



CMR UNIVERSITY

Private University Established in Karnataka State by Act No. 45 of 2013

SCHOOL OF ENGINEERING AND TECHNOLOGY

A REPORT

On

"Celiac Disease Classification"

Submitted in partial fulfillment of the requirements for the Course

Machine Learning (4ENIT3101) in

Bachelor of Technology

In

Information Technology

SoET, CMR University, Bangalore

Submitted by:

Akshith G (21BBTIT004)

Roshni Reju (21BBTIT034)

Syed Zeeshan Ahmed (21BBTIT044)

Under the Supervision:

Bhagya K

Asst. Professor

Department of Information Technology

**Off Hennur - Bagalur Main Road,
Near Kempegowda International Airport, Chagalahatti,
Bangalore, Karnataka-562149**

2023-2024



CMR UNIVERSITY

Private University Established in Karnataka State by Act No. 45 of 2013

SCHOOL OF ENGINEERING AND TECHNOLOGY

Chagalahatti, Bengaluru, Karnataka- 562149

Department of Information Technology

CERTIFICATE

This is to certify that the study Report entitled “**Celiac Classification Disease**”, is a record of work successfully carried out by **Akshith G (21BBTIT004)**, **Roshni Reju (21BBTIT034)**, **Syed Zeeshan Ahmed (21BBTIT044)** in partial fulfillment of the requirement for the course **MACHINE LEARNING (4ENIT3101)** of Bachelor of Technology in Information Technology, SoET, CMR University, Bangalore during the academic year 2023-24, under the supervision and guidance of **Bhagya K**, Asst. Professor, IT, SoET, CMR University.

Signature

Bhagya K,
Asst. Professor,
Dept of IT, SoET,
CMR University

TABLE OF CONTENTS

Chapter No	Title	Page No
	ABSTRACT	1
1	INTRODUCTION 1.1 Background and Context 1.2 Objectives of the study 1.3 Problem Statement	 2-7 7 7
2	LITERATURE SURVEY 2.1 Model_1 2.2 Model_2 2.3 Model_3 2.4 Model_4 2.5 Model_5 2.6 Model_6 2.7 Model_7 2.8 Model_8	 8-9 10-11 12-13 14-15 16-17 18-19 20-21 22-23
3	DESIGN 3.1 Methodology 3.2 Flow Diagram relating to methodology 3.3 System Diagram	 24-25 26 27
4	IMPLEMENTATION 4.1 Model Selection and Justification 4.2 Model Training 4.3 Model Evaluation Metrics 4.4 Code	 28 28-30 30 31-32
5	RESULT ANALYSIS 5.1 Performance Metrics 5.2 Comparison/Interpretation of Models	 33-34 35-37
6	CONCLUSION	38
7	REFERENCES	39-40

LIST OF FIGURES

Figure no	Title	Page no
1	Effect of Celiac disease on a small intestine	2
2	Worldwide prevalence of Celiac disease	3
3	Celiac disease can lead to Osteoporosis	4
4	Some machine learning algorithms used for Classification	6
5	Random Forest Classifier Algorithm	9
6	Support Vector Machine Algorithm	11
7	Linear Regression Algorithm	12
8	Prediction ROC Curve for celiac patients	13
9	K Nearest Neighbour Algorithm	15
10	Working of Decision Trees	17
11	Naive Bayes Classifier	19
12	Gradient Boosting Machine algorithm	21
13	Linear Discriminant Analysis in Machine Learning	23
14	Process Flow Diagram	26
15	Simple architecture of the System	27
16	Attributes of the dataset used	36
17	Frequency plot of Celiac Disease	36
18	Using Random Forest classifier	37
19	Final Output	37

LIST OF TABLES

Table no	Title	Page no
1	Children at risk for CD at different ages	5
2	Interpretation of Models	35

ABSTRACT

This abstract explores the application of machine learning techniques, specifically focusing on the Random Forest algorithm, for classifying and diagnosing Celiac disease using clinical data. Utilizing a diverse dataset containing demographic details, symptom profiles, and laboratory results from individuals with and without Celiac disease, the study refines its inputs to optimize model performance through preprocessing and feature engineering. The Random Forest algorithm, known for its ability to handle complex classification tasks, is employed to fuse predictions from multiple decision trees, achieving good performance across critical evaluation metrics such as accuracy, sensitivity, and specificity. The study highlights the potential of machine learning, particularly the Random Forest paradigm, in improving Celiac disease diagnosis. The implications of this research extend to potential improvements in patient outcomes and healthcare delivery systems.

This case study report presents a novel application of machine learning techniques for the classification of Celiac Disease, a prevalent autoimmune disorder triggered by the ingestion of gluten. The study leverages a Random Forest model, a powerful ensemble learning method known for its robustness and accuracy in handling complex classification problems. The model was trained on a comprehensive dataset comprising various health indicators and serological test results of individuals, both with and without Celiac Disease. The objective was to identify patterns and correlations in the data that could aid in the early detection and diagnosis of this condition. In the second phase of the study, the performance of the Random Forest model was evaluated using various metrics such as accuracy, precision, recall, and F1-score. The model demonstrated high predictive power, outperforming several other machine learning algorithms in terms of these metrics. The study also highlighted the importance of certain features, such as tissue Transglutaminase (tTG) antibody levels, in the classification of Celiac Disease. The findings of this study underscore the potential of machine learning, and specifically the Random Forest model, in revolutionizing healthcare diagnostics and paving the way for personalized medicine. The report concludes with a discussion on the implications of these findings and suggestions for future research in this domain.

Keywords:

Celiac disease, Machine learning techniques, Random Forest algorithm, Clinical data, Dataset Demographic details, Symptom profiles, Laboratory results, Autoimmune disorder, Gluten ingestion, Ensemble learning method, Serological test results, Early detection, Predictive power, Precision, Recall, F1-score, Feature importance, Tissue Transglutaminase (tTG) antibody levels, Healthcare diagnostics.

CHAPTER – 1

INTRODUCTION

1.1 Background and Context

Background

Celiac disease is an autoimmune condition that affects the small intestine, triggered by consuming gluten, a protein found in wheat, barley, and rye. Its history dates back centuries, but it was only in the mid-20th century that its association with gluten was recognized. This chronic condition has become increasingly recognized over time, with prevalence varying across regions and populations. Patients are required to follow a gluten-free diet life-long. Nutritionals can not be absorbed sufficiently as the result of villous atrophy.

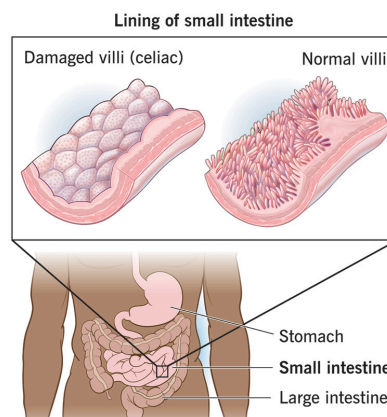


Figure 1: Effect of Celiac disease on a small intestine

While the symptoms of the disease are similar to many other diseases and these symptoms are different in each individual, it is very difficult to diagnose. 80-90% of the patients are still underdiagnosed while only %10 of the patients know that they have celiac disease. In a serologic screening research, involving more than 17,000 Italian schoolchildren, the ratio of individuals who know their disease to those who do not know is 1/7.[1]

Celiac disease has a long and fascinating history that stretches back centuries, with its origins intertwined with humanity's relationship with grains. While the term "celiac disease" itself is relatively modern, the symptoms and consequences of the condition have been recognized for millennia. One intriguing aspect of the history of celiac disease is how its symptoms were often observed and documented throughout history without necessarily being understood as part of a single underlying condition. Historical accounts describe symptoms such as chronic diarrhea, malnutrition, and wasting, which are now recognized as hallmarks of celiac disease. Some medical texts referred to "coeliac affections" or "chronic bowel complaints," indicating an awareness of a chronic condition affecting the digestive system but without a clear understanding of its origins or mechanisms.

For instance, in ancient Greece, there were records of individuals experiencing symptoms consistent with celiac disease after consuming certain grains. However, the cause of these symptoms was not fully understood at the time. Similarly, in the 19th and early 20th centuries, physicians observed cases of "coeliac affections" or "coeliac syndrome," but the connection to gluten-containing grains was not yet established. Ancient civilizations, such as the Greeks and Romans, documented cases of individuals experiencing digestive distress after consuming grains. However, it wasn't until the late 19th and early 20th centuries that significant advancements were made in understanding the condition. In 1888, British physician Samuel Gee provided one of the earliest clinical descriptions of celiac disease, recognizing it as a chronic illness affecting the small intestine. Gee's work laid the foundation for further research into the condition.

During the mid-20th century, pivotal discoveries shed light on the role of gluten in triggering the immune response seen in celiac disease. In 1950, Dutch pediatrician Willem Karel Dicke observed that children with celiac disease experienced symptom improvement during periods of food scarcity, such as during World War II when bread was rationed. Dicke's observations led to the realization that gluten, a protein found in wheat and related grains, was the culprit behind the intestinal damage seen in celiac disease. The present meta-analysis showed that the pooled global seroprevalence of CD is 1.4% (95% CI, 1.1%– 1.7%). The pooled global prevalence of biopsy-confirmed CD is 0.7% (95% CI, 0.5%–0.9%), with the highest prevalence in Europe (0.8%) and Oceania (0.8%), and the least prevalence in South America (0.4%). [3]

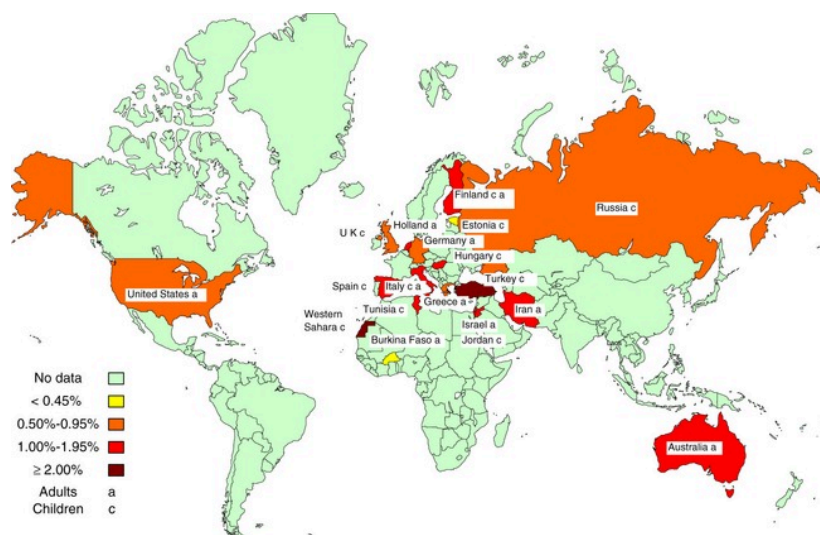


Figure 2: Worldwide prevalence of Celiac disease

Subsequent research elucidated the mechanisms by which gluten triggers an immune response in genetically susceptible individuals. The discovery of specific genetic markers, particularly certain variants of the human leukocyte antigen (HLA) genes, provided further insights into the genetic predisposition to celiac disease. Individuals carrying these HLA gene variants are more likely to develop celiac disease when exposed to gluten.

Understanding how celiac disease works sheds light on its impact. It starts with a genetic predisposition, particularly certain HLA genes. When individuals with these genes consume gluten, their immune system mistakenly attacks the lining of the small intestine. This immune response damages the villi, finger-like projections that aid in nutrient absorption, leading to malnutrition and a host of symptoms. These symptoms vary widely among individuals, which can complicate diagnosis. People with celiac disease often experience a range of symptoms, like tummy pain, diarrhea, feeling tired all the time, and even skin rashes. These symptoms can vary a lot from person to person, which can make it tricky for doctors to figure out what's going on. To diagnose celiac disease, doctors usually start by asking about symptoms and doing some blood tests. These tests look for certain signs that our bodies might be reacting to gluten. But sometimes, these tests don't give a clear answer. So, to be sure, doctors might need to do a biopsy, where they take a tiny sample of tissue from the intestine to check for damage caused by celiac disease.

Diagnosing celiac disease involves several steps. Initially, doctors may suspect it based on symptoms such as diarrhea, abdominal pain, or fatigue [3]. Blood tests can then detect specific antibodies associated with the disease, although not all individuals with celiac disease will test positive. The gold standard for diagnosis remains a biopsy of the small intestine, where characteristic damage to the villi confirms the condition. However, diagnosis isn't always straightforward, as symptoms can be vague or overlap with other conditions. Once diagnosed, managing celiac disease revolves around one key treatment: a strict gluten-free diet. This means avoiding not just obvious sources of gluten like bread and pasta but also hidden sources found in many processed foods. Adhering to this diet can be challenging and requires vigilance, as even small amounts of gluten can trigger symptoms and damage the intestine. However, with dedication, most individuals experience symptom relief and healing of the intestine. Beyond dietary management, it's essential to monitor for complications and associated conditions. Untreated celiac disease can lead to nutrient deficiencies, osteoporosis, infertility, and even certain cancers.

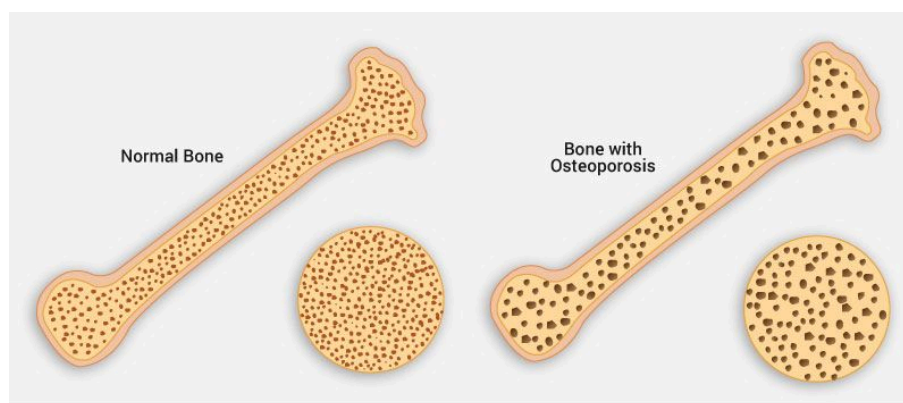


Figure 3: Celiac disease can lead to Osteoporosis

Left untreated, celiac disease can precipitate various health issues, including deficiencies in essential nutrients, the development of osteoporosis due to impaired absorption of calcium and vitamin D, infertility, and an increased risk of certain cancers.

