UNIVERSITY OF LONDON

INTERNATIONAL PROGRAMMES

BSc Computer Science and Related Subjects

CM3070 Final Year Project (FYP) Report

Deep Learning for Healthy Eating Recommendations Using Nutrition Data

Author: Marcus Tan Lai He
Student Number: 220218906
Date of Submission: 31st March 2025
Supervisor: Desmond Lee Yong Meng
Github Link: https://github.com/mlhtan001/CM3070/tree/main

Table of Contents

Chapter 1: Introduction

1.1 Concept and Motivation

The project, **"Deep Learning for Healthy Eating Recommendations Using Nutrition Data,"** is motivated by the increasing need for **personalised dietary recommendations** for the ageing population, particularly for individuals managing chronic conditions such as **hypertension, diabetes, and high cholesterol**. Proper dietary management plays a crucial role in controlling these conditions. Yet, many seniors struggle with making informed food choices due to the **complexity of nutritional guidelines** and the **overwhelming variety of food options** available.

This project seeks to bridge the gap by providing a **deep-learning-powered recommendation system** that can analyse **recipes**, **suggest healthier ingredient substitutions**, and **provide meal recommendations** tailored to a user's dietary needs.

The inspiration for this project is both **personal and societal**. My mother's recent **diagnosis of high cholesterol** highlighted the importance of **dietary modifications** in reducing health risks. This experience, combined with **societal concerns** regarding the rise of diet-related chronic diseases among the elderly, reinforced the need for a **practical**, **AI-driven tool** that could support **seniors and caregivers** in making healthier food choices.

With the advancement of artificial intelligence (AI) and deep learning (DL), this project applies Neural Networks (NNs) and Natural Language Processing (NLP) to develop a personalised dietary recommendation system. Unlike existing systems, which provide generalised health recommendations, this system offers **individualised guidance** by analysing both **structured nutritional data and unstructured ingredient lists** from recipes. It ensures that suggestions are accurate, actionable, and practical, making it **accessible for everyday use**.

1.2 Research Questions, Aims and Objectives

This project seeks to address the following key research questions:

1. How can **DL techniques** be applied to classify recipes based on **nutritional content**?
2. What **methods** can effectively recommend **healthier alternatives** for seniors with specific dietary restrictions?
3. How can **ingredient-level analysis** support **practical dietary decisions** for seniors?

The **primary goal** of this project is to develop a **personalised dietary recommendation system** that integrates **Machine Learning (ML), DL, and NLP** to generate **health-conscious meal suggestions** tailored to seniors' **dietary needs and restrictions**.

To achieve this, the project focuses on the following **objectives** (Table 1).

Objective	Description
Develop a **multi-layer neural network (MLNN)** to classify recipes	Train a DL model to categorise recipes as "healthy" or "unhealthy" based on nutritional content and ingredient composition.

Provide tailored dietary recommendations	Generate personalised meal suggestions aligned with individual health conditions, such as **low-sodium diets for hypertension** or **low-fat diets for high cholesterol management**.
Suggest healthier ingredient substitutions	Recommend alternatives for high-risk ingredients (e.g. replacing butter with olive oil) while maintaining **flavour and practicality**.
Analyze recipes for dietary concerns	Identify **nutritional risks** (e.g. high cholesterol, sodium, or sugar content) and propose modifications to meet dietary goals.

Table 1: Summary of project objectives and descriptions

These objectives serve as **building blocks** for an AI-driven recommendation system that ensures both **health compliance** and **practical meal planning** for seniors.

1.3 Deliverables

To implement the stated objectives, the following key **deliverables** have been outlined:

1. **Recipe Classification Model**
    - A **DL model** was trained to classify recipes as either **healthy or unhealthy** based on **nutritional analysis** and **ingredient composition**.
2. **Personalised Health Recommendations**
    - A **recommendation system** that tailors dietary advice **based on a user's health conditions** (e.g. recommending **low-sodium recipes** for hypertensive individuals).
3. **Healthier Recipe Alternatives**
    - Ingredient substitution suggestions to **improve the nutritional profile** of meals while maintaining **taste and practicality**.
4. **Nutrient and Ingredient Analysis Tool**
    - A **detailed analysis framework** that identifies **potential dietary risks** in recipes and suggests appropriate adjustments.

By implementing these deliverables, the system provides **actionable dietary insights** that enable seniors to make **informed food choices** without requiring extensive nutritional knowledge.

1.4 Justification of Objectives and Deliverables

Each objective and deliverable plays a **crucial role** in making the system **practical, useful, and impactful** for seniors. Table 2 below shows the justifications for each deliverable.

Deliverable	Justification
Recipe Classification	Ensures the system can accurately evaluate recipes using nutrient and ingredient data, enabling precise dietary recommendations
Personalised Health Recommendations	Bridge the gap between generic advice and individualised dietary guidance, making the system highly relevant for seniors managing chronic conditions

Healthier Recipe Alternatives	Focus on practical solutions by suggesting ingredient substitutions, empowering users to make healthier choices without compromising flavour.
Ingredient and Nutrient Analysis	Provides in-depth insights into recipes, helping users identify and address potential dietary concerns.

Table 2: Summary of project deliverables and justifications

By aligning the project's objectives with real-world dietary challenges, the system ensures that seniors receive **targeted**, **effective meal recommendations**, improving their long-term health and well-being.

1.5 Project Template and Methodology

This project follows the **CM3015 Machine Learning and Neural Networks** template, which includes:

Data Preprocessing
- **Cleaning and structuring** nutritional and ingredient data from publicly available datasets (e.g. Kaggle).
- Handling **missing values, outliers, and class imbalances** to ensure robust model training.

Model Development
- Training an **MLNN** that incorporates **structured nutritional features** and **unstructured ingredient text** for recipe classification.
- Integrating **NLP** techniques such as **Word2Vec or GloVe** to enhance ingredient analysis.

Recommendation System Implementation
- Developing a **hybrid recommendation system** using:
    - **Content-Based Filtering:** Suggesting recipes based on **nutrient profiles** and **ingredient similarities**.
    - **Rule-Based Filtering:** Enforcing dietary constraints (e.g. **low sodium, low sugar, or high protein recommendations**).
    - **Hybrid Approach:** Combining **content-based** and **rule-based** techniques for personalised recommendations.

Evaluation Metrics
- **Classification Performance:** Evaluating accuracy, precision, recall and F1-score.
- **User Satisfaction Testing:** Assessing the **practicality and effectiveness** of dietary recommendations.

This project represents a **meaningful application of AI and DL** in healthcare, addressing a **critical need** for personalised dietary recommendations tailored to seniors. By focusing on recipe classification, personalised health guidance, and ingredient analysis, it bridges the gap between **AI advancements** and **real-world dietary challenges**.

The **outcomes** of this project have the potential to:

- Enhance dietary decision-making for seniors.
- Contribute to advancements in AI-driven healthcare tools.
- Inspire future research on AI-powered personalised nutrition systems.

Word Count: 855

Chapter 2: Literature Review

Dietary recommendation systems have evolved significantly, leveraging AI, ML, and NLP to enhance meal planning, nutrition tracking, and ingredient analysis. Early systems primarily focused on calorie counting and generic diet planning, but recent advancements have introduced personalised meal recommendations based on individual health metrics and dietary restrictions.

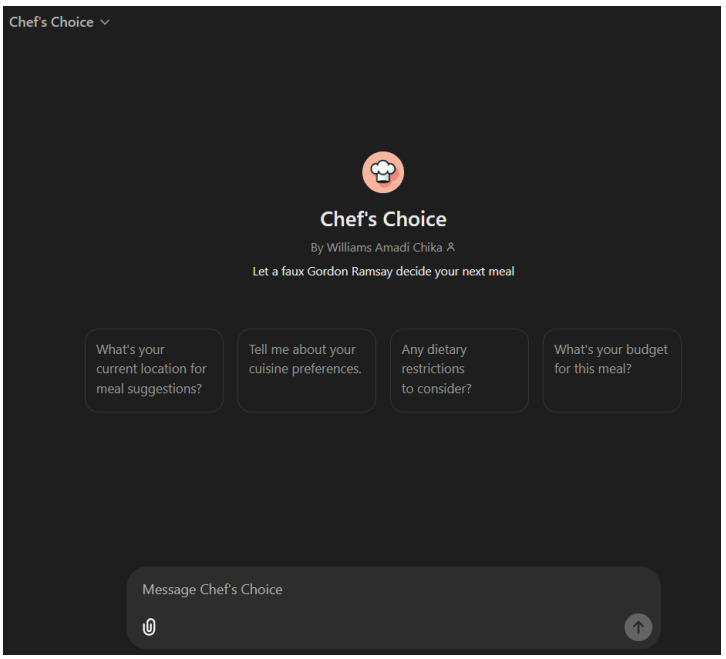
However, despite these innovations, most existing systems do not cater specifically to seniors who require dietary modifications due to chronic health conditions like diabetes, hypertension, and high cholesterol. This project aims to address these limitations by integrating DL models, structured nutrient data, and NLP-enhanced ingredient analysis to deliver personalised dietary recommendations tailored for seniors.

2.1 Existing Systems

Three AI-powered dietary recommendation systems — Chef's Choice, NutrifyMe, and EatLove — are assessed for their strengths, limitations, and applicability to this project.

2.1.1 Chef's Choice - Personalised Meal Guidance

The **Chef's Choice** system is an AI-powered **culinary assistant** designed to assist users in making **informed meal decisions** by considering their **cuisine preferences, dietary restrictions, available cooking time, and budget constraints**. Inspired by the **culinary expertise of Gordon Ramsay**, it provides **personalised meal recommendations** that align with users' tastes and constraints [1].



Sourced from: https://theresanaiforthat.com/gpt/chef-s-choice/

The system is powered by ChatGPT-4o, allowing it to interact conversationally with users, gather their preferences and dietary needs, and generate tailored meal suggestions. This interactive capability makes it an effective tool for enhancing the meal-planning experience. The core features include:

- **Cuisine Recommendation**: Suggests dishes from **various global cuisines**, introducing users to new and diverse meal options.
- **Dietary Adaptations**: Accommodates **specific dietary restrictions**, such as **gluten-free, vegan, or low-carb diets**, ensuring users receive suitable recommendations.
- **Time-Sensitive Recipes**: Provides meals that can be prepared within a user-specified time frame, making it ideal for those with **limited cooking time**.
- **Budget Considerations**: Recommends meals that fit within the user's **budget**, ensuring cost-effective meal planning.

