Cairo University
Faculty of Computers & Artificial Intelligence
Operations Research & Decision Support Dept.

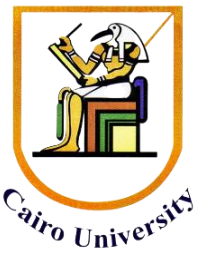
Non-Alcoholic Fatty Liver Disease (NAFLD) Prediction

The Graduation Project Submitted to
The Faculty of Computers and Artificial Intelligence,
Cairo University
In Partial Fulfillment of the Requirements
for the Bachelor Degree

In
Operations Research and Decision Support

Under Supervision of:
Prof. Dr. Tarek H. M. Abou-El-Enien

CAIRO UNIVERSITY
FEBRUARY/2025



NON-ALCOHOLIC FATTY LIVER DISEASE (NAFLD) PREDICTION

Ahmed Mohamed Mageed El-Zeedy	**20200046**
Khalid Ehab Shoukry Ibrahim	**20200167**
Abdalrahman Sameh Mohamed Abdelgawad	**20200286**
Ziad Ayman Mohamed Mohamed	**20200198**
Ziad Nasser Amin Ismael	**20200210**

Supervised By

Prof. Dr. Tarek H. M. Abou-El-Enien

CAIRO UNIVERSITY
FEBRUARY, 2025

ABSTRACT

Non-alcoholic fatty liver disease (NAFLD) is a common liver disorder characterized by fat accumulation in hepatocytes among individuals with minimal or no alcohol consumption. It is closely linked to metabolic syndrome, obesity, type 2 diabetes, and dyslipidemia. NAFLD ranges from simple steatosis to non-alcoholic steatohepatitis (NASH), which can progress to fibrosis, cirrhosis, and even liver cancer. The disease is largely asymptomatic in its early stages, making early detection crucial for effective management.

To address this, a predictive application was developed using machine learning algorithms to estimate NAFLD risk. The system enables users, including healthcare professionals and patients, to input relevant health data for risk assessment. Various algorithms were explored to optimize predictive accuracy, which are: k-Neighbors Classifier, Support Vector Machine, Random Forest, XGBoost and Bayesian Network. The methodology involved data preprocessing, feature selection, model training, and cross-validated hyperparameter tuning to ensure robust performance.

The system was tested using multiple evaluation metrics, confirming high prediction accuracy. The developed tool provides a user-friendly interface for seamless data input and analysis. By facilitating early diagnosis and intervention, this predictive model has the potential to reduce the burden of NAFLD-related complications and improve patient outcomes.

DECLARATION

We hereby declare that our dissertation is entirely our work and genuine / original. We understand that in case of discovery of any PLAGIARISM at any stage, our group will be assigned an F (FAIL) grade and it may result in withdrawal of our Bachelor's degree.

Group members:

Name	**Signature**
Ahmed Mohamed Mageed El-Zeedy	_____
Khalid Ehab Shoukry Ibrahim	_____
Abd Al-Rahman Sameh Mohamed	_____
Ziad Ayman Mohamed	_____
Ziad Nasser Amin Ismael	_____

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.0 Introduction

Non-alcoholic fatty liver disease (NAFLD) is a common condition characterized by the accumulation of fat in the liver of individuals who consume little to no alcohol. It encompasses a spectrum of liver conditions ranging from simple steatosis to non-alcoholic steatohepatitis (NASH), which can lead to fibrosis, cirrhosis, and even hepatocellular carcinoma.

1.0.1 Importance of Prediction

Predicting NAFLD is crucial for early intervention and management. Early detection can prevent disease progression and associated complications. The prediction models often utilize a combination of clinical, biochemical, and imaging data to assess the risk of developing NAFLD.

1.0.2 Factors Influencing NAFLD Prediction

- **Demographic Factors**: Age, sex, and ethnicity can influence the prevalence and risk factors associated with NAFLD.
- **Metabolic Factors**: Conditions such as obesity, type 2 diabetes, and dyslipidemia are significant predictors.
- **Lifestyle Factors**: Diet, physical activity, and sedentary behavior play critical roles in liver fat accumulation.
- **Genetic Predisposition**: Certain genetic variations may increase the risk of developing NAFLD.

1.0.3 Methods of Prediction

- **Clinical Assessment**: Evaluating symptoms, medical history, and physical examination findings.
- **Biochemical Tests**: Liver function tests, lipid profiles, and insulin resistance markers.
- **Imaging Techniques**: Ultrasound, CT scans, and MRI can help visualize fat accumulation in the liver.
- **Machine Learning Models**: Advanced algorithms analyze large datasets to identify patterns and predict NAFLD risk.

1.1 Problem domain

The problem domain of Non-Alcoholic Fatty Liver Disease (NAFLD) prediction encompasses various aspects of the disease, from understanding its etiology to applying predictive models. Here's a detailed overview:

1.1.1 Epidemiology and Public Health Impact

- **Prevalence**: NAFLD is a leading cause of chronic liver disease globally, affecting a significant percentage of the population, particularly in developed countries.

- **Health Burden**: The increasing prevalence of obesity and metabolic syndrome contributes to a rise in NAFLD cases, leading to complications such as liver cirrhosis and cancer.

1.1.2 Pathophysiology

- **Fat Accumulation**: The primary issue is the excessive fat accumulation in liver cells, which can trigger inflammation and cellular damage.
- **Metabolic Dysfunction**: NAFLD is closely associated with metabolic syndrome, insulin resistance, and dyslipidemia, complicating its management.

1.1.3 Risk Factor Identification

- **Demographics**: Age, gender, and ethnicity influence the risk of developing NAFLD.
- **Lifestyle Factors**: Diet, physical inactivity, and alcohol consumption play pivotal roles.
- **Comorbidities**: Conditions such as diabetes, hypertension, and dyslipidemia need to be considered in risk assessments.

1.2 Problem Statement

Non-Alcoholic Fatty Liver Disease (NAFLD) has become a significant public health issue globally, with an estimated 25% of the population affected. This condition, characterized by the accumulation of fat in the liver in individuals who consume little to no alcohol, is often asymptomatic in its early stages. If left untreated, NAFLD can progress to more severe liver diseases, such as non-alcoholic steatohepatitis (NASH), cirrhosis, and even liver cancer. Despite its growing prevalence, NAFLD remains underdiagnosed, primarily due to the lack of early detection methods that are both cost-effective and non-invasive.

Steatosis, often known as simple fatty liver, is the initial stage.

Even if there isn't any inflammation or damage yet, this happens as the liver cells begin to accumulate fat. Many people are unaware that they have a fatty liver because there are frequently no symptoms in this early stage. Many people do not develop fatty liver, and the excess fat in liver cells can be decreased with regular exercise and a good diet. Non-alcoholic steatohepatitis, or NASH, is estimated to develop in 20% of individuals with simple fatty liver.

Non-alcoholic steatohepatitis (NASH) is the second stage of NAFLD.

This stage happens when inflammation and fat accumulation in the liver cells coexist. Up to 5% of the UK population, or one in every twenty, is believed to be affected by this stage. When the liver repairs damaged tissue, inflammation happens. The liver may eventually find it difficult to heal the injured tissue quickly enough, and the inflammatory tissue may leave a scar. Fibrosis is the term for the onset of scar tissue.

Fibrosis is the third stage of NAFLD.

This happens when the liver and the blood arteries around it have chronic scar tissue. At this point, the liver can still operate rather effectively, and addressing the underlying source of the inflammation may stop the damage from getting worse or even reverse part of it. The liver's ability to function is impacted, though, if scar tissue gradually begins to replace a large portion of the healthy liver tissue. Cirrhosis may result from this.

Cirrhosis is the fourth stage of NAFLD.

At this point, symptoms including yellowing of the skin and whites of the eyes and a dull aching in the lower right side of the ribs begin to show up as the liver ceases functioning normally. Although it is challenging to eliminate the scar tissue in liver cirrhosis, removing the cause of the liver damage can stop the disease's growth.

Traditional diagnostic techniques for NAFLD, such as liver biopsies and imaging methods, although accurate, come with substantial drawbacks. Liver biopsies are invasive, expensive, and carry a risk of complications, while imaging methods, such as ultrasound or MRI, may not always provide definitive results in early-stage NAFLD. These limitations hinder the ability to detect the disease early and initiate timely interventions that could prevent disease progression. Additionally, the reliance on such methods in clinical settings contributes to the high cost and resource burden associated with managing NAFLD.

Given these challenges, there is an urgent need for more accessible, non-invasive, and cost-effective solutions for detecting and monitoring NAFLD. The use of machine learning (ML) techniques presents a promising avenue for the development of predictive models that can accurately identify individuals at risk of developing NAFLD before it progresses to more severe stages. Machine learning algorithms, particularly supervised learning methods, can leverage a variety of data sources—such as demographic information, lab results, lifestyle factors, and medical history—to identify patterns and predict the likelihood of NAFLD. By utilizing such predictive models, healthcare providers could identify at-risk individuals early, enabling timely interventions and potentially reducing the burden of NAFLD-related complications.

In conclusion, the development of a reliable, non-invasive, and efficient machine learning-based model to predict NAFLD risk is essential in addressing the growing prevalence of the disease and its associated complications. Such a model could revolutionize the early detection and management of NAFLD, reducing both the healthcare burden and the risk of progression to more severe liver diseases

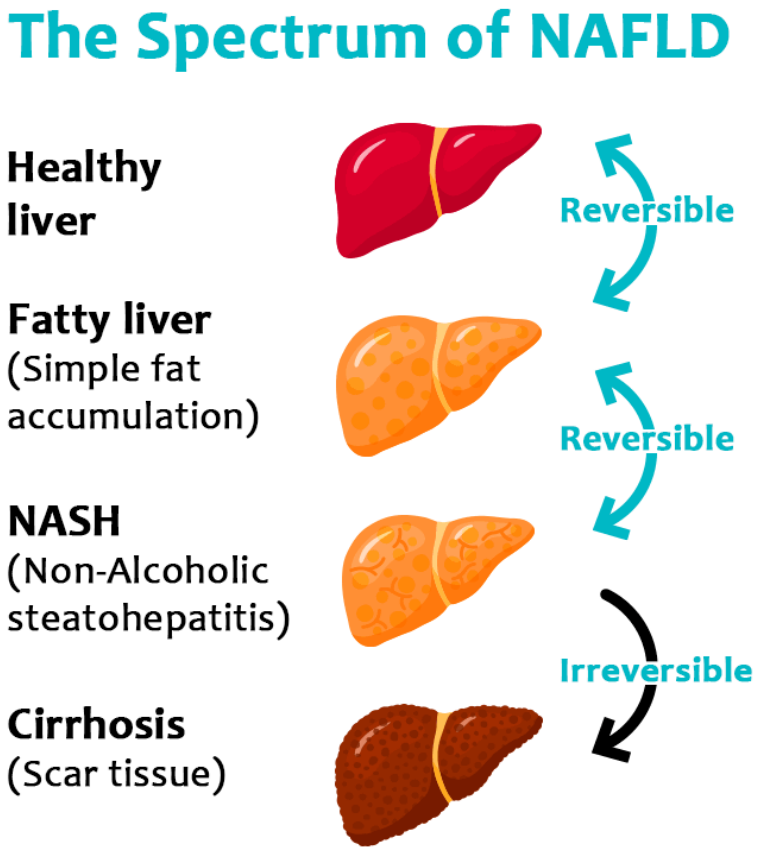


Figure 1 Stages of NAFLD

1.3 Proposed System

The proposed system aims to develop a machine learning-based predictive model for NAFLD diagnosis. It will use clinical, demographic, metabolic, and lifestyle data to estimate the probability of NAFLD in an individual. The system will feature a user-friendly interface where healthcare professionals can input patient data and receive an instant risk assessment.

To enhance the accuracy and robustness of the model, advanced machine learning algorithms will be employed. Techniques such as logistic regression, decision trees, random forests, and support vector machines (SVM) will be explored to find the most effective model for NAFLD prediction. These algorithms will be trained on historical data from diverse patient populations, allowing the system to generalize well to new, unseen cases. Feature engineering and selection will be used to optimize the model's performance, ensuring that only the most relevant variables contribute to the final prediction.

The system will provide healthcare professionals with an intuitive, easy-to-use interface, allowing them to input patient information quickly and receive immediate feedback. This interface will be designed to be minimalistic and user-friendly, with dropdown menus, checkboxes, and simple input fields to capture the necessary data points. Once the data is entered, the system will analyze the inputs in real-time and generate a risk score, which will indicate whether the patient is at low, moderate, or high risk for developing NAFLD. The

results will be accompanied by recommendations for next steps, such as further diagnostic testing or lifestyle interventions.

In addition to risk scoring, the system will provide interpretability features to ensure healthcare professionals can understand the factors contributing to the risk assessment. For example, the model could highlight the most influential variables that led to the prediction, such as high BMI or elevated blood glucose levels. This transparency will increase trust in the system, allowing medical practitioners to make informed decisions based on the results.

Furthermore, the system will be designed for scalability and adaptability. It will be able to integrate with existing electronic health record (EHR) systems, facilitating seamless data exchange and real-time updates. This integration will ensure that the model's predictions are based on the most up-to-date patient information. As new data becomes available or as research on NAFLD evolves, the model will be regularly updated and retrained to incorporate the latest findings. This continuous learning approach will ensure the system stays relevant and effective as new diagnostic techniques and risk factors emerge.

Privacy and data security will be top priorities in the development of the system. Given the sensitivity of patient data, robust encryption methods will be implemented to protect the data during transmission and storage. The system will also comply with relevant data protection regulations, such as HIPAA or GDPR, ensuring that patient information remains confidential and secure. Additionally, the system will provide user authentication and role-based access controls, allowing only authorized healthcare professionals to access and input patient data.

In summary, the proposed machine learning-based system for NAFLD diagnosis aims to provide an accessible, efficient, and accurate tool for early detection of this prevalent liver disease. By leveraging a wide range of patient data and cutting-edge machine learning techniques, the system will help healthcare providers identify individuals at risk, make informed clinical decisions, and implement timely interventions. This approach has the potential to reduce the overall healthcare burden of NAFLD and improve patient outcomes by enabling earlier detection and more personalized treatment plans.

1.3.1 Aims and Objectives

Aims:

- To develop an AI-based predictive model for early detection of NAFLD.
- To create a user-friendly interface for healthcare professionals.

Objectives:

- Collect and preprocess clinical datasets related to NAFLD.
- Train and validate machine learning models using state-of-the-art algorithms.
- Integrate explainable AI techniques to enhance model interpretability.
- Implement the model into a web-based or desktop application for clinical use.

1.3.2 Proposed System Features

- Data Collection & Processing: Use clinical and demographic data for training the model.
- Machine Learning Model: Train and validate various algorithms like Random Forest, XGBoost, and Deep Learning.
- Feature Selection & Engineering: Identify the most relevant predictors of NAFLD.
- User Interface: Provide a simple input-output system for risk prediction.
- Visualization & Reporting: Generate reports for healthcare professionals.
- Data Security & Compliance: Ensure adherence to medical data privacy regulations.

1.4 Development Methodology

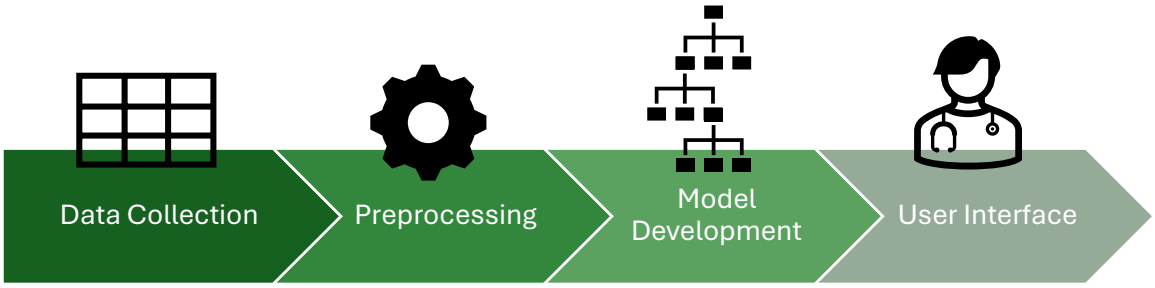


Figure 2 Development Methodology

1.4.1 Data Collection

Using a dataset published at Dryad, collected at Murakami Memorial Hospital during health checkups programs.

1.4.2 Data Preprocessing

- Basic Feature Selection: removing features that are obviously redundent and also removing one feature from every pair of features with very strong correlation.
- Startified Splitting: sampling while preserving real class distribution.

1.4.3 Model Development

- Data Preprocessing
  - Undersampling: performed to balance the training data based on output variable's classes.
  - Feature Scaling: performed to prevent bias as a result of features having different intervals.
  - Feature Projection: projecting the dataset using FAMD and NMF techniques.
- Cross–Validated Hyperparameter Tuning
- Wrapper Feature Selection

1.4.4 Building User Interface

- Building an interface allows the user to enter all information required and, based on entered information, predict the probability of being infected.