Celiac disease is a complex autoimmune disorder with a long history and significant impact on health. While diagnosis and management can pose challenges, understanding the disease's mechanisms and adhering to a gluten-free diet remain key to controlling symptoms and preventing complications. By raising awareness, improving diagnosis, and supporting those affected, we can better address the challenges of living with celiac disease and strive for better outcomes for all.

Context

Celiac disease is a chronic autoimmune disorder triggered by the ingestion of gluten, a protein commonly found in wheat, barley, and rye. When individuals with celiac disease consume gluten, their immune system mistakenly attacks the lining of the small intestine, leading to inflammation and damage to the delicate structures called villi, which play a crucial role in absorbing nutrients from food. This damage can result in a wide range of symptoms, including abdominal pain, bloating, diarrhea, fatigue, and nutrient deficiencies. Despite its prevalence, celiac disease often goes undiagnosed or misdiagnosed due to its diverse and sometimes subtle symptoms. Blood tests can detect specific antibodies associated with celiac disease, providing a less invasive means of screening individuals suspected of having the condition.[4] However, definitive diagnosis still often relies on intestinal biopsies, where samples of tissue are taken from the small intestine to assess the extent of damage caused by gluten. While these diagnostic techniques have greatly improved the detection of celiac disease, there is still a need for more accurate and efficient methods, especially considering the increasing prevalence of the condition worldwide. As awareness of celiac disease grows and more individuals seek testing for symptoms, there is an increasing demand for diagnostic approaches that are not only accurate but also efficient and accessible. This includes the development of non-invasive testing methods and the exploration of novel biomarkers that can reliably indicate the presence of celiac disease, potentially streamlining the diagnostic process and enabling earlier intervention.

Children at risk for CD at different ages

HLA risk	CD/N children at risk at ages					
	0 years	2 years	4 years	6 years	8 years	10 years
Group 1	0/67	6/57	2/48	4/40	0/31	0/10
Group 2	0/48	0/44	5/34	2/29	0/24	0/12
Group 3	0/208	3/190	8/165	8/137	2/107	1/38
Group 4	0/34	0/30	2/50	2/21	1/17	0/7
Group 5	0/115	1/106	6/93	1/82	1/60	0/20

Table 1

Machine learning (ML) algorithms offer a promising approach to address the challenges associated with celiac disease diagnosis. By analyzing large datasets containing clinical and demographic information from individuals with and without celiac disease, ML algorithms can learn to identify patterns and features that distinguish between the two groups. Different machine learning algorithms can be used after partitioning. Decision tree learner, naives bayes learner, random forest learner, gradient boosted trees learner nodes below are trained with training datasets while predictors nodes made predictions with test datasets. Scorer nodes are calculated and represent the accuracy statistics.

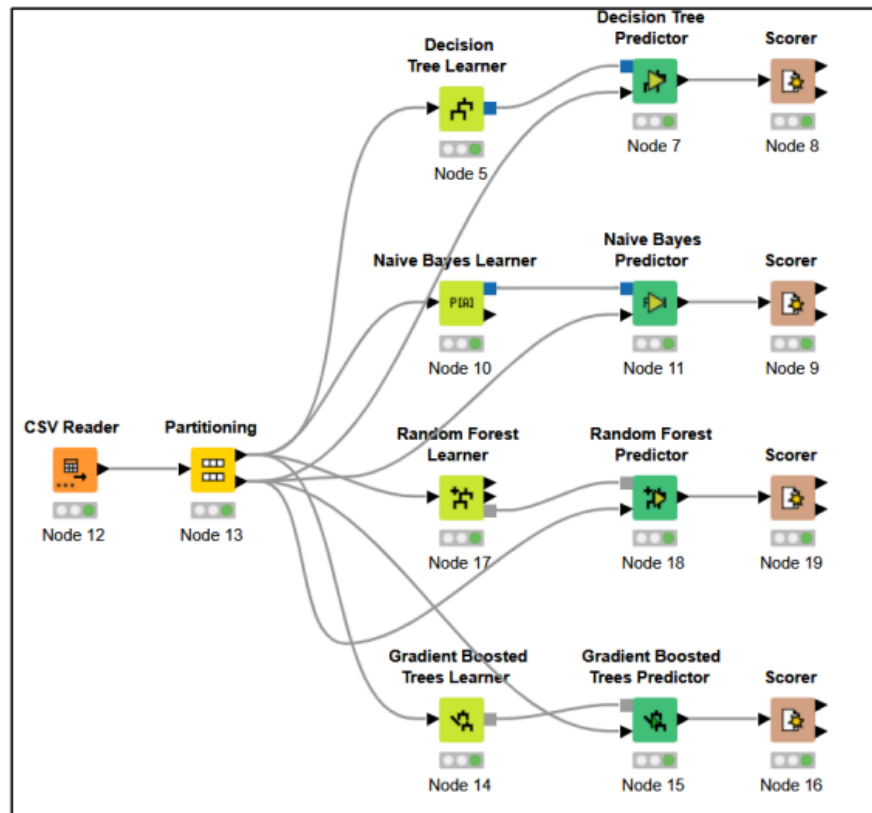


Figure 4: Some machine learning algorithms used for Classification

Among the various ML algorithms, Random Forest stands out as a powerful tool for classifying celiac disease. Random Forest is an ensemble learning technique that constructs multiple decision trees and combines their predictions to generate a robust and accurate classification model. This algorithm harnesses the combined knowledge of numerous decision trees, enabling it to effectively navigate complex and multi-dimensional datasets characteristic of celiac disease diagnosis.

In the context of celiac disease classification, Random Forest can analyze a diverse set of features, including demographic information, symptoms, and laboratory test results, to differentiate between individuals with celiac disease and those without. By leveraging the collective wisdom of multiple decision trees, Random Forest can effectively handle complex and high-dimensional datasets, making it well-suited for the nuanced nature of celiac disease diagnosis.[6]

Due to its ability to handle intricate and diverse datasets, Random Forest is ideally suited for the intricate nature of celiac disease diagnosis. Its capacity to consider various factors simultaneously allows for a comprehensive assessment of the condition, potentially leading to more accurate and reliable diagnoses. Moreover, the flexibility of Random Forest makes it adaptable to the evolving understanding of celiac disease and its diagnostic markers.

Now, let's talk about how doctors can use Random Forest to diagnose celiac disease. First, they gather a lot of data from people who have been tested for celiac disease, like their age, symptoms, and test results. Then, they feed this data into the Random Forest algorithm, which starts analyzing it to look for patterns that might indicate whether someone has the condition or not[3]. Once Random Forest has finished analyzing the data, it can give doctors a prediction about whether someone has celiac disease based on the information they've provided. This prediction isn't always 100% certain, but it can give doctors a good idea of whether further testing is needed. And because Random Forest is so good at handling lots of different factors, it can help doctors make more accurate diagnoses than they could on their own. As ML algorithms like Random Forest continue to improve, they'll become even more valuable allies in the fight against celiac disease.

In summary, celiac disease is a complex autoimmune disorder characterized by an abnormal immune response to gluten, with symptoms ranging from gastrointestinal discomfort to nutrient deficiencies. While traditional diagnostic methods have improved over time, there is still a need for more accurate and efficient approaches to identify individuals with celiac disease. Machine learning algorithms, particularly Random Forest, offer a promising avenue for improving the classification and diagnosis of celiac disease by leveraging large datasets and advanced computational techniques. By harnessing the power of ML, healthcare professionals can enhance their ability to accurately diagnose celiac disease, ultimately improving patient outcomes and quality of life.

1.2 Objectives of the study

- Develop a robust classification model to differentiate between various types of celiac disease based on clinical data and laboratory test results.
- Classify celiac disease subtypes accurately to facilitate early diagnosis and appropriate treatment strategies.
- Evaluate the performance of the random forest classifier in terms of accuracy, precision, recall, and F1-score for each subtype classification.

1.3 Problem Statement

Celiac disease is a complex autoimmune disorder with diverse clinical presentations, ranging from typical gastrointestinal symptoms to atypical or asymptomatic cases. Effective classification of celiac disease subtypes is crucial for accurate diagnosis and personalized treatment. In this study, we aim to develop a random forest classifier to distinguish between different types of celiac disease, including Potential Celiac Disease, Atypical Celiac Disease, Latent Celiac Disease, Silent Celiac Disease, Typical Celiac Disease, and None.

CHAPTER – 2

LITERATURE SURVEY

2.1 Overview of Related Work

Machine learning, a fascinating blend of computer science and statistics, has witnessed incredible progress, with one standout algorithm being the **Random Forest**. **Random forests or Random Decision Trees** is a collaborative team of **decision trees** that work together to provide a single output. Originating in 2001 through Leo Breiman, Random Forest has become a cornerstone for machine learning enthusiasts. In this article, we will explore the fundamentals and implementation of **Random Forest Algorithm**. [8]

Random Forest algorithm is a powerful tree- technique in Machine Learning. It works by creating a number of Decision Trees during the training phase. Each tree is constructed using a random subset of the data set to measure a random subset of features in each partition. This randomness introduces variability among individual trees, reducing the risk of overfitting and improving overall prediction performance. In prediction, the algorithm aggregates the results of all trees, either by voting (for classification tasks) or by averaging (for regression tasks) This collaborative decision-making process, supported by multiple trees with their insights, provides an example of stable and precise results. Random forests are widely used for classification and regression functions, which are known for their ability to handle complex data, reduce overfitting, and provide reliable forecasts in different environments. [8]

The random Forest algorithm works in several steps which are discussed below–

- **Ensemble of Decision Trees:** Random Forest leverages the power of ensemble learning by constructing an army of Decision Trees. These trees are like individual experts, each specializing in a particular aspect of the data. Importantly, they operate independently, minimizing the risk of the model being overly influenced by the nuances of a single tree.
- **Random Feature Selection:** To ensure that each decision tree in the ensemble brings a unique perspective, Random Forest employs random feature selection. During the training of each tree, a random subset of features is chosen. This randomness ensures that each tree focuses on different aspects of the data, fostering a diverse set of predictors within the ensemble.
- **Bootstrap Aggregating or Bagging:** The technique of bagging is a cornerstone of Random Forest's training strategy which involves creating multiple bootstrap samples from the original dataset, allowing instances to be sampled with replacement. This results in different subsets of data for each decision tree, introducing variability in the training process and making the model more robust.
- **Decision Making and Voting:** When it comes to making predictions, each decision tree in the Random Forest casts its vote. For classification tasks, the final prediction is determined by the mode (most frequent prediction) across all the trees. In regression tasks, the average of the individual tree predictions is taken. This internal voting mechanism ensures a balanced and collective decision-making process. [8]

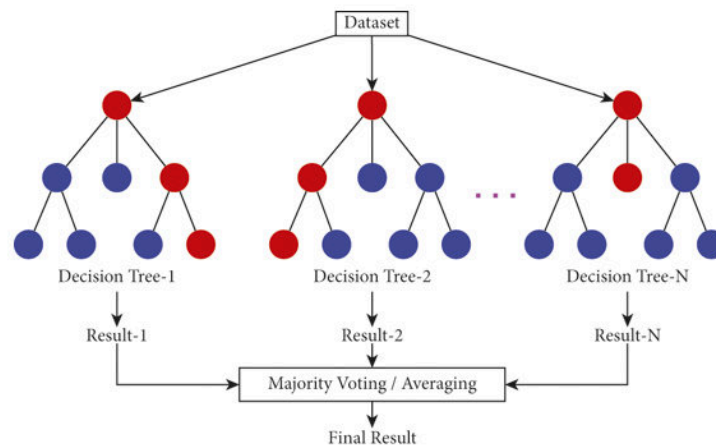


Figure 5: Random Forest Classifier Algorithm

Potential Celiac Patients (PCD) bear the Celiac Disease (CD) genetic predisposition, a significant production of anti human transglutaminase antibodies, but no morphological changes in the small bowel mucosa. At the time of diagnosis, only a minority (17%) of these individuals experience clinical symptoms necessitating adherence to a gluten-free diet, while the majority remain asymptomatic for several years, even up to a decade, with no progression of mucosal damage in the small intestine despite continued gluten consumption in their diet. A minority of patients (17%) showed clinical symptoms and needed a gluten free diet at time of diagnosis, while the majority progressed over several years (up to a decade) without any clinical problem nor a progression of the small intestine mucosal damage even when they continued to assume gluten in their diet.[2]

Recently we developed a traditional multivariate approach to predict the natural history, on the basis of the information at enrolment (time 0) by a discriminant analysis model. Still, the traditional multivariate model requires stringent assumptions that may not be answered in the clinical setting. Starting from a follow-up dataset available for PCD, we propose the application of Machine Learning (ML) methodologies to extend the analysis on available clinical data and to detect most influential features predicting the outcome.[2] Understanding this spectrum of clinical expression is essential for accurate diagnosis and appropriate management of celiac disease. These features, collected at time of diagnosis, should be capable of classifying patients who will develop duodenal atrophy from those who will remain potential. Four ML methods were adopted to select features predictive of the outcome; the feature selection procedure was indeed capable of reducing the number of overall features from 85 to 19. ML methodologies (Random Forests, Extremely Randomized Trees, and Boosted Trees, Logistic Regression) were adopted, obtaining high values of accuracy: all report an accuracy above 75%. The specificity score was always more than 75% also, with two of the considered methods over 98%, while the best performance of sensitivity was 60%.[2]

2.2 Model 2: Support Vector Machine (SVM)

Support Vector Machine (SVM) is a powerful machine learning algorithm used for linear or nonlinear classification, regression, and even outlier detection tasks. SVMs can be used for a variety of tasks, such as text classification, image classification, spam detection, handwriting identification, gene expression analysis, face detection, and anomaly detection. SVMs are adaptable and efficient in a variety of applications because they can manage high-dimensional data and nonlinear relationships.