The **integration of AI-driven conversation** in Chef's Choice enhances the meal selection process, making it a **user-friendly and interactive tool** for dietary decision-making. Its conversational approach to understanding user needs and delivering tailored suggestions serves as a valuable model for developing systems that aim to provide personalised dietary guidance. However, it does not specifically address the unique dietary needs of seniors managing chronic health conditions. Additionally, it lacks features that provide nutrient-specific analysis and real-time recipe modifications tailored to individual health requirements.

2.1.2 NutrifyMe – Web-Based Nutrition Management

NutrifyMe is a **web-based dietary and nutrition management application** designed to help users track their **nutrient intake and overall fitness** through **personalised diet plans**. It employs **structured health data** — such as **age, weight, height, gender, and activity levels** — to generate **customised dietary recommendations**. The system primarily focuses on **caloric intake, weight management, and fitness goals**, making it a practical tool for users looking to maintain **general health and wellness**.

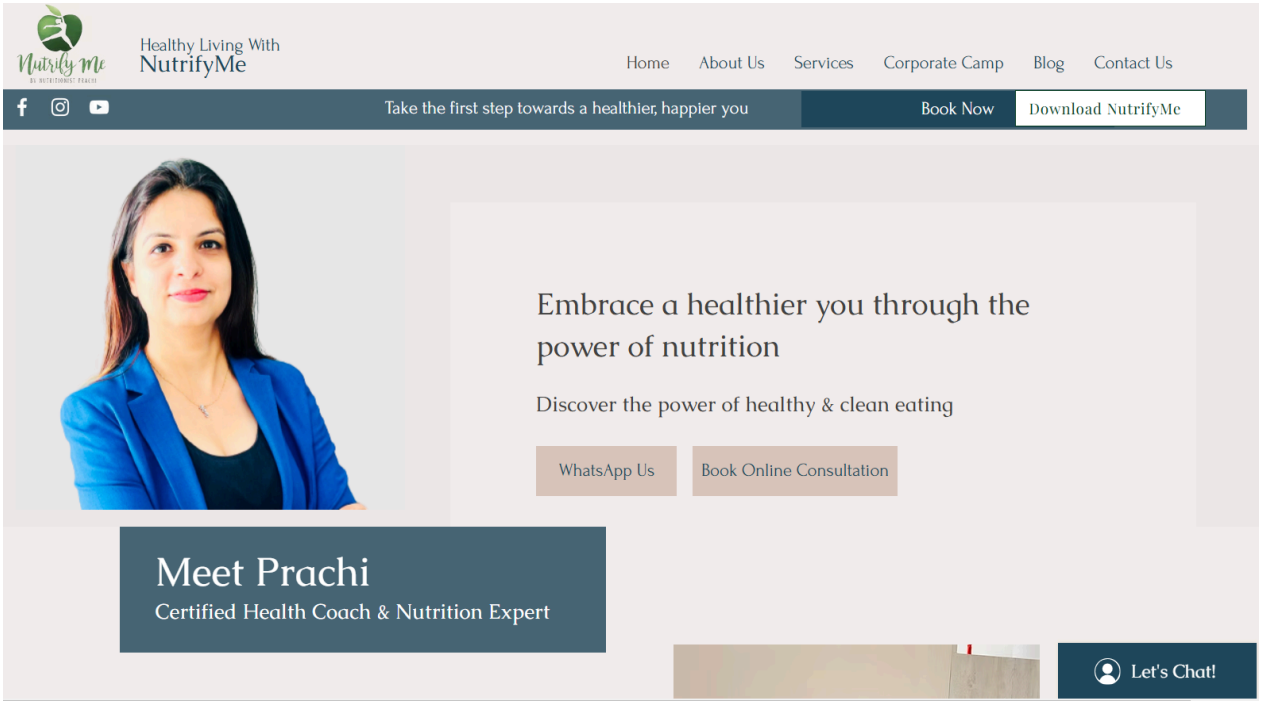
NutrifyMe collects user data and applies **health metrics calculations** such as:

- **Body Mass Index (BMI)** – Used to assess weight status and determine dietary needs.
- **Basal Metabolic Rate (BMR)** – Determines the number of calories required for basic bodily functions, guiding meal recommendations.

Using this data, NutrifyMe provides:

- **Personalised Diet Plans** – Generate structured **meal plans and calorie intake suggestions** based on user-specific goals (e.g. weight loss, muscle gain, or maintenance).
- **Recipe Recommendations** – Offer meals that align with caloric and macronutrient requirements.
- **Graphical Progress Tracking** – Visual tools that help users monitor their nutritional intake, weight trends, and fitness progress.
- **Expert Guidance** – Includes **blogs and advice from nutritionists** to provide additional dietary insights.

The application is built on React and uses Firebase for user authentication and data storage, ensuring scalability and responsiveness across devices [2].



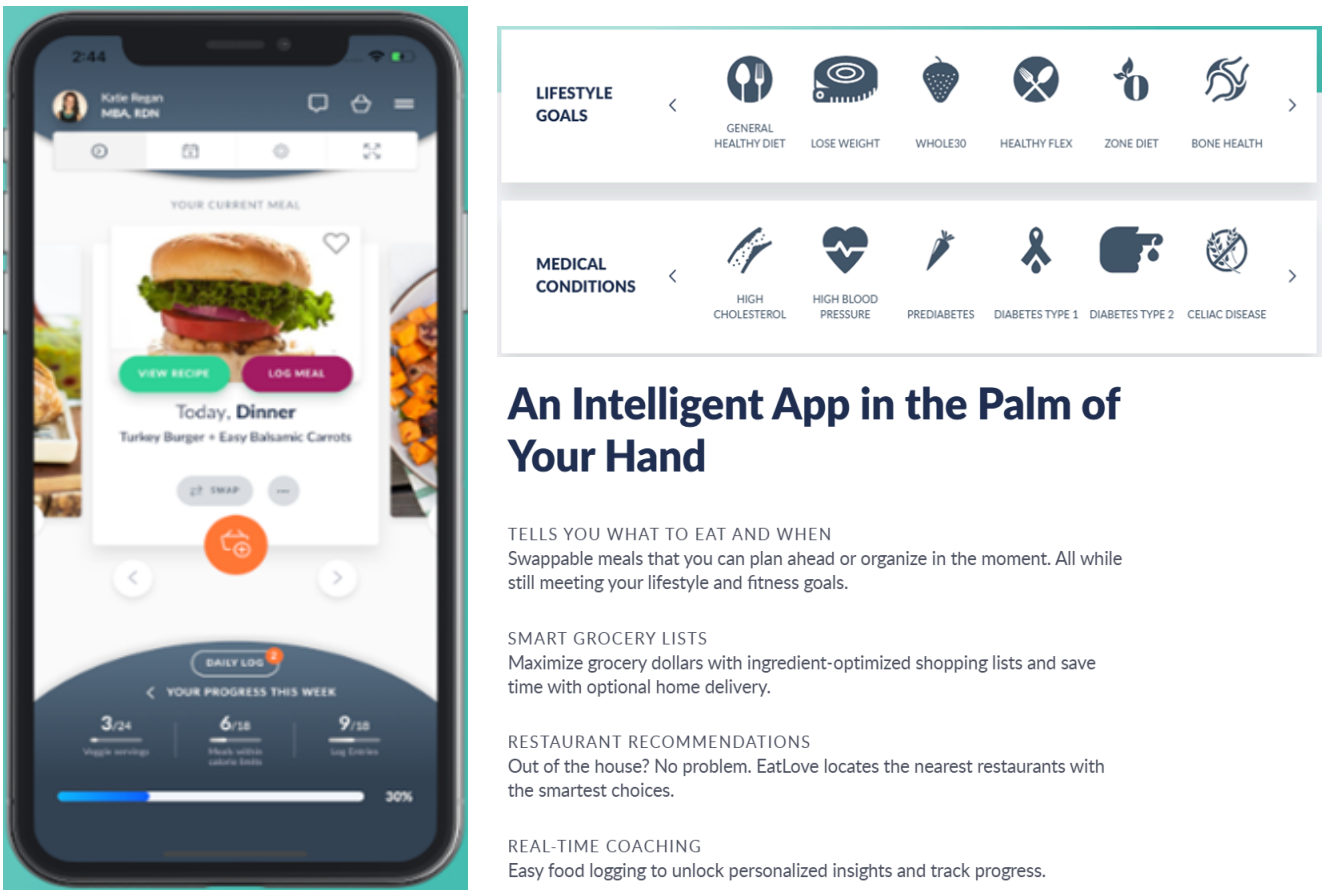
Sourced from: https://www.nutrifyme.in/

NutrifyMe highlights the importance of integrating user-specific health metrics into dietary systems. Its ability to generate personalised diet plans demonstrates the value of using structured data for health-driven recommendations. The application also emphasises user engagement by including progress tracking and recipes, which aligns with this project's focus on providing actionable dietary guidance. However, NutrifyMe lacks dynamic, recipe-level modifications that account for real-time user preferences or specific dietary restrictions. Furthermore, the application does not cater specifically to seniors managing chronic conditions, and its recommendations focus on high-level meal plans rather than ingredient-level analysis and substitutions.

2.1.3 EatLove: NLP-Powered Dietary Recommendations

Natural Language Processing (NLP) has become a crucial component in modern dietary recommendation systems, allowing them to analyse **unstructured data** such as **ingredient lists, recipe descriptions, and user-generated dietary logs**. These AI-driven systems use **text-processing techniques** to extract **meaningful nutritional insights**, offering features such as **ingredient-based recommendations, personalised meal planning, and real-time dietary tracking**. NLP-based dietary solutions are particularly effective in bridging the gap between **raw textual data and actionable dietary recommendations**, making them increasingly relevant in personalised nutrition.

One of the most advanced NLP-powered dietary recommendation systems is **EatLove**, which **integrates AI and nutrition science** to generate **customised meal plans based on user health data**. Unlike traditional systems that rely solely on structured nutrient data, EatLove **processes natural language inputs**, offering a more **intuitive and flexible approach to meal planning**.