1.5 Resource Requirement

1.5.1 Human Resources

- Data Scientists/AI Specialists
  - Expertise in machine learning, feature engineering, and algorithm implementation.
  - Familiarity with healthcare datasets and predictive modeling.

- Domain Experts (Hepatology/Healthcare)
    - Insights into NAFLD pathophysiology and clinical data interpretation.
- Software Developers
    - To build the interface and integrate the machine learning model into a user-friendly application.
- Project Manager
    - To coordinate the project activities, manage timelines, and communicate with stakeholders.
- Research Assistants
    - To assist in literature reviews and data collection.

1.5.2 Hardware Resources

- Computational Infrastructure
    - High-performance workstations or cloud-based computational resources (e.g., AWS, Google Cloud, Azure) for training machine learning models.
    - GPUs for accelerating computations in algorithms like Random Forest, XGBoost, and deep learning models (if explored).
- Storage
    - Reliable and secure storage systems for the dataset and results.
    - Backup solutions for data integrity.

1.5.3 Software and Tools

- Programming Languages
    - Python or R for model development and statistical analysis.
- Integrated Development Environments (IDEs)
    - IntelliJ IDEA, Jupyter Notebook, or PyCharm for coding and debugging.
- Machine Learning Libraries
    - Scikit-learn, TensorFlow, PyTorch, or XGBoost.
- Data Visualization Tools
    - Matplotlib, Seaborn, or Tableau for data exploration and presentation.
- Database Management
    - SQL or NoSQL databases for structured/unstructured data handling.
- Version Control
    - GitHub or GitLab for managing code versions and collaborations.

1.5.4 Data

- Clinical Datasets
    - Datasets mentioned in the related documents (e.g., anthropometric and body composition data).
    - Supplementary data from electronic health records, wearable devices, and genetic sources.
- Data Cleaning and Preprocessing Tools
    - Tools for handling missing values, scaling features, and integrating data from different sources.

1.5.5 Budget

- Human Resources Costs
  - Salaries for team members.
- Infrastructure Costs
  - Cloud computing expenses (e.g., for GPU/TPU usage).
  - Licenses for specific software tools, if required.
- Training Costs
  - Workshops or training sessions on machine learning and NAFLD for team members.

1.5.6 Research and Documentation Resources

- Access to Research Papers
  - Subscriptions to journals (if full access to some references is needed).
- Collaboration Tools
  - Slack, Microsoft Teams, or similar tools for communication and documentation.
- Guidelines and Standards
  - Compliance with healthcare regulations (e.g., HIPAA, GDPR) for handling sensitive clinical data.

1.5.7 Validation Resources

- External Experts for Model Validation
  - Collaborate with healthcare providers to validate the AI model.
- Testing Infrastructure
  - Simulated environments to test the integration of the model in real-world scenarios.

1.5.8 Post-Development Resources

- Deployment
  - Web hosting for the user interface or app-based deployment.
- Maintenance
  - Continuous monitoring and updates of the model based on user feedback.
- Marketing and Education
  - Creating awareness about the tool among healthcare providers and patients.

1.6 Report Layout

- Chapter 1: Introduction
- Chapter 2: Background and Existing Work
- Chapter 3: Data Exploration and Preprocessing
- Chapter 4: Algorithms and Techniques
- Chapter 5: Implementation
- Chapter 6: Conclusion and Future Work
- References

CHAPTER 2
BACKGROUND AND EXISTING WORK

2.0 Introduction

Non-Alcoholic Fatty Liver Disease (NAFLD) has garnered significant attention in medical research and public health due to its rising prevalence and potential complications. This section explores the existing work and background related to the prediction of NAFLD, highlighting key studies, methodologies, and advancements in the field.

2.0.1 Understanding NAFLD

- **Definition and Classification**: NAFLD encompasses a range of liver conditions, primarily characterized by fat deposition in the liver without significant alcohol consumption. It is typically classified into simple steatosis and NASH, with the latter indicating inflammation and liver cell damage.
- **Epidemiological Studies**: Research has shown that NAFLD affects approximately 25% of the global population, with higher rates observed in individuals with obesity and metabolic syndrome.

2.0.2 Predictive Models and Tools

- **Traditional Risk Scores**: Early work in predicting NAFLD risk involved the development of scoring systems, such as the NAFLD fibrosis score and the AST-to-ALT ratio, which rely on clinical and biochemical parameters.
- **Imaging Techniques**: Advances in non-invasive imaging modalities, such as ultrasound, MRI, and elastography, have improved the ability to assess liver fat content and fibrosis, aiding in the prediction and diagnosis of NAFLD.

2.0.3 Machine Learning and Artificial Intelligence

- **Emerging Technologies**: Recent advancements in machine learning and artificial intelligence have led to the development of sophisticated predictive models. Studies have utilized algorithms to analyze large datasets, integrating clinical, biochemical, and imaging data to enhance prediction accuracy.
- **Notable Studies**: Research has demonstrated the efficacy of machine learning approaches in predicting NAFLD. For example, models trained on electronic health records have shown promise in identifying high-risk individuals based on patterns in the data.

2.1 Project Overview

Non-Alcoholic Fatty Liver Disease (NAFLD) is a prevalent liver disorder that affects millions of people globally, characterized by the accumulation of fat in the liver of individuals who consume minimal to no alcohol. As NAFLD progresses, it can evolve into more severe conditions, such as non-alcoholic steatohepatitis (NASH), cirrhosis, and liver cancer. Early detection is crucial for effective management and prevention of complications; however, NAFLD is often asymptomatic in its early stages, making it challenging to diagnose. This project aims to leverage machine learning techniques to develop a system that predicts the risk of NAFLD using clinical, demographic, metabolic, and lifestyle data. By creating an AI model capable of accurately identifying NAFLD, the system will assist healthcare professionals in making well-informed decisions about patient

care, ultimately facilitating early diagnosis and intervention, reducing the incidence of NAFLD, and improving patient outcomes. It also aims to wrap the final model in a user friendly interface to be used by doctors.

2.2 Limitations of project

Data-Related Limitations

- **Data Quality and Completeness**: Missing or incomplete data in clinical records, which can impact model accuracy and reliability.
- **Sample Size**: If the dataset is small or lacks diversity, it may not generalize well to larger or different populations.
- **Imbalanced Data**: A dataset with a disproportionate number of NAFLD-positive and NAFLD-negative cases can lead to biased models.

Algorithmic Limitations

- **Model Interpretability**: Some machine learning algorithms (e.g., Random Forest, XGBoost) are complex and lack transparency, making it difficult to explain predictions to healthcare professionals.
- **Overfitting**: Risk of the model performing well on training data but poorly on unseen data due to overfitting.
- **Feature Selection Bias**: Incorrectly selected features might reduce model performance or ignore important predictors.

Resource Constraints

- **Computational Costs**: Training models with large datasets and complex algorithms can require significant computational power and time.
- **Infrastructure**: Deployment of the model may need specialized systems that healthcare facilities might lack.

2.2.1 Innovations in Project

The innovations of this project lie in leveraging advanced machine learning techniques to address the growing problem of Non-Alcoholic Fatty Liver Disease (NAFLD) and improve its early detection and prediction. Here are the key innovations:

- Machine Learning for Early Detection:
    - The project focuses on utilizing machine learning (ML) algorithms to predict the risk of NAFLD. This approach allows for early identification of individuals at risk of developing the disease, well before it progresses to severe stages such as cirrhosis or liver cancer.

    - By combining clinical, biochemical, and lifestyle factors, the model can identify patterns in large datasets, facilitating more accurate predictions than traditional methods.
- Non-Invasive, Cost-Effective Prediction:
    - Traditional diagnostic techniques, like liver biopsies and imaging methods, are costly, invasive, and not always effective for early-stage detection. The innovation here is in creating a non-invasive, cost-effective predictive model using machine learning, which reduces the reliance on expensive diagnostic methods.
- Application of Advanced Algorithms:
    - The project explores various machine learning algorithms such as K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Random Forest (RF), Extreme Gradient Boosting (XGB), and Bayesian Networks (BN). These are state-of-the-art techniques that have been proven to work well in complex prediction tasks.
    - The innovation lies in applying these advanced techniques specifically to the prediction of NAFLD, which has not been widely explored.

Overall, the innovation of this project lies in the integration of machine learning with clinical, biochemical, and lifestyle data to create a predictive model that is non-invasive, cost-effective, and accurate, ultimately leading to better early detection and management of NAFLD.

2.2.2 Design of the Project

- **System Architecture**: The system consists of a backend machine learning model, a database for storing patient data, and a frontend user interface.
- **Data Flow**: Data is collected, preprocessed, analyzed using machine learning algorithms, and presented as a risk score to users.
- **Technology Stack**: Python (for ML), Flask/Django (for backend), React/Angular (for frontend), SQL/NoSQL (for database), and cloud computing resources for deployment.

2.3 Existing Work

The following are the journal articles that used the dataset of NAGALA cohort study [1]:

- Ectopic fat obesity presents the greatest risk for incident type 2 diabetes: a population-based longitudinal study (2018), International Journal of Obesity [2]
    - Objective: study the predictive value of obesity phenotype for diabetes
    - Tools: Cox proportional hazard model
- Association between hypertriglyceridemic-waist phenotype and non-alcoholic fatty liver disease: a general population-based study (2022), Lipids in Health and Disease [3]
    - Objective: study the predictive value of triglyceride waist circumference phenotype for non-alcoholic fatty liver disease
    - Tools: stepwise logistic regression
- Abdominal obesity phenotypes are associated with the risk of developing non-alcoholic fatty liver disease: insights from the general population (2022), BMC Gastroenterology [4]
    - Objective: study the predictive value of abdominal obesity phenotype for non-alcoholic fatty liver disease
    - Tools: stepwise logistic regression
- The triglyceride glucose-body mass index: a non-invasive index that identifies non-alcoholic fatty liver disease in the general Japanese population, Journal of Translational Medicine (2022) [5]
    - Objective: study the predictive value of triglyceride glucose-body mass index for non-alcoholic fatty liver disease
- Assessing temporal differences of baseline body mass index, waist circumference, and waist-height ratio in predicting future diabetes (2023), Frontiers in Endocrinology [6]
    - Objective: compare the predictive value of body mass index, waist circumference and waist-height ratio for the occurrence of diabetes at different time points in the future
    - Tools: stepwise Cox proportional hazards regression
- The role of predicted lean body mass and fat mass in non-alcoholic fatty liver disease in both sexes: Results from a secondary analysis of the NAGALA study (2023), Frontiers in Nutrition [7]
    - Objective: study the predictive value of lean body mass and fat mass for non-alcoholic fatty liver disease
    - Tools: stepwise logistic regression and restricted cubic spline regression
- Remnant cholesterol/high-density lipoprotein cholesterol ratio is a new powerful tool for identifying non-alcoholic fatty liver disease (2022), BMC Gastroenterology [8]
    - Objective: study the predictive value of remnant cholesterol/high-density lipoprotein cholesterol ratio for non-alcoholic fatty liver disease
    - Tools: stepwise logistic regression
- Association between weight-adjusted waist index and non-alcoholic fatty liver disease: a population-based study (2024), BMC Endocrine Disorders [9]

- Objective: study the predictive value of weight-adjusted waist index for non-alcoholic fatty liver disease
- Tools: stepwise logistic regression and restricted cubic spline regression

CHAPTER 3
DATA EXPLORATION AND PREPROCESSING

3.1 Data Collection

A dataset published at Dryad website has been used. It was collected at Murakami Memorial Hospital in Gifu city in Japan for the NAGALA cohort study. NAFLD was diagnosed by abdominal ultrasonography. [1]

3.2 Data Definition

Feature	Description	Type
Sex	Person's sex (0: Female, 1: Male)	Categorical – Binary
Age	Person's age	Numerical – Discrete
Body Mass Index (BMI)	Weight in relation to height (kg/m^2)	Numerical – Continuous
Weight	Body weight (kg)	Numerical – Continuous
Height	Body height (cm)	Numerical – Continuous
Lean Body Mass	Body mass without fats (kg)	Numerical – Continuous
Fat Mass	Mass of body fat (kg)	Numerical – Continuous
Waist Circumference	Circumference of the waist (cm)	Numerical – Continuous
Diabetes Mellitus (DM)	A group of diseases caused by increased levels of blood sugar (0: Not Infected, 1: Infected)	Categorical – Binary
Alanine Aminotransferase (ALT)	An enzyme mainly found in liver (U/L)	Numerical – Discrete
Aspartate Transferase (AST)	An enzyme found in liver and other body components (U/L)	Numerical – Discrete
Gamma-Glutamyl Transferase (GGT)	An enzyme mainly found in liver (U/L)	Numerical – Discrete
High-Density Lipoprotein (HDL)	A type of liboprotein that removes fat molecules from body cells (mmol/L)	Numerical – Continuous
Total Cholestrol (TC)	The total amount of all types of cholesterol (mmol/L)	Numerical – Continuous
Triglyceride (TG)	A chemical compound that is main component of body fats (mmol/L)	Numerical – Continuous
Fasting Plasma Glucose (FPG)	A measure for blood sugar level after fasting for 10 to 16 hours while from everything with the exception of water (mmol/L)	Numerical – Continuous
Glycated Hemoglobin (Hb1Ac)	A form of hemoglobin that bonds with sugar in the blood without the help of enzymes. It is measured to determine the average blood sugar level over the last three months	Numerical – Continuous
Systolic Blood Pressure (SBP)	Amount of force exerted on arteries during a heartbeat, i.e., when heart contracts (mmHg)	Numerical – Continuous
Diastolic Blood Pressure (DBP)	Amount of force exerted on arteries between two heartbeats, i.e., when heart rests (mmHg)	Numerical – Continuous
Exercise	Whether the person does exercise for at least once a week (1: Yes, 0: No)	Categorical – Binary
Drinking	Drinking status (1: None/Small, 2: Light, 3: Moderate)	Categorical – Ordinal
Smoking	Smoking status (1: None, 2: Past, 3: Present)	Categorical – Ordinal
Ethanol consumption	Amount of ethanol consumed per week (g/wk)	Numerical – Continuous
NAFLD	Whether the person has NAFLD or not (1: Yes, 0: No)	Categorical – Binary

Table 1 Data definition

3.3 Exploration

3.3.1 Measures

AST (U/L)	
Mean	18.22601923
Standard Error	0.072618114
Median	17
Mode	17
Standard Deviation	8.668972671
Sample Variance	75.15108716
Kurtosis	1348.358605
Skewness	22.48366312
Range	587
Minimum	3
Maximum	590
Sum	259739
Count	14251
Confidence Level(95.0%)	0.142340978

Table 2 Descriptive statistics of AST

LBM (kg)	
Mean	42.6441294
Standard Error	0.080717202
Median	42.9052
Mode	48.0386
Standard Deviation	9.635821947
Sample Variance	92.8490646
Kurtosis	-0.919440152
Skewness	0.301948677
Range	71.2501
Minimum	24.6789
Maximum	95.929
Sum	607721.4881
Count	14251
Confidence Level(95.0%)	0.158216246

Table 6 Descriptive statistics of LBM

FM (kg)	
Mean	16.13398675
Standard Error	0.043264111
Median	15.7037
Mode	8.667599999
Standard Deviation	5.164763707
Sample Variance	26.67478415
Kurtosis	1.806811288
Skewness	0.750754756
Range	52.0986
Minimum	0.621199998
Maximum	52.7198
Sum	229925.4452
Count	14251
Confidence Level(95.0%)	0.084803303

Table 3 Descriptive statistics of FM

Height (cm)	
Mean	164.7961617
Standard Error	0.071047935
Median	164.6
Mode	160
Standard Deviation	8.481528578
Sample Variance	71.93632702
Kurtosis	-0.586896137
Skewness	0.075497621
Range	65.79999999
Minimum	127.6
Maximum	193.4
Sum	2348510.1
Count	14251
Confidence Level(95.0%)	0.139263222

Table 5 Descriptive statistics of Height

WC (cm)	
Mean	76.18522209
Standard Error	0.076232251
Median	76
Mode	70
Standard Deviation	9.100419531
Sample Variance	82.81763563
Kurtosis	0.320204715
Skewness	0.407763406
Range	81.5
Minimum	48.5
Maximum	130
Sum	1085715.6
Count	14251
Confidence Level(95.0%)	0.149425158

Table 7 Descriptive statistics of WC

ALT (U/L)	
Mean	19.76394639
Standard Error	0.121179809
Median	16
Mode	13
Standard Deviation	14.46614903
Sample Variance	209.2694678
Kurtosis	797.692619
Skewness	16.0303552
Range	854
Minimum	2
Maximum	856
Sum	281656
Count	14251
Confidence Level(95.0%)	0.237528237