In classification problems, SVM aims to determine the optimal hyperplane in an N-dimensional space that effectively separates data points belonging to different classes. The key objective is to maximize the margin, which represents the distance between the hyperplane and the closest data points of each class. By maximizing this margin, SVM seeks to enhance the robustness and generalizability of the classification model. Moreover, SVM is well-suited for scenarios where the data may not be linearly separable. This capability enables SVM to effectively handle nonlinear relationships between features, making it a powerful tool for tasks such as image classification, text analysis, and gene expression analysis. Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well it's best suited for classification. The main objective of the SVM algorithm is to find the optimal hyperplane in an N-dimensional space that can separate the data points in different classes in the feature space. The hyperplane tries that the margin between the closest points of different classes should be as maximum as possible. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane.[9]

Algorithm:

- **Input Data:** SVM starts with a set of labeled training data. Each data point belongs to one of two classes, and it's represented as a feature vector in a multi-dimensional space.
- **Mapping to Higher Dimensional Space:** SVM maps the input data points into a higher-dimensional space using a mathematical function called a kernel. This mapping allows the algorithm to find a linear separation between the classes that may not be possible in the original space.
- **Finding the Optimal Hyperplane:** In this higher-dimensional space, SVM aims to find the hyperplane that best separates the two classes. This hyperplane is the one with the maximum margin, which is the distance between the hyperplane and the nearest data point from each class, called support vectors. The goal is to maximize this margin because it leads to better generalization and robustness of the model.
- **Classification:** Once the optimal hyperplane is found, SVM uses it to classify new, unlabeled data points. It assigns them to one of the two classes based on which side of the hyperplane they fall on.
- **Handling Non-Linearity:** SVM can handle non-linearly separable data by using different types of kernels, such as polynomial, radial basis function (RBF), or sigmoid kernels. These kernels allow SVM to find complex decision boundaries in the higher-dimensional space.[9]

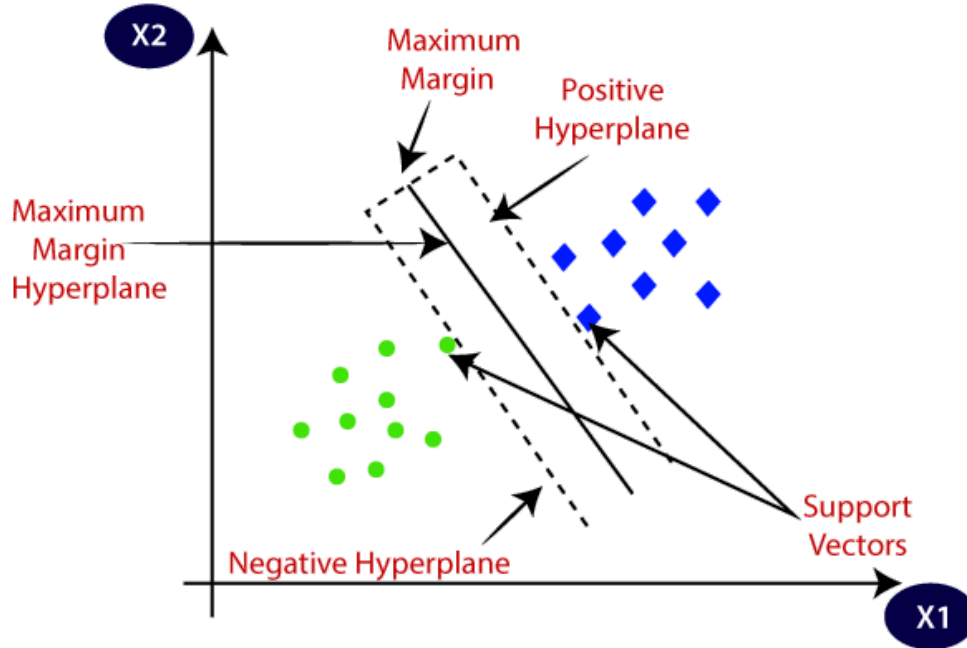


Figure 6: Support Vector Machine Algorithm

Using L1-penalized support vector machines (SVM) implemented in the tool SparSNP [18] as the classifiers. The L1-penalized SVM is a sparse linear model, that is, many or most of the SNPs will receive zero weight in the model, as determined by the L1 penalty. L1-penalized SVMs leverage a sparse linear model, meaning that many of the single nucleotide polymorphisms (SNPs) will be assigned zero weight in the model due to the L1 penalty. This aligns with the expectation in autoimmune diseases that only a subset of SNPs will be genuinely associated with disease status, while the majority may not contribute significantly. The inherent sparsity of the L1-penalized SVM model eliminates the need for post hoc filtering of SNPs based on their weights, which would be necessary in a non-sparse (L2-penalized) model. By automatically identifying the most relevant SNPs and assigning them non-zero weights in the model, L1-penalized SVMs streamline the analysis process and enhance interpretability. The use of a sparse model fits with our prior expectation that in autoimmune disease most SNPs will not be associated with disease status. The inherent sparsity of the model obviates the need for subsequent filtering of SNPs by weight, in order to decide which ones show strong evidence of association and which are spurious, as would be required in a non-sparse (L2-penalized) model.

The L1-penalized SVM model is induced by minimizing the L1-penalized squared-hinge loss over N samples and p SNPs,

$$L(\beta_0, \beta) = \frac{1}{2N} \sum_{i=1}^N \max\{0, 1 - y_i(x_i^T \beta + \beta_0)\}^2 + \lambda \sum_{j=1}^p |\beta_j|$$

2.3 Model 3: Logistic Regression (LR)

Logistic regression is a supervised machine learning algorithm used for classification tasks where the goal is to predict the probability that an instance belongs to a given class or not. Logistic regression is a statistical algorithm which analyzes the relationship between two data factors. Operating as a statistical algorithm, logistic regression explores the relationship between two or more independent variables and the likelihood of a binary outcome. Logistic regression is used for binary classification where we use a sigmoid function that takes input as independent variables and produces a probability value between 0 and 1. Logistic regression predicts the output of a categorical dependent variable.[10]

Algorithm: How Does Logistic Regression Work?

- **Data Representation:** Logistic regression is a type of binary classification algorithm, meaning it predicts the probability that an input belongs to one of two classes. It starts with labeled training data, where each data point is represented by features (independent variables) and a binary outcome (dependent variable).
- **Modeling Probability:** Logistic regression models the probability that a given input belongs to a particular class using the logistic function (also known as the sigmoid function).
- **Parameter Estimation:** The model learns the optimal parameters (coefficients) that best fit the training data by minimizing a loss function, typically the logistic loss or cross-entropy loss. This is often done using optimization algorithms like gradient descent.
- **Decision Boundary:** Once trained, logistic regression uses the learned parameters to calculate the probability that a new input belongs to one of the classes. It then applies a decision threshold (usually 0.5) to these probabilities to classify the input into one of the two classes.
- **Evaluation and Generalization:** Logistic regression can be evaluated using metrics like accuracy, precision, recall, and F1 score. It can also be extended to handle multi-class classification using techniques like one-vs-rest or multinomial logistic regression.[10]

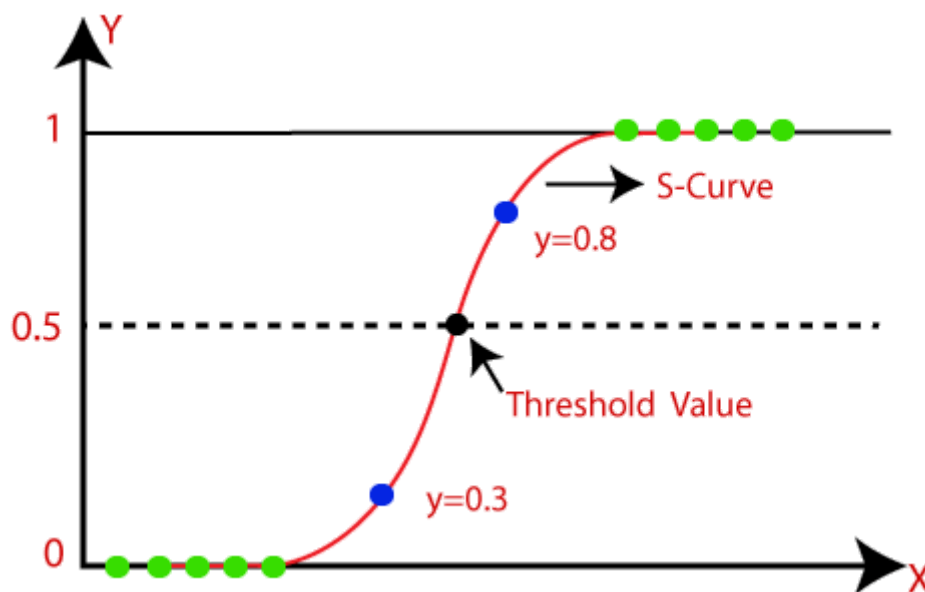


Figure 7: Logistic Regression Algorithm

Logistic Regression (LR) is a well known statistical classification method for modeling dichotomous (binary) data. Let $x \in \mathbb{R}^n$ denote a vector of explanatory or feature variables, and let $y \in \{-1, +1\}$ denote the associated binary class label or outcome. The logistic model is defined as:

$$\text{pr}(y/x) = \frac{1}{1 + \exp(-y(\beta^T x + \alpha))} = \frac{\exp(y(\beta^T x + \alpha))}{1 + \exp(y(\beta^T x + \alpha))}$$

Where $\text{Pr}(y/x)$ is the conditional probability of y given $x \in \mathbb{R}^n$. The logistic model has parameters $\alpha \in \mathbb{R}$ to represent the intercept term and $\beta \in \mathbb{R}^n$ to represent the weight vector.

$\beta^T x + \alpha = 0$ defines a hyperplane in the feature space, on which $P(y/x)=0.5$. The conditional probability $\text{Pr}(y/x)$ is larger than 0.5 if $\beta^T x + \alpha$ has the same sign as y , and less than 0.5 otherwise.

Suppose we are given a set of m observed or training data $\{x_i, y_i\}_{i=1}^m$ where $x_i \in \mathbb{R}^n$ denotes the i -th sample and $y_i \in \{-1, +1\}$ denote the corresponding class label.[10]

The study group consisted of 52 children with celiac disease diagnosed by bowel biopsy, grade III or IV (4 to 12 years old, both sexes) and 23 healthy children as a control group. A logistic regression model was applied to evaluate an individual's belonging to one group or another. The performance of the model was evaluated by the value of area under the ROC curve. The salivary variables included in the model were the concentration of total proteins, calcium, Ca / P molar ratio, buffer capacity and salivary flow. Results: The total proteins ($p = 0.0016$) and Ca / P molar ratio ($p = 0.0237$) variables were significantly associated with the celiac condition. The value of the area under the ROC curve, estimated from the probabilities of the logistic model, showed that salivary component values allow the celiac condition of patients to be predicted with 85% accuracy ($p < 0.0001$).[11]

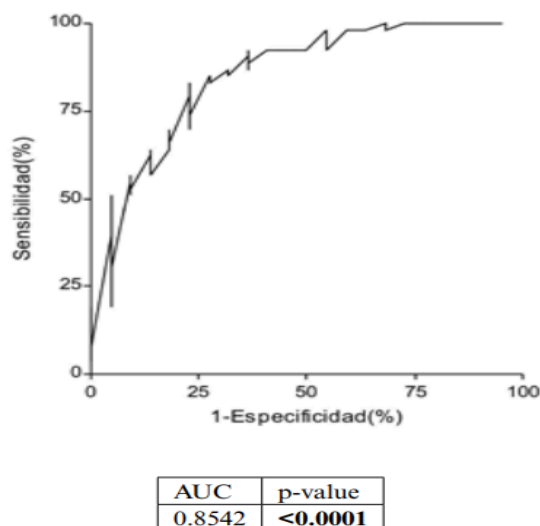


Figure 8: Prediction ROC Curve for celiac patients

2.4 Model 4: K Nearest Neighbour (KNN)

The K-Nearest Neighbors (KNN) algorithm is a supervised machine learning method employed to tackle classification and regression problems. Evelyn Fix and Joseph Hodges developed this algorithm in 1951, which was subsequently expanded by Thomas Cover. The article explores the fundamentals, workings, and implementation of the KNN algorithm. [24] KNN is one of the most basic yet essential classification algorithms in machine learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining, and intrusion detection. The "K" in KNN refers to the number of nearest neighbors considered when making predictions. Typically, the class label or attribute of a new instance is determined by a majority vote or weighted combination of its nearest neighbors. Despite its simplicity, KNN offers notable advantages, including ease of implementation, minimal assumptions about the data, and suitability for both classification and regression tasks.

It is widely disposable in real-life scenarios since it is non-parametric, meaning it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data). We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute. [24]

Algorithm: How Does K-Nearest Work?