Sourced from: https://www.eatlove.is/

EatLove provides a **comprehensive, AI-enhanced dietary system** with several **key features**:

- **Personalised Meal Planning**: The system tailors **meal plans and recipes** to accommodate **dietary restrictions and preferences**, such as **gluten-free, vegan, or low-sodium diets**.
- **Dynamic Nutritional Insights**: It **analyses recipes** for detailed **nutrient content**, providing users with **caloric breakdowns, macronutrient composition (protein, fats, carbohydrates), and micronutrient details**.
- **Real-Time Adjustments**: Users can **modify meal plans dynamically**, allowing the system to **update recommendations instantly** based on evolving dietary preferences and health goals.
- **Collaborative Tools**: EatLove enables **nutritionists and dietitians** to **collaborate with users**, refining their dietary plans to improve adherence and effectiveness.

These functionalities make EatLove **a powerful AI-driven dietary tool**, particularly for **users who require flexible, adaptable meal plans** tailored to **individual health needs**.

EatLove demonstrates the potential of NLP in dietary recommendation systems by showing how unstructured data can be processed and transformed into personalised health insights. However, it lacks the integration of deep learning (DL) techniques to analyse structured nutrient data comprehensively. This gap restricts its ability to classify recipes based on health-specific criteria such as cholesterol content or sodium levels. Additionally, it does not sufficiently focus on providing recipe modifications or alternatives tailored to chronic conditions. EatLove primarily offers meal plans but does not classify recipes as "healthy" or "unhealthy" or suggest real-time ingredient substitutions to meet specific health goals, such as reducing saturated fat or sodium content.

2.2 Addressing Gaps with the Proposed Project

While existing dietary recommendation systems such as **Chef's Choice, NutrifyMe, and EatLove** offer personalised meal guidance, they have notable limitations when catering to **seniors with chronic health conditions**. These systems primarily focus on **user preferences, calorie tracking, or general meal planning**, but they do not provide **recipe-level modifications, structured nutrient profiling, or DL-based health classification**. The proposed project builds upon these strengths while **filling the gaps** by integrating **DL, structured nutrient analysis, and NLP-powered ingredient substitutions** to deliver **a more adaptive and health-driven dietary recommendation system**.

2.2.1 Beyond Preference-Based Meal Selection in Chef's Choice

Chef's Choice is an **AI-powered meal recommendation system** that **conversationally gathers user preferences** and suggests meals based on cuisine type, available cooking time, and budget. However, it does not assess **nutritional content** or tailor recommendations for individuals managing conditions like **high cholesterol or hypertension**. The system lacks **nutrient-based classification** and the ability to **suggest modifications to improve a meal's health profile**.

In contrast, the proposed system integrates **DL to classify recipes** based on health-specific criteria such as **sodium, fat, and calorie levels**. Additionally, **NLP-powered ingredient modifications** allow users to receive **real-time suggestions for healthier substitutions**. For example, **butter in a recipe could be replaced with olive oil** for a heart-healthier option. This ensures that recommendations go beyond **taste preferences** to **support dietary goals that align with medical needs**.

2.2.2 Introducing Recipe-Level Modifications Missing in NutrifyMe

NutrifyMe personalises meal plans based on **BMI, BMR, and activity levels**, but its approach is **high-level meal planning** rather than **recipe-specific adjustments**. It does not provide **real-time ingredient modifications** or dynamically adapt recommendations based on **changing health needs**.

To address this, the proposed project introduces **recipe-level modifications** that allow users to **adjust specific ingredients** rather than being limited to **fixed meal plans**. For instance, if a **recipe contains high sodium**, the system will recommend **herbal alternatives** instead of salt. This ensures **greater flexibility and adaptability**, allowing users to maintain **nutritional compliance without sacrificing taste**.

2.2.3 Strengthening NLP-Based Dietary Analysis in EatLove

EatLove is an **NLP-powered system** that provides **personalised meal plans**, but it lacks **structured nutrient profiling and DL capabilities**. While it processes **unstructured text data**, it does not incorporate **scientific dietary constraints** into its recommendations. Additionally, it does not classify recipes as **healthy or unhealthy** based on structured data.

The proposed project combines **NLP with DL** to evaluate **both structured nutrient data and unstructured ingredient descriptions**. Unlike EatLove, which focuses on **meal plans**, this project ensures **scientifically backed dietary recommendations** that classify recipes and suggest modifications **based on long-term health goals**.

2.2.4 A Comprehensive Health-Driven Dietary System

By **integrating DL, structured data analysis, and NLP-powered modifications**, this project ensures that **dietary recommendations are both scientifically sound and practically useful**. Unlike existing systems, which focus on **either user preferences or general meal planning**, this project delivers **personalised, health-conscious recommendations that dynamically adapt to individual dietary needs**.

This **comprehensive and adaptive approach** not only **fills the gaps** in current systems but also enhances **AI-driven nutritional guidance**, making it a **practical and medically relevant tool for seniors managing chronic health conditions**.

2.3 Techniques and Methods

This section outlines the core AI techniques, ML algorithms, and NLP methodologies used in the project.

2.3.1 Software Libraries

These are the software libraries required for this project (Table 3):

Name of software library	Application
Pandas	It manages and preprocesses data, including cleaning nutrient datasets, normalising values, and engineering health-specific features.
Scikit-learn	It supports implementing and testing ML algorithms such as **Random Forest** and **Logistic Regression**. It will also provide evaluation metrics for model assessment.
TensorFlow (Keras)	Enables the development of **MLNNs** to classify recipes and analyse nutrient profiles alongside ingredient lists. It also supports both structured and unstructured data inputs.
Natural Language Toolkit (NLTK) or spaCy	NLP libraries such as NLTK or spaCy will process unstructured ingredient descriptions, enabling tasks like: a. **Tokenisation**: Breaking down text into analysable components, such as individual ingredients. b. **Semantic Analysis**: Understanding ingredient roles (e.g. differentiating "low-fat butter" from "butter"). c. **Ingredient Substitutions**: Suggesting healthier alternatives based on ingredient context, such as replacing butter with olive oil.

Table 3: Summary of software libraries and applications

2.3.2 Algorithms

1. **Random Forest**

Random Forest will classify recipes as "healthy" or "unhealthy" based on structured nutrient data. Its feature importance analysis will help identify the most critical nutrients (e.g. sodium or cholesterol) affecting health classifications.

2. **Logistic Regression**
   This algorithm provides a baseline for binary classification tasks, such as predicting whether a recipe aligns with a specific dietary goal. Its simplicity aids interpretability but may be limited to complex relationships.

3. **Multi-Layer Neural Networks (MLNNs)**
   MLNNs will process structured nutrient profiles and unstructured ingredient descriptions simultaneously. They are well-suited for identifying non-linear relationships and multi-modal data interactions, enabling tasks like low-sodium or low-cholesterol recipe classification.

4. **NLP-Augmented Models**
   Incorporating NLP techniques into MLNNs will allow the system to dynamically modify recipes. For example, it can identify high-sodium ingredients and recommend substitutions.

2.3.3 Research Methodologies

1. **Data Preprocessing**
   Preprocessing ensures data quality through the following:
   - Cleaning and normalising nutrient datasets.
   - Tokenising and vectorising ingredient descriptions for NLP models.
   - Engineering features to highlight health-critical metrics, such as fat-to-protein ratios.

2. **Evaluation and Validation**
   Evaluation uses metrics (accuracy, precision, recall, F1-score) to assess classification performance. Cross-validation ensures models generalise well to new data.

3. **Explainability and Usability**
   Feature importance analysis (e.g. using Random Forest) and interpretable NLP outputs enhance system transparency, enabling users to understand the rationale behind recommendations.

2.4 Supporting Research Studies

The development of the proposed dietary recommendation system is further supported by a wealth of research demonstrating the effectiveness of dietary interventions and AI-driven methodologies in health management. Diet plays a vital role in reducing cholesterol levels, improving cardiovascular health, and managing chronic conditions. At the same time, AI and ML have proven effective in personalising nutritional analysis and recommendations, enabling tools that adapt to user-specific health needs.

2.4.1 The Role of Diet in Managing Cholesterol and Cardiovascular Health

Dietary interventions have been widely recognised as a cornerstone of chronic disease management, particularly in controlling cholesterol levels and reducing cardiovascular risk factors. Key studies include:

a. **Anderson, T.J., et al. (2017)**
This study emphasises the importance of adopting a heart-healthy diet that limits saturated fats and includes fibre-rich foods such as whole grains and fruits. It concludes that such dietary changes can significantly lower cholesterol levels and reduce cardiovascular risks. These findings reinforce the project's goal of providing personalised recommendations that incorporate nutrient-specific analysis to address chronic conditions like high cholesterol [3].

b. **Jenkins et al. (2003)**
This research demonstrated that a dietary portfolio of cholesterol-lowering foods, including plant sterols, soy protein, and soluble fibre, could achieve cholesterol reductions comparable to those of lovastatin, a common cholesterol-lowering medication. The study underscores the power of diet in managing health conditions and validates the project's approach of integrating evidence-based dietary guidelines into personalised meal plans [4].

2.4.2 AI and Machine Learning (ML) in Nutritional Analysis and Recommendation Systems

The intersection of AI and healthcare has opened new possibilities for creating adaptive and personalised dietary tools. The following studies highlight the potential of AI techniques in nutritional analysis:

a. **Esteva, A., et al. (2019)**
This guide explores the application of DL in healthcare, emphasising its ability to analyse complex datasets and provide customised recommendations. The study highlights the potential of AI-driven dietary tools in addressing specific health needs, such as cholesterol management, which aligns with the proposed system's use of MLNNs for recipe classification and nutrient-specific analysis [5].

b. **Kaur, P., et al. (2020)**
This research discusses an ML-based food recommendation system that leverages predictive analytics to offer personalised meal options. It demonstrates how user preferences and dietary needs can inform algorithmic recommendations, providing foundational insights into building personalised nutrition systems like the proposed project [6].

c. **Jitendra, M.S.N.V., et al. (2023)**
The study introduces an ML-powered personalised food recommendation system employing algorithms such as Random Forest and Logistic Regression. While effective for general recommendations, the study highlights opportunities to enhance classification accuracy and nutrient-specific recommendations through advanced AI techniques, which the proposed project integrates [7].

d. **Mitchell, E. G., et al. (2021)**

> This research combines ML with expert knowledge to recommend nutrition goals tailored to individual health needs. The study emphasises the importance of blending algorithmic predictions with domain expertise, which the proposed system adopts by integrating evidence-based dietary guidelines with AI-driven analysis for actionable recommendations [8].