Table 4 Descriptive statistics of ALT

GGT (U/L)	
Mean	19.12553505
Standard Error	0.135152687
Median	15
Mode	12
Standard Deviation	16.13419691
Sample Variance	260.31231
Kurtosis	62.80417986
Skewness	5.790597325
Range	372
Minimum	3
Maximum	375
Sum	272558
Count	14251
Confidence Level(95.0%)	0.2649169

Table 8 Descriptive statistics of GGT

TG (mmol/L)	
Mean	0.891410898
Standard Error	0.005296792
Median	0.72256
Mode	0.50805
Standard Deviation	0.632318022
Sample Variance	0.399826081
Kurtosis	15.46305533
Skewness	2.807563256
Range	10.20616
Minimum	0.06774
Maximum	10.2739
Sum	12703.49671
Count	14251
Confidence Level(95.0%)	0.010382403

Table 9 Descriptive statistics of TG

FPG (mmol/L)	
Mean	5.147538783
Standard Error	0.003448795
Median	5.10692
Mode	5.05141
Standard Deviation	0.411708746
Sample Variance	0.169504091
Kurtosis	-0.271981445
Skewness	-0.027641903
Range	3.21958
Minimum	2.83101
Maximum	6.05059
Sum	73357.5752
Count	14251
Confidence Level(95.0%)	0.006760089

Table 10 Descriptive statistics of FPG

TC (mmol/L)	
Mean	5.123624325
Standard Error	0.007271739
Median	5.06856
Mode	4.80996
Standard Deviation	0.868082408
Sample Variance	0.753567067
Kurtosis	0.668844554
Skewness	0.437059038
Range	9.72336
Minimum	2.56014
Maximum	12.2835
Sum	73016.77026
Count	14251
Confidence Level(95.0%)	0.014253557

Table 11 Descriptive statistics of TC

HbA1c (%)	
Mean	5.177758754
Standard Error	0.002688775
Median	5.2
Mode	5.2
Standard Deviation	0.320979319
Sample Variance	0.103027723
Kurtosis	-0.051369323
Skewness	0.082372343
Range	2.7
Minimum	3.7
Maximum	6.4
Sum	73788.24
Count	14251
Confidence Level(95.0%)	0.005270349

Table 12 Descriptive statistics of HbA1c

HDL (mmol/L)	
Mean	1.459071973
Standard Error	0.003366443
Median	1.411956
Mode	1.37058
Standard Deviation	0.401877777
Sample Variance	0.161505748
Kurtosis	0.765333123
Skewness	0.668842489
Range	3.56868
Minimum	0
Maximum	3.56868
Sum	20793.23468
Count	14251
Confidence Level(95.0%)	0.006598668

Table 13 Descriptive statistics of HDL

Ethanol consumption (g/wk)	
Mean	28.3143273
Standard Error	0.391476521
Median	1
Mode	0
Standard Deviation	46.73350887
Sample Variance	2184.020851
Kurtosis	1.852892609
Skewness	1.707983509
Range	207
Minimum	0
Maximum	207
Sum	403507.4783
Count	14251
Confidence Level(95.0%)	0.767345059

Table 18 Descriptive statistics of Ethanol Consumption

DBP (mmHg)	
Mean	71.1218511
Standard Error	0.086979826
Median	70.5
Mode	70
Standard Deviation	10.38343876
Sample Variance	107.8158004
Kurtosis	0.843449223
Skewness	0.495335077
Range	117
Minimum	39
Maximum	156
Sum	1013557.5
Count	14251
Confidence Level(95.0%)	0.170491808

Table 19 Descriptive statistics of DBP

Age	
Mean	43.53301523
Standard Error	0.074497438
Median	42
Mode	35
Standard Deviation	8.893321825
Sample Variance	79.09117308
Kurtosis	-0.117884276
Skewness	0.409192243
Range	61
Minimum	18
Maximum	79
Sum	620389
Count	14251
Confidence Level(95.0%)	0.146024699

Table 14 Descriptive statistics of Age

Weight (kg)	
Mean	60.26223423
Standard Error	0.097289493
Median	59.1
Mode	60.6
Standard Deviation	11.61418157
Sample Variance	134.8892136
Kurtosis	0.747132898
Skewness	0.66934638
Range	118.5
Minimum	32.3
Maximum	150.8
Sum	858797.1
Count	14251
Confidence Level(95.0%)	0.190700101

Table 15 Descriptive statistics of Weight

SBP (mmHg)	
Mean	113.9345309
Standard Error	0.12416357
Median	112.5
Mode	111.5
Standard Deviation	14.82234306
Sample Variance	219.7018538
Kurtosis	1.345907977
Skewness	0.642247792
Range	180
Minimum	71
Maximum	251
Sum	1623681
Count	14251
Confidence Level(95.0%)	0.243376797

Table 17 Descriptive statistics of SBP

BMI(kg/m^2)	
Mean	22.06466053
Standard Error	0.026274579
Median	21.72839506
Mode	19.25532814
Standard Deviation	3.136594941
Sample Variance	9.838227826
Kurtosis	1.576561282
Skewness	0.815867704
Range	36.15692582
Minimum	13.76620316
Maximum	49.92312898
Sum	314443.4772
Count	14251
Confidence Level(95.0%)	0.051501603