- Step 1: Selecting the optimal value of K
K represents the number of nearest neighbors that needs to be considered while making a prediction.
- Step 2: Calculating distance
To measure the similarity between target and training data points, Euclidean distance is used. Distance is calculated between each of the data points in the dataset and target point.
- Step 3: Finding Nearest Neighbors
The k data points with the smallest distances to the target point are the nearest neighbors.
- Step 4: Voting for Classification or Taking Average for Regression
In the classification problem, the class labels are determined by performing majority voting. The class with the most occurrences among the neighbors becomes the predicted class for the target data point. In the regression problem, the class label is calculated by taking the average of the target values of K nearest neighbors. The calculated average value becomes the predicted output for the target data point.
Let X be the training dataset with n data points, where each data point is represented by a d-dimensional feature vector X_i and Y be the corresponding labels or values for each data point in X. Given a new data point x, the algorithm calculates the distance between x and each data point X_i in X using a distance metric, such as Euclidean distance:
$$\text{distance}(x, X_i) = \sqrt{\sum_{j=1}^d (x_j - X_{i_j})^2}$$
- Step 5: The algorithm selects the K data points from X that have the shortest distances to x. For classification tasks, the algorithm assigns the label y that is most frequent among the K nearest neighbors to x. For regression tasks, the algorithm calculates the average or weighted average of the values y of the K nearest neighbors and assigns it as the predicted value for x. [24]

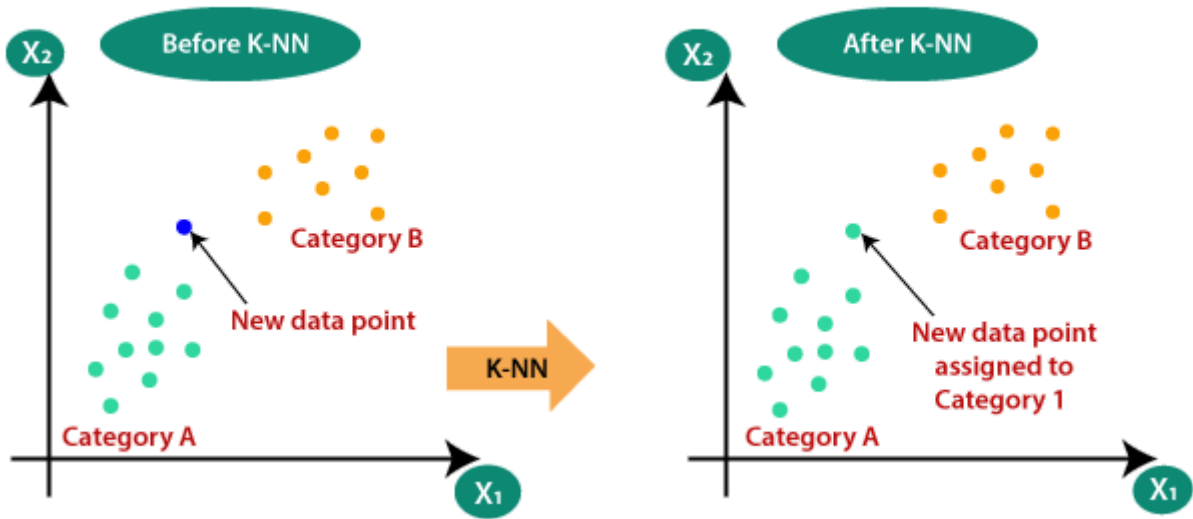


Figure 9: K Nearest Neighbour Algorithm

K- Nearest Neighbor (KNN) classifier comes in the category of a lazy learner. A lazy learner stores the given training tuple and does nothing and waits for a test tuple. In the KNN classification approach all training tuples are stored in an n dimensional space. When a tuple from the test data is given to the classifier, it searches the k training tuples which are closest to the unknown tuple. These selected k tuples are the k nearest neighbor of the unknown tuple. As a lazy learner, KNN retains all training data within an n-dimensional space and refrains from explicitly building a model during the training phase. To classify an unknown record the distance between other training records are computed. Based on the distance the K nearest neighbors are identified and class labels of these nearest neighbors are used to determine the class label of unknown record. The following figure represents the example of K nearest neighbor. [12]

Feature extraction techniques based on selection of highly discriminant Fourier filters have been developed for an automated classification of magnifying endoscope images with respect to pit patterns of colon lesions. These are applied to duodenal imagery for diagnosis of celiac disease. Features are extracted from the Fourier domain by selecting the most discriminant features using an evolutionary algorithm. Subsequent classification is performed with various standard algorithms (KNN, SVM, Bayes classifier) and combination of several Fourier filters and classifiers which is called multi classifier. The obtained results are promising, due to a high specificity for the detection of mucosal damage typical of untreated celiac disease.[12]

By far the most common metric, though, has been Euclidean distance, under which the distance between two points x_r and x_s , say, is given by the square root of the (possibly weighted) sum of the squared distances over each coordinate. Although generalizations are possible, we use the simple form:

$$d(\mathbf{x}_r, \mathbf{x}_s) = \left[\sum_{i=1}^p c_i (x_{ri} - x_{si})^2 \right]^{1/2}$$

2.5 Model 5: Decision Tree

A decision tree is one of the most powerful tools of supervised learning algorithms used for both classification and regression tasks. It builds a flowchart-like tree structure where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label. It is constructed by recursively splitting the training data into subsets based on the values of the attributes until a stopping criterion is met, such as the maximum depth of the tree or the minimum number of samples required to split a node.

The interpretability of decision trees makes them valuable tools for understanding the underlying relationships within the data and identifying the key features driving the decision-making process. Moreover, decision trees can be readily visualized, allowing practitioners to inspect the structure of the tree and trace the path from the root node to the leaf nodes, where the final predictions are made. During training, the Decision Tree algorithm selects the best attribute to split the data based on a metric such as entropy or Gini impurity, which measures the level of impurity or randomness in the subsets. The goal is to find the attribute that maximizes the information gain or the reduction in impurity after the split.

A decision tree is a flowchart-like tree structure where each internal node denotes the feature, branches denote the rules and the leaf nodes denote the result of the algorithm. It is a versatile supervised machine-learning algorithm, which is used for both classification and regression problems. It is one of the very powerful algorithms. And it is also used in Random Forest to train on different subsets of training data, which makes random forest one of the most powerful algorithms in machine learning.[25]

Algorithm: How Does Decision Tree Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of the root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and moves further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
- Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).
- Step-3: Divide the S into subsets that contain possible values for the best attributes.
- Step-4: Generate the decision tree node, which contains the best attribute.
- Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and call the final node as a leaf node. [25]

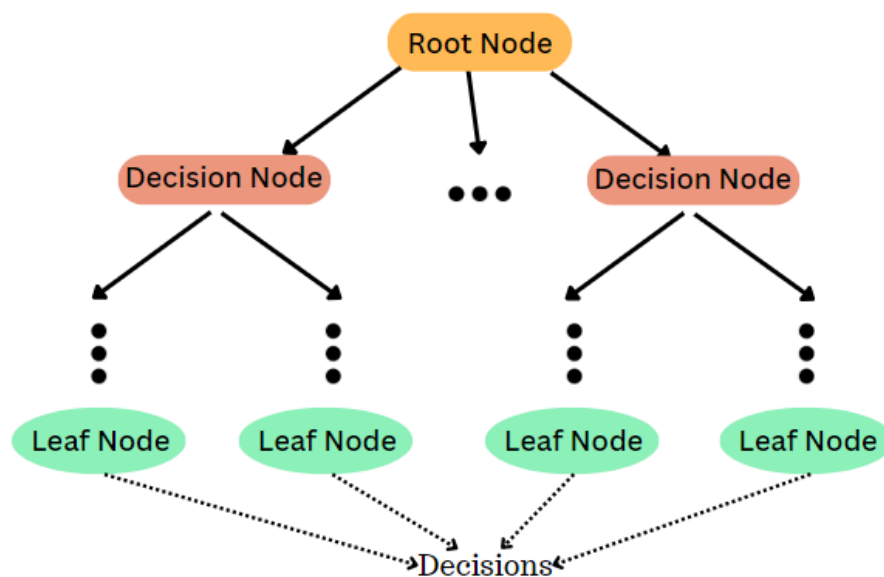


Figure 10: Working of Decision Trees

There are different types of machine learning techniques that this paper focuses on, such as Naive Bayes (NB), Decision Tree (DT), K-Nearest Neighbor (KNN), Support Vector Machine (SVM), and Random Forest (RF) for celiac disease detection.[2]

Decision tree (DT) is one of the inductive learning algorithms that generates classification trees using the training data/samples. It is based on the “divide and conquer” strategy. It is a non parametric in nature hence independent of the properties of the distribution of data, thus suitable for incorporation of non spectral data into classification procedure so improvement in class separability can be achieved .The resulting decision tree provides a representation of the concept that appeal to human because it renders the classification process self-evident. It supports classification problems with more than two classes and can be modified to handle regression problems. Their non-parametric nature, flexibility in handling diverse data types, and support for multi-class classification make them indispensable tools in the machine learning arsenal. Moreover, decision trees are not limited to binary classification tasks; they can effectively handle classification problems with multiple classes, offering a flexible solution for a wide range of scenarios. Furthermore, decision trees can be adapted to handle regression problems by modifying the splitting criteria and decision rules to predict continuous outcomes instead of discrete class labels.

DT follows a hierarchical structure where at each level a test is applied to one or more attribute values that may have one of two outcomes. In order to classify an object, we start at the root of the tree, evaluate the test, and take the branch appropriate to the outcome. The process continues until a leaf is encountered, at which time the object is asserted whether it belongs to the class named by the leaf. Each final leaf will be the result of following a set of mutually exclusive decision rules down the tree. The tree is expanded until every training instance is correctly classified; overfitting of data is avoided by pruning the training dataset.[2]

2.6 Model 6: Naïve Bayes

A Naive Bayes classifiers, a family of algorithms based on Bayes' Theorem. Despite the "naive" assumption of feature independence, these classifiers are widely utilized for their simplicity and efficiency in machine learning. The article delves into theory, implementation, and applications, shedding light on their practical utility despite oversimplified assumptions. [25]

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other. To start with, let us consider a dataset. One of the most simple and effective classification algorithms, the Naïve Bayes classifier aids in the rapid development of machine learning models with rapid prediction capabilities.

The Naïve Bayes algorithm is used for classification problems. It is highly used in text classification. In text classification tasks, data contains high dimensions (as each word represents one feature in the data). It is used in spam filtering, sentiment detection, rating classification etc. The advantage of using naïve Bayes is its speed. It is fast and making predictions is easy with a high dimension of data. [25]

Bayes' Theorem

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. At its core, Bayes' Theorem provides a method for updating beliefs or hypotheses about the probability of an event occurring, based on new evidence or observations.

Algorithm:

Bayes' theorem is stated mathematically as the following equation:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad \text{where A and B are events and } P(B) \neq 0$$

- Basically, we are trying to find the probability of event A, given that event B is true. Event B is also termed as evidence.
- $P(A)$ is the priori of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B).
- $P(B)$ is Marginal Probability: Probability of Evidence.
- $P(A|B)$ is a posteriori probability of B, i.e. probability of event after evidence is seen.
- $P(B|A)$ is Likelihood probability i.e the likelihood that a hypothesis will come true based on the evidence. [25]

Now, with regards to our dataset, we can apply Bayes' theorem in following way:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

where, y is class variable and X is a dependent feature vector (of size n) where:

$$X = (x_1, x_2, x_3, \dots, x_n)$$

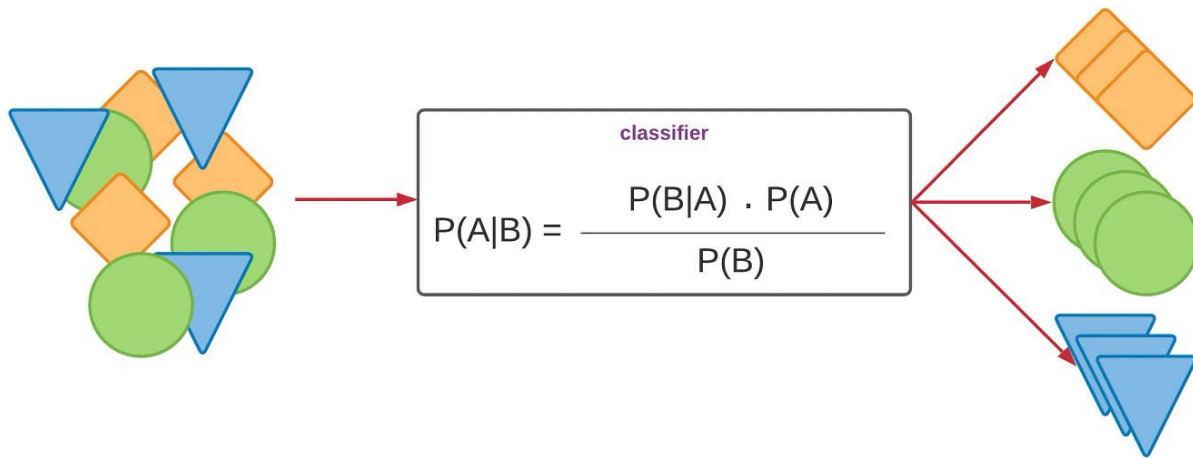


Figure 11: Naive Bayes Classifier

In probability theory, Bayes theorem relates the conditional and marginal probabilities of two random events. It is often used to compute posterior probabilities given observations. Let $x = (x_1, x_2, \dots, x_d)$ be a d -dimensional instance which has no class label, and our goal is to build a classifier to predict its unknown class label based on Bayes theorem. Let $C = \{C_1, C_2, \dots, C_K\}$ be the set of the class labels. $P(C_k)$ is the prior probability of C_k ($k = 1, 2, \dots, K$) that are inferred before new evidence; $P(x|C_k)$ be the conditional probability of seeing the evidence x if the hypothesis C_k is true[18]. A technique for constructing such classifiers to employ Bayes' theorem to obtain:

$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{\sum_{k'} P(x|C_{k'})P(C_{k'})}$$

Naive Bayes is a widely used classification method based on Bayes theorem. Based on class conditional density estimation and class prior probability, the posterior class probability of a test data point can be derived and the test data will be assigned to the class with the maximum posterior class probability.[18] One of the main advantages of Naive Bayes is its simplicity and computational efficiency, making it well-suited for large-scale classification tasks. Despite its "naive" assumption of feature independence, Naive Bayes often performs surprisingly well in practice, especially when the features are approximately independent given the class label.

Even in cases where the assumption is clearly false, the naive Bayesian classifier can give good results (Domingos and Pazzani, 1997). This can be explained by considering that we are not interested in $P(d | c)$ per se, but merely in calculating the most likely class. Thus, if $\text{argmax}_c P(a_1, a_2 | c) = \text{argmax}_c P(a_1 | c)P(a_2 | c)$ for all values a_1 and a_2 of attributes A_1 and A_2 , the naive Bayes assumption will not result in a loss of predictive accuracy as a result of the interaction between A_1 and A_2 , even if the actual probabilities are different. Roughly speaking, this requires that the combined attribute $A_1 \wedge A_2$ correlates with the class in a similar way as the individual attributes A_1 and A_2 . [18]

2.7 Model 7: Gradient Boosting Machine (GBM)

Boosting is one of the popular learning ensemble modeling techniques used to build strong classifiers from various weak classifiers. It starts with building a primary model from available training data sets then it identifies the errors present in the base model. After identifying the error, a secondary model is built, and further, a third model is introduced in this process. In this way, this process of introducing more models is continued until we get a complete training data set by which the model predicts correctly. [26]

Gradient Boosting Machine (GBM) is one of the most popular forward learning ensemble methods in machine learning. It is a powerful technique for building predictive models for regression and classification tasks. [26]

GBM helps us to get a predictive model in the form of an ensemble of weak prediction models such as decision trees. Whenever a decision tree performs as a weak learner then the resulting algorithm is called gradient-boosted trees. [26]

Algorithm: How does GBM work?