The proposed system builds on these findings by integrating evidence-based dietary practices with cutting-edge AI techniques. By leveraging DL, predictive analytics, and expert knowledge, the project aims to deliver a comprehensive solution that bridges the gap between theoretical health insights and practical dietary tools. This synthesis ensures that the system not only aligns with proven health principles but also offers innovative, real-world applications to improve seniors' quality of life through personalised nutrition. As such, this project stands at the intersection of technology and healthcare, poised to make meaningful contributions to personalised dietary management.

Word Count: 2409

Chapter 3: Project Design

3.1 Domain and Target Users

3.1.1 Domain of the Project

This project operates within the **healthcare and nutrition** domain, where dietary habits directly influence health outcomes, particularly for individuals with **chronic conditions**. The integration of **DL** and **NLP** with **nutritional science** enables intelligent dietary recommendations, addressing **structured (nutrient values) and unstructured (ingredient descriptions) data**. The project aligns with three key areas (Table 4):

Involvement	Explanation
Nutritional Science	Use **evidence-based dietary strategies**, such as cholesterol-lowering diets, to create health-specific meal plans.
Artificial Intelligence in Healthcare	Employs **ML models** to analyse structured and unstructured data, enhancing dietary recommendations.
User-Centric Health Management	Transforms **theoretical nutritional guidelines** into **practical meal suggestions**, making healthy eating more accessible.

Table 4: Summary of key involvements and explanations

The healthcare and nutrition sector benefits significantly from AI advancements, and this project presents an **innovative approach** to **personalised dietary management**.

3.1.2 Target Users and Their Needs

The system serves **three user groups**, consisting of **one primary and two secondary users**, each with distinct needs, as shown in Table 5 below.

User	Needs and Challenges	How the System Addresses These

		Gaps
Seniors Managing Chronic Conditions (Primary)	Struggle with complex dietary guidelines and identifying suitable meals for diabetes, hypertension, or high cholesterol.	Provides **tailored recipe suggestions and ingredient modifications** to align with dietary restrictions.
Caregivers and Nutritionists (Secondary)	Need tools to simplify meal planning for patients or dependents.	Offers **personalised dietary recommendations**, enabling structured meal plans.
General Health-Conscious Individuals (Secondary)	Want to make healthier eating choices but lack nutritional expertise.	Provides **nutrient-rich recipe suggestions** and **ingredient-level analysis** for informed decision-making.

Table 5: Summary of users, their needs and how the system addresses them

This system provides an accessible, **scientifically backed dietary tool** for seniors and health-conscious individuals.

3.2 Overall Project Structure

This section details the **workflow and design** of the project, encompassing **research questions, data preprocessing, and model implementation**.

3.2.1 Research Questions

As mentioned in the literature review, despite advancements, significant gaps remain in existing systems that address nuanced health conditions. Existing research has also underscored the importance of dietary interventions in managing chronic illnesses and the transformative potential of AI techniques, such as DL and NLP, in healthcare and nutrition [3][5]. Therefore, the following research questions and hypotheses (Table 6) guiding the project are:

Research Question	Hypothesis
How can deep learning (DL) techniques be applied to classify recipes based on nutritional content?	**MLNNs** can accurately classify recipes into **health-focused categories** by analysing structured and unstructured data.
What methods can effectively recommend healthier alternatives for seniors with specific dietary restrictions?	NLP-enhanced ML models can **dynamically identify ingredient substitutions** to meet health requirements.
How can ingredient-level analysis support practical dietary decisions for seniors?	NLP-powered **ingredient analysis** can improve **usability and personalisation**, allowing seniors to make **informed meal choices**.

Table 6: Summary of research questions and hypotheses

These questions ensure that the project meets its primary objective — **creating an effective AI-driven dietary recommendation system**.

3.2.2 Data Collection and Preprocessing

The dataset, "**Epicurious - Recipes with Rating and Nutrition**", is sourced from Kaggle, and its recipe information is lifted from the website "Epicurious" [9]. It contains over 20,000 recipes, each with structured and semi-structured data such as recipe rating, nutritional information and assigned category (sparse) [10]. The preprocessing steps (Table 7) involved include:

Preprocessing Step	Explanation
Data Cleaning	Missing values are imputed via **mean/median imputation**, and duplicate recipes are removed.
Text Preprocessing	**Tokenisation, lemmatisation, and stopword removal** refine ingredient lists for NLP analysis.
Feature Scaling	Normalisation ensures numerical attributes remain consistent across the dataset.
Class Balancing	**Random Oversampling** is applied to balance the **healthy vs. unhealthy** class distribution.
Data Splitting	The dataset is divided into **training (70%)**, **validation (15%)**, and **test (15%)** sets.

Table 7: Summary of preprocessing steps and explanations

3.2.3 Model Design

To effectively classify recipes based on nutritional content and provide personalised dietary recommendations, a two-stage approach is employed in this project: implementing baseline models for initial evaluation and developing enhanced models that integrate advanced techniques, including DL and NLP.

3.2.3.1 Stage 1: Baseline Models

The baseline models serve as interpretable and computationally efficient benchmarks to evaluate the dataset and lay the foundation for advanced modelling. They focus on structured data, such as nutrient values and categorical labels.

Random Forest (Figure 1) is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes for classification tasks. It can also handle high-dimensional data and capture complex interactions between features, thus evaluating feature importance and identifying key nutrients (e.g. sodium, cholesterol, calories) that significantly influence recipe classification. These algorithms have been applied in healthcare and are known to predict disease risks from imbalanced data, demonstrating their robustness in handling complex datasets [11].

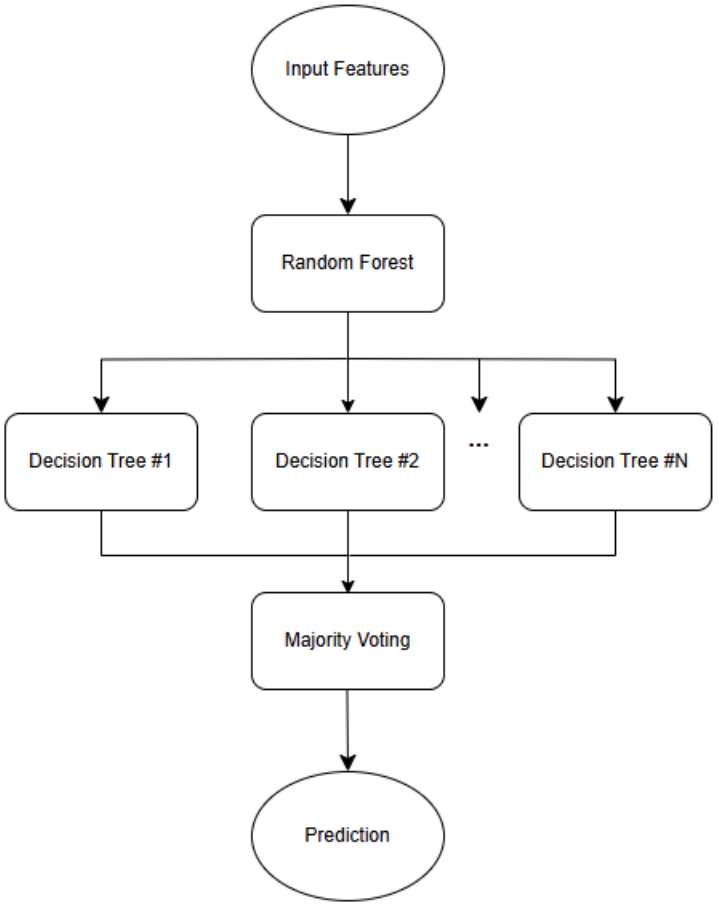


Figure 1: Flowchart of Random Forest architecture

Logistic Regression (Figure 2) is a statistical model that applies a logistic function to model binary dependent variables. It has been utilised in personalised food recommendation systems, providing a foundation for more complex predictive models [7]. Hence, the model can be used to determine whether a recipe meets a specific dietary criterion and compare its performance with more advanced models.

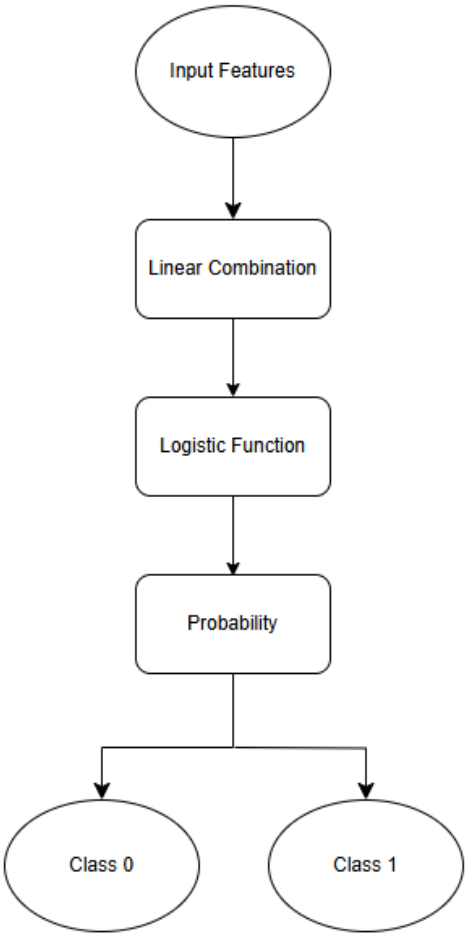


Figure 2: Flowchart of Logistic Regression architecture

3.2.3.2 Stage 2: Enhanced Models with NLP Integration

Building on the insights and structure established by baseline models, the enhanced models incorporate DL techniques and NLP for advanced analysis. These models are designed to process both structured nutritional data and unstructured ingredient text. These are the steps required to do so (Table 8):

Step	Explanation
Designing the Multi-Layer Neural Network (MLNN)	The input layer accepts structured features (numerical nutritional data) and unstructured features (vectorised ingredient text). Employs multiple dense layers with activation functions (e.g. ReLU) to learn non-linear relationships and hierarchical patterns within the data. Dropout layers are introduced between hidden layers to reduce overfitting by randomly deactivating a portion of neurons during training. The output layer uses a softmax activation function to output probabilities across multiple recipe health categories.
Training the MLNN	For multi-class classification tasks, cross-entropy loss is used to measure the discrepancy between predicted and actual class labels.