Table 16 Descriptive statistics of BMI

3.3.2 Distributions

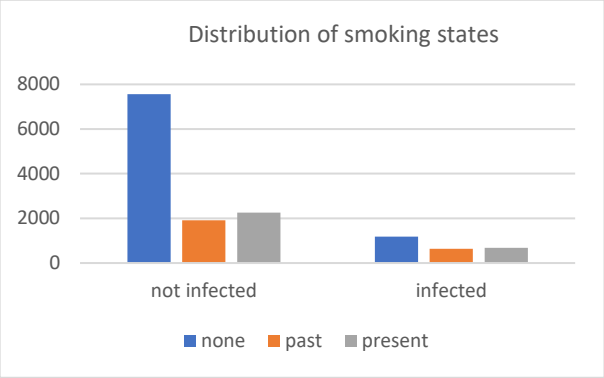


Figure 3 Distribution of smoking states

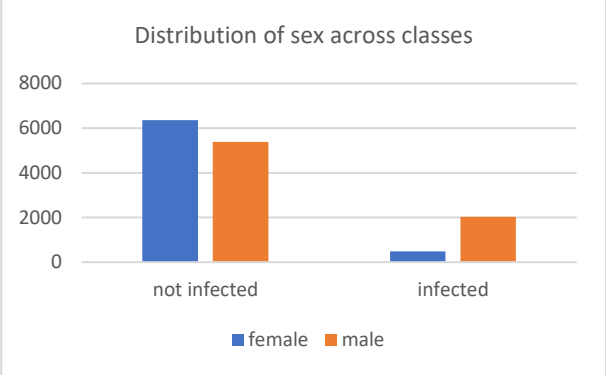


Figure 4 Distribution of sex across classes

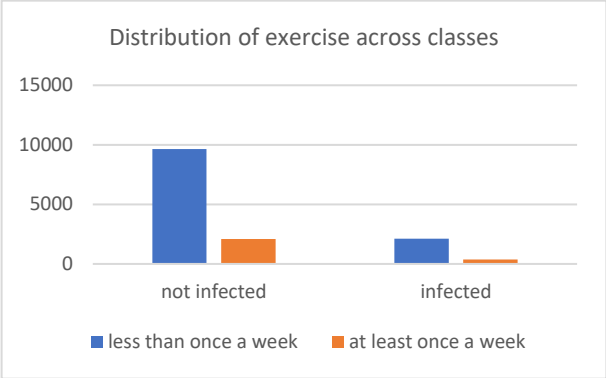


Figure 5 Distribution of exercise across classes

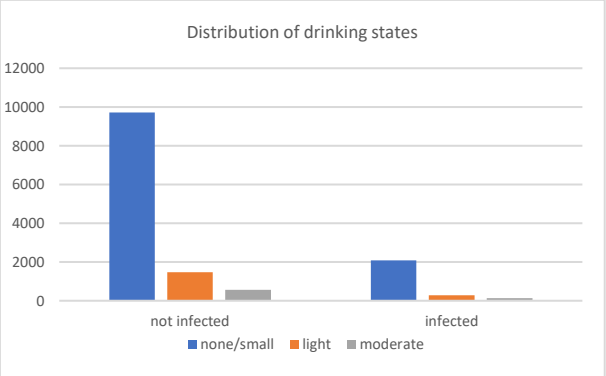


Figure 6 Distribution of drinking states

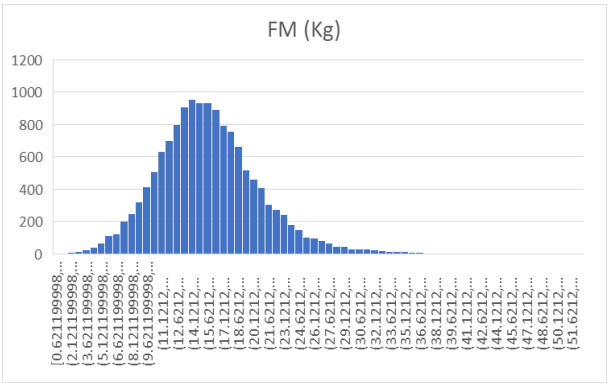


Figure 14 Distribution of FM

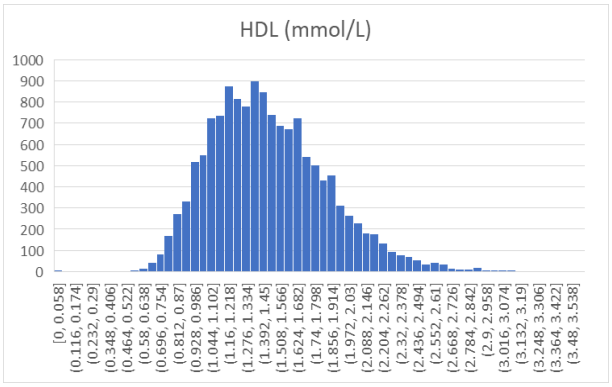


Figure 13 Distribution of HDL

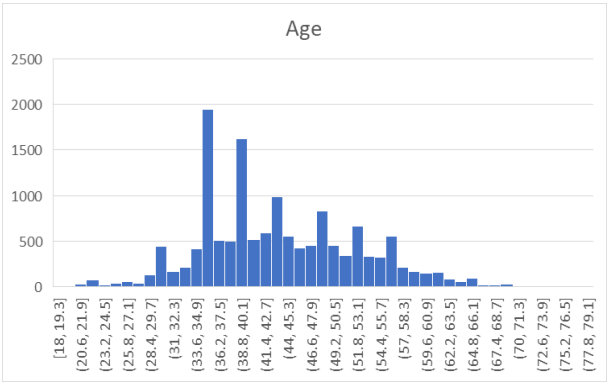


Figure 12 Distribution of Age

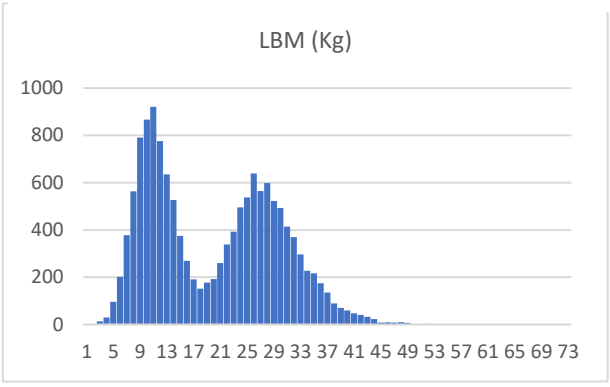


Figure 11 Distribution of LBM

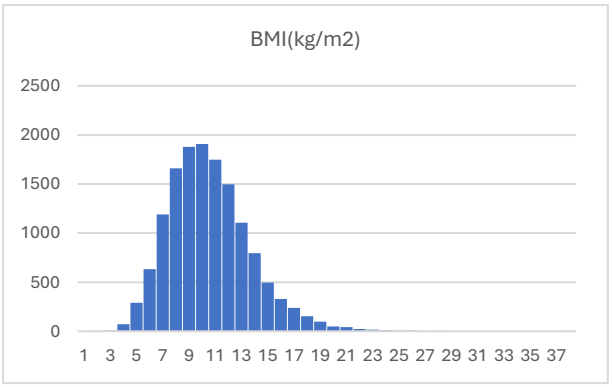


Figure 10 Distribution of BMI

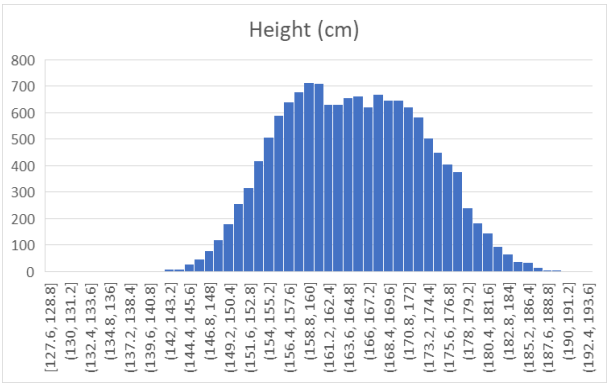


Figure 9 Distribution of Height

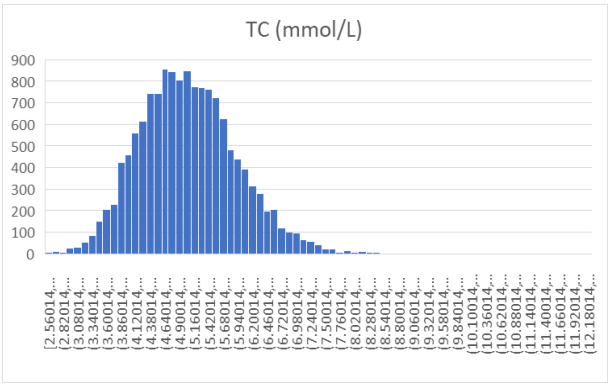


Figure 8 Distribution of TC

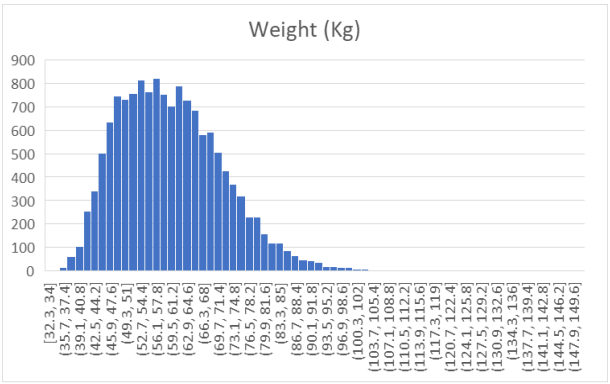


Figure 7 Distribution of Weight

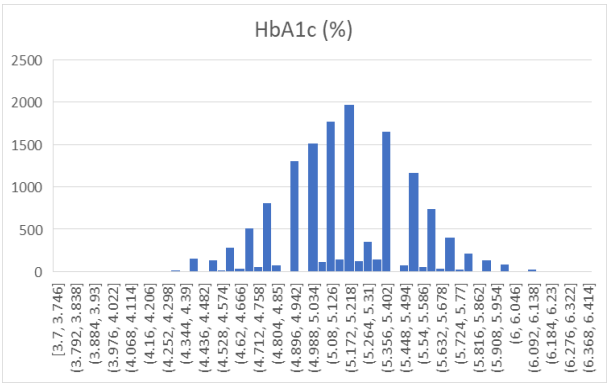


Figure 20 Distribution of HbA1c

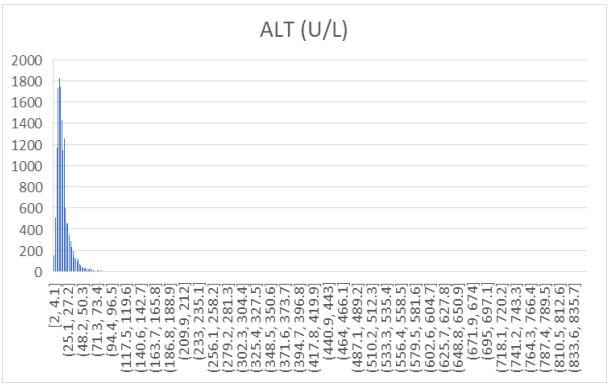


Figure 19 Distribution of ALT

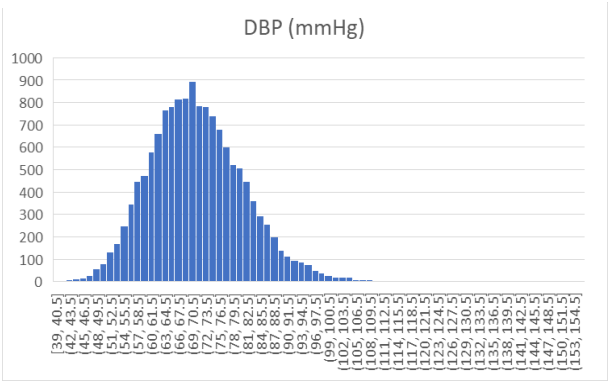


Figure 18 Distribution of DBP

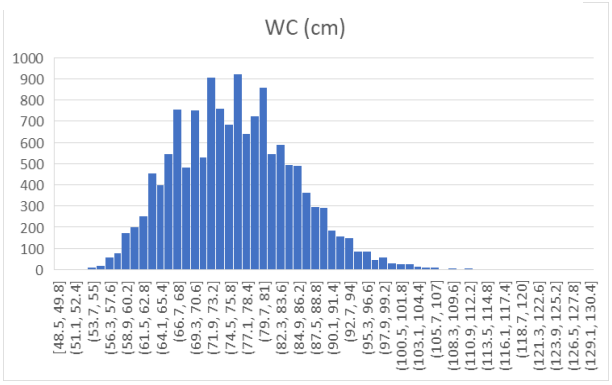


Figure 17 Distribution of WC

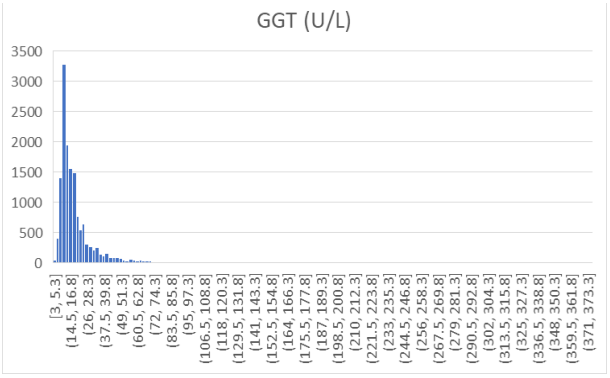


Figure 15 Distribution of GGT

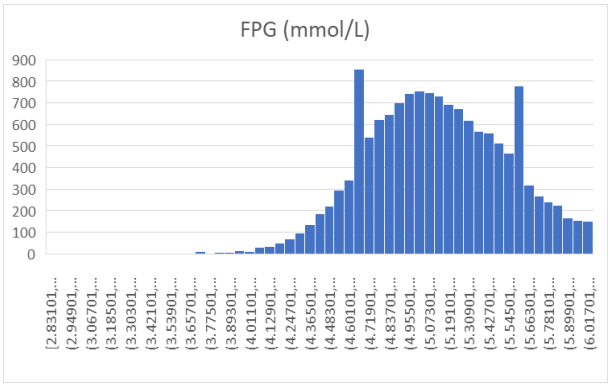


Figure 16 Distribution of FPG

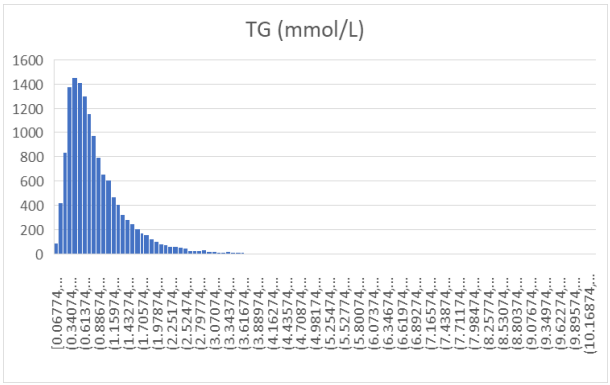


Figure 22 Distribution of TG

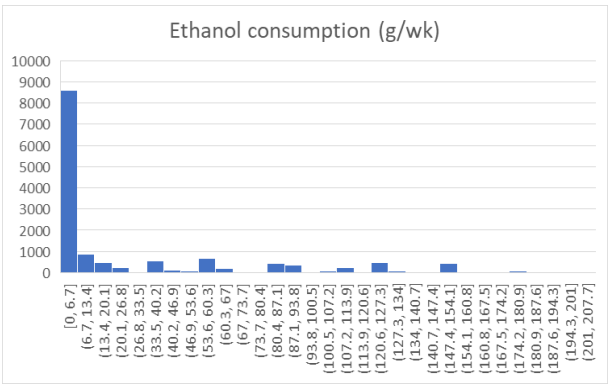


Figure 21 Distribution of Ethanol Consumption

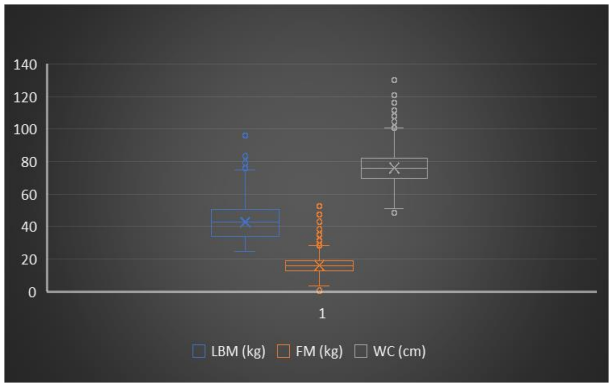


Figure 23 Box Plots 1

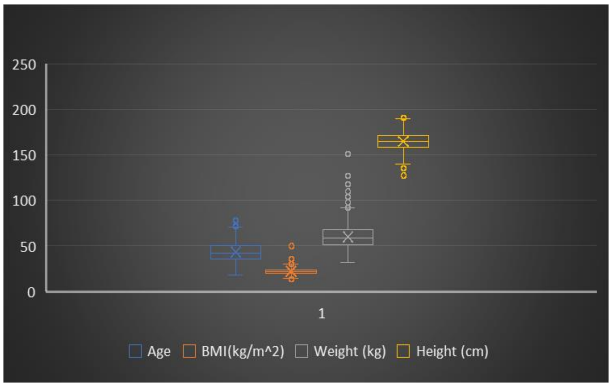


Figure 24 Box Plots 2

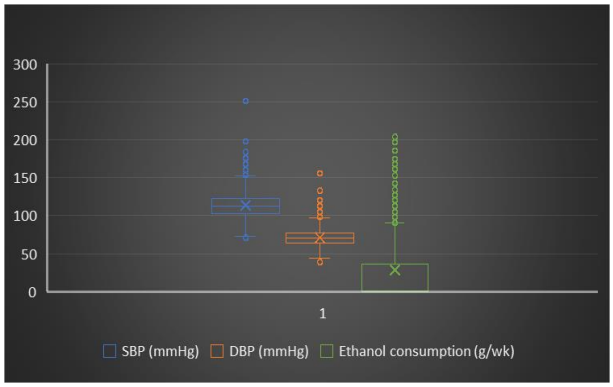


Figure 25 Box Plots 3

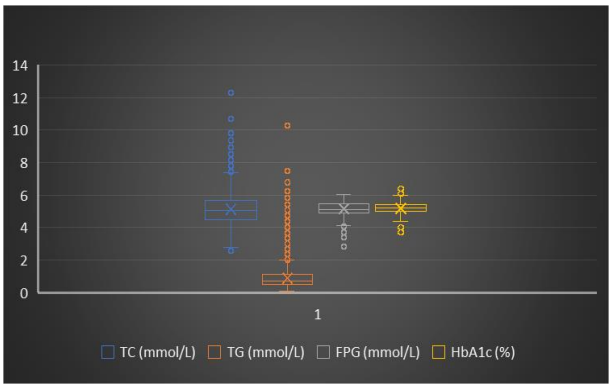


Figure 26 Box Plots 4

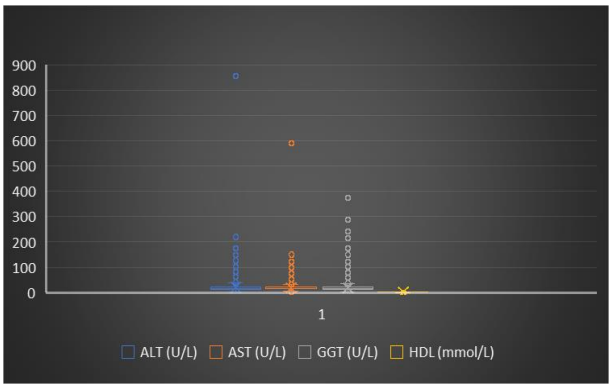


Figure 27 Box Plots 5

3.3.3 Correlations

Correlation is a statistical measure that describes the relationship between two or more variables. It indicates how one variable changes in relation to another. Correlation is commonly expressed using a correlation coefficient, which ranges from -1 to 1:

- +1 (Perfect Positive Correlation): When one variable increases, the other also increases in a perfectly linear way.
- −1 (Perfect Negative Correlation): When one variable increases, the other decreases in a perfectly linear way.
- 0 (No Correlation): No relationship between the variables.

The following is the pairwise correlation matrix:

	Age	BMI (kg/m^2)	Weight (kg)	Height (cm)	LBM (kg)	FM (kg)	WC (cm)	ALT (U/L)	AST (U/L)	GGT (U/L)	HDL (mmol/l)	TC (mmol/l)	TG (mmol/l)	FPG (mmol/l)	HbA1c (%)	SBP (mmHg)	DBP (mmHg)	Ethanol consumption (g/wk)
Age	1.00	0.08	-0.03	-0.16	-0.07	0.08	0.14	-0.01	0.06	0.08	-0.03	0.30	0.15	0.17	0.22	0.19	0.23	0.09
BMI (kg/m^2)	0.08	1.00	0.85	0.20	0.60	0.77	0.86	0.37	0.20	0.28	-0.41	0.20	0.40	0.34	0.15	0.45	0.44	0.10
Weight (kg)	-0.03	0.85	1.00	0.69	0.90	0.52	0.87	0.39	0.20	0.33	-0.47	0.11	0.41	0.39	0.08	0.44	0.43	0.22
Height (cm)	-0.16	0.20	0.69	1.00	0.85	-0.10	0.43	0.21	0.09	0.21	-0.32	-0.07	0.21	0.26	-0.06	0.19	0.20	0.27
LBM (kg)	-0.07	0.60	0.90	0.85	1.00	0.10	0.68	0.37	0.19	0.34	-0.48	0.05	0.38	0.40	0.01	0.37	0.38	0.30
FM (kg)	0.08	0.77	0.52	-0.10	0.10	1.00	0.64	0.17	0.09	0.08	-0.15	0.16	0.19	0.11	0.17	0.26	0.23	-0.10
WC (cm)	0.14	0.86	0.87	0.43	0.68	0.64	1.00	0.38	0.22	0.33	-0.46	0.19	0.43	0.39	0.16	0.46	0.45	0.18
ALT (U/L)	-0.01	0.37	0.39	0.21	0.37	0.17	0.38	1.00	0.82	0.47	-0.24	0.14	0.31	0.21	0.09	0.23	0.23	0.06
AST (U/L)	0.06	0.20	0.20	0.09	0.19	0.09	0.22	0.82	1.00	0.35	-0.10	0.11	0.17	0.09	0.11	0.16	0.17	0.05
GGT (U/L)	0.08	0.28	0.33	0.21	0.34	0.08	0.33	0.47	0.35	1.00	-0.17	0.19	0.33	0.25	0.08	0.24	0.25	0.24
HDL (mmol/l)	-0.03	-0.41	-0.47	-0.32	-0.48	-0.15	-0.46	-0.24	-0.10	-0.17	1.00	0.13	-0.46	-0.23	-0.01	-0.20	-0.22	-0.05
TC (mmol/l)	0.30	0.20	0.11	-0.07	0.05	0.16	0.19	0.14	0.11	0.19	0.13	1.00	0.33	0.17	0.21	0.18	0.19	0.01
TG (mmol/l)	0.15	0.40	0.41	0.21	0.38	0.19	0.43	0.31	0.17	0.33	-0.46	0.33	1.00	0.30	0.11	0.29	0.31	0.13
FPG (mmol/l)	0.17	0.34	0.39	0.26	0.40	0.11	0.39	0.21	0.09	0.25	-0.23	0.17	0.30	1.00	0.32	0.35	0.33	0.17
HbA1c (%)	0.22	0.15	0.08	-0.06	0.01	0.17	0.16	0.09	0.11	0.08	-0.01	0.21	0.11	0.32	1.00	0.12	0.10	-0.10
SBP (mmHg)	0.19	0.45	0.44	0.19	0.37	0.26	0.46	0.23	0.16	0.24	-0.20	0.18	0.29	0.35	0.12	1.00	0.90	0.15
DBP (mmHg)	0.23	0.44	0.43	0.20	0.38	0.23	0.45	0.23	0.17	0.25	-0.22	0.19	0.31	0.33	0.10	0.90	1.00	0.18
Ethanol consumption (g/wk)	0.09	0.10	0.22	0.27	0.30	-0.10	0.18	0.06	0.05	0.24	-0.05	0.01	0.13	0.17	-0.10	0.15	0.18	1.00

Figure 28 Correlation Matrix

3.4 Preprocessing

Data preprocessing is a crucial step in data analysis and machine learning, ensuring that raw data is cleaned and structured for better performance. Without proper preprocessing, raw data may lead to inaccurate models and misleading insights.

Additionally, data preprocessing enhances the efficiency of machine learning algorithms by reducing computational complexity and improving model accuracy. By investing time in preprocessing, analysts and data scientists can achieve more reliable and meaningful results from their datasets.

3.4.1 Basic Feature Selection

In the beginning, basic feature selection has been performed based on redundency.

First, features that represented perfect redundency have been removed, which are:

- Body Mass Index (BMI), because it can be calculated from other features as $\text{Body Mass Index} = \frac{\text{Weight}}{\text{Height}^2}$
- Weight, because it can be calculated from other features as $\text{Weight} = \text{Lean Body Mass} + \text{Fat Mass}$
- Ethanol comsumption, due to the existense of Drinking feature, which is a categorized version of it

Second, feature selection has been performed based on the correlation matrix.

One feature has been selected from every pair of features with correlation ≥ 0.7, which are:

- LBM, because it has 0.85 correlation with height and height has less correlation with other features
- ALT, because it has 0.82 correlation with AST and AST has less correlation with other features
- DBP, because it has 0.9 correlation with SBP and SBP has less correlation with other features

3.4.2 Stratified Splitting

When splitting data into training and testing sets, it is essential to ensure that the distribution of the target variable remains consistent in both subsets. Stratified splitting achieves this by preserving the proportion of each class (e.g., NAFLD vs. Non-NAFLD) in both the training (80%) and testing (20%) sets.

This technique is particularly useful for imbalanced datasets, where one class significantly outnumbers the other. Without stratification, random splitting may result in a training set dominated by the majority class and a test set with very few instances of the minority class, leading to biased or inaccurate model performance.

3.4.3 Feature Scaling

Scaling based on the interquartile range and centering based on the median has been performed. This way of scaling and centering, known as robust scaling, is more robust to outliers than other scaling techniques. [10]

Scaling has been performed based on the training data before every training operation. This means that testing set remains the same, and also during cross-validation, validation set remains the same.

3.4.4 Feature Projection

To make the models benefit more easily from the data, feature projection techniques has been used to reduce the dataset into a more compact but useful form.

The techniques used are Non-negative Matrix Factorization (NMF) and Factor Analysis of Mixed Data (FAMD).

For NMF, it thas been performed on numerical features while the categorical features have been concatenated as is.

Different number of components has been evaluated on the training data as shown below. The selected number of components for each projection technique is highlighted.

Number of Components	NMF – Reconstruction Error	FAMD – Percentage of Explained Variance
1	2686.311593270351	23.273345936090656
2	2040.6406820971727	35.723748655544796
3	1579.2680977394318	44.436205068041986
4	1262.1135986882746	52.0927074354854
5	903.427159178197	59.08877844315688
6	370.2564943792498	65.01714714476294
7	169.00173371298072	70.30435143902204
8	142.34770133088207	75.4132853463928
9	115.67279371575016	79.85390304331582
10	100.91704675547462	83.88447399102986
11	82.68468231810327	87.66725196451891

Table 20 Comparing NMF and FAMD results on different number of components

CHAPTER 4
ALGORITHMS AND TECHNIQUES

4.1 Machine Learning

4.1.1 Definition

Machine Learning (ML) is a tool of artificial intelligence (AI) that deals with creating and examining statistical algorithms that facilitate computers in identifying patterns and making decisions with little to no human interaction.

ML is beneficial in many areas such as, but not limited to, natural languages, computer-generated visual imagery, voice recognition, email categorization, farming, and healthcare. The use of ML aimed towards solving business challenges is referred to as predictive analytics.

Statistical and mathematical optimization methods (mathematical programming) are at the core of machine learning development. It is also associated with data mining which emphasizes unsupervised exploratory data analysis (EDA).