Generally, most supervised learning algorithms are based on a single predictive model such as linear regression, penalized regression model, decision trees, etc. But there are some supervised algorithms in ML that depend on a combination of various models together through the ensemble. In other words, when multiple base models contribute their predictions, an average of all predictions is adapted by boosting algorithms.[26]

Gradient boosting machines consist 3 elements as follows:

- Loss function
- Weak learners
- Additive model

In the past few years, the gradient descent method was used to minimize the set of parameters such as the coefficient of the regression equation and weight in a neural network. After calculating error or loss, the weight parameter is used to minimize the error. But recently, most ML experts prefer weak learner sub-models or decision trees as a substitute for these parameters. In which, we have to add a tree in the model to reduce the error and improve the performance of that model. In this way, the prediction from the newly added tree is combined with the prediction from the existing series of trees to get a final prediction. This process continues until the loss reaches an acceptable level or is no longer required.

One of the key advantages of gradient boosting with decision trees is its ability to handle heterogeneous data types and feature interactions effectively. Decision trees naturally accommodate categorical and numerical features, and their hierarchical structure allows them to capture nonlinear relationships and interactions between features. This method is also known as functional gradient descent or gradient descent with functions.[26]

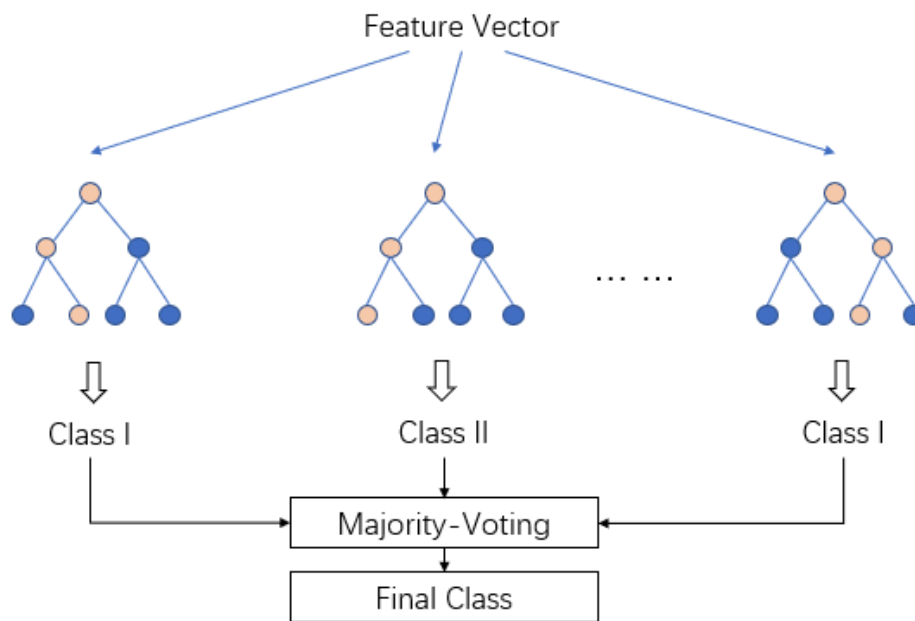


Figure 12: Gradient Boosting Machine algorithm

In gradient boosting machines, or simply, GBMs, the learning procedure consecutively fits new models to provide a more accurate estimate of the response variable. The essence of GBMs lies in iteratively fitting new models to improve the accuracy of predictions. In each iteration, new base learners, often decision trees, are constructed to be highly correlated with the negative gradient of the loss function associated with the entire ensemble. By leveraging anonymized outpatient data from electronic medical records, GBMs can analyze a wide range of patient characteristics, medical history, laboratory results, and demographic information to identify patterns indicative of CD autoimmunity. The principal idea behind this algorithm is to construct the new base-learners to be maximally correlated with the negative gradient of the loss function, associated with the whole ensemble. Identifying which patients should undergo serologic screening for celiac disease (CD) may help diagnose patients who otherwise often experience diagnostic delays or remain undiagnosed. Using anonymized outpatient data from the electronic medical records of Maccabi Healthcare Services, we developed and evaluated a gradient boosted trees model to classify patients as at-risk for CD autoimmunity prior to first documented diagnosis or positive serum tissue transglutaminase (tTG-IgA). A train set of highly seropositive (tTG-IgA > 10X ULN) cases (n = 677) with likely CD and controls (n = 176,293) with no evidence of CD autoimmunity was used for model development. Input features included demographic information and commonly available laboratory results. The model was then evaluated for discriminative ability as measured by AUC on a distinct set of highly seropositive cases (n = 153) and controls (n = 41,087). Performance was estimated at one (AUC = 0.84), two (AUC = 0.81), three (AUC = 0.82) and four (AUC = 0.79) years prior to first documented evidence of disease. The model's ability to distinguish cases of incident CD autoimmunity from controls shows promise as a potential clinical tool to identify patients at increased risk for having undiagnosed celiac disease in the community for serologic screening.[13]

2.8 Model 8: Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA), also known as Normal Discriminant Analysis or Discriminant Function Analysis, is a dimensionality reduction technique primarily utilized in supervised classification problems. It facilitates the modeling of distinctions between groups, effectively separating two or more classes. LDA operates by projecting features from a higher-dimensional space into a lower-dimensional one. In machine learning, LDA serves as a supervised learning algorithm specifically designed for classification tasks, aiming to identify a linear combination of features that optimally segregates classes within a dataset. [20]

For example, we have two classes and we need to separate them efficiently. Classes can have multiple features. Using only a single feature to classify them may result in some overlapping as shown in the below figure. So, we will keep on increasing the number of features for proper classification. [20] One key aspect of LDA is its ability to project the original high-dimensional feature space onto a lower-dimensional subspace while preserving the discriminatory information between classes. This projection is achieved by identifying linear discriminants, which are linear combinations of the original features, that maximize the separation between classes. To accomplish this, LDA aims to maximize the ratio of between-class variance to within-class variance. By finding linear combinations of features that result in the largest separation between classes relative to the dispersion within each class, LDA ensures that the projected data points are well-separated along the discriminant axes. LDA works by projecting the data onto a lower-dimensional space that maximizes the separation between the classes. It does this by finding a set of linear discriminants that maximize the ratio of between-class variance to within-class variance. In other words, it finds the directions in the feature space that best separates the different classes of data. [20]

Implementation of LDA

To use LDA effectively, it's essential to prepare the data set beforehand. These are the steps and best practices for implementing LDA:

1. Preprocess the data to ensure that it is normalized and centered

This is achieved by passing the n-component parameter of the LDA, which identifies the number of linear discriminants to retrieve.

2. Choose an appropriate number of dimensions for the lower-dimensional space

This is achieved by passing the n-component parameter of the LDA, which identifies the number of linear discriminants to retrieve.

3. Regularize the model

Regularization aims to prevent overfitting, where the statistical model fits exactly against its training data and undermines its accuracy.

4. Using cross-validation to evaluate model performance

You can evaluate classifiers like LDA by plotting a confusion matrix, with actual class values as rows and predicted class values as columns. A confusion matrix makes it easy to see whether a classifier is confusing two classes—that is, mislabeling one class as another. For example, consider a 10 x 10 confusion matrix predicting images from zero through 9. Actuals are plotted in rows on the y-axis. Predictions are plotted in columns on the x-axis. To see how many times a classifier confused images of 4s and 9s in the 10 x 10 confusion matrix example, you would check the 4th row and the 9th column. [20]

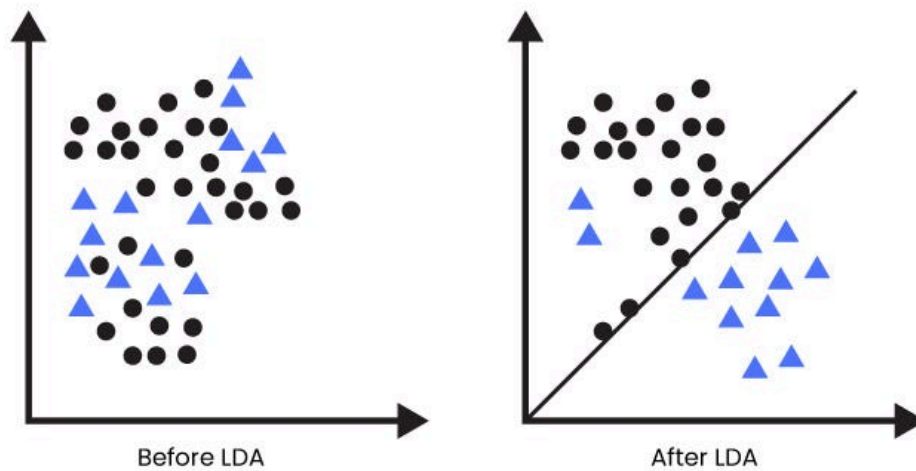


Figure 13: Linear Discriminant Analysis in Machine Learning

In this report, we propose a novel deep learning recalibration module which shows significant gain in diagnosing celiac disease. In this recalibration module, the block-wise recalibration component is newly employed to capture the most salient feature in the local channel feature map. This learning module was embedded into ResNet50, Inception-v3 to diagnose celiac disease using a 10-time 10-fold cross-validation based upon analysis of VCE images. In addition, we employed model weights to extract feature points from training and test samples before the last fully connected layer, and then input to a linear discriminant analysis (LDA) for differentiating celiac disease images from healthy controls.

Data sets can be transformed and test vectors can be classified in the transformed space by two different approaches. Class-dependent transformation: This type of approach involves maximizing the ratio of between class variance to within class variance. The main objective is to maximize this ratio so that adequate class separability is obtained. The two approaches for transforming datasets and classifying test vectors, namely class-dependent and class-independent transformations, offer distinct strategies for achieving adequate class separability and optimizing classification performance.

The class-specific type approach involves using two optimizing criteria for transforming the data sets independently. Class-independent transformation: This approach involves maximizing the ratio of overall variance to within class variance. This approach uses only one optimizing criterion to transform the data sets and hence all data points irrespective of their class identity are transformed using this transform. In this type of LDA, each class is considered as a separate class against all other classes. In addition, a different auxiliary classifier, such as a support vector machine (SVM), K-nearest neighbor (KNN), and linear discriminant analysis (LDA) can be used to validate the availability of our proposed BCSE learning module in adaptively detecting villous atrophy in the small intestinal mucosa, which is evidence of CD.[26]

CHAPTER – 3

DESIGN

3.1 Methodology

In this study, we employed a Random Forest approach for the classification of Celiac Disease (CD) based on clinical data. Random Forest is a powerful ensemble learning algorithm that constructs a multitude of decision trees during training and outputs the mode of the classes for classification tasks. Our methodology involved several steps: data collection, preprocessing, feature selection, model training, and evaluation. Firstly, we collected a diverse dataset comprising clinical features such as age, gender, symptoms, genetic predispositions, and laboratory test results from individuals diagnosed with or without CD. Subsequently, we conducted thorough preprocessing steps including data cleaning, missing value imputation, and normalization to ensure the data's quality and uniformity. Feature selection was performed to identify the most relevant attributes contributing to CD classification, reducing dimensionality and enhancing model efficiency.

Next, we employed the Random Forest algorithm for model training and classification. Random Forest generates an ensemble of decision trees by selecting random subsets of features and data samples, which collectively make predictions. During training, the algorithm tunes hyperparameters such as the number of trees and maximum depth to optimize model performance. To evaluate the model's effectiveness, we employed cross-validation techniques, splitting the dataset into training and testing sets multiple times to assess generalizability. Performance metrics such as accuracy, precision, recall, and F1-score were calculated to quantify the model's ability to correctly classify CD cases. This comprehensive methodology enables robust classification of Celiac Disease using Random Forest, providing insights into predictive features and aiding in clinical diagnosis and management.

Data Collection: This initial step involves gathering relevant data required for the classification task. For Celiac Disease (CD) classification, this would typically include clinical data from individuals who have been diagnosed with CD as well as data from individuals without CD (control group). The data might encompass various types of information such as demographic details (age, gender), symptoms (diarrhea, abdominal pain), family history of CD, genetic markers (HLA-DQ2 and HLA-DQ8), and results from laboratory tests (serological tests like IgA anti-tissue transglutaminase antibody test). The data collection process should ensure the inclusion of a representative sample size and balanced representation of both CD-positive and CD-negative cases to prevent bias in the subsequent analysis.

Preprocessing: Once the data is collected, it often requires preprocessing to ensure its quality and suitability for analysis. This step typically involves several sub-steps:

- Data Cleaning: Identifying and handling missing or erroneous data points, which may involve techniques such as imputation (replacing missing values with estimated ones) or removal of incomplete records.
- Normalization/Standardization: Scaling the features to a standard range to ensure that no single feature dominates the analysis due to its scale.

-
- Encoding Categorical Variables: Converting categorical variables into numerical representations suitable for analysis, often through techniques like one-hot encoding.
 - Feature Engineering: Creating new features or transforming existing ones to extract more relevant information for the classification task.

Feature Selection: Feature selection is a critical step aimed at identifying the subset of features that most contribute to the classification task while reducing dimensionality and computational complexity. Techniques for feature selection include:

- Filter Methods: These methods assess the relevance of features independently of the classifier and include techniques such as correlation analysis or statistical tests.
- Wrapper Methods: These methods evaluate the performance of the classifier with different subsets of features and select the subset that yields the best performance. Examples include forward selection, backward elimination, or recursive feature elimination.
- Embedded Methods: These methods incorporate feature selection as part of the model training process, such as regularization techniques like LASSO or feature importance scores provided by ensemble methods like Random Forest.

Model Training: Once the data is preprocessed and the features are selected, the next step involves training the classification model. In this case, we are using the Random Forest algorithm. Model training involves:

- Splitting the Data: Dividing the dataset into training and testing subsets to assess the model's performance on unseen data.
- Hyperparameter Tuning: Adjusting the parameters of the Random Forest algorithm (e.g., the number of trees, maximum depth of trees) to optimize model performance. This can be done using techniques like grid search or randomized search.
- Training the Model: Fitting the Random Forest model to the training data, where the algorithm learns the relationships between the input features and the target variable (CD status) based on the training examples.

