.

```
try:
    # Fit data to model
    history = model.fit(data_reshaped, target,
                        batch_size=batch_size,
                        epochs=no_epochs,
                        verbose = verbosity,
                        validation_split=0.2,
                        callbacks = EarlyStopping(monitor='val_loss',
                                                  patience=20, verbose=verbosity,
                                                  mode='auto',
                                                  restore_best_weights=True))
```

Figure 31: 1D-CNN Fit data to model

Finally, after all folds complete, the script aggregates results by printing the list of per-fold accuracies, computing their mean, and generating summary statistics with pandas. Model.summary() is called to display the network's final layer structure and parameter count. Throughout, prediction arrays, true labels, and timing information are written to log files, ensuring full reproducibility and enabling detailed post-hoc analysis of where and why the network succeeds or fails on different spectral samples.

Model Architecture

- input_shape: (n_wavenumbers, 1) — reshaped Raman spectrum
- kernel_size: **5** — size of each Conv1D filter
- filters: **[80, 40, 20]** — number of convolutional filters per layer
- activation: **ReLU** — applied after each Conv1D and Dense layer
- kernel_regularizer: **L2(0.03)** — regularization penalty to reduce overfitting
- dense_units: **200** — number of neurons in the fully connected layer
- output_units: **1** — sigmoid neuron for binary classification

Training Configuration

- loss_function: **Binary Crossentropy**
- optimizer: **Adam**
- learning_rate: **1e-4**
- batch_size: **2** — chosen for high update frequency on small datasets
- epochs: **300** — maximum training limit
- early_stopping_patience: **20** — stop training if no improvement on validation loss

Validation Strategy

- validation_split: **0.2** — 20% of each fold's data held out during training

- shuffle: **True** — ensures data is randomly distributed before training each fold
- num_folds: **50** — number of full re-trainings for statistical robustness

4.3.2 Accuracy with Unprocessed data

Here is the accuracy of 1D-CNN with the unprocessed data

Run No	Date	Result
1	23/06/2025	57%
2	23/06/2025	65%
3	23/06/2025	63.5%
4	23/06/2025	59%
5	23/06/2025	66.6%
Average		62.22%

Table 11: 1D-CNN Accuracy results with unprocessed data

This shown that 1D-CNN have the highest accuracy in this project when comparing trained model with unprocessed data with the accuracy ranging from 57% to 66.6% and an average of 62.22%. This is a better result compare to SVM and ExtraTree but as shown later in the project, it is still low to when I combine the training model with data processing technique.

Below is the hyperparameter that was choosen for the 1D-CNN training model:

4.4 Results

Below is the table that shown the accuracy of each model with the unprocessed data:

SVM	ExtraTree	1D-CNN
54.4%	61.2%	62.22%

Table 12: Baseline results

Despite all models working with the same input dataset, the 1D-CNN edges ahead in performance. This reflects its strength in learning localized spectral patterns via convolutional filters, even when trained on a small dataset. ExtraTrees performs

better than SVM, likely due to its ability to capture non-linear relationships and its ensemble-based averaging that reduces variance. SVM trails behind, possibly because it struggles with signal noise and limited sample size, especially without optimal kernel tuning or feature transformations.

I have created a table that shows the accuracy of each training models corresponded with the polynomial order of Penalized Poly and IMP, here are the results:

Poly Order	SVM (PenPoly)	SVM (IMP)	Extratree (PenPoly)	Extratree (IMP)	1D-CNN (PenPoly)	1D-CNN (IMP)
3	64.9	65	60.6	62.16	75.6	81.27
4	57.5	72.75	66.25	70.5	77	76.29
5	65.5	73	67	72	79.1	83
6	68.5	72	67.75	69.16	81	75.69
7	71.5	75.5	74.25	71.5	83	86.39
8	69.8	76	74	72.5	79.4	75.79
9	74	77.5	74.5	80.9	76.7	80.7
10	77.2	77	75	79.33	79	79.89
11	76	80.6	75.5	75.66	81	82
12	75.5	79.5	74.25	78.83	81.56	82.6
13	74.5	79.8	76.8	79	78.8	83.7
14	76.2	78.4	75.5	76.16	76.3	75.69
15	74	78.3	72	76.83	78.34	77.98
16	74.7	77.7	73	77.83	78.14	78.55

Table 13: Overal Results

Across polynomial expansions from degree 3 up to 16, I evaluated three learner types support vector machines (SVM), ExtraTrees ensembles, and a one-dimensional convolutional neural network (1D-CNN) each trained on two preprocessing schemes: penalized-polynomial features (“PenPoly”) and imputed features (“IMP”). Charting accuracy versus polynomial order reveals how each model interacts with feature complexity and which combination yields the best performance on the Raman spectra dataset.