	The Adam Optimiser is chosen for its ability to adapt learning rates dynamically, improving convergence. Evaluation metrics are calculated during training to monitor performance.
Integration of NLP Techniques	Pre-trained models like GloVe are fine-tuned to understand ingredient context (e.g. distinguishing "low-fat yoghurt" from "full-fat yoghurt"). Semantic similarity is calculated using cosine similarity between word embeddings to suggest healthier alternatives.
Tuning of hyperparameters	Adjust hyperparameters such as the number of layers, number of neurons per layer, learning rate, and dropout rate to optimise model performance.

Table 8: Summary of steps and explanations

3.2.3.3 Recommendation Engine

The recommendation engine is a critical component of the project, designed to translate the outputs of the ML and NLP models into actionable dietary guidance. It utilises predictions, classifications and ingredient analyses to provide personalised recipe suggestions, healthier ingredient substitutions, and tailored meal plans that align with users' dietary needs and preferences. Two algorithms have been proposed for the system to work.

First, **content-based filtering** helps to recommend recipes based on their features (e.g. nutrient profiles, ingredient lists, health categories etc.) and their similarities to specific user dietary requirements such as low cholesterol or low sodium. Cosine similarity is used to compute similarity scores between recipes and rank them based on how similar they are to pre-defined user preferences. The second one is **rule-based filtering**, which implements rules or modifications according to health guidelines. For instance, conditional statements or decision trees can be applied to enforce dietary constraints.

A **hybrid** recommendation system is also considered for this component as it offers a balance between user preferences and health requirements. By blending similarity-based ranking with rule-based adjustments, this approach helps to refine recommendations and further enhances personalisation, ensuring dietary compliance.

The flowchart below summarises the architecture of the hybrid recommendation system (Figure 3).

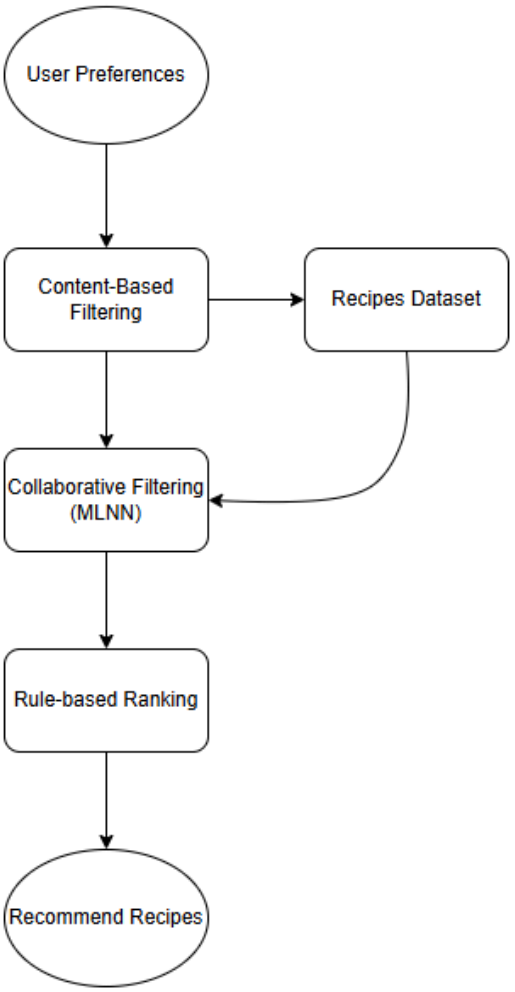


Figure 3: Flowchart of hybrid recommendation system architecture

3.2.4 Evaluation Strategy

The evaluation strategy ensures that the models accurately classify recipes, recommend suitable ingredient modifications, and provide personalised dietary suggestions. The assessment consists of **classification metrics**, **user testing**, and **comparative analysis** with existing systems.

3.2.4.1 Model Performance Evaluation

The following metrics (Table 9) are used to measure classification accuracy and predictive reliability:

Metric	Purpose
Accuracy	Measures how often recipes are correctly classified as “healthy” or “unhealthy.”
Precision	Ensures minimal false positives, particularly for **healthy** recipe classifications.
Recall	Ensures the model correctly identifies all relevant recipes, preventing false negatives.
F1-Score	Balances precision and recall to provide a **comprehensive evaluation** for imbalanced datasets.

Cross-Entropy Loss	Evaluates the discrepancy between predicted probabilities and actual class labels to optimise model performance.

Table 9: Summary of metrics and purposes

For initial benchmarking, **Random Forest and Logistic Regression** are evaluated before transitioning to **MLNN with NLP integration**. Hyperparameter tuning is performed to optimise learning rates, dropout rates, and neural network structures.

3.2.4.2 User Testing and Usability Assessment

User testing assesses the practical usability of the system among **seniors, caregivers, and health-conscious individuals**. A **survey-based evaluation** collects feedback on key usability factors (Table 10):

Factor	Description
Ease of Use	Measures how intuitive and accessible the system is, especially for seniors.
Relevance of Recommendations	Assesses how well recipe suggestions align with health conditions and dietary needs.
Ingredient Substitution Effectiveness	Evaluates whether suggested modifications improve nutritional quality while maintaining meal taste and usability.
Overall Satisfaction	Captures user perceptions of the system's effectiveness.

Table 10: Summary of factors and descriptions

A **two-week pilot test** tracks user interactions, measuring how often users follow recommendations and whether modifications align with dietary restrictions.

3.3 Important Technologies

3.3.1 IDE and Programming Language

Jupyter Notebook is used as the primary **integrated development environment (IDE)** for its interactive capabilities, which are ideal for data analysis, model training, and debugging. **Python**, known for its simplicity and extensive library ecosystem, serves as the **programming language** for this project. Its versatility ensures the seamless implementation of ML and NLP tasks.

3.3.2 Libraries and Frameworks

Several libraries and frameworks are employed to address specific stages of the pipeline:

1. **Pandas** is used for data manipulation and preprocessing, enabling efficient handling of structured data through its DataFrame structure.
2. **Scikit-learn** supports baseline models along with utilities for preprocessing, feature selection, and evaluation metrics.
3. **TensorFlow**, known for its scalability and deployment capabilities, is utilised for building and training advanced DL models.

4. **SpaCy** and **NLTK** handle text preprocessing, including tokenisation, lemmatisation, and stopword removal, which are essential for processing ingredient lists.
5. **Gensim** is employed to create or import pre-trained word embeddings (e.g. Word2Vec, GloVe), which enhance the contextual understanding of ingredient data.
6. **Matplotlib** and **Seaborn** enable data visualisation, offering tools for creating insightful plots that aid in performance evaluation and exploratory analysis.

3.3.3 Development Tools

The system was developed using Python-based tools to enable both backend model deployment and a user-friendly web interface.

Flask was used to handle backend operations, including loading the trained MLNN model and exposing it via RESTful Application Programming Interfaces (APIs). Its lightweight nature made it ideal for local deployment, allowing secure and efficient communication between the model and interface.

Streamlit was used for the web-based interface, chosen for its simplicity and speed in building interactive applications. It allowed real-time user input and dynamic display of recipe recommendations with minimal front-end coding. Users could easily adjust dietary preferences and view predictions directly from structured data frames and charts.

Both enabled a responsive, private and accessible system, which is well-suited for seniors and caregivers seeking practical and personalised nutrition advice.

3.4 Work Plan

The project follows a structured 20-week timeline, visualised in a **Gantt chart** (Figure 4), covering **data preprocessing, model development, evaluation, and deployment**. Table 11 summarises the project work plan and activities.

Weeks	Objectives	Activities
1 - 2	Project Planning and Ideation	- Brainstorm project goals and scope - Identify resources needed - Outline initial research questions and aims
2 - 4	Literature Review and Project Proposal	- Review relevant papers on diet management and DL in healthcare - Identify existing systems and research gaps - Draft and finalise the project proposal
4 - 5	Data Acquisition and Preprocessing	- Identify and download the selected dataset (Epicurious dataset) - Clean and preprocess data - Perform Exploratory Data Analysis (EDA) - Split the dataset into training, validation, and test sets
6 - 7	Feature Engineering and Model Planning	- Create derived features (e.g. sodium-to-calorie ratio) - Tokenise and vectorise ingredient text using TF-IDF or word embeddings.

		- Design model architectures for baseline and enhanced models
7 - 9	Baseline Model Implementation and Evaluation	- Train and test models on structured data - Evaluate using metrics (accuracy, precision, recall, F1-score) - Identify limitations of baseline models and refine data features as needed
9 - 10	Preliminary Report	- Document methods for data preprocessing and baseline model implementation - Summarise results of baseline model evaluation - Provide insights on planned improvements for enhanced models
11 - 13	Enhanced Model Development	- Build and train MLNNs for recipe classification - Integrate NLP techniques for processing ingredient data - Experiment with model hyperparameters to improve performance
13 - 14	Enhanced Model Testing and Refinement	- Validate enhanced models on the validation dataset - Fine-tune hyperparameters (e.g. learning rate, number of layers, dropout rate) - Compare enhanced models with baseline results for performance improvement - Analyse errors and refine models for better accuracy
15 - 16	Recommendation Engine Development	- Design algorithms for content-based filtering and rule-based filtering - Implement a recommendation engine to suggest recipes and ingredient substitutions - Test the recommendation engine for accuracy and relevance of recommendations
17 - 18	Deployment and Usability Testing	- Build a web-based interface using Flask and Streamlit - Deploy the recommendation system to a local or cloud server - Conduct usability testing with target users (seniors, caregivers) - Gather feedback on system functionality and user experience
19 - 20	Final Report	- Compile all findings and methodologies - Include evaluation results, challenges faced, and lessons learned - Suggest future directions and potential enhancements for the project

Table 11: Summary of work plan with key objectives and activities

	Weeks																			
Tasks	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Project Planning and Ideation																				
Literature Review and Project Proposal																				
Data Acquisition and Preprocessing																				
Feature Engineering and Model Planning																				
Baseline Model Implementation and Evaluation																				
Preliminary Report																				
Enhanced Model Development																				
Enhanced Model Testing and Refinement																				
Recommendation Engine Development																				
Deployment and Usability Testing																				
Final Report																				

Figure 4: Gantt chart visualising work plan

Word Count: 2086

Chapter 4: Implementation

The prototype demonstrates the feasibility of applying ML and NLP techniques to analyse structured nutritional data and unstructured ingredient text. Through the implementation of various ML models and an NLP-enhanced classification system, the prototype focuses on building a functional dietary recommendation engine that supports personalised, health-conscious decision-making.