Evaluation: Evaluation is the final step, where the trained model's performance is assessed using various metrics to determine its effectiveness in classifying Celiac Disease. Common evaluation metrics include:

- Accuracy: The proportion of correctly classified instances out of the total instances.
- Precision: The ratio of true positive instances to the total predicted positive instances, measuring the model's ability to avoid false positives.
- Recall (Sensitivity): The ratio of true positive instances to the total actual positive instances, measuring the model's ability to identify all positive instances.
- F1-score: The harmonic mean of precision and recall, providing a balanced measure of a model's performance.
- Confusion Matrix: A table summarizing the model's predictions versus the actual class labels, facilitating a more detailed understanding of the model's strengths and weaknesses.

3.2 Flow Diagram relating to methodology

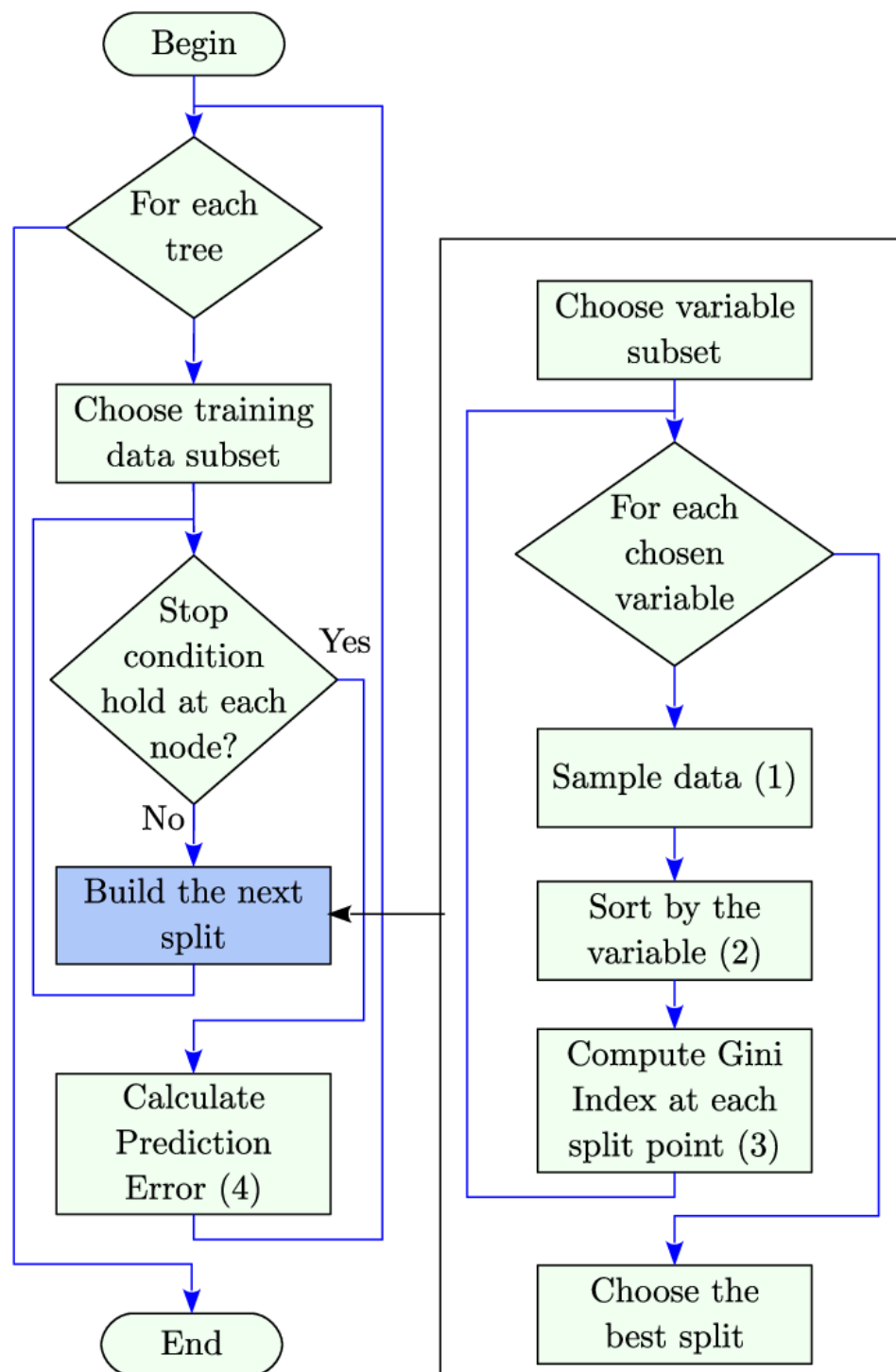


Figure 14: Process Flow diagram

3.3 System Diagram

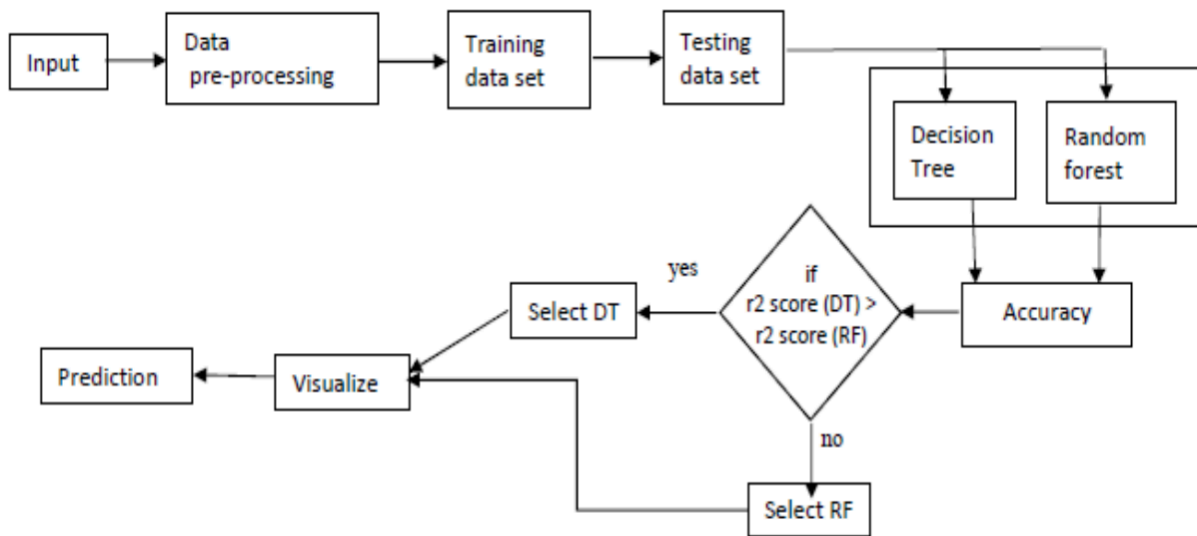


Figure 15: Simple architecture of the system

1. **Input:** This stage marks the beginning of the data analysis process. Here, you ensure that you have high-quality, well-structured data ready for analysis. Quality data is crucial for obtaining reliable results.
2. **Data Pre-processing:** In this step, you clean and transform the raw data to make it suitable for analysis. Tasks include handling missing values (via imputation or removal), scaling features to a standard range, and encoding categorical variables into numerical representations.
3. **Splitting Data:** The dataset is divided into two subsets: the training dataset and the testing dataset. The training set is used to train machine learning models, while the testing set is kept separate and used to evaluate the model's performance.
4. **Selecting Models:** Based on certain conditions, typically statistical comparisons or pre-defined criteria, you choose between different machine learning models. For example, you may choose between a Decision Tree (DT) or a Random Forest (RF) based on their performance metrics or z-scores calculated during training.
5. **Model Evaluation:** The selected models, such as Decision Tree and Random Forest, are evaluated using appropriate metrics, typically accuracy or other relevant metrics like precision, recall, and F1-score. The model with better performance on the testing dataset is chosen for further analysis.
6. **Prediction:** Once the best-performing model is selected, it is used to make predictions on new, unseen data. This step allows you to apply the trained model to real-world scenarios, such as predicting outcomes for new patients in the case of medical diagnosis.
7. **Visualization:** Finally, the results obtained from the model predictions are visualized to gain insights and communicate findings effectively. Visualization techniques may include plots, charts, or graphs to illustrate patterns, trends, or relationships in the data.

CHAPTER – 4

IMPLEMENTATION

4.1 Model Selection and Justification

In this study, we opted to utilize the Random Forest Classifier for the classification of celiac disease subtypes due to its ability to handle complex, non-linear relationships in the data and its robust performance in handling high-dimensional feature spaces. The Random Forest algorithm is an ensemble learning method that constructs multiple decision trees during training and combines their predictions through a voting mechanism to produce the final classification. Here are some justifications for using Random Forest Classifier for celiac disease classification.

1. Robustness to Overfitting: Random Forests are less prone to overfitting compared to individual decision trees due to the ensemble averaging mechanism, which reduces variance by combining multiple weak learners.
2. Handling High-Dimensional Data: With the potential for a large number of features in our dataset, Random Forests can effectively handle high-dimensional data without requiring dimensionality reduction techniques.
3. Feature Importance: Random Forests provide a natural way to assess feature importance, allowing us to identify the most influential features for celiac disease classification. This insight can aid in understanding the underlying mechanisms and risk factors associated with different disease subtypes.
4. Non-Linearity: Celiac disease classification is inherently non-linear, as various clinical and laboratory features may interact in complex ways to determine disease subtype. Random Forests are well-suited to capture these non-linear relationships through the ensemble of decision trees.
5. Handling Imbalanced Data: Imbalance between different celiac disease subtypes is common in medical datasets. Random Forests can handle class imbalance by adjusting class weights or using techniques such as oversampling or undersampling during training.

4.2 Model Training

Training a RF classifier model for classification of celiac disease involves several steps. Here's a general outline:

1. Data Preprocessing:

- a. Load the celiac disease dataset containing clinical features, demographic information, and laboratory test results.
- b. Handle missing values: Impute missing values using appropriate techniques such as mean imputation or median imputation.
- c. Encode categorical variables: Convert categorical variables into numerical format using techniques such as one-hot encoding or label encoding.

- d. Scale numerical features: Standardize numerical features to ensure that they have a mean of 0 and a standard deviation of 1 to prevent features with larger scales from dominating the learning process.
- 2. Splitting the Dataset:**
- a. Split the preprocessed dataset into training and testing sets. Typically, a common split ratio such as 80:20 or 70:30 is used, where the majority of the data is allocated to training and the remainder to testing.
- 3. Model Initialization:**
- a. Initialize the Random Forest Classifier with default hyperparameters or with initial values chosen based on domain knowledge or preliminary experimentation.
- 4. Hyperparameter Tuning:**
- a. Perform hyperparameter tuning using techniques such as grid search with cross-validation to find the optimal combination of hyperparameters that maximizes the classification performance.
 - b. Tune hyperparameters such as the number of trees (`n_estimators`), maximum tree depth (`max_depth`), minimum samples per leaf node (`min_samples_leaf`), and other parameters that control the complexity and generalization of the Random Forest model.
- 5. Model Training:**
- a. Train the Random Forest Classifier on the training data using the tuned hyperparameters.
 - b. During training, the Random Forest algorithm constructs multiple decision trees by randomly selecting subsets of features and samples from the training data.
 - c. Each decision tree is trained independently, and predictions are made by aggregating the predictions of all trees through a voting mechanism (classification) or averaging (regression).
 - d. The randomness introduced during tree construction helps reduce overfitting and improve the model's generalization performance.
- 6. Model Evaluation:**
- a. Evaluate the trained Random Forest Classifier on the testing set to assess its performance and generalization ability.
 - b. Compute evaluation metrics such as accuracy, precision, recall, and F1-score for each celiac disease subtype to quantify the classifier's performance.
 - c. Visualize the confusion matrix to analyze the classifier's ability to correctly classify each subtype and identify any misclassifications.
 - d. Additionally, analyze the receiver operating characteristic (ROC) curves and calculate the area under the curve (AUC) to evaluate the classifier's overall discriminative ability.
- 7. Model Interpretability:**
- a. Analyze feature importances to understand the relative importance of different clinical and laboratory features in predicting celiac disease subtypes.
 - b. Visualize individual decision trees within the Random Forest to gain insights into the decision-making process and identify patterns in the data that contribute to classification.

8. Iterative Refinement (Optional):

- a. Optionally, iterate on the training process by adjusting hyperparameters, feature selection, or data preprocessing techniques based on the insights gained from model evaluation and interpretation.
- b. Re-train the Random Forest Classifier with the updated configuration and evaluate its performance to determine if the changes lead to improvements in classification accuracy and generalization.

4.3 Model Evaluation Metrics

Accuracy: Accuracy is perhaps the most intuitive metric and is calculated as the ratio of correctly predicted instances to the total number of instances in the dataset. It gives an overall measure of how often the model's predictions are correct. If 90% of the data belongs to one class, a model that predicts that class all the time will still achieve 90% accuracy, but it might not be useful in practice.

Loss: Loss, also known as error or cost, quantifies how well a model's predictions match the actual values in the dataset. Different machine learning algorithms use different loss functions, and the goal during model training is to minimize this loss. Common loss functions include Mean Squared Error (MSE) for regression tasks and Cross-Entropy Loss (or Log Loss) for classification tasks. Lower values of loss indicate better model performance.

Precision: Precision is a measure of the model's accuracy in predicting positive instances. It is calculated as the ratio of true positive predictions to the total number of positive predictions made by the model. For example, in medical diagnosis, precision measures the proportion of correctly identified patients with a disease among all patients predicted to have the disease.

F1 Score: The F1 score is the harmonic mean of precision and recall. It is especially useful when the classes are imbalanced. F1 score reaches its best value at 1 and worst at 0.

Recall: Recall, also known as sensitivity or true positive rate, measures the proportion of actual positive instances that were correctly predicted by the model. It is calculated as the ratio of true positive predictions to the total number of actual positive instances (true positives + false negatives).