Model	Preprocessing	Peak Accuracy(%)	Peak Degree
SVM	PenPoly	77.2	10
SVM	IMP	80.6	11
ExtraTrees	PenPoly	76.8	12
ExtraTrees	IMP	80.9	9
1D-CNN	PenPoly	83	7-8
1D-CNN	IMP	86.39	5

Table 14: Peak Accuracy Results

When comparing the baseline results (unprocessed Raman spectra) to the processed results (after baseline correction and feature engineering), the performance gains are substantial across all three classifiers:

- SVM improved from 54.4% (unprocessed) to 80.6% when trained on imputed data with a polynomial order of 11.
- ExtraTrees rose from 61.2% (unprocessed) to 80.9% at polynomial order 9 under the same imputed preprocessing.
- 1D-CNN advanced from 62.22% (unprocessed) to a peak of 86.39% using imputed data and a 7th-degree polynomial baseline correction.

These improvements highlight just how critical effective preprocessing is in Raman spectral classification. Without baseline correction, the classifiers struggled to separate signal from noise, especially SVM, which relies heavily on feature scaling and noise-free boundaries. In contrast, once baseline trends were corrected and the data enriched through polynomial expansion especially when paired with imputation each model was able to extract more robust and discriminative features.

Notably, the 1D-CNN gained the most from preprocessing, jumping more than 24 percentage points. This reinforces the idea that convolutional architectures benefit greatly from well-aligned, denoised spectral inputs, allowing their filters to detect meaningful local patterns. Overall, the processed pipeline proves essential for unlocking the full classification potential of these models.

SVM models show a gradual rise in accuracy as I increase polynomial degree. SVM with PenPoly features climbs from 64.9 % at degree 3 to about 77.2 % around degree 10, then settles in the mid-70s. Its imputed counterpart (SVM IMP) begins higher and peaks at 80.6 % at degree 11 before dipping slightly. Across all degrees, imputation consistently outperforms the penalized-poly approach by roughly 3–5 points, and the SVM's optimal range appears between degrees 10 and 12.

ExtraTrees exhibits a similar trend but reaches its apex at lower degrees. ExtraTrees PenPoly improves from 60.6 % up to ~76.8 % near degree 12, whereas ExtraTrees IMP surges to 80.9 % at degree 9 before oscillating in the high-70s. Again, imputed features confer a clear advantage, and tree-based learners see diminishing or negative returns once the polynomial order exceeds 10.

The 1D-CNN configurations deliver the strongest results, showing sensitivity to polynomial degree but outperforming both SVMs and ExtraTrees. The CNN trained on PenPoly features starts at 75.6 % for degree 3, peaks at 83 % around degrees 7–8, then tapers off toward the high-70s by degree 16. More notably, the 1D-CNN IMP model achieves its top accuracy 86.39 % at degree 7 and maintains strong performance (around 79–83 %) for higher degrees. This early peak indicates that a moderate polynomial expansion best suits the CNN's capacity without introducing redundant or noisy features.

In direct comparison, 1D-CNN IMP at polynomial degree 7 is the clear winner, surpassing the next best configuration (ExtraTrees IMP at 80.9 %) by over 5 points. Overall model ranking by peak accuracy is: 1D-CNN IMP > 1D-CNN PenPoly ≈ SVM IMP ≈ ExtraTrees IMP > SVM PenPoly > ExtraTrees PenPoly. While SVM and ExtraTrees favor higher orders (8–12), the CNN peaks early (5–7) and can suffer from over-polynomialization beyond that.