4.1.2 Types of Machine Learning

- **Supervised Learning:**
  - In supervised learning, the model is trained on a labeled dataset, meaning each input is paired with a corresponding output.
  - Examples: Regression, Classification.
  - Common Algorithms: Linear Regression, Logistic Regression, Support Vector Machines (SVM), Decision Trees, Random Forest (RF), XG Boost (XGB).
- **Unsupervised Learning:**
  - Unsupervised learning works with unlabeled data, aiming to find hidden structures or patterns within it.
  - Examples: Clustering, Dimensionality Reduction.
  - Common Algorithms: K-Means Clustering, Hierarchical Clustering, Principal Component Analysis (PCA).
- **Reinforcement Learning:**
  - Involves training an agent to make decisions by rewarding desirable behaviors and penalizing undesirable ones.
  - Applications: Robotics, Game AI, Autonomous Vehicles. [11]

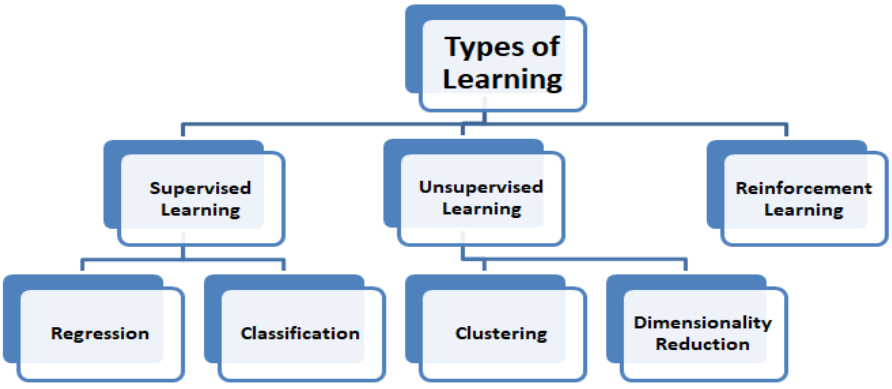


Figure 29 Types of ML

4.2 Algorithms

4.2.1 Definition

A machine learning algorithm is a collection of guidelines or procedures that an AI system uses to carry out tasks, usually to predict output values from a given set of input variables or to find new patterns and insights in data. Machine learning (ML) can learn thanks to algorithms.

ML algorithms that are trained on more data will typically yield more accurate results than those that are trained on less data. Algorithms are trained using statistical techniques to identify categories or forecast outcomes, as well as to reveal important information in data mining initiatives. These insights can then help you make better decisions to increase important growth indicators.

The capacity to examine data to spot trends and anticipate problems before they arise is one use case for machine learning algorithms. More sophisticated AI can provide speech recognition, speed up response times, enable more individualized service, and increase customer satisfaction. Supply chain management, logistics and transportation, retail, and manufacturing are among the industries that embrace generative AI because of its capacity to automate processes, boost productivity, and offer insightful information, even to novices. These sectors especially benefit from machine learning algorithms that generate new content from massive volumes of data.

4.2.2 Selected Algorithms

Numerous machine learning algorithms are chosen because of their proven efficacy in earlier research and their compatibility with our objectives:

- K-Nearest Neighbors (KNN)
- Support Vector Machines (SVM)
- Random Forest (RF)
- Extreme Gradient Boosting (XGB)
- Bayesian Networks (BN)

4.2.2.1 K-Nearest Neighbors (KNN)

A non-parametric, supervised learning classifier, the k-nearest neighbors (KNN) method employs closeness to classify or forecast how to group a single data point. It is among the most widely used and straightforward regression and classification classifiers in machine learning today.

The KNN algorithm, which operates on the premise that comparable points may be located close to one another, is commonly employed as a classification technique, however it can be applied to regression or classification issues.

A majority vote determines the class label for classification issues; that is, the label most often associated with a particular data point is adopted. The phrase "majority vote" is more frequently used in literature, even though this is properly referred to as "plurality voting." The difference between these terms is that "majority voting," which mostly functions when there are only two categories, officially calls for a majority of more than 50%. You don't

necessarily need 50% of the vote to decide on a class when there are several classes, such four categories; you can designate a class with a vote of more than 25%. [12]

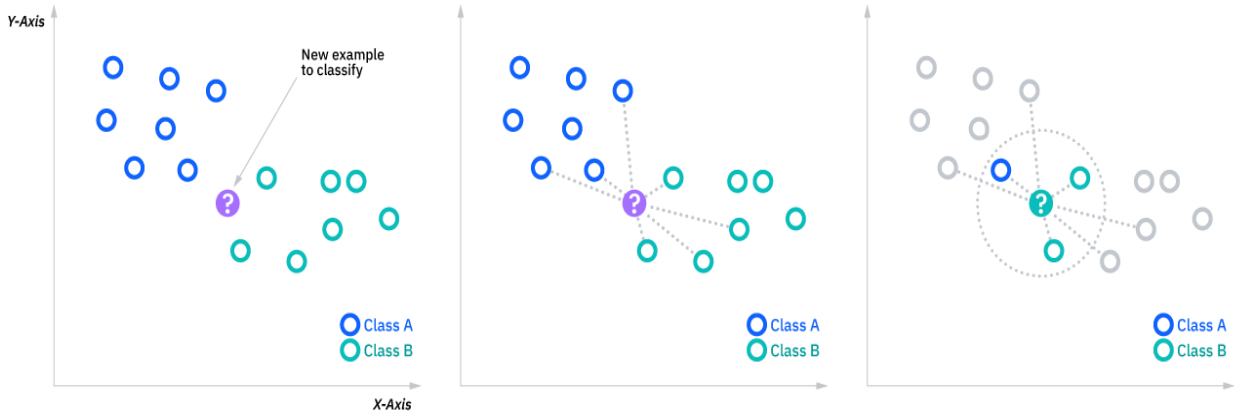


Figure 30 KNN

4.2.2.2 Support Vector Machine (SVM)

A supervised machine learning approach called a support vector machine (SVM) uses an N-dimensional space to identify the best line or hyperplane for classifying data by maximizing the distance between each class.

SVMs are frequently applied to classification issues. By identifying the best hyperplane that optimizes the margin between the nearest data points of opposing classes, they are able to differentiate between two classes. Whether the hyperplane is a line in two dimensions or a plane in n dimensions depends on the quantity of features in the input data. Since there are several hyperplanes that may be used to distinguish between classes, the method can determine the optimal decision border between classes by maximizing the margin between points. As a result, it can effectively generalize to fresh data and generate precise categorization forecasts. Since they pass through the data points that establish the greatest margin, the lines that are next to the ideal hyperplane are referred to as support vectors.

Because it can handle both linear and nonlinear classification tasks, the SVM method is often employed in machine learning. However, in situations when the data cannot be separated linearly, the data is transformed into a higher-dimensional space using kernel functions. This usage of kernel functions is sometimes referred to as the "kernel trick." The particular use case and the properties of the data determine whether kernel function—linear, polynomial, radial basis function (RBF), or sigmoid—is best. [13]

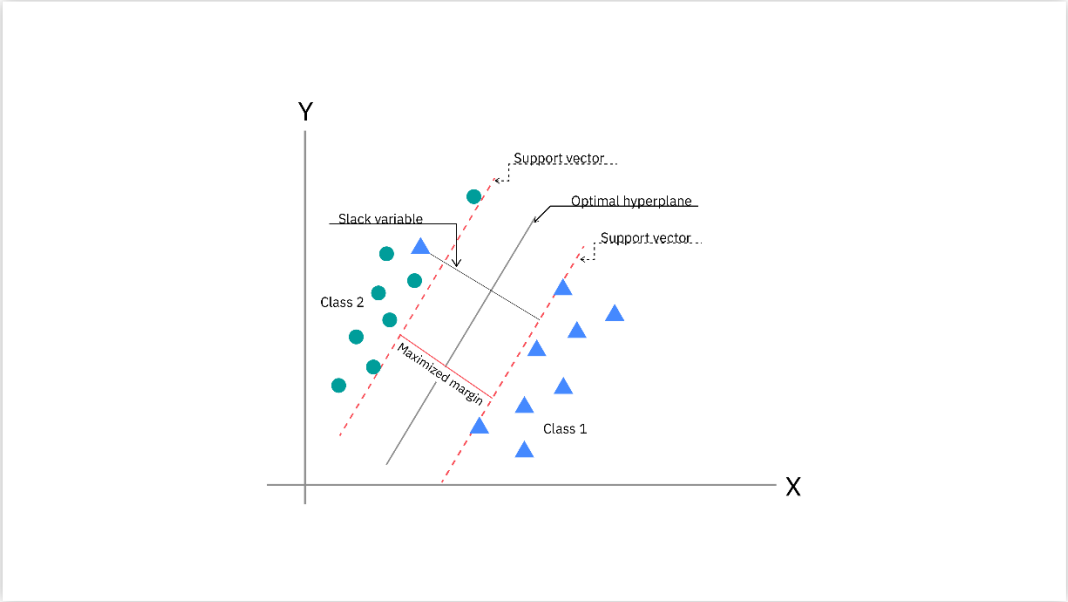


Figure 31 SVM

4.2.2.3 Random Forest (RF)

In order to produce predictions, we vote on all the trees using the Random Forest algorithm, a potent tree learning method in machine learning. They are frequently employed for tasks involving regression and classification.

This kind of classifier makes predictions by utilizing a large number of decision trees.

Each tree is trained using several random segments of the dataset, and the outcomes are then averaged. This method contributes to increased forecast accuracy. Ensemble learning is the foundation of Random Forest.

Consider consulting a group of friends about potential holiday destinations. Based on their own viewpoints and tastes, each buddy makes a suggestion (decision trees trained on distinct subsets of data). After that, you decide by taking into account the opinions of the majority or averaging their recommendations (ensemble prediction).

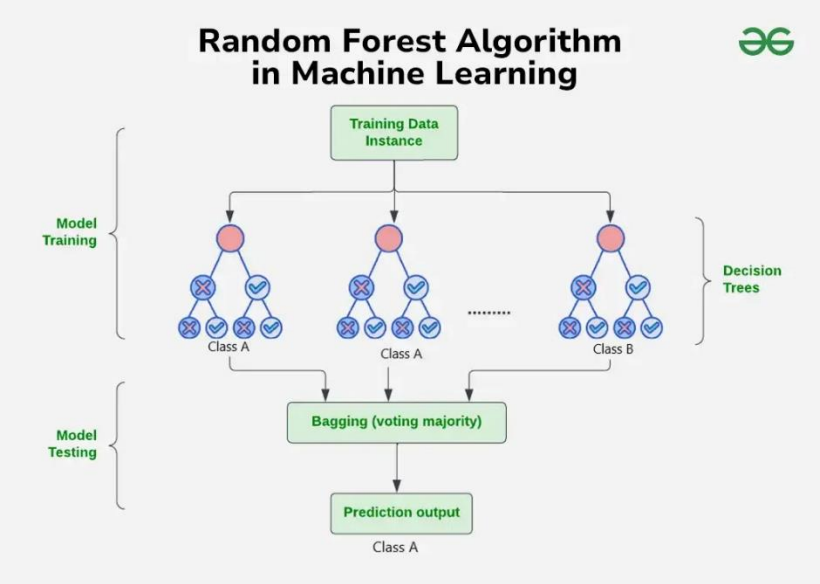


Figure 32 Random Forest

As illustrated in the image: The process begins with a dataset that contains rows and the class labels (columns) that correlate to those rows.

The training data is then used to generate several decision trees. A random selection of features and a random subset of data (with replacement) are used to train each tree. This method is referred to as bootstrap aggregating or bagging.
Every Decision Tree in the group gains the ability to anticipate on its own. Every Decision Tree in the ensemble makes a prediction when it is shown a new, unseen instance.
All of the Decision Trees' projections are combined to get the final forecast. Usually, a majority vote is used for classification, while averaging is used for regression. [14]

4.2.2.4 Extreme Gradient Boosting (XGB)

Extreme Gradient Boosting, or XG Boost, is a potent machine learning method renowned for its accuracy, speed, and efficiency. It is a member of the family of ensemble learning methods known as boosting algorithms, which aggregate the predictions of several weak learners.

Extreme Gradient Boosting, or XG Boost, is a cutting-edge machine learning technique that is well-known for its outstanding prediction capabilities. When it comes to gradient-boosting algorithms in particular, it is the industry standard for ensemble learning. It creates a dependable and accurate prediction model by gradually developing a number of poor learners.
In essence, XG Boost aggregates the predictions of several weak learners, often decision trees, to create a powerful predictive model. By using a boosting strategy, each weak learner following it fixes the errors of its predecessors, resulting in an incredibly accurate ensemble model.

By adjusting the model's parameters periodically in response to the gradients of the errors, the gradient optimization approach optimizes a cost function. Additionally, the technique

introduces the concept of "gradient boosting with decision trees," which reduces the objective function by determining the relative relevance of each decision tree added to the ensemble. XG Boost goes one step farther and increases accuracy and efficiency by including a regularization term and employing a more sophisticated optimization technique.
Because it can handle big datasets in a range of machine-learning tasks, such as regression and classification, it has become more well-known and widely used. [15]

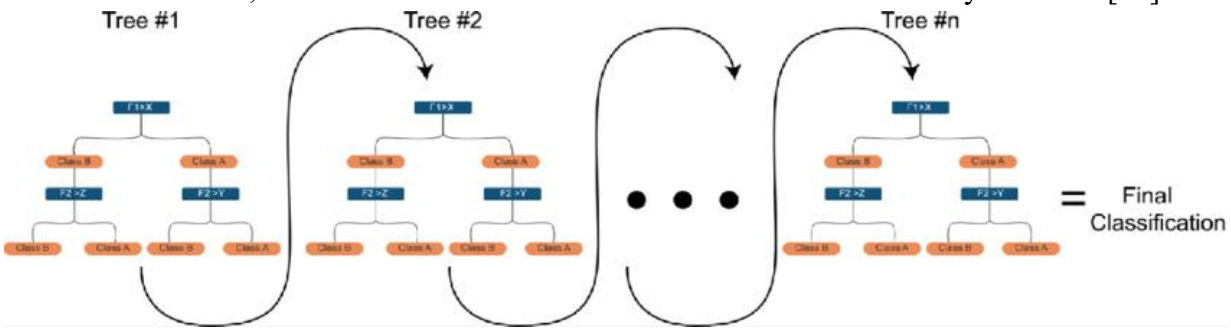


Figure 33 XGBoost

4.2.2.5 Bayesian Networks (BN)

A Bayesian network (also known as a Bayes network, Bayes net, belief network, or judgment network) is a probabilistic graphical model that depicts a collection of variables and their conditional connections using a directed acyclic graph (DAG). While causal networks are a type of causal notation, they are distinct from Bayesian networks. Bayesian networks are perfect for estimating the chance that an event occurred and that any of multiple potential known reasons contributed to it. For example, a Bayesian network might reflect the probability correlations between illnesses and symptoms. Given symptoms, the network can calculate the likelihood of the existence of certain illnesses.

Bayesian networks are directed acyclic graphs (DAGs) with nodes representing variables in the Bayesian sense, which might be observable quantities, latent variables, unknown parameters, or hypotheses. Each edge indicates a direct conditional reliance. Any pair of unconnected nodes (i.e., no route links one node to the other) represents variables that are conditionally independent of one another. Each node is coupled with a probability function that accepts a specific set of values for the node's parent variables and returns (as output) the probability (or probability distribution, if appropriate) of the variable represented by the node. For example, if there are m Boolean variables, the probability function might be represented by a table with 2 power m potential parent combinations. [16]

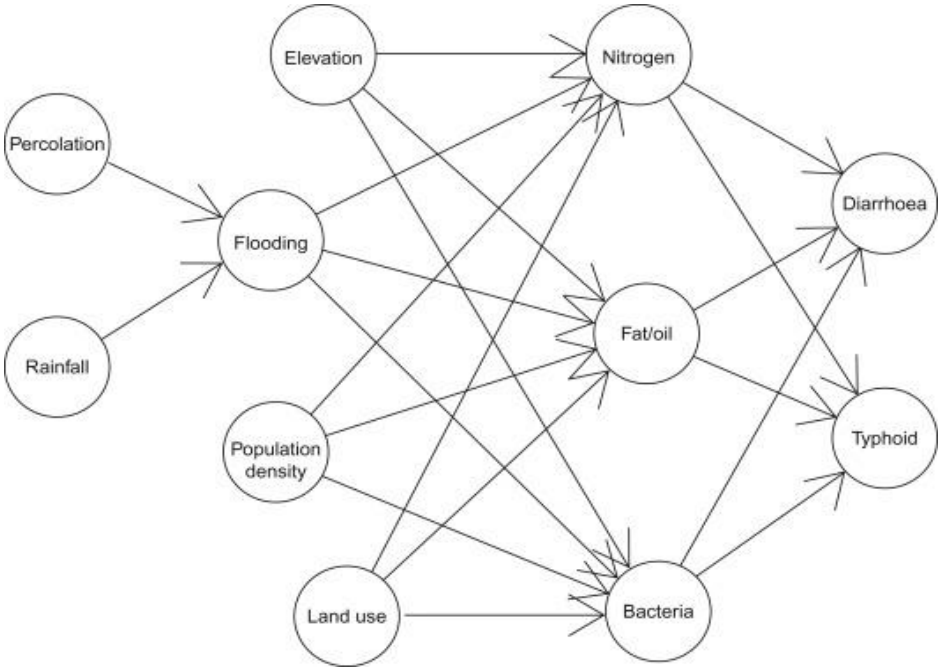


Figure 34 Bayesian Network

4.3 Model Developement

Three separate instances for each model has been developed, each following different preprocessing track. One for the original data with scaling, and two for two projection techniques.