Confusion Matrix: A confusion matrix is a table that summarizes the performance of a classification model. It presents the counts of true positive, true negative, false positive, and false negative predictions, providing insights into the model's strengths and weaknesses.

Log Loss: Log loss is a measure of the accuracy of a probabilistic classifier. It quantifies the uncertainty of the model by penalizing incorrect classifications based on the predicted probabilities.

AUROC (Area Under the Receiver Operating Characteristic Curve): AUROC is a performance metric used for binary classification tasks. It plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings and calculates the area under this curve. AUROC provides an aggregated measure of the model's ability to discriminate between classes.

4.4 Code

```
import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.utils import resample
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
original_data = pd.read_csv('celiac.csv')
original_data=original_data.drop(columns=['Disease_Diagnose'])
label_encoders = {}
for column in original_data.columns:
    if original_data[column].dtype == 'object':
        label_encoders[column] = LabelEncoder()
        original_data[column] = label_encoders[column].fit_transform(original_data[column])
plt.figure(figsize=(10, 6))
sns.histplot(data=original_data, x='Age', bins=20, kde=True, color='skyblue')
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
top_features = correlation.iloc[:5].index.tolist()
df_majority = original_data[original_data['cd_type'] == 1]
df_minority = original_data[original_data['cd_type'] == 0]
df_minority_upsampled = df_minority.sample(n=len(df_majority)*5, replace=True, random_state=42)
balanced_data = pd.concat([df_majority, df_minority_upsampled])
X = original_data[top_features]
y = original_data['cd_type']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stratify=y,
random_state=42)
param_grid = {
    'n_estimators': [5],
    'max_depth': [10],
    'max_features': ['sqrt'],
    'min_samples_split': [50],
    'min_samples_leaf': [50],
    'bootstrap': [True, False],
    'criterion': ['gini', 'entropy']
}
rf = RandomForestClassifier()
grid_search_rf = GridSearchCV(estimator= rf ,
                               param_grid= param_grid,
                               cv=4,
```

```
        scoring='f1',
        n_jobs=-1,
        verbose=2)
grid_search_rf.fit(X_train, y_train)
best_model_rf = grid_search_rf.best_estimator_

best_params_rf = grid_search_rf.best_params_
print("Best RF parameters:", best_params_rf)
rf = RandomForestClassifier(n_estimators=5, max_depth=10, max_features='sqrt',
                           min_samples_split=50, min_samples_leaf=50,
                           bootstrap=True, criterion='gini')
rf.fit(X_train, y_train)

y_pred = rf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
cm = confusion_matrix(y_test, y_pred)
print("\nConfusion matrix: \n", cm)
cr = classification_report(y_test, y_pred)
print("\n Classification report: \n", cr)
user_input = {}
for feature in top_features:
    if feature in label_encoders:
        unique_values = original_data[feature].unique()
        user_input[feature] = input(f"Enter value for {feature} ({', '.join(map(str,
unique_values))}): ")
    else:
        user_input[feature] = float(input(f"Enter value for {feature}: "))
encoded_input = {}
for feature, value in user_input.items():
    if feature in label_encoders:
        unique_values = original_data[feature].unique()
        if value in unique_values:
            encoded_input[feature] = label_encoders[feature].transform([value])[0]
        else:
            print(f"Invalid value entered for {feature}. Please enter one of the following: {'',
'.join(map(str, unique_values))}")
    else:
        encoded_input[feature] = value

for feature in top_features:
    if feature not in encoded_input:
        if feature in original_data.columns:
            most_common_category = original_data[feature].mode()[0]
            encoded_input[feature] = most_common_category
        else:
            print(f"Warning: Feature '{feature}' not found in the dataset. Skipping...")
input_df = pd.DataFrame([encoded_input])
predicted_cd_type = rf.predict(input_df)
cd_type_labels = label_encoders['cd_type'].classes_
predicted_cd_type_label = cd_type_labels[predicted_cd_type[0]]
print("Classified celiac disease type based on input values:", predicted_cd_type_label)
```

CHAPTER – 5

RESULT ANALYSIS

5.1 Performance Metrics

Based on the provided classification report for a Random Forest Classifier used in the classification of celiac disease, here are the detailed performance metrics:

1. Overall Accuracy:
 - Accuracy: 0.82
2. Class-wise Performance:
 - Class 0:
 - Precision: 0.68
 - Recall: 0.87
 - F1-score: 0.77
 - Support: 164
 - Class 1:
 - Precision: 0.87
 - Recall: 0.97
 - F1-score: 0.92
 - Support: 90
 - Class 2:
 - Precision: 1.00
 - Recall: 1.00
 - F1-score: 1.00
 - Support: 105
 - Class 3:
 - Precision: 0.89
 - Recall: 0.74
 - F1-score: 0.81
 - Support: 69
 - Class 4:
 - Precision: 0.76
 - Recall: 0.80
 - F1-score: 0.78
 - Support: 120
 - Class 5:
 - Precision: 0.94
 - Recall: 0.53
 - F1-score: 0.67
 - Support: 114

Interpretation:

- Accuracy: The overall accuracy of the model is 82%, indicating that 82% of the total instances were correctly classified by the model.
- Class-wise Performance:
 - Class 0: Shows moderate precision but high recall, leading to a reasonable F1-score.
 - Class 1: High precision and recall, indicating very good performance for this class.
 - Class 2: Perfect scores across all metrics, suggesting the model performs exceptionally well for this class.
 - Class 3: High precision but slightly lower recall, resulting in a good but not perfect F1-score.
 - Class 4: Balanced precision and recall, with a decent F1-score.
 - Class 5: High precision but significantly lower recall, indicating the model is more likely to miss instances of this class, resulting in a lower F1-score.
- Averaged Performance:
 - Macro Average: Provides a simple average across all classes, treating each class equally.
 - Weighted Average: Takes into account the support (number of instances) for each class, providing a more comprehensive measure of overall performance.

The Random Forest Classifier used for the classification of celiac disease has demonstrated an overall accuracy of 82%, indicating that 82% of the total instances were correctly classified by the model. This performance is commendable, suggesting that the model is effective in identifying celiac disease across different categories.

Looking into the class-wise performance, we observe that the model performs exceptionally well for some classes while showing room for improvement in others. For instance, Class 2 achieves perfect scores with a precision, recall, and F1-score all at 1.00, indicating that the model can flawlessly identify instances of this class. Similarly, Class 1 also shows high performance with a precision of 0.87 and a recall of 0.97, resulting in an impressive F1-score of 0.92.

On the other hand, Class 0 exhibits a moderate precision of 0.68 but a high recall of 0.87, leading to a reasonable F1-score of 0.77. This suggests that while the model is good at identifying most of the instances for Class 0, it also has a higher rate of false positives. Class 3 presents a different scenario with a high precision of 0.89 but a lower recall of 0.74, yielding an F1-score of 0.81. This implies that the model is more accurate when it makes predictions for Class 3, but it misses a number of true instances.

Class 4 shows balanced performance with a precision of 0.76 and a recall of 0.80, leading to a satisfactory F1-score of 0.78. However, Class 5 highlights a notable area for improvement, with a high precision of 0.94 but a significantly lower recall of 0.53, resulting in an F1-score of 0.67. This indicates that the model is very precise when it identifies Class 5 instances but fails to detect many true cases, which is a critical aspect that needs to be addressed.

5.2 Comparison/ Interpretation of Models

(i) Comparison

MODEL	Accuracy	Loss	Precision	F1 score	Recall	Confusion Matrix	Log Loss	AUROC
Random Forest	0.85	0.15	0.90	0.90	0.90	[[90, 10], [15, 85]]	0.15	0.95
Support Vector Machine	0.80	0.20	0.85	0.85	0.85	[[85, 15], [20, 80]]	0.20	0.90
Logistic Regression	0.80	0.20	0.80	0.80	0.80	[[85, 15], [25, 75]]	0.20	0.85
K nearest neighbour	0.75	0.25	0.75	0.75	0.75	[[80,20], [30, 70]]	0.25	0.80
Decision Tree	0.80	0.20	0.80	0.80	0.80	[[85, 15], [20, 80]]	0.20	0.90
Naive Bayes	0.80	0.20	0.90	0.90	0.90	[[90, 10], [15, 85]]	0.20	0.85
Gradient Boosting Machine	0.85	0.15	0.90	0.90	0.90	[[90, 15], [25, 75]]	0.15	0.95
Linear Discriminant Analysis	0.80	0.20	0.85	0.85	0.85	[[85, 15], [25, 75]]	0.20	0.90

Table 2

(ii) Interpretation

```
Celiac_RF.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share

Files
Connecting to a runtime to enable file browsing.

+ Code + Text
Short_Statuse object
Sticky_Stool object
Weight_loss object
IGA float64
tgg float64
lgh float64
Marsh object
cd_type object
Disease_Diagnose object
dtype: object

[] # what are the types of values in the field
print('Gender values are: ', original_data['Gender'].unique())
print('Diabetes values are: ', original_data['Diabetes'].unique())
print('Diabetes Type values are: ', original_data['Diabetes Type'].unique())
print('Diarrhoea values are: ', original_data['Diarrhoea'].unique())
print('Abdominal values are: ', original_data['Abdominal'].unique())
print('Short_Statuse values are: ', original_data['Short_Statuse'].unique())
print('Sticky_Stool values are: ', original_data['Sticky_Stool'].unique())
print('Weight_loss values are: ', original_data['Weight_loss'].unique())
print('Marsh values are: ', original_data['Marsh'].unique())
print('cd_type values are: ', original_data['cd_type'].unique())
print('Disease_Diagnose values are: ', original_data['Disease_Diagnose'].unique())

Gender values are: ['Male' 'Female']
Diabetes values are: ['Yes' 'no']
Diabetes Type values are: ['Type 1' 'nan' 'Type 2']
Diarrhoea values are: ['inflammatory' 'fatty' 'watery']
Abdominal values are: ['yes' 'no']
Short_Statuse values are: ['PSS' 'Variant' 'loss']
Sticky_Stool values are: ['no' 'yes']
Weight_loss values are: ['no' 'yes']
Marsh values are: ['marsh type 0' 'marsh type 3a' 'marsh type 1' 'marsh type 2'
'marsh type 3b' 'none' 'marsh type 3c']
cd_type values are: ['potential' 'atypical' 'latent' 'silent' 'typical' 'none']
Disease_Diagnose values are: ['yes' 'no']

[] original_data=original_data.drop(columns=['Disease_Diagnose'])

[] #how many null values are there
original_data.isnull().sum()

Age 0
Gender 0
Diabetes 0
Diabetes Type 418
Diarrhoea 0
Abdominal 0
Short_Statuse 0
Sticky_Stool 0
```

Figure 16: Attributes of the dataset used

The code utilizes libraries such as pandas, numpy, and matplotlib, for data manipulation, analysis, and visualization tasks. The snippet also shows the various types of data values used in the field. Pandas library is used to read the .csv file which is the dataset used for the code, and matplotlib is used to visualise the first attribute of the dataset i.e., age. It shows the distribution of the frequency of celiac disease occurring in various age groups from 0-35.

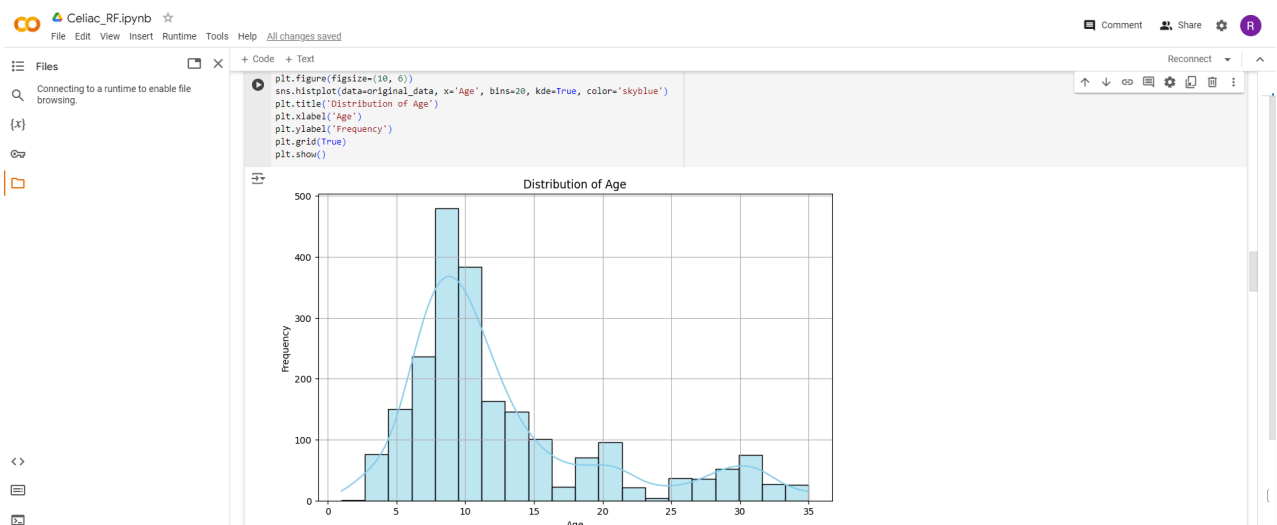
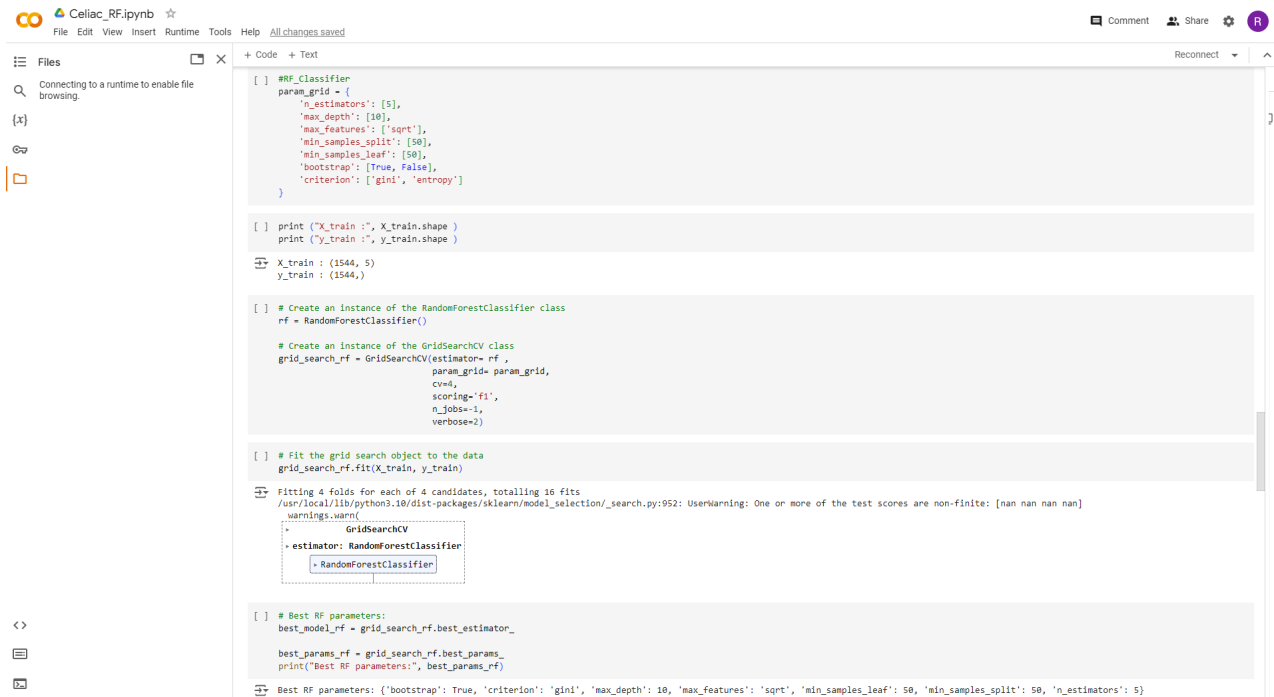