Chapter 5: CONCLUSION

5.1 Concluding Remarks

In this research, the primary objective was to establish effective methods for processing noisy Raman signals, particularly focusing on baseline correction using polynomial regression. Different polynomial orders from 3 to 16 were explored to determine their impact on signal processing. The study identified that low orders (e.g. 3) tend to underfit and lose essential spectral detail, while very high orders (e.g. 16) overfit and introduce spurious noise. Crucially, there is no "one-size-fits-all" degree: optimal order must be chosen in light of each signal's characteristics.

Unprocessed Raman data were fed into Support Vector Machines (SVM) and ExtraTrees classifiers to benchmark performance without preprocessing. As expected, raw spectra yielded poor accuracy, underscoring the need for baseline correction and feature engineering. We implemented SVM (linear, polynomial, RBF, sigmoid kernels) and ExtraTrees via scikit-learn in Jupyter Notebook, shuffling data prior to training to eliminate ordering bias.

Baseline-corrected spectra obtained by polynomial regression of orders 3–16 under two schemes (direct L2-penalized "PenPoly" vs. prior missing-value imputation "IMP") were then re-classified. ExtraTrees(PenPoly) improved from an average of 60.6 % at order 3 to 76.8 % at order 12; ExtraTrees(IMP) peaked at 80.9 % at order 9. Likewise, SVM(PenPoly) rose from 64.9 % to 77.2 % (order 10), while SVM(IMP) attained 80.6 % at order 11.

Extending this comparison to a one-dimensional convolutional neural network (1D-CNN) an ANN architecture with Conv1D layers revealed even stronger gains. The CNN(PenPoly) peaked at 83.0 % accuracy around orders 7–8. Most notably, the CNN(IMP) model achieved 86.39 % at 5th-degree expansion the highest accuracy of all configurations before gradually tapering at higher degrees.

Taken together, these results demonstrate:

- **Imputation (IMP)** consistently outperforms direct penalized-poly (PenPoly) preprocessing by 3–5 %.
- **Tree-based and kernel methods** (ExtraTrees, SVM) favor higher polynomial orders (9–12).
- **Convolutional ANNs** extract rich local features with only moderate expansion, peaking early (order 5) and avoiding over-polynomialization.

The research concluded that careful observation and evaluation of changes in each dataset are essential for selecting the most suitable polynomial order, ensuring accurate baseline correction and improved signal classification.

5.2 Limitations

Despite the significant findings and contributions of this research, several limitations should be acknowledged.

Firstly, the quality of the Raman signal data used in this study may vary, and some datasets might contain inherent noise or artifacts. The availability of high-quality, consistent data is crucial for accurate baseline correction and signal classification. Additionally, while the research focused on specific polynomial orders for baseline correction, the findings may not be universally applicable to all types of Raman signals or other spectroscopic data. Further research is needed to validate the results across different datasets and experimental conditions.

Another limitation lies in the choice of polynomial orders. The study explored polynomial orders ranging from 3 to 16 for baseline correction. Although this range was chosen based on preliminary observations, it may not cover all potential polynomial orders that could be effective. A more exhaustive exploration of polynomial orders could yield different insights. The implementation of machine learning models, especially for large datasets, requires substantial computational resources. The limitations in computational power may have constrained the scope and scale of the experiments conducted in this research.

While the study employed machine learning techniques for baseline correction and signal classification, some manual adjustments were still necessary. The reliance on visual observation and manual fine-tuning introduces subjectivity and potential biases. Furthermore, the research utilized specific machine learning models and hyperparameters. The choice of models and their settings could influence the outcomes, and different models or hyperparameter configurations might produce varying results.

Finally, external factors such as environmental noise, experimental setup, and instrument calibration can impact the quality of Raman signal data. These factors were not exhaustively controlled or accounted for in this research, which may affect the reproducibility of the results.

Acknowledging these limitations is essential for understanding the context and boundaries of the findings. It also provides a foundation for future research to build upon and address the identified constraints.

5.3 Recommendations

Future research should focus on collecting a larger and more diverse dataset of Raman signals from various sources and conditions to enhance the robustness of the findings. This expanded dataset will allow for a more comprehensive analysis and help improve the generalizability of the results.