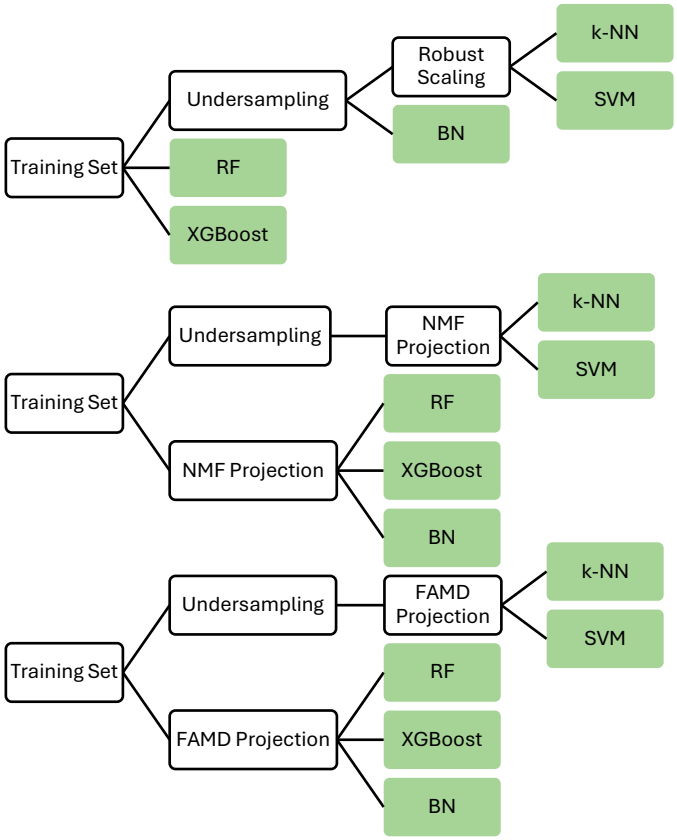


Figure 35 Overview of different preprocessing tracks in the project

4.3.1 Handling Imbalance

The data is imbalanced, which means that there are much more instances with infection than instances with no infection. This would cause many algoritms to be biased towards predicting the majority class (i.e., not having NAFLD) more often that they should. This can give an acceptable overall accuracy, but the accuracy of the minority class will be low compared to the accuracy of the majority class. This is a problem since, in our domain, misclassifying an individual as not having NAFLD (i.e., false negative) is worse than misclassifying an individual as having NAFLD.

This has been handled by setting class weights and, for models that don't support class or sample weights, performing random undersampling on the training set (before every training operation).

4.3.2 Hyperparameter Tuning

There are various techniques for hyperparameter tuning. The following figure shows an overview of some of the common techniques.

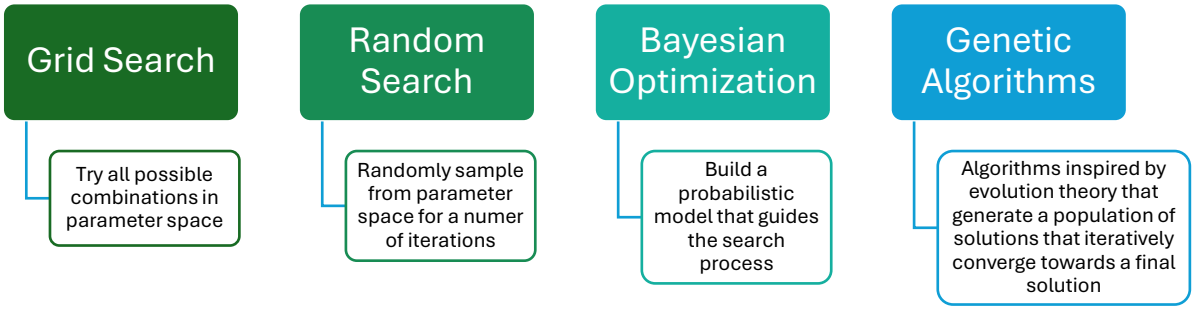


Figure 36 Common hyperparameter tuning techniques

In this project, Bayesian optimization is used, so, let's discuss Bayesian optimization in brief.

Bayesian optimization is a heuristic optimization method for black-box functions that is able to reach global optimal solution. It starts with an assummed distribution of the objective function $f(X)$ given the values of the variables $P(f(X)|X)$ and iteratively updates it through Gaussian process regression, also known as kriging. [17]

Gaussian process regression works by placing a Gaussian distribution of the value of $f(X)$ at every point x in the variables space, resulting in an infinite number of distributions (if at least one variable is continuous). The mean of the distribution at point x represents the expected value of $f(x)$ and the variance represents the degree of uncertainty.

On selecting a point x_i at iteration i, this point is evaluated and the posterior distributions are updated as follows:

- For the distribution at point x_i, the mean is set to $f(x_i)$ and variance is set to 0
- For other distributions, the mean and variance are also updated by means of a kernel function

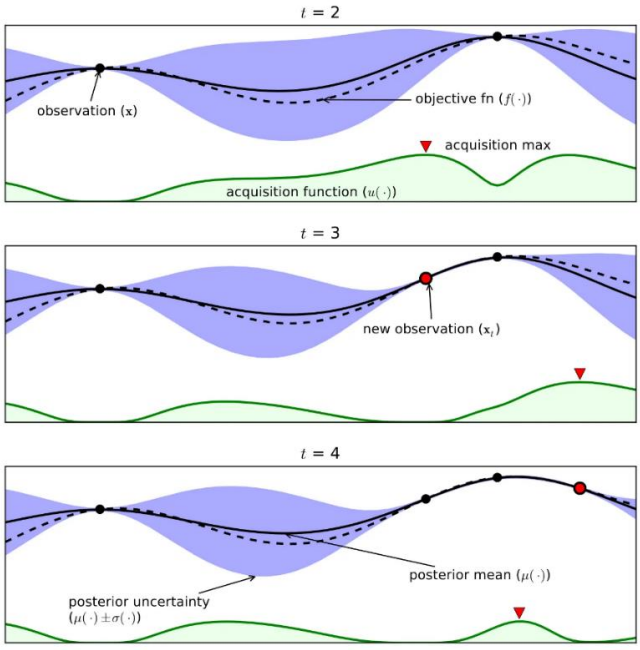


Figure 37 Few iterations of Bayesian optimizaiton with 1 variable

This posterior joint distribution of $f(X)$ is then used to form something called acquisition function, which manages the exploration and exploitation depending on the mean and variance values. This function is then maximized to determine the next point.

In the context of hyperparameter tuning, the variables are the hyperparameters and the objective function is the training and the subsequent scoring. The optimizer doesn't have any information about the underlying process of obtaining the score, it only works on the inputs an observes the output.

The following are the search spaces for each classifier:

- KNN

Hyperparameter	Possible Values
Number of neighbors	$[1, 200]$
Minkowski distance degree	$[1, 4]$

Table 21 KNN Search Space

- SVM

Hyperparameter	Possible Values
C	$[0.01, 100]$

Table 22 SVM Search Space

- Random Forest

Hyperparameter	Possible Values
Maximum number of leafs	$[2, 100]$
Criterion	Gini Impurity or Entropy

Table 23 Random Forest Search Space

- XGBoost

Hyperparameter	Possible Values
Maximum number of leafs	$[2, 100]$
Learning rate	$[0.01, 0.99]$

Table 24 XGBoost Search Space

- Bayesian Network

Space	Hyperparameter	Possible Values
1	DAG structure	Tree Augmented Naïve Bayes
	Number of discretization bins	$[2, 100]$
2	DAG structure	Bayesian Augmented Naïve Bayes and General Bayesian Network
	Structure score	K2, BDeu, BDs, BIC and AIC
	Number of discretization bins	$[1, 100]$
	Maximum allowed indegree	$[2, 10]$

Table 25 Bayesian Network Search Spaces

4.3.3 Ensemble Models

After models are trained, they are combined into a single ensemble model. An ensemble models have been developed for each preprocessing track.

The ensembles used are soft voting ensembles, which make predictions by averaging probabilities predicted by each included classifier. Weights can also assigned to each classifier, which makes final prediction affected differently by different classifiers. [18]

This can be mathematically expressed as follows:

$$\hat{y} = \underset{i}{\operatorname{argmax}} \sum_{\forall j} w_i p_{ij}$$

$\hat{y}$ is the prediction
i represents indices of classes
j represents indices of models

In this project, weights have been assigned for each classifier by dividing each classifier's score by the summation of the scores of all classifiers. The score used will be discussed later.

4.3.4 Wrapper Feature Selection

After building the ensembles, wrapper feature selection has been performed for the ensemble trained on original features based on features importances.

Features importances have been calculated using permutation importance method, which works by shuffling the values of one feature and calculating the difference in score to determine how much this feature contributes to the identification of the correct class. This is repeated for all features and can also be repeated multiple times for every feature multiple times to ensure the importance values are reliable. [19]

The following are average importance values for each feature, where importance is calculated 5 times. Based on the following feature importances, the most important 13 features have been selected and the ensemble has been rebuilt.

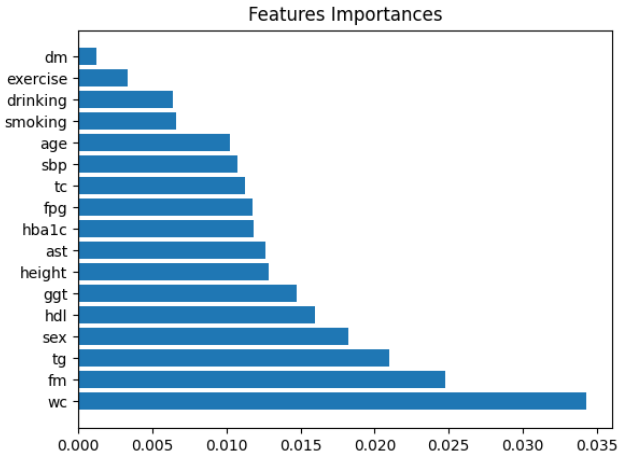


Figure 40 Bar chart of feature importances

4.4 Model Evaluation

There are various scores for evaluating classification models, such as accuracy, recall and F-scores. However, these scores only evaluate the correctness of predictions, which makes then inappropriate for evaluating models that predict probabilities.

There are scores for evaluating the confidence of the model rather than just the correctness. These are called scoring rules.

In this project, Brier score has been used. It is a loss function, so, the lower it is the better the predictions are. Its values are from 0 to 1. It is calculated for 2 classes as follows:

$$BS = \sum_{\forall i} (P(Y_i = 1) - y_i)^2$$

i represent indices of data records

For example, consider the probability of having NAFLD is 0.8:

- If the person has NAFLD, then $BS = (0.8 - 1)^2 = 0.04$
- If the person doesn't have NAFLD, then $BS = (0.8 - 0)^2 = 0.64$

Due to the class imbalance in testing and validation sets, Brier score can't be used as is. This can be handled by calculating the Brier score for positive instances and negative instances separately, then using their average.

Finally, since the Bayesian optimization implementation used performs maximization, the score has been modified as follows:

$$1 - \frac{BS^+ + BS^-}{2}$$

4.5 Results

After the hyperparamters had been selected, they were used to train models on the whole training set.

The following are Brier scores and the confusion matrices for each ensemble:

1st Ensemble (Scaling + Selection):

$BS^{+} = 0.11106636937574234$

$BS^{-} = 0.13253513265823616$

	True 0	True 1
Predicted 0	1870	72
Predicted 1	479	430
Recall	0.8	0.86

Table 26 Confusion Matrix of 1st Ensemble

2nd Ensemble (NMF):

$BS^{+} = 0.25951212066870105$

$BS^{-} = 0.08060367980423089$

	True 0	True 1
Predicted 0	2117	231
Predicted 1	232	271
Recall	0.9	0.54

Table 27 Confusion Matrix of 2nd Ensemble

3rd Ensemble (FAMD):

$BS^{+} = 0.11725455459548668$

$BS^{-} = 0.1363602160264297$

	True 0	True 1
Predicted 0	1856	76
Predicted 1	493	426
Recall	0.79	0.85

Table 28 Confusion Matrix of 3rd Ensemble

Based on the previous results, 1st ensemble has been selected for predictions.

CHAPTER 5
IMPLEMENTATION

In this chapter, we will discuss our implementation. Python programming language was used to implement the project.

5.1 Dependencies

- Handling Data and Preprocessing
  - Pandas: needed for better handling of the data.
  - Prince: implements various feature projection algorithms.
- Machine Learning
  - Scikit-learn: needed for various machine learning and preprocessing algorithms.
  - XGBoost
  - PGMPy: needed for the Bayesian network.
  - MLxtend: needed for ensembling the models.
- Training
  - Scikit-optimize: needed for Bayesian optimization.
  - Imbalanced-learn: needed for undersampling.

5.2 Modules

5.2.1 Constants

Maximum number of parallel jobs

```
NJOBS = 5
```

Feature sets

```
# ordinal features
ORDS = ['smoking', 'drinking']
# nominal features
NOMS = ['sex', 'dm', 'exercise']
# categorical features
CATS = NOMS + ORDS

# discrete features
DISCS = ['age', 'ast', 'ggt']
# continuous features
CONTS = ['height', 'fm', 'wc', 'hdl', 'tc', 'tg', 'fpg', 'hba1c', 'sbp']
# numercial features
NUMS = DISCS + CONTS
```

A dictionary of categorical variables and textual representations of their categories (needed for GUI)

```
CAT_DICT = {
    'drinking': ('None/Small', 'Light', 'Moderate'),
    'smoking': ("None", "Past", "Present"),
    'sex': ("Female", "Male"),
    'exercise': ("Plays exercies less often once a week", "Plays exercise at least once a week"),
    'dm': ("Doesn't have diabetes", "Has diabetes"),
    'nafld': ("Doesn't have NAFLD", "Has NAFLD")
}
```

A dictionary of categorical variables and text displayed in the GUI

```
TEXT_DICT = {
    'drinking': 'Drinking status',
    'smoking': 'Smoking status',
    'sex': 'Sex',
    'age': 'Age',
    'ast': 'AST (U/L)',
    'ggt': 'GGT (U/L)',
    'height': 'Height (cm)',
    'fm': 'Fat Mass (kg)',
    'wc': 'Waist Circumference (cm)',
    'hdl': 'HDL (mmol/L)',
    'tc': 'TC (mmol/L)',
    'tg': 'TG (mmol/L)',
    'fpg': 'FPG (mmol/L)',
    'hba1c': 'HbA1c (%)',
    'sbp': 'SBP (mmHg)'
}
```

5.2.2 Stratified Splitting

Import splitting function

```
from sklearn.model_selection import train_test_split
```

A function for stratified splitting

```
def split(X, y, test_size):
    return train_test_split(X, y, test_size=test_size, stratify=y)
    # test_size is the fraction of testing data
```

5.2.3 Classifiers Objects

Importing classifiers classes

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from modules.BNClassifier import BNClassifier
```

Classifiers objects dictionary:

```
clfs_dict = {
    'knn': KNeighborsClassifier(weights='distance', algorithm='brute'),
    'svm': SVC(probability=True, class_weight='balanced', kernel='linear'),
    'rf': RandomForestClassifier(class_weight='balanced'),
    'xgb': XGBClassifier(enable_categorical=True, scale_pos_weight=4.7),
    # scale_pos_weight is the weight of positive class
    'bn': BNClassifier()
}
```

5.2.4 Hyperparameter Tuning Constants

Import search spaces classes

```
from skopt.space import Categorical, Integer, Real
```

Real for hyperparameters that take continuous values

Integer for hyperparameters that take iteger values

Categorical for hyperparameters that have a limited set of possible values

Dictionary of search spaces *search_spaces*:

- *k*-NN

```
'knn': {'clf__n_neighbors': Integer(1, 200), 'clf__p': Integer(1, 4)},
```

- SVM

```
'svm': {'clf__C': Real(0.01, 100)},
```

- Random Forest

```
'rf': {
    'clf__max_leaf_nodes': Integer(2, 100),
    'clf__criterion': Categorical(['gini', 'entropy'])
},
```

- XGBoost

```
'xgb': {
    'clf__max_leaves': Integer(2, 100),
    'clf__learning_rate': Real(0.01, 0.99)
},
```

- Bayesian Network