4.1 Data Preprocessing

Data preprocessing is a fundamental step to ensure data quality, consistency, and compatibility with machine learning (ML) models.

4.1.1 Handling Missing Values

Several numerical attributes in the **“Epicurious - Recipes with Rating and Nutrition”** dataset contain missing values, including **calories, fat, and sodium**. Since most **ML and DL models require complete numerical inputs** [12], these missing values (Figure 5) must be addressed appropriately.

```
In [10]: # Check for missing values
         missing_values = df.isnull().sum()
         missing_values

Out[10]: title            0
         rating           0
         calories      4117
         protein       4162
         fat           4183
                        ...
         cookbooks        0
         leftovers        0
         snack            0
         snack week       0
         turkey           0
         Length: 680, dtype: int64

In [11]: # Filter columns with null values
         missing_cols_with_nulls = missing_values[missing_values > 0]
         missing_cols_with_nulls

Out[11]: calories    4117
         protein     4162
         fat         4183
         sodium      4119
         dtype: int64
```

Figure 5: Code snippet of checking missing values from the dataset

To maintain data integrity while preventing bias, missing values are **replaced with the median** of their respective columns. **Median imputation** (Figure 6) is chosen as it is less sensitive to extreme values compared to mean imputation.

```
In [12]: # Impute missing values (using median)
         for col in ['calories', 'protein', 'fat', 'sodium']:
             df[col].fillna(df[col].median(), inplace=True)

In [13]: df.isnull().sum()

Out[13]: title         0
         rating        0
         calories      0
         protein       0
         fat           0
                      ..
         cookbooks     0
         leftovers     0
         snack         0
         snack week    0
         turkey        0
         Length: 680, dtype: int64
```

Figure 6: Code snippet of median imputation

4.1.2 Handling Outliers

Outliers in **calories, sodium, and fat content** can distort model performance and lead to inaccurate predictions. These outliers may arise due to **errors in data collection** or the presence of **extremely high-calorie and high-fat recipes**, which can bias the classification of healthy and unhealthy meals (Appendix A.1).

Removing outliers could result in a significant loss of data, which might weaken the model's ability to generalise, especially for smaller datasets [13]. Instead of removing these outliers, an **outlier-capping strategy** is employed using the **Interquartile Range (IQR) method** (Figure 7). This adjusts extreme values while preserving data integrity, making the dataset more suitable for ML algorithms [14].

```
In [18]: # Define the outlier handling function
def cap_outliers(df_cleaned, column_name):
    Q1 = df[column_name].quantile(0.25)
    Q3 = df[column_name].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    df_cleaned[column_name] = np.clip(df_cleaned[column_name], lower_bound, upper

# Apply the function to numerical columns
numerical_cols = ['rating', 'calories', 'protein', 'fat', 'sodium']
for col in numerical_cols:
    cap_outliers(df, col)

# Check the results
print(df[numerical_cols].describe())

# Save the dataset with handled outliers
print("Outliers handled and dataset saved")
```

```
             rating      calories       protein           fat        sodium
count  18251.000000  18251.000000  18251.000000  18251.000000  18251.000000
mean       3.995911    401.373541     14.619117     21.147143    420.958632
std        0.667415    244.884604     14.802906     15.607316    388.823859
min        2.812500      0.000000      0.000000      0.000000      0.000000
25%        3.750000    238.000000      4.000000     11.000000    132.000000
50%        4.375000    331.000000      8.000000     17.000000    294.000000
75%        4.375000    516.500000     21.000000     28.000000    589.000000
max        5.000000    934.250000     46.500000     53.500000   1274.500000
Outliers handled and dataset saved
```

Figure 7: Code snippet of outer-capping strategy using IQR

4.1.3 Exploratory Data Analysis (EDA)

EDA provides **insights into patterns and relationships** within the dataset, guiding **feature engineering and model selection**.

4.1.3.1 Correlation Analysis

Correlation analysis helps identify **relationships** between different numerical features. This is crucial because **nutritional attributes are often interdependent**. For instance, recipes with **higher calorie content** may also have **higher fat content**, and sodium levels may be associated with the type of cuisine.

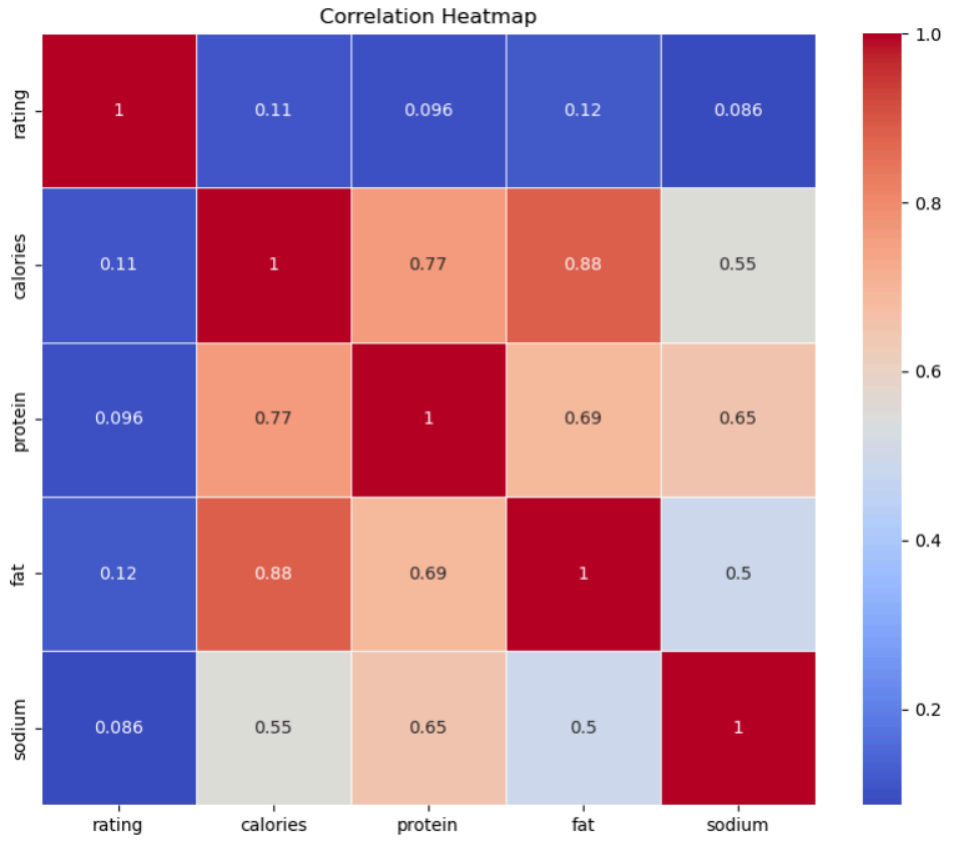


Figure 8: Correlation heatmap depicting relationships of numerical features

Key insights from the correlation heatmap (Figure 8) include:

- **Calories and Fat (r = 0.88)**: Recipes with higher calorie content typically have greater fat content, as fat significantly contributes to caloric values.
- **Calories and Protein (r = 0.77)**: High-protein recipes generally contain higher calorie counts, reflecting macronutrient influence on dietary categorisation.
- **Fat and Sodium (r = 0.5)**: High-fat recipes often also contain elevated sodium levels, potentially due to processed ingredients such as cheese and cured meats.

These findings are used to **enhance feature selection**, ensuring that the ML models leverage meaningful variables while avoiding redundancy.

4.1.3.2 Distribution Analysis

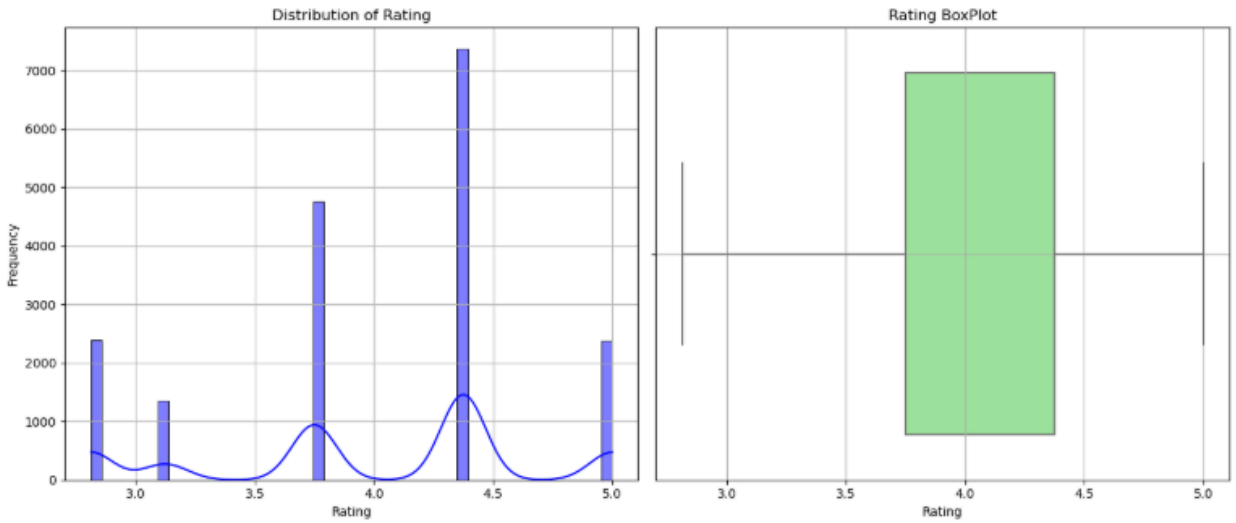


Figure 9: Distribution plot and box plot of ratings

A distribution analysis (Figure 9) was performed to **examine user preferences** through recipe ratings. Most recipes received **ratings between 4.0 and 4.5**, indicating prevalent high satisfaction levels among users. This suggests that seniors managing chronic conditions and health-conscious users are **more inclined** toward well-rated recipes, thus aligning recommendation outcomes with user taste and satisfaction preferences.