Additionally, investigating other baseline correction methods, such as wavelet transforms or adaptive smoothing, and comparing their effectiveness with polynomial regression is crucial. Exploring these alternative methods can provide insights into their relative strengths and weaknesses, leading to improved signal processing techniques.

Advanced machine learning techniques, such as deep learning or ensemble methods, should also be explored to enhance signal classification accuracy. These techniques have the potential to capture complex patterns in the data and improve the overall performance of the classification models.

Implementing automated hyperparameter optimization techniques, like grid search or Bayesian optimization, is another important recommendation. These optimization methods can help identify the best model configurations, leading to more efficient and accurate model training.

Ensuring controlled experimental setups to account for external factors like noise and temperature is essential for improving data quality. By controlling these environmental factors, researchers can obtain more reliable and reproducible results.

Finally, focusing on creating real-time signal processing algorithms for immediate analysis and feedback is crucial. Developing real-time systems will enable researchers to process and analyze signals on the fly, providing timely insights and enhancing the practical applications of the research.

REFERENCES

1. World Health Organization, n.d. Diabetes. [Online]
   Available at: https://www.who.int/news-room/fact-sheets/detail/diabetes#:~:text=In%202019%2C%20diabetes%20was%20the,of%20cardiovascular%20deaths%20(1)
2. Jack R. Leonards, P. R. D., 1962. ELECTRODE SYSTEMS FOR CONTINUOUS MONITORING IN CARDIOVASCULAR SURGERY. Annals of the New York Academy of Sciences, 102(1), p. 29–45.
3. Narkhede P, Dhalwar S, Karthikeyan B (2016) Nir based noninvasive blood glucose measurement. Indian J Sci Technol 9. https://indjst.org/articles/nir-based-non-invasive-blood-glucose-measurement
4. Pande MC, Joshi A (2015) Non-invasive blood glucose measurement. Int J Comput Eng Res 5:26–28
5. Guo D, Zhang D, Zhang L, Lu G (2012) Non-invasive blood glucose monitoring for diabetics by means of breath signal analysis. Sensors Actuators B Chem 173:106–113
6. Shaker G, Smith K, Omer AE, Liu S, Csech C, Wadhwa U, SafaviNaeini S, Hughson R (2018) Non-invasive monitoring of glucose level changes utilizing a mm-wave radar system. Int J Mob Hum Comput Interact (IJMHCI) 10:10–29
7. Caduff A, Hirt E, Feldman Y, Ali Z, Heinemann L (2003) First human experiments with a novel non-invasive, non-optical continuous glucose monitoring system. Biosens Bioelectron 19:209–217
8. Briganti.G, L. M., 2020. Artificial intelligence in medicine: today and tomorrow. Front medicine, Volume 7, pp. 7-27.
9. Cowie CC, C. S. M. A. e. a., 2018. Diabetes in America. 3rd edition. August: NIDDK.
10. American Diabetes Association, n.d. Understanding Type 1 Diabetes. [Online]
    Available at: https://diabetes.org/about-diabetes/type-1?form=MG0AV3
    [Accessed 2024].
11. WHO, 2024. Diabetes. [Online]
    Available at: https://www.who.int/news-room/fact-sheets/detail/diabetes?form=MG0AV3
    [Accessed 2024].
12. Bessesen, D., 2023. What Is Diabetes?. [Online]
    Available at: https://www.niddk.nih.gov/health-information/diabetes/overview/what-is-diabetes
    [Accessed 2024].
13. WHO, 2011. Diabetes fact sheet no. 312. 2011. s.l.:WHO.