```
'bn': [
    ({ # search space 1
        'clf__dag_structure': Categorical(['tan']),
        'clf__n_bins': Integer(1, 100)
    }, 25), # 25 iterations
    ({ # search space 2
        'clf__dag_structure': Categorical(['ban', 'gbn']),
        'clf__scoring_method': Categorical(
            ["k2score", "bdeuscore", "bdsscore", "bicscore", "aicscore"]
        ),
        'clf__n_bins': Integer(1, 100),
        'clf__max_indegree': Integer(2, 10)
    }, 100) # 100 iterations
]
```

Dictionary of the number of Bayesian optimization iterations for each variable

```
# unused for algorithms whose search spaces have specified number of iterations
n_iters = {
    'knn':  50,
    'svm':  25,
    'rf':   50,
    'xgb':  50,
```

```
    'bn':   None
}
```

5.2.5 Bayesian Network Classifier

```
from modules.constants import *
import pandas as pd
from sklearn.base import ClassifierMixin
from pgmpy.estimators import HillClimbSearch, TreeSearch
from pgmpy.models import BayesianNetwork
from pgmpy.estimators import BayesianEstimator
```

- *BNClassifier* Class

```
class BNClassifier(BayesianNetwork, ClassifierMixin):
```

- Constructor

```
def __init__(self, scoring_method='k2score', n_bins=6,
dag_structure='ban', max_indegree=3):
    super().__init__()
    self.scoring_method = scoring_method
    self.n_bins = n_bins
    self.dag_structure = dag_structure
    self.max_indegree = max_indegree
```

- Training function

```
def fit(self, X, y):
    # ensure all columns names have same datatype (string)
    X = X.rename(columns={col: str(col) for col in X.columns})

    # convert NAFLD to a Pandas Series of categorical datatype
    y = pd.Series(y, name='nafld', dtype=pd.CategoricalDtype())

    df = pd.concat([X, y], axis=1)

    # constructing discretization bins
    self.bins = {}
    for col in df.columns:
        # if feature is not already categorical
        if not isinstance(df[col].dtype, pd.CategoricalDtype):
            self.bins[col] = pd.qcut(df[col], self.n_bins, retbins=True, duplicates='drop')[1]
            # set the the first bin to -inf and last bin to inf
            # to accept any value
            self.bins[col][0] = -float("inf")
            self.bins[col][-1] = float("inf")

    # discretization
    df = self.discretize(df)

    # structure learning
    self.learn_structure(df, X.columns)

    # parameter learning
    super().fit(df, estimator=BayesianEstimator)

    return self
```

- Function to learn structure of the network

```
def learn_structure(self, df, features):
    if self.dag_structure == 'gbn':
        # learn the structure of a general bayesian network
```

```
        # through hill climbing
        learned_dag = HillClimbSearch(df).estimate(
            scoring_method=self.scoring_method,
            show_progress=False,
            max_indegree=self.max_indegree
        )

    elif self.dag_structure == 'ban':
        # learn the structure of a bayesian augmented naive bayes
        # through hill climbing with forced naive bayes edges
        learned_dag = HillClimbSearch(df).estimate(
            scoring_method=self.scoring_method,
            # force edges from NAFLD to all the features
            fixed_edges=[('nafld', col) for col in features],
            show_progress=False,
            max_indegree=self.max_indegree
        )

    elif self.dag_structure == 'tan':
        # learn the structure of a tree augmented naive bayes
        # through the chow-liu algorithm
        learned_dag = TreeSearch(df, n_jobs=1).estimate(
            estimator_type='tan',
            class_node='nafld'
        )

    # reset the network if already fitted
    while self.nodes():
        node = list(self.nodes)[0]
        self.remove_node(node)

    # add the learned edges
    self.add_edges_from(learned_dag.edges)

    return self
```

- Function to perform discretization

```
def discretize(self, df):
    # ensure all columns names have same datatype (string)
    df = df.rename(columns={col: str(col) for col in df.columns})

    discretized_df = df.copy()
    for col in df.columns:
        if col in self.bins.keys():
            discretized_df[col] = pd.cut(df[col], self.bins[col])
    return discretized_df
```

- Function to predict class

```
def predict(self, X):
    X = self.discretize(X)
    y_pred = super().predict(X).to_numpy()
    return y_pred
```

- Function to predict probability

```
def predict_proba(self, X):
    X = self.discretize(X)
    probs = super().predict_probability(X).to_numpy()
    return probs
```

- Function to predict probabilites of different categories of any variable

```
def predict_probability(self, df):
    df = self.discretize(df)
    return super().predict_probability(df)
```

- Function to get hyperparameters values

```
def get_params(self, deep=True):
    return {
        'scoring_method': self.scoring_method,
        'n_bins': self.n_bins,
        'dag_structure': self.dag_structure,
        'max_indegree': self.max_indegree
    }
```

- Function to set hyperparameters

```
def set_params(self, **params):
    for param in params:
        setattr(self, param[0], param[1])
    return self
```

5.2.6 Training and Testing

```
from modules.constants import *
from modules.constants_tuning import *
from modules.clfs import clfs_dict

from imblearn.pipeline import Pipeline
from imblearn.under_sampling import RandomUnderSampler
from skopt import BayesSearchCV
from mlxtend.classifier import EnsembleVoteClassifier
from sklearn.metrics import brier_score_loss
```

- Function to make a pipeline for each classsifer, pipelines allow to automate preprocessing by performing multiple steps sequentially

```
def make_pipe(clf_name, clf, preprocessing_step):
```

- *k*-NN and SVM

```
if (clf_name == 'knn' or clf_name == 'svm'):
    # pipeline steps: undersampling -> preprocessing -> classifier
    return Pipeline([('sampler', RandomUnderSampler()), preprocessing_step, ('clf', clf)])
```

- Random Forest and XGBoost

```
elif (clf_name == 'rf' or clf_name == 'xgb'):
    # if the preprocessing step is scaling
    if preprocessing_step[0] == 'scaler':
        # pipeline steps: classifier
        return Pipeline([('clf', clf)])
    # if the preprocessing step is not scaling
    else:
        # pipeline steps: preprocessing -> classifier
        return Pipeline([preprocessing_step, ('clf', clf)])
```

- BN

```
elif (clf_name == 'bn'):
    # if the preprocessing step is scaling
    if preprocessing_step[0] == 'scaler':
        # pipeline steps: undersampling -> classifier
        return Pipeline([('sampler', RandomUnderSampler()), ('clf', clf)])

    # if the preprocessing step is not scaling
    else:
        # pipeline steps: undersampling -> preprocessing -> classifier
        return Pipeline([('sampler', RandomUnderSampler()), preprocessing_step, ('clf', clf)])
```

- Function to perform hyperparameter tuning and training

```
def tune_train(X_train, y_train, X_test, y_test, preprocessing_step):
    preprocessing_step[1].set_output(transform='pandas')

    clf_pipes = []
    for clf_name, clf in clfs_dict.items():
        clf_pipe = make_pipe(clf_name, clf, preprocessing_step)
        optimizer = BayesSearchCV(
            clf_pipe,
            search_spaces[clf_name],
            # perform startified 5-fold cross-validation
            cv=5,
            n_iter=n_iters[clf_name],
            scoring=custom_brier,
            refit=False,
            n_jobs=NJOBS
        )
        optimizer.fit(X_train, y_train)

        # set hyperparameters to learned ones
        clf_pipe.set_params(**optimizer.best_params_)

        # create a new instance of FAMD because it fails to refit
        if preprocessing_step[0] == 'famd':
            from prince import FAMD
            famd = FAMD(preprocessing_step[1].n_components)
            clf_pipe.steps[-2] = ('famd', famd)

        clf_pipe.fit(X_train, y_train)
```

```
        clf_pipes.append(clf_pipe)

    # calculate scores
    scores = [custom_brier(clf_pipe, X_test, y_test) for clf_pipe in clf_pipes]

    # combining models into an ensemble
    weights = [score/sum(scores) for score in scores]
    ensemble = EnsembleVoteClassifier(
        clf_pipes,
        voting="soft",
        weights=weights,
        fit_base_estimators=False,
        use_clones=False
    ).fit(X_train, y_train)

    return ensemble
```

- Function to calculate the Brier score discussed

```
def custom_brier(clf, X, y):
    y_pred = clf.predict_proba(X)

    # calculate brier score for positive samples only
    brier_pos = brier_score_loss(y[y==1], y_pred[:,1][y==1])

    # calculate brier score for negative samples only
    brier_neg = brier_score_loss(y[y==0], y_pred[:,1][y==0])

    return 1 - (brier_pos + brier_neg)/2
```

5.2.7 Wrapper Feature Selection

```
from sklearn.inspection import permutation_importance
from modules.constants import NJOBS
from modules.train_test import custom_brier
import matplotlib.pyplot as plt
```

- Calulating permutation importance

```
def get_permutation_importances(est, some_X, some_y):
    results = permutation_importance(
        est,
        some_X,
        some_y,
        n_jobs=NJOBS,
        scoring=custom_brier
    )
    # get the average importance across all 5 repititions
    return results['importances_mean']
```

- Function to get list of columns sorted descendingly by their importance

```
def get_sorted_cols(cols, importances):
    return [index for _, index in sorted(zip(importances, cols), reverse=True)]
```

- Function to plot a bar chart of feature importances

```
def plot(cols, importances):
    sorted_cols = get_sorted_cols(cols, importances)
    plt.barh(sorted_cols, sorted(importances, reverse=True))
    plt.title("Features Importances")
    plt.show()
```

5.3 Basic Preprocessing

```
from modules.splitting import *
import pandas as pd
```

- Reading the data

```
df = pd.read_csv("datasets/raw.csv")
```

- Feature selection

```
redundent_features = [
    'bmi',
    'weight',
    'ethanol',
    'lbm',
    'alt',
    'dbp'
]

# delete the features
for feature in redundent_features:
    del df[feature]
```

- Making drinking and smoking start from 0

```
df['smoking'] -= 1
df['drinking'] -= 1
```

- Splitting the data into features and target variable

```
X = df.iloc[:,:-1]
y = df['nafld']
```

- Splitting the data into training and testing (1/5 for testing)

```
X_train, X_test, y_train, y_test = split(X, y, 0.2)
```

- Saving the data into CSV files

```
X_train.to_csv("datasets/X_train.csv", index=False)
X_test.to_csv("datasets/X_test.csv", index=False)
y_train.to_csv("datasets/y_train.csv", index=False)
y_test.to_csv("datasets/y_test.csv", index=False)
```

5.4 Reading Data

This code is used to read the data after it has overgone basic preprocessing

```
from modules.constants import *

import pandas as pd

# reading the data

X_train = pd.read_csv("datasets/X_train.csv")
X_test = pd.read_csv("datasets/X_test.csv")
y_train = pd.read_csv("datasets/y_train.csv")['nafld']
y_test = pd.read_csv("datasets/y_test.csv")['nafld']

# converting categorical features to datatypes expected by scikit-learn

X_train[NOMS] = X_train[NOMS].astype(pd.CategoricalDtype())
X_test[NOMS] = X_test[NOMS].astype(pd.CategoricalDtype())
X_train[ORDS] = X_train[ORDS].astype(pd.CategoricalDtype(ordered=True))
X_test[ORDS] = X_test[ORDS].astype(pd.CategoricalDtype(ordered=True))
```

5.5 Feature Projection - Determining Number of Components

This code has been used to test NMF and FAMD with different number of components to help decide what number of components to use

```
from modules.constants import *
from _read import *

from sklearn.decomposition import NMF
from prince import FAMD
```

- NMF

```
# perform nmf for a number of components from 1 to 11
for n in range(1, 12):
    nmf = NMF(n, init='nndsvd', max_iter=100000)
    nmf.fit_transform(X_train, y_train)
    # print reconstruction error
    print(nmf.reconstruction_err_)
```

- FAMD

```
# converting nominal features to integer type to be treated as categorical
X_train[CATS] = X_train[CATS].astype("int")
# converting numerical features to integer type to be treated as numerical
X_train[NUMS] = X_train[NUMS].astype("float")
# converting ordinal features to integer type to be treated as numerical
X_train[ORDS] = X_train[ORDS].astype("float")
# perform famd for a number of components from 1 to 11
for n in range(1, 12):
    famd = FAMD(n)
    famd.fit_transform(X_train, y_train)
```

```
    # print total percentage of explained variance
    print(famd.cumulative_percentage_of_variance_[-1])
```

5.6 Training Ensembles

```
from modules.train_test import *
from modules.selection import *
from _read import *

from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import RobustScaler
from sklearn.decomposition import NMF
from prince import FAMD
import pickle
```

- 1st Ensemble (Scaling)

```
# hyperparameter tuning and training
ensemble_scaled = tune_train(
    X_train,
    y_train,
    X_test,
    y_test,
    ('scaler', ColumnTransformer([('scaler', RobustScaler(), NUMS+ORDS)], remainder='passthrough', verbose_feature_names_out=False))
)

# saving ensemble
with open("../models/ensemble_scaled.pkl", 'wb') as f:
    pickle.dump(ensemble_scaled, f)
```

- 2nd Ensemble (NMF)

```
# NMF object
nmf = NMF(7, init='nndsvd', max_iter=100000)

# hyperparameter tuning and training
ensemble_nmf = tune_train(
    X_train,
    y_train,
    X_test,
    y_test,
    ('nmf', ColumnTransformer([('nmf', nmf, NUMS)], remainder='passthrough', verbose_feature_names_out=False))
)

# saving ensemble
with open("../models/ensemble_nmf.pkl", 'wb') as f:
    pickle.dump(ensemble_nmf, f)
```

- 3rd Ensemble (FAMD)

```
# converting nominal features to integer type to be treated as categorical
X_train[NOMS] = X_train[NOMS].astype("int")
X_test[CATS] = X_test[CATS].astype("int")
# converting numerical features to integer type to be treated as numerical
X_train[NUMS] = X_train[NUMS].astype("float")
X_test[NUMS] = X_test[NUMS].astype("float")
# converting ordinal features to integer type to be treated as numerical
X_train[ORDS] = X_train[ORDS].astype("float")
X_test[ORDS] = X_test[ORDS].astype("float")
# FAMD object
famd = FAMD(11)

# hyperparameter tuning and training
ensemble_famd = tune_train(X_train, y_train, X_test, y_test, ('famd', famd))

# saving ensemble
with open("../models/ensemble_famd.pkl", 'wb') as f:
    pickle.dump(ensemble_famd, f)
```

5.7 Wrapper Feature Selection

```
from modules.train_test import *
import modules.selection as selection
from _read import *
from modules.train_test import *

from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import RobustScaler
import pickle
```

- Read 1st Ensemble

```
with open("../models/ensemble_scaled.pkl", 'rb') as f:
    ensemble_scaled = pickle.load(f)
```

- Perform Wrapper Feature Selection

```
# get features importance
importances = selection.get_permutation_importances(ensemble_scaled, X_train, y_train)

# plot feature importances
selection.plot(X_train.columns, importances)

# prompt to get number of features desired
n = int(input("Enter number of features to select: "))

sorted_cols = selection.get_sorted_cols(X_train.columns, importances)

SELECTION = sorted_cols[:n]

X_selection_train = X_train.loc[:,SELECTION]
X_selection_test = X_test.loc[:,SELECTION]
```

- Retrain 1st Ensemble

```
NUMS_SELECTION = [col for col in NUMS if col in SELECTION]
ORDS_SELECTION = [col for col in ORDS if col in SELECTION]

# hyperparameter tuning and training
ensemble_selection = tune_train(
    X_selection_train,
    y_train,
    X_selection_test,
    y_test,
    ('scaler', ColumnTransformer([('scaler', RobustScaler(), NUMS_SELECTION+ORDS_SELECTION)], remainder='passthrough', verbose_feature_names_out=False))
)

# saving ensemble
with open("../models/ensemble_selection.pkl", 'wb') as f:
    pickle.dump(ensemble_selection, f)
```

5.8 Results

```
from _read import *
from modules.train_test import custom_brier

from sklearn.metrics import ConfusionMatrixDisplay
import pickle
```

- 1st Ensemble (Scaling + Selection)

```
# read ensemble 1
with open("../models/ensemble_selection.pkl", 'rb') as f:
    ensemble_selection = pickle.load(f)

SELECTION = ['wc', 'fm', 'tg', 'sex', 'hdl', 'ggt', 'height', 'ast', 'hba1c', 'fpg', 'tc', 'sbp', 'age']

# calculate Brier score for each class
```

```
y_pred = ensemble_selection.predict_proba(X_test)
print(brier_score_loss(y_test[y_test==1], y_pred[:,1][y_test==1]))
print(brier_score_loss(y_test[y_test==0], y_pred[:,1][y_test==0]))
# construct confusion matrix
print(confusion_matrix(y_test, ensemble_selection.predict(X_test)))
```

- 2nd Ensemble (NMF)