4.1.3.3 Cooking Categories Analysis

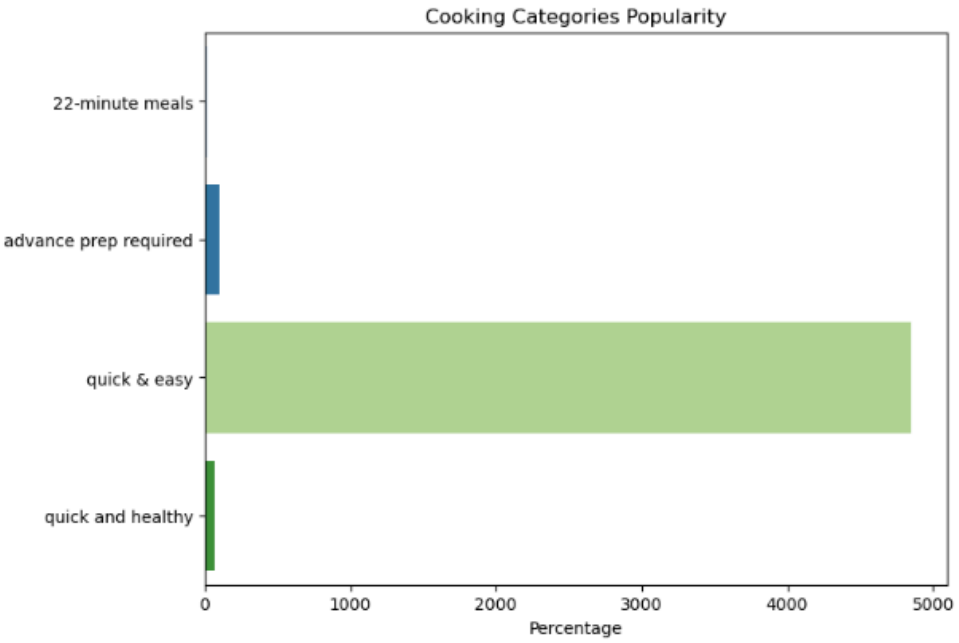


Figure 10: Bar chart of cooking categories' popularity

Additionally, analysing cooking categories (Figure 10) was instrumental in **identifying popular recipe types**. The dominance of categories such as "**quick & easy**" highlights a user preference for recipes requiring minimal preparation and cooking effort. This insight is particularly valuable for tailoring the recommendation system towards seniors who may have limited energy or time for cooking, as well as caregivers looking for convenient, healthy meal options.

4.2 Text Preprocessing

To enable the **integration of unstructured ingredient text** into the classification model, **NLP techniques** are applied. This preprocessing step (Appendix A.2) is essential for transforming raw text into a structured format suitable for ML.

Key Text Processing Steps:

- **Tokenisation**: Splits ingredient text into individual words.
- **Lowercasing**: Converts text to lowercase to avoid case-sensitive mismatches.
- **Stopword Removal**: Eliminates common words such as "and" or "with" to retain meaningful terms.
- **Lemmatisation**: Reduces words to their root forms (e.g. "chopped" to "chop").
- **Vectorisation**: Translates processed text into numerical values using **TF-IDF (Term Frequency-Inverse Document Frequency)**.

4.3 Creating a Target Variable

To build a **classification model**, recipes are labeled as either **"Healthy"** or **"Unhealthy"** based on **nutritional thresholds** (Figure 11). These thresholds align with dietary guidelines for seniors managing chronic conditions like **high cholesterol, diabetes, and hypertension**.

Criteria for a Recipe to be Considered Healthy:

- **Sodium < 500 mg:** Helps prevent high blood pressure.
- **Fat < 15 g:** Limits unhealthy fat intake.
- **Calories < 400 kcal:** Encourages balanced portion sizes.

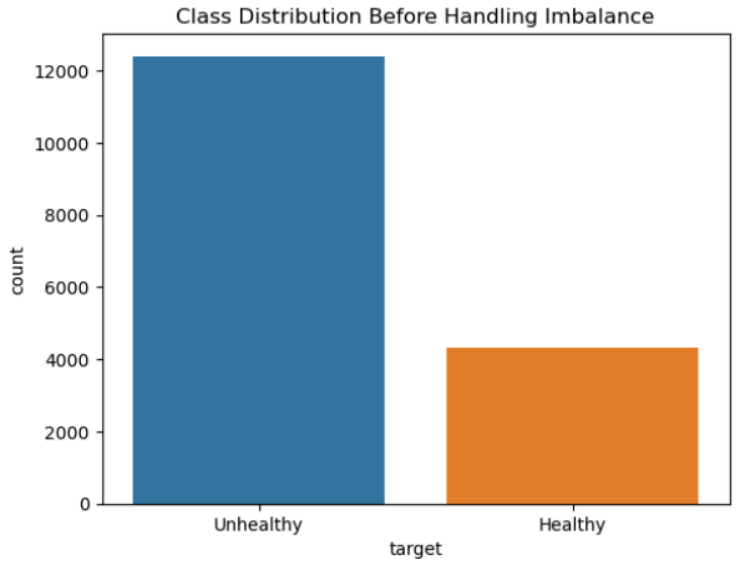
Each recipe is evaluated based on these criteria, and a **binary target variable** is assigned. This classification is critical for **training supervised learning models** that predict the healthiness of recipes.

```
In [34]: # Define thresholds for healthy recipes
         def classify_recipe(row):
             if row['sodium'] < 500 and row['fat'] < 15 and row['calories'] < 400:
                 return 'Healthy'
             else:
                 return 'Unhealthy'

         # Apply classification logic
         df_processed_recipe['target'] = df.apply(classify_recipe, axis=1)
         y = df_processed_recipe['target']
```

Figure 11: Code snippet of defining thresholds through a function

4.4 Handling Class Imbalance

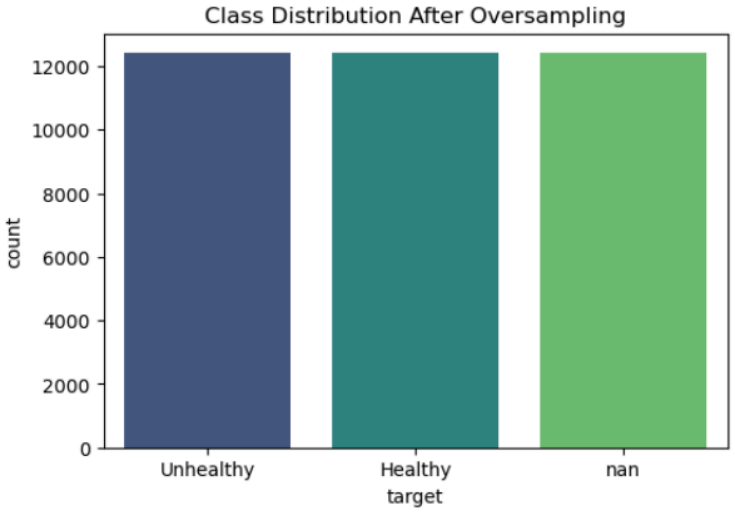
Class imbalance occurs frequently in real-world datasets, where one category (e.g. "Unhealthy") significantly outnumbers another (e.g. "Healthy"). Such an imbalance can skew model predictions towards the dominant class (Figure 12).




```
Unhealthy    12404
Healthy       4327
Name: target, dtype: int64
```

Figure 12: Count plot depicting distribution before handling imbalance

To address this, **Random Oversampling** was applied, duplicating instances of the minority "Healthy" class until it matched the majority class distribution (Figure 13).




```
Class Distribution After Oversampling:
Unhealthy    12404
Healthy      12404
nan          12404
Name: target, dtype: int64
```

Figure 13: Count plot depicting distribution after oversampling

This method was preferred over **SMOTE (Synthetic Minority Oversampling Technique)** for two key reasons:

- **Suitability for mixed data:** Unlike SMOTE, Random Oversampling doesn't generate synthetic samples, making it appropriate for datasets containing structured nutritional features alongside textual ingredient data.

- **Interpretability preservation:** Random Oversampling maintains authentic data characteristics, ensuring dietary recommendations and insights remain practically valid.

Balancing the dataset effectively mitigates bias, allowing the model to accurately and fairly classify recipes as healthy or unhealthy.

4.5 Data Scaling and Splitting of Data

To ensure **consistent feature representation**, **Min-Max Scaling** is applied to transform numerical features (e.g. calories, fat, sodium) into a common range of **0 to 1** (Figure 14). This prevents **models with gradient-based learning** (e.g. NNs) from being influenced by disproportionately large numerical values.

```
# Initialize MinMaxScaler
scaler = MinMaxScaler()

# Scale numeric features
df_scaled_numeric = pd.DataFrame(scaler.fit_transform(df_balanced[numeric_cols]), columns=numeric_cols

# Concatenate the scaled numeric features and TF-IDF transformed ingredient features
df_final_features = pd.concat([df_scaled_numeric, tfidf_df], axis=1)
```

Figure 14: Code snippet of data scaling using MinMaxScaler

The dataset is then split into **training, validation, and testing subsets** to evaluate model performance. A **70-15-15% split** is used:

- **70% Training Set:** Used to train the model.
- **15% Validation Set:** Optimizes hyperparameters and prevents overfitting.
- **15% Testing Set:** Evaluates final model performance on unseen data.

A **stratified split** is applied to maintain the **same proportion of healthy and unhealthy recipes** across all subsets, ensuring **fair evaluation** (Figure 15).

```
# Extract final features and target
X = df_final_features
y = df_balanced['target']  # Use the corrected target column

# Split data into train (70%) and temp set (30%)
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)

# Split temp set equally into validation (15%) and test set (15%)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42, strati

# Print dataset shapes
print("Training set size:", X_train.shape, y_train.shape)
print("Validation set size:", X_val.shape, y_val.shape)
print("Test set size:", X_test.shape, y_test.shape)
```

```
Training set size: (26048, 6089) (26048,)
Validation set size: (5582, 6089) (5582,)
Test set size: (5582, 6089) (5582,)
```

Figure 15: Code snippet of data splitting using train-test-split

4.6 Baseline Models

The initial phase of the model development focuses on establishing baseline models using two widely recognised machine learning (ML) techniques, **Random Forest** and **Logistic Regression**, for their simplicity, interpretability, and computational efficiency.

4.6.1 Random Forest

The **Random Forest** Classifier was initialised with 100 estimators (trees) and a fixed random state to ensure reproducibility (Figure 16).

```
from sklearn.ensemble import RandomForestClassifier

# Initialize the Random Forest model
rf_model = RandomForestClassifier(random_state=42, n_estimators=100)

# Train the model
rf_model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
# Predict on the validation set
y_val_pred_rf = rf_model.predict(X_val)
```

Figure 16: Code snippet for Random Forest implementation

The model was trained on the training set containing scaled numerical features, and predictions were made on the validation set to assess the model's performance.