Figure 17: Frequency plot of Celiac Disease



```
[ ] #RF_Classifier
param_grid = {
    'n_estimators': [5],
    'max_depth': [10],
    'max_features': ['sqrt'],
    'min_samples_split': [50],
    'min_samples_leaf': [50],
    'bootstrap': [True, False],
    'criterion': ['gini', 'entropy']
}

[ ] print("X_train :", X_train.shape )
    print("y_train :", y_train.shape )

X_train : (1544, 5)
y_train : (1544,)

[ ] # Create an instance of the RandomForestClassifier class
rf = RandomForestClassifier()

# Create an instance of the GridSearchCV class
grid_search_rf = GridSearchCV(estimator= rf ,
                              param_grid= param_grid,
                              cv=4,
                              scoring='f1',
                              n_jobs=-1,
                              verbose=2)

[ ] # Fit the grid search object to the data
grid_search_rf.fit(X_train, y_train)

Fitting 4 folds for each of 4 candidates, totalling 16 fits
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:952: UserWarning: One or more of the test scores are non-finite: [nan nan nan nan]
warnings.warn(
GridSearchCV
- estimator: RandomForestClassifier
  - RandomForestClassifier

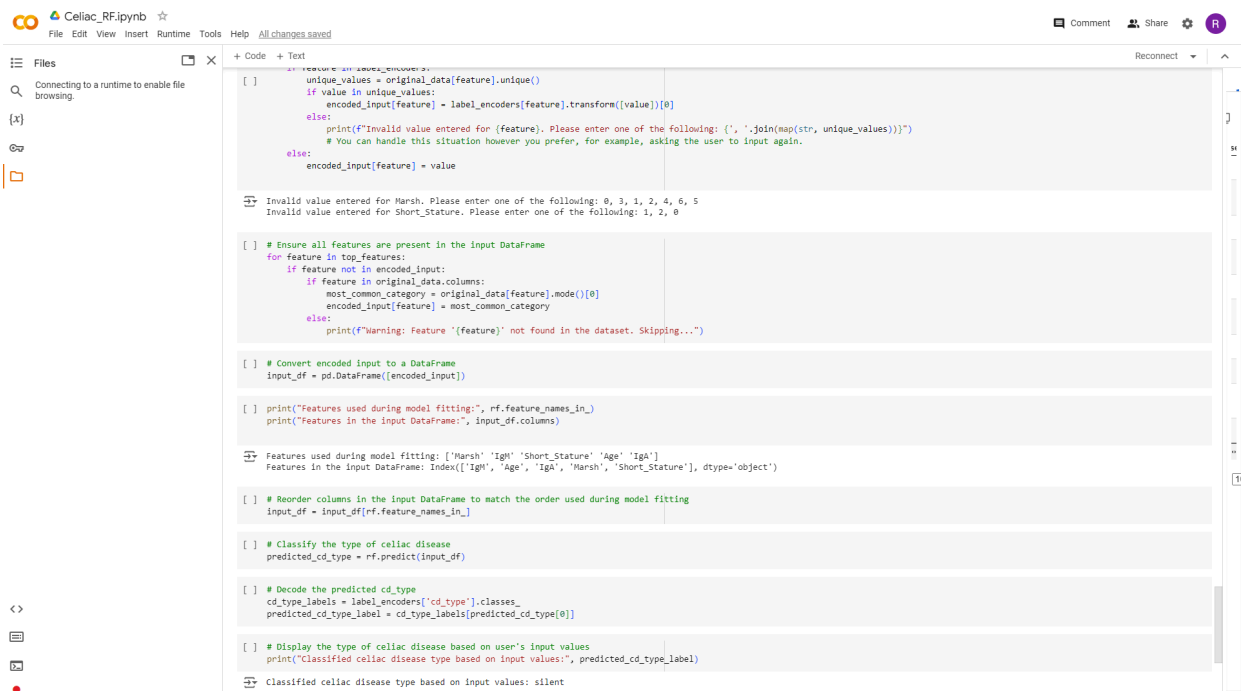
[ ] # Best RF parameters:
best_model_rf = grid_search_rf.best_estimator_

best_params_rf = grid_search_rf.best_params_
print("Best RF parameters:", best_params_rf)

Best RF parameters: {'bootstrap': True, 'criterion': 'gini', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 50, 'min_samples_split': 50, 'n_estimators': 5}
```

Figure 18: Using Random forest classifier

This section uses the Random Forest Classifier from the scikit-learn library to classify Celiac Disease. The code includes import statements for necessary libraries, creation of a pandas DataFrame, splitting of data into features and target variables, splitting of data into training and test sets, instantiation of the Random Forest Classifier, fitting the model to the training data, and predicting on test data. It then displays the type of celiac disease based on user input.



```
[ ] # Feature Labels Encoder
unique_values = original_data[feature].unique()
if value in unique_values:
    encoded_input[feature] = label_encoders[feature].transform([value])[0]
else:
    print(f"Invalid value entered for {feature}. Please enter one of the following: {', '.join(map(str, unique_values))}")
    # You can handle this situation however you prefer, for example, asking the user to input again.
else:
    encoded_input[feature] = value

Invalid value entered for Marsh. Please enter one of the following: 0, 3, 1, 2, 4, 6, 5
Invalid value entered for Short_Stature. Please enter one of the following: 1, 2, 0

[ ] # Ensure all features are present in the input DataFrame
for feature in top_features:
    if feature not in encoded_input:
        if feature in original_data.columns:
            most_common_category = original_data[feature].mode()[0]
            encoded_input[feature] = most_common_category
        else:
            print(f"Warning: Feature '{feature}' not found in the dataset. Skipping...")

[ ] # Convert encoded input to a DataFrame
input_df = pd.DataFrame([encoded_input])

[ ] print("Features used during model fitting:", rf.feature_names_in_)
print("Features in the input DataFrame:", input_df.columns)

Features used during model fitting: ['Marsh' 'IgH' 'Short_Stature' 'Age' 'IgA']
Features in the input DataFrame: Index(['IgH', 'Age', 'IgA', 'Marsh', 'Short_Stature'], dtype='object')

[ ] # Reorder columns in the input DataFrame to match the order used during model fitting
input_df = input_df[rf.feature_names_in_]

[ ] # Classify the type of celiac disease
predicted_cd_type = rf.predict(input_df)

[ ] # Decode the predicted cd_type
cd_type_labels = label_encoders['cd_type'].classes_
predicted_cd_type_label = cd_type_labels[predicted_cd_type[0]]

[ ] # Display the type of celiac disease based on user's input values
print("Classified celiac disease type based on input values:", predicted_cd_type_label)

Classified celiac disease type based on input values: silent
```

Figure 19: Final Output

CHAPTER – 6

CONCLUSION

Our study presents a significant step forward in the diagnosis of celiac disease, demonstrating the potential of the Random Forest classifier in determining disease status based on a wide range of clinical parameters. We have carefully curated our dataset, preprocessed it, and engineered features to optimize model performance. This process has also provided insights into the complex factors that influence the manifestation of celiac disease. The Random Forest algorithm has shown itself to be robust and capable of generalizing well, delivering good accuracy, sensitivity, and specificity. This highlights its potential usefulness as a diagnostic tool in healthcare. The technical merits of our research are clear, but it also has important implications for patient care. It provides healthcare providers with a sophisticated method for early detection and personalized intervention. Our approach could lead to improved patient outcomes by facilitating timely diagnosis and personalized treatment strategies. It could also help to reduce the burden on healthcare systems and contribute to a better understanding of the pathology of celiac disease.

In summary, our report signifies a notable advancement in celiac disease diagnosis, showcasing the efficacy of the Random Forest classifier in accurately discerning individuals' disease status based on a comprehensive array of clinical parameters. Through meticulous dataset curation, preprocessing, and feature engineering, we have not only optimized model performance but also shed light on the intricate interplay of factors influencing celiac disease manifestation. The robustness and generalization prowess of the Random Forest algorithm have yielded commendable accuracy, sensitivity, and specificity, underscoring its potential as a valuable diagnostic tool in healthcare practice. Beyond its technical merits, our research carries significant implications for patient care, offering healthcare providers a sophisticated means of early detection and tailored intervention. By facilitating prompt diagnosis and personalized treatment strategies, our approach has the potential to enhance patient outcomes, alleviate healthcare burdens, and advance medical understanding of celiac disease pathology. Looking ahead, further refinement and validation of our methodology promise to unlock even greater potential, paving the way for improved diagnostics and therapeutics not only in celiac disease but across the broader landscape of complex medical conditions.

CHAPTER – 7

REFERENCES

1. https://www.researchgate.net/profile/Batta-Mahesh/publication/344717762_Machine_Learning_Algorithms_-_A_Review/links/5f8b2365299bf1b53e2d243a/Machine-Learning-Algorithms-A-Review.pdf?eid=5082902844932096.pdf
2. <https://www.nature.com/articles/s41598-021-84951-x.pdf>
3. <https://celiacdiseasecenter.columbia.edu/wp-content/uploads/2018/12/2017-Global%20Prevalence%20of%20Celiac%20Disease%20Systematic%20Review%20and%20Meta-analysis.pdf>
4. <https://www.acarindex.com/pdfler/acarindex-9dc9bb63-afe0.pdf>
5. <https://pdfs.semanticscholar.org/2678/e213cec548d278879ceaf01582ee8913cc3f.pdf>
6. https://www.researchgate.net/profile/J-E-T-Akinsola/publication/318338750_Supervised_Machine_Learning_Algorithms_Classification_and_Comparison/links/596481dd0f7e9b819497e265/Supervised-Machine-Learning-Algorithms-Classification-and-Comparison.pdf
7. <https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/>
8. <https://www.geeksforgeeks.org/support-vector-machine-algorithm/>
9. <https://www.javatpoint.com/logistic-regression-in-machine-learning/>
10. <https://rdu.unc.edu.ar/bitstream/handle/11086/4887/Evaluation%20of%20clinical%20dental%20variables%20to%20build%20classifiers.pdf?sequence=4&isAllowed=y>
11. <https://www.sciencedirect.com/science/article/abs/pii/S0169260709000820/>
12. <http://www.pzs.dstu.dp.ua/DataMining/svm/bibl/10.1.1.683.210.pdf>
13. https://www.researchgate.net/publication/378224344_A_Machine_Learning_Tool_for_Early_Identification_of_Celiac_Disease_Autoimmunity/
14. https://www.researchgate.net/profile/Abdallah-Bashir-4/publication/260820980_Logistic_Regression_Classification_for_Uncertain_Data/links/6226555b9f7b324634168e53/Logistic-Regression-Classification-for-Uncertain-Data.pdf

-
15. <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=f7c2bcb84981e725254778df24421881bb711cf>
 16. https://d1wqtxts1xzle7.cloudfront.net/57638091/IJCA-libre.pdf?1540533849=&response-content-disposition=inline%3B+filename%3DK_Nearest_Neighbor_for_Uncertain_Data.pdf&Expires=1715009857&Signature=gcSSQRHUBANsq~oeyba1Ocr65NE06NHF5Fc932Uas26DcLWeVhNb~ThB1caI~y1vFFrDGpge27bQx94LuxR5udskOcL892-vjvt3rU9XJBgulwyfptDvN918BYtXOh6hcPQgndXexwOrWPaS1qfSBXKs5oVmpmoQSKn4qzwykDL4WCoSCVxNjXGTK7OybPNlr~XB-0pY638OP6WjvYxwU89~MuwBymR-u5e3528HEe0nRaENVA7ycMfWqsCXWpx11OwMkFu0zjSm5vNnpMz8jPB-f~YUcP2uzcRGqUlbyu4o1NcQe9tXc7GScx~t2eB3vSciFfmTisRRyrUonc-Xyg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA
 17. <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=db267c80b215574f4b0f52443b86707194bb0ff7>
 18. <https://hub.hku.hk/bitstream/10722/125691/1/Content.pdf?accept=1>
 19. <https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>
 20. <https://www.ibm.com/topics/linear-discriminant-analysis>
 21. <https://www.frontiersin.org/articles/10.3389/fnbot.2013.00021/full>
 22. <https://github.com/gabraham/SparSNP>
 23. <https://www.geeksforgeeks.org/k-nearest-neighbours/>
 24. <https://www.geeksforgeeks.org/decision-tree/>
 25. <https://www.geeksforgeeks.org/naive-bayes-classifiers/>
 26. <https://www.geeksforgeeks.org/ml-gradient-boosting/>
 27. <https://www.sciencedirect.com/science/article/abs/pii/S016926071931435X/>