```
# read ensemble 2
with open("../models/ensemble_nmf.pkl", 'rb') as f:
    ensemble_nmf = pickle.load(f)

# calculate Brier score for each class
y_pred = ensemble_nmf.predict_proba(X_test)
print(brier_score_loss(y_test[y_test==1], y_pred[:,1][y_test==1]))
print(brier_score_loss(y_test[y_test==0], y_pred[:,1][y_test==0]))
# construct confusion matrix
print(confusion_matrix(y_test, ensemble_nmf.predict(X_test)))
```

- 3rd Ensemble (FAMD)

```
# read ensemble 3
with open("../models/ensemble_famd.pkl", 'rb') as f:
    ensemble_famd = pickle.load(f)

# converting nominal features to integer type to be treated as categorical
X_test[CATS] = X_test[CATS].astype("int")
# converting numerical features to integer type to be treated as numerical
X_test[NUMS] = X_test[NUMS].astype("float")
# converting ordinal features to integer type to be treated as numerical
X_test[ORDS] = X_test[ORDS].astype("float")

# calculate Brier score for each class
y_pred = ensemble_famd.predict_proba(X_test)
print(brier_score_loss(y_test[y_test==1], y_pred[:,1][y_test==1]))
print(brier_score_loss(y_test[y_test==0], y_pred[:,1][y_test==0]))
# construct confusion matrix
print(confusion_matrix(y_test, ensemble_famd.predict(X_test)))
```

5.9 User Interface

```
from tkinter import Tk, IntVar, Listbox, Toplevel, Text
from tkinter import ttk
from tkinter.messagebox import showinfo, showerror
from modeling.modules.constants import *

# all features
SELECTION = ['wc', 'fm', 'tg', 'sex', 'hdl', 'ggt', 'height', 'ast', 'hba1c', 'fpg', 'tc', 'sbp', 'age']

# all columns
COLS = SELECTION + ['nafld']

# updated feature sets
NOMS = [col for col in NOMS if col in SELECTION]
ORDS = [col for col in ORDS if col in SELECTION]
CATS = NOMS + ORDS
DISCS = [col for col in DISCS if col in SELECTION]
CONTS = [col for col in CONTS if col in SELECTION]
NUMS = DISCS + CONTS
```

- Application Class
    - Constructor

```
def __init__(self):
    super().__init__()
    self.columnconfigure(0, weight=1)
    self.rowconfigure(0, weight=1)
    self.title("NAFLD Prediction")
    self.geometry('600x450')

    self.notebook = ttk.Notebook(self)
```

```
    self.notebook.grid(sticky="wens")

    self.columns_widgets = {}
    self.columns_vars = {}
```

- Main Frame

```
frame_main = ttk.Frame(self)
frame_main.columnconfigure(0, weight=1)
frame_main.columnconfigure(1, weight=1)
frame_main.rowconfigure(0, weight=1)
self.notebook.add(frame_main, text="Main")

frame_general = ttk.LabelFrame(frame_main, text="General")
frame_general.columnconfigure(0, weight=1)
frame_general.grid(sticky="wens", padx=5, pady=5)

frame_laboratory = ttk.LabelFrame(frame_main, text="Laboratory")
frame_laboratory.columnconfigure(0, weight=1)
frame_laboratory.grid(sticky="wens", row=0, column=1, padx=5, pady=5)

ttk.Button(frame_main, text="Predict", command=self.classify).grid(sticky="we", padx=5, columnspan=2)
```

- General Frame

```
ttk.Label(frame_general, text=TEXT_DICT['sex']).grid(sticky='w', padx=5, pady=(5, 0))
self.columns_widgets['sex'] = ttk.Combobox(frame_general, values=CAT_DICT['sex'])
self.columns_widgets['sex'].grid(sticky="we", padx=5)

ttk.Label(frame_general, text=TEXT_DICT['age']).grid(sticky='w', padx=5)
self.columns_widgets['age'] = ttk.Spinbox(frame_general, to=float("inf"))
self.columns_widgets['age'].grid(sticky="we", padx=5)

ttk.Label(frame_general, text=TEXT_DICT['height']).grid(sticky='w', padx=5)
self.columns_widgets['height'] = ttk.Spinbox(frame_general, to=float("inf"))
self.columns_widgets['height'].grid(sticky="we", padx=5)

ttk.Label(frame_general, text=TEXT_DICT['fm']).grid(sticky='w', padx=5)
self.columns_widgets['fm'] = ttk.Spinbox(frame_general, to=float("inf"))
self.columns_widgets['fm'].grid(sticky="we", padx=5)

ttk.Label(frame_general, text=TEXT_DICT['wc']).grid(sticky='w', padx=5)
self.columns_widgets['wc'] = ttk.Spinbox(frame_general, to=float("inf"))
self.columns_widgets['wc'].grid(sticky="we", padx=5)
```

- Laboratory Frame

```
ttk.Label(frame_laboratory, text=TEXT_DICT['ast']).grid(sticky='w', padx=5)
self.columns_widgets['ast'] = ttk.Spinbox(frame_laboratory, to=float("inf"))
self.columns_widgets['ast'].grid(sticky="we", padx=5)

ttk.Label(frame_laboratory, text=TEXT_DICT['ggt']).grid(sticky='w', padx=5)
self.columns_widgets['ggt'] = ttk.Spinbox(frame_laboratory, to=float("inf"))
self.columns_widgets['ggt'].grid(sticky="we", padx=5)

ttk.Label(frame_laboratory, text=TEXT_DICT['hdl']).grid(sticky='w', padx=5)
self.columns_widgets['hdl'] = ttk.Spinbox(frame_laboratory, to=float("inf"))
self.columns_widgets['hdl'].grid(sticky="we", padx=5)

ttk.Label(frame_laboratory, text=TEXT_DICT['tc']).grid(sticky='w', padx=5)
self.columns_widgets['tc'] = ttk.Spinbox(frame_laboratory, to=float("inf"))
self.columns_widgets['tc'].grid(sticky="we", padx=5)

ttk.Label(frame_laboratory, text=TEXT_DICT['tg']).grid(sticky='w', padx=5)
self.columns_widgets['tg'] = ttk.Spinbox(frame_laboratory, to=float("inf"))
self.columns_widgets['tg'].grid(sticky="we", padx=5)

ttk.Label(frame_laboratory, text=TEXT_DICT['fpg']).grid(sticky='w', padx=5)
self.columns_widgets['fpg'] = ttk.Spinbox(frame_laboratory, to=float("inf"))
```

```
self.columns_widgets['fpg'].grid(sticky="we", padx=5)

ttk.Label(frame_laboratory, text=TEXT_DICT['hba1c']).grid(sticky='w', padx=5)
self.columns_widgets['hba1c'] = ttk.Spinbox(frame_laboratory, to=float("inf"))
self.columns_widgets['hba1c'].grid(sticky="we", padx=5)

ttk.Label(frame_laboratory, text=TEXT_DICT['sbp']).grid(sticky='w', padx=5)
self.columns_widgets['sbp'] = ttk.Spinbox(frame_laboratory, to=float("inf"))
self.columns_widgets['sbp'].grid(sticky="we", padx=5)
```

- Prediction Frame

```
frame_pred = ttk.Frame(self)
frame_pred.columnconfigure(0, weight=1)
frame_pred.rowconfigure(2, weight=1)
self.notebook.add(frame_pred, text="Prediction")

ttk.Label(frame_pred, text=
            "Here, you can predict any selected variable(s) given "
            "the values of other variables by using Bayesian Network."
).grid(sticky="w", padx=5, pady=(5, 0))

ttk.Label(frame_pred, text="Select what to predict").grid(sticky='w', padx=5)
self.prediction_selection = Listbox(frame_pred, selectmode='multiple')
self.prediction_selection.grid(sticky="wens", padx=5)
for col in reversed(COLS):
    self.prediction_selection.insert(0, col)

self.columns_vars['nafld'] = IntVar(self)
self.columns_widgets['nafld'] = ttk.Checkbutton(frame_pred, variable=self.columns_vars['nafld'],
text="Has NAFLD")
self.columns_widgets['nafld'].grid(sticky="we", padx=5)

ttk.Button(frame_pred, text="Show Bayesian Network", command=self.plot).grid(sticky="wes", padx=5)
ttk.Button(frame_pred, text="Predict", command=self.predict).grid(sticky="we", padx=5)
```

- Load Ensemble

```
import sys, os, pickle
sys.path.append(os.path.join(os.getcwd(), "modeling"))
with open("models/ensemble_selection.pkl", 'rb') as f:
    self.ensemble = pickle.load(f)
```

- Function to bring data from input widgets into a Pandas DataFrame

```
def get_data(self):
    import pandas as pd
    sample_df = pd.DataFrame()
    try:
        for col in COLS:
            widget = self.columns_widgets[col]
            if type(widget) is ttk.Checkbutton:
                sample_df[col]=[self.columns_vars[col].get()]
            else:
                if col in CAT_DICT.keys():
                    sel = widget.get()
                    if sel: sample_df[col]=[CAT_DICT[col].index(sel)]
                else:
                    val = widget.get()
                    sample_df[col]=[val]

        sample_df[DISCS] = sample_df[DISCS].astype('int')
        sample_df[CONTS] = sample_df[CONTS].astype('float')
        sample_df[NOMS] = sample_df[NOMS].astype(pd.CategoricalDtype())
        sample_df[ORDS] = sample_df[ORDS].astype(pd.CategoricalDtype(ordered=True))
    except ValueError:
        showerror("Invalid Input", "One or more variables have invalid input.")
        return
    for col in NUMS:
```

```
        if (sample_df[col] < 0).any():
            showerror("Invalid Input", f"Value of {TEXT_DICT[col]} is negative")
            return
    return sample_df
```

- Function to perform classification by providing the probability of positive class

```
def classify(self):
    sample_df = self.get_data()
    if sample_df is None: return
    del sample_df['nafld']

    probs = self.ensemble.predict_proba(sample_df)[0]
    showinfo("Result", f"Probability of having NAFLD is {round(probs[1], 4)}")
```

- Function to predict probabilities of different categories of desired variable(s) using Bayesian Network

```
def predict(self):
    sample_df = self.get_data()
    if sample_df is None: return

    # construct a list of variables to predict
    selected_vars = []
    # for every desired variable
    for sel in self.prediction_selection.curselection():
        # delete variable from dataframe
        del sample_df[COLS[sel]]
        selected_vars.append(COLS[sel])

    # predict variables missing from dataframe (i.e., desired variables)
    predictions = self.ensemble.clfs[-1]['clf'].predict_probability(sample_df)

    # Results Window

    res = Toplevel(self)
    res.title("Results")
    res.columnconfigure(0, weight=1)
    res.rowconfigure(0, weight=1)

    # Results Widget

    res_text = Text(res, width=30, height=10, background=self['background'])
    # for every desired variable
    for var in selected_vars:
        # append variable's name to the widget's text
        res_text.insert('end', var+'\n')

        # for every column in the prediction dataframe
        for pred_col in predictions.columns:
            # if this column is a category of this variable
            if pred_col.startswith(var):
                # extract category name
                category = pred_col.removeprefix(var)[1:]
                # add the probability of the category to the text
                if var in CAT_DICT.keys():
                    text = "{:<80}{}\n".format(
                        "  Probability of "+CAT_DICT[var][int(category)],
                        round(predictions[pred_col][0], 4)
                    )
                else:
```

```
                    text = "{:<80}{}\n".format(
                        "  Probability of being within "+category,
                        round(predictions[pred_col][0], 4)
                    )
                # append the text to the widget's text
                res_text.insert('end', text)

    # disable editing of the widget's text
    res_text['state'] = 'disabled'

    res_text.grid(sticky='wens')
```

- Function to plot the learned Bayesian Network

```
def plot(self):
    import networkx as nx
    import matplotlib.pyplot as plt
    bn = self.ensemble.clfs[-1]['clf']
    pos = nx.shell_layout(bn)
    nx.draw_networkx(bn, pos, node_size=2300)
    plt.tight_layout()
    plt.axis(False)
    plt.show()
```

- Create the main window an wait for interaction

```
app = App()
app.mainloop()
```

CHAPTER 6
CONCLUSION AND FUTURE WORK

6.1 Conclusion

The development of a machine learning-based predictive system for Non-Alcoholic Fatty Liver Disease (NAFLD) represents a significant step forward in the early detection and management of this increasingly prevalent condition. By leveraging advanced data preprocessing techniques, feature selection, and ensemble modeling, this project aims to provide a reliable and accurate tool for predicting NAFLD risk. The integration of clinical, biochemical, and imaging data into a user-friendly interface will enable healthcare providers to make informed decisions and intervene early, potentially preventing the progression of NAFLD to more severe liver conditions such as cirrhosis and hepatocellular carcinoma.

Key accomplishments of this project include:

- Improved Early Detection:
    - The system will enable early identification of individuals at risk of NAFLD, allowing for timely intervention and management. This can significantly reduce the risk of disease progression and associated complications.
- Cost-Effective Solution
    - By providing a non-invasive and automated prediction tool, the system can reduce the need for expensive and time-consuming diagnostic procedures, making it accessible to a wider population.
- Enhanced Decision-Making:
    - Healthcare providers will benefit from a tool that offers interpretable predictions, helping them understand the factors contributing to a patient's risk and guiding them in developing personalized treatment plans.
- Scalability and Accessibility:
    - The system is designed to be scalable and can be deployed in various healthcare settings, from large hospitals to smaller clinics. This ensures that the benefits of the tool can reach a broad audience, including underserved populations.
- Advancement in Healthcare Technology:
    - This project contributes to the growing field of AI and machine learning in healthcare, demonstrating the potential of these technologies to address complex medical challenges. It also sets a foundation for future research and development in predictive modeling for other chronic diseases.
- Personal and Professional Growth:
    - As a graduation project, this work provides an opportunity to apply theoretical knowledge in machine learning and data science to a real-world problem. It enhances skills in data preprocessing, model development, and system design, which are highly valuable in the fields of AI and healthcare technology.

6.2 Future Work

While this project has successfully developed a predictive system for Non-Alcoholic Fatty Liver Disease (NAFLD), several areas can be explored to enhance its effectiveness and impact:

- Integration of Real-Time Data
    - Incorporating real-time patient data from wearable devices and electronic health records can improve prediction accuracy and enable continuous monitoring of at-risk individuals.
- Deep Learning for Enhanced Accuracy
    - Future iterations of this model can explore deep learning techniques such as Convolutional Neural Networks (CNNs) for medical imaging analysis or Recurrent Neural Networks (RNNs) for sequential patient history data.
- Explainable AI for Better Interpretability
    - Implementing Explainable AI (XAI) techniques will make the model's predictions more transparent and interpretable, allowing healthcare professionals to trust and understand the decision-making process.
- Incorporation of Genomic Data
    - Integrating genetic and epigenetic data can enhance risk assessment by identifying hereditary factors contributing to NAFLD susceptibility.
- Personalized Treatment Recommendations
    - Beyond risk prediction, the system can be expanded to provide personalized lifestyle and treatment recommendations based on a patient's risk profile, medical history, and response to interventions.
- Mobile Application for Patient Engagement
    - Developing a mobile app can allow patients to track their liver health, receive lifestyle recommendations, and stay engaged in their own care through AI-powered insights.
- Multi-Disease Prediction System
    - Extending the model to predict other liver-related diseases or metabolic disorders, such as Type 2 Diabetes or Metabolic Syndrome, could significantly increase its utility in preventive healthcare.
- Clinical Validation and Deployment
    - Collaborating with medical institutions for real-world testing and validation will be crucial for improving the model's reliability and securing regulatory approval for clinical use.

REFERENCES

[1] https://datadryad.org/stash/dataset/doi:10.5061/dryad.8q0p192

[2] https://doi.org/10.1038/s41366-018-0076-3

[3] https://doi.org/10.1186/s12944-022-01660-8

[4] https://doi.org/10.1186/s12876-022-02393-9

[5] https://doi.org/10.1186/s12967-022-03611-4

[6] https://doi.org/10.3389/fendo.2022.1020253

[7] https://doi.org/10.3389/fnut.2023.1103665

[8] https://doi.org/10.1186/s12876-022-02216-x

[9] https://doi.org/10.1186/s12902-024-01554-z

[10] https://en.wikipedia.org/wiki/Feature_scaling#Robust_Scaling

[11] https://www.studytrigger.com/article/types-of-learning-in-machine-learning/

[12] https://www.ibm.com/think/topics/knn

[13] https://www.ibm.com/think/topics/support-vector-machine

[14] https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/

[15] https://www.geeksforgeeks.org/ml-xgboost-extreme-gradient-boosting/

[16] https://en.wikipedia.org/wiki/Bayesian_network

[17] https://en.wikipedia.org/wiki/Bayesian_optimization

[18] https://rasbt.github.io/mlxtend/user_guide/classifier/EnsembleVoteClassifier/

[19] https://scikit-learn.org/stable/modules/permutation_importance.html