14. Simon D, C. M. T. N. S. C. E. E., 1985. Com- parison of glycosylated hemoglobin and fasting plasma glucose with two-hour post-load plasma glucose in the detection of diabetes mellitus. Am Journal Epidemiology, pp. 589-593.
15. IE, C., 2011. International expert committee report on the role of the a1c assay in the diagnosis of diabetes. Diabetes Care, pp. 1327-1334.
16. Sacks, D. B., Kirkman, M. S. & Little, R. R., 2024. Point-of-Care HbA1c in Clinical Practice: Caveats and Considerations for Optimal Use. Diabetes Care, 47(7), pp. 1104-1110.
17. Whitley, H. P., Yong, E. V. & Rasinen, C., 2015. Selecting an A1C Point-of-Care Instrument. Diabetes Spectrum, 28(3), pp. 201-208.
18. Schaffert, L.-N.et al., 2016. Point-of-care HbA1c tests - diagnosis of diabetes, United Kingdom: IHR Community Healthcare MIC.
19. Manzella, D., 2024. What Is the Fasting Plasma Glucose Test?. [Online] Available at: https://www.verywellhealth.com/understanding-the-fasting-plasma-glucose-test-1087680 [Accessed 2025].
20. Anon., n.d. Glucose tolerance test. [Online] Available at: https://www.mayoclinic.org/tests-procedures/glucose-tolerance-test/about/pac-20394296?form=MG0AV3 [Accessed 2025].
21. Close, K., 2023. Oral Glucose Tolerance Test: Uses and Results. [Online] Available at: https://www.verywellhealth.com/the-oral-glucose-tolerance-test-1087684 [Accessed 2024].
22. NHS, 2024. Continuous glucose monitoring and hybrid closed loop for diabetes. [Online] Available at: https://www.nhs.uk/conditions/cgm-and-hcl-for-diabetes/?form=MG0AV3 [Accessed 2024].
23. Hadhazy, A., 2024. Can AI Improve Medical Diagnostic Accuracy?. [Online] Available at: https://hai.stanford.edu/news/can-ai-improve-medical-diagnostic-accuracy?form=MG0AV3 [Accessed 2024].
24. Guevara E et al (2022) Feasibility of Raman spectroscopy as a potential in vivo tool to screen for pre-diabetes and diabetes. J Biophotonics 15:9
25. Ho, B., 2021. Kernel methods and support vector machines. Japan. [Online] Available at: https://dokumen.tips/documents/kernel-methods-and-support-

vector-baoviasmsmllecturel3-kernel-linearsupport.html?page=1
[Accessed 2024].
26. Pierre Geurts, D. E. L. W., 2006. Extremely randomized trees. Machine Learning, 63(1), pp. 3-42.
27. Nomura, A., Noguchi, M., Kometani, M. et al, 2021. Artificial Intelligence in Current Diabetes Management and Prediction. [Online]
Available at: https://link.springer.com/article/10.1007/s11892-021-01423-2?form=MG0AV3
[Accessed 2024].
28. SpectralAI, 2024. Artificial Intelligence in Medical Diagnosis: How Medical Diagnostics are Improving through AI. [Online]
Available at: https://www.spectral-ai.com/blog/artificial-intelligence-in-medical-diagnosis-how-medical-diagnostics-are-improving-through-ai/?form=MG0AV3
[Accessed 2024].
29. Jupyter, 2024. Project Jupyter Documentation. [Online]
Available at: https://docs.jupyter.org/en/latest/?form=MG0AV3
30. Gonzale, W. V., Mobashsher, A. T. & Abbosh, A., 2019. The Progress of Glucose Monitoring-A Review of Invasive to Minimally and Non-Invasive Techniques, Devices and Sensors. Sensors.
31. Tang, L., Chen, S. J. C. C.-J. & Liu, J.-T., 2020. Non-Invasive Blood Glucose Monitoring Technology: A Review. [Online]
Available at: https://pmc.ncbi.nlm.nih.gov/articles/PMC7731259/
[Accessed 2024].
32. Krishnan, S., Vinupritha, H. & Kathirvelu, D., 2020. Non-invasive glucose monitoring using machine learning. 2020 International Conference on Communication and Signal Processing (ICCSP). IEEE.
33. E, G., 2022. Feasibility of Raman spectroscopy as a potential in vivo tool to screen for pre-diabetes and diabetes.. J Biophotonics 15:9.
34. JHB, I., 2022. Prevalence of diabetic macular edema based on optical coherence tomography in people with diabetes: A systematic review and meta-analysis. Surv Ophthalmol, 67(4).
35. A, Y., 2022. Glucose Content Analysis using Image Processing and Machine Learning Techniques. International Conference on Information and Communications, Issue 5.
36. M, S., DP, C., RC, R. & S, Q., 2021. Non- invasive glucose monitoring using optical sensor and machine learning techniques for diabetes applications. IEEE.
37. Podgornyy, A., 2025. Five AI Innovations That Will Redefine Healthcare In 2025. [Online]
Available at:

https://www.forbes.com/councils/forbestechcouncil/2025/02/28/five-ai-innovations-that-will-redefine-healthcare-in-2025/?form=MG0AV3&form=MG0AV3
[Accessed 2025].
38. Aayush, 2025. AI in Healthcare 2025: Real-World Data, Impact Analysis, and Future Trends. [Online]
Available at: https://www.alltechnerd.com/ai-in-healthcare-2025-real-world-data/?form=MG0AV3&form=MG0AV3
[Accessed 2025].
39. Dadoo, J. K., 2025. Can AI Cheat Death? Future Of Healthcare In Algorithmic World. [Online]
Available at: https://www.businessworld.in/article/can-ai-cheat-death-future-of-healthcare-in-algorithmic-world-549084?form=MG0AV3&form=MG0AV3
[Accessed 2025].
40. Raman CV, Krishnan KS (1928) A new type of secondary radiation. Nature 121:501–502
41. Das, R. S. & Agrawal, Y. K, 2011. Raman spectroscopy: Recent advancements, techniques and applications. Vibrational Spectroscopy, 57(2), pp. 163-176.
42. Willard, et al., 1988. Instrumental methods of analysis. Issue 7.
43. OG, R. et al., 2015. Fluorescence modeling for optimized-binary compressive de- tection raman spectroscopy. Opt Express , Issue 23, p. 23935–23951.
44. J, S. et al., 2012. In vivo blood.
45. E, G. et al., 2018. Use of raman spectroscopy to screen diabetes mellitus with machine learning tools. Biomed Opt.
46. Jianhua Zhao, Harvey Lui, David I. MCLean, Haishan Zeng, 2007. Automated Autofluorescence Background Subtraction Algorithm. The Laboratory for Advanced Medical Photonics (LAMP), Department of Dermatology and Skin Science, University of British Columbia: s.n.
47. Valentina Teodolinda Vincoli, A. C. G. A. B. M. B., 2022. Electrospun Silk Fibroin Scaffolds for Tissue Regeneration: Chemical, Structural, and Toxicological Implications of the Formic Acid-Silk Fibroin Interaction, s.l.: Frontiers in Bioengineering and Biotechnology.
48. Do Cong Tuan, 2024. Digital signal processing combined with machine learning in diabetes diagnosis
49. Chen, X., Wu, X., Chen, C. et al, 2023. Raman spectroscopy combined with a support vector machine algorithm as a diagnostic technique for primary Sjögren’s syndrome. [Online]

Available at: https://doi.org/10.1038/s41598-023-29943-9
[Accessed 2025].
50. Yan, L., Su, H., Liu, J. et al., 2024. Rapid detection of lung cancer based on serum Raman spectroscopy and a support vector machine: a case-control study. [Online]
Available at: https://doi.org/10.1186/s12885-024-12578-y
[Accessed 2025].
51. dos Santos, D.P., Sena, M.M., Almeida, M.R. et al., 2023. Unraveling surface-enhanced Raman spectroscopy results through chemometrics and machine learning: principles, progress, and trends. [Online]
Available at: https://doi.org/10.1007/s00216-023-04620-y
[Accessed 2025].
52. Kiranyaz, S. et al., 2021. 1D convolutional neural networks and applications: A survey. [Online]
Available at: https://doi.org/10.1016/j.ymssp.2020.107398
[Accessed 2025].
53. Cacciari, I. & Ranfagni, A., 2024. Hands-On Fundamentals of 1D Convolutional Neural Networks A Tutorial for Beginner Users. [Online]
Available at: https://doi.org/10.3390/app14188500
[Accessed 2025].
54. Kiranyaz, S., Ince, T. & Gabbouj, M., 2016. Real-Time Patient-Specific ECG Classification by 1-D Convolutional Neural Networks. [Online]
Available at: 10.1109/TBME.2015.2468589
[Accessed 2025].