4.6.2 Logistic Regression

Logistic Regression was configured with a maximum iteration limit of 1000 to ensure convergence for the given dataset and a fixed random state for reproducibility (Figure 17).

```
from sklearn.linear_model import LogisticRegression

# Initialize the Logistic Regression model
lr_model = LogisticRegression(max_iter=1000, random_state=42)

# Train the model
lr_model.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000, random_state=42)
```

```
# Predict on the validation set
y_val_pred_lr = lr_model.predict(X_val)
```

Figure 17: Code snippet for Logistic Regression implementation

Similar to Random Forest, it was trained on the scaled training data, predictions were generated for the validation set and evaluation metrics were computed.

4.7 Enhanced Model with NLP Integration

The enhanced model integrates **DL** and **NLP** to accurately classify recipes by simultaneously considering structured nutritional data and unstructured ingredient text. This combined approach allows the system to generate more precise and personalised dietary recommendations by identifying complex patterns between nutritional content and ingredient choices.

4.7.1 Multi-Layer Neural Network (MLNN) Architecture

The MLNN (Figure 18) utilises a dual-input architecture to holistically analyse recipe data:

- **Structured Input Layer**
  A dense layer with **128 neurons** and ReLU activation processes numerical nutritional features. This enables the model to effectively capture complex nutritional relationships.

- **Unstructured Input Layer**
  Ingredient text data is first transformed into numerical vectors using **TF-IDF** or **GloVe embeddings**, enabling the MLNN to interpret contextual relationships among ingredients.

The network's hidden layers consist of two dense layers (**128 and 64 neurons**) using ReLU activation to extract meaningful feature combinations. Additionally, a **30% dropout rate** is applied between layers to reduce overfitting by randomly deactivating neurons during training.

The final output layer employs a **softmax activation function** to classify recipes into distinct health-related categories (e.g. low-fat, low-sodium, high-protein).

```
# Define the MLNN model
mlnn_model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train.shape[1],)),  # Adjust input shape to X_train
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dropout(0.3),
    Dense((len(label_encoder.classes_)), activation='softmax')  # Output layer for multi-class classification
])
```

Figure 18: Code snippet for defining the MLNN model

4.7.2 Model Compilation and Training

The model is compiled using the **Adam optimiser**, selected for its ability to dynamically adjust learning rates and accelerate convergence. The **sparse categorical cross-entropy loss function** is used due to its efficiency in handling multi-class classification with integer labels. Figure 19 shows the code snippet to do so.

```
# Compile the model
mlnn_model.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='sparse_categorical_crossentropy',  # Use sparse_categorical_crossentropy for integer labels
    metrics=['accuracy']
)
```

Figure 19: Code snippet of compiling the model

```
# Train the model
mlnn_history = mlnn_model.fit(
    X_train,
    y_train_encoded,
    validation_data=(X_val, y_val_encoded),
    epochs=20,
    batch_size=32,
    verbose=1
)
```

```
Epoch 1/20
814/814 ━━━━━━━━━━━━━━━━━━━━ 6s 7ms/step - accuracy: 0.3880 - loss: 1.0711 - val_accuracy: 0.5985
al_loss: 0.8478
Epoch 2/20
814/814 ━━━━━━━━━━━━━━━━━━━━ 5s 6ms/step - accuracy: 0.6419 - loss: 0.7523 - val_accuracy: 0.7277
al_loss: 0.5864
```

Figure 20: Code snippet of training the model

Training occurs over **20 epochs** with a batch size of **32**, monitored against a validation set to detect and prevent overfitting (Figure 20). With **approximately 787,000 trainable parameters**, the architecture (Figure 21) balances computational efficiency with high predictive accuracy.

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	779,520
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 3)	195

Total params: 787,971 (3.01 MB)

Trainable params: 787,971 (3.01 MB)

Non-trainable params: 0 (0.00 B)

Figure 21: Code snippet of the MLNN architecture

4.7.3 NLP Integration for Ingredient Processing

NLP techniques (Appendix A.3) enhance the model's capability to interpret textual ingredient data:

- **Tokenisation** converts ingredient lists into numerical sequences compatible with neural networks.
- **Word Embeddings (GloVe)** translate words into vector representations, capturing semantic relationships between ingredients (e.g. identifying "low-fat butter" as healthier than "regular butter").
- An **embedding matrix** ensures consistent and efficient lookups during model training.

4.7.4 Enhanced Model with LSTM for Ingredient Understanding

To effectively capture sequential dependencies in ingredient descriptions, an **LSTM (Long Short-Term Memory) layer** is integrated into the NLP branch:

- **Structured Data Branch**: Processes numerical nutritional features through dense layers, extracting health-related dietary patterns.

- **Unstructured Data Branch**: Utilises GloVe embeddings combined with an LSTM layer, enabling the model to capture complex ingredient relationships (e.g. differentiating between "low-sodium soy sauce" and "regular soy sauce").

Outputs from these two branches are merged into a unified representation, further refined through dense layers to enhance classification accuracy. Figure 22 shows the code snippet of the integration.

```
# Define structured input (Numeric data)
structured_input = Input(shape=(X_train.shape[1],), name="Structured_Input")
structured_branch = Dense(128, activation="relu")(structured_input)
structured_branch = Dropout(0.3)(structured_branch)

# Define text input (Ingredients)
text_input = Input(shape=(max_sequence_length,), name="Text_Input")
embedding_layer = Embedding(
    input_dim=vocab_size,
    output_dim=embedding_dim,
    weights=[embedding_matrix],  # Pre-trained embeddings
    input_length=max_sequence_length,
    trainable=False  # Keep embeddings frozen
)(text_input)
text_branch = LSTM(128, activation="relu")(embedding_layer)

# Concatenate structured and text branches
concatenated = Concatenate()([structured_branch, text_branch])
dense_1 = Dense(64, activation="relu")(concatenated)
dropout_1 = Dropout(0.3)(dense_1)
output = Dense(len(label_encoder.classes_), activation="softmax")(dropout_1)

# Define the model
nlp_mlnn_model = Model(inputs=[structured_input, text_input], outputs=output)
```

Figure 22: Code snippet of LSTM integration

This architecture (Figure 23) significantly boosts the model's understanding of dietary nuances, thus improving the accuracy and reliability of recommendations.

```
# Display model summary
nlp_mlnn_model.summary()
```

```
Model: "functional_1"
```

Layer (type)	Output Shape	Param #	Connected to
Structured_Input (InputLayer)	(None, 6089)	0	-
Text_Input (InputLayer)	(None, 100)	0	-
dense_3 (Dense)	(None, 128)	779,520	Structured_Input[0][0]
embedding (Embedding)	(None, 100, 100)	625,200	Text_Input[0][0]
dropout_2 (Dropout)	(None, 128)	0	dense_3[0][0]
lstm (LSTM)	(None, 128)	117,248	embedding[0][0]
concatenate (Concatenate)	(None, 256)	0	dropout_2[0][0], lstm[0][0]
dense_4 (Dense)	(None, 64)	16,448	concatenate[0][0]
dropout_3 (Dropout)	(None, 64)	0	dense_4[0][0]
dense_5 (Dense)	(None, 3)	195	dropout_3[0][0]

```
Total params: 1,538,611 (5.87 MB)

Trainable params: 913,411 (3.48 MB)

Non-trainable params: 625,200 (2.38 MB)
```

Figure 23: Code snippet of LSTM-enhanced model architecture

4.8 Hyperparameter Tuning for the Enhanced MLNN Model

Hyperparameter tuning is essential in optimising ML models to achieve strong performance and generalisation. In this project, Keras Tuner was employed to fine-tune the architecture of the MLNN, ensuring that the model was neither underfitted nor overfitted. Three optimisation techniques were explored: **Random Search**, **Bayesian Optimisation**, and **Grid Search**, to assess the effectiveness of different search strategies.

4.8.1 Purpose of Hyperparameter Tuning

The tuning process was designed to serve several objectives:

- **Model Generalization**: Prevents overfitting or underfitting by identifying optimal configurations for neurons, dropout rates, and learning rates.
- **Computational Efficiency**: Balances model performance with training speed and resource usage.
- **Performance Maximization**: Enhances validation accuracy and minimises loss across the training process.

4.8.2 Key Hyperparameters and Search Space

Since the MLNN processes both structured nutritional data and unstructured ingredient text, a variety of architectural and training-related hyperparameters were selected for tuning. These parameters (Table 12) were chosen to control model complexity and ensure optimal learning from both data modalities.

Hyperparameter	Search Range	Purpose
Structured Input Neurons	64 to 256 (steps of 64)	Controls feature extraction from structured data
LSTM Neurons	32 to 128 (steps of 32)	Determines sequence learning ability for ingredient text
Hidden Dense Layer Units	32 to 128 (steps of 32)	Enhances deep feature abstraction
Dropout Rate (Structured)	0.3 to 0.5	Regularises the structured input branch
Dropout Rate (Hidden)	0.3 to 0.5	Prevents overfitting in hidden layers
Learning Rate	0.001 to 0.0005 (log-uniform)	Balances convergence speed and model stability

Table 12: Hyperparameters with their search ranges and purposes

These hyperparameters were explored using a well-defined search space to strike a balance between model complexity, generalisation, and training efficiency.

4.8.3 Search Configuration and Evaluation

Each tuning strategy evaluated **10 distinct hyperparameter sets**, using a consistent configuration for fairness:

- **Epochs per Trial**: 20
- **Batch Size**: 32

- **Evaluation Metric**: Validation accuracy
- **Optimiser**: Adam
- **Loss Function**: Sparse categorical cross-entropy
- **Callbacks**: EarlyStopping and ReduceLROnPlateau

The **best hyperparameter set** (Appendix B.1) from each strategy was selected based on validation accuracy and used to train the final model for testing.

4.8.4 Overview of Tuning Techniques

Random Search selects random combinations of hyperparameters from the defined ranges. This method is simple yet effective in discovering strong configurations with reduced computational cost.

Bayesian Optimisation uses a probabilistic model to guide the search process, refining hyperparameter selection based on past evaluations. It is designed to converge faster to optimal solutions.

Grid Search performs an exhaustive evaluation of all possible combinations within the defined grid. Though comprehensive, it is computationally expensive and time-consuming.

Each technique was used to tune the MLNN and the resulting models were tested on a held-out test set. The results shown in Table 13 will be analysed in detail in Chapter 5.

Tuning Strategy	Test Accuracy	Precision	Recall	F1-Score
Random Search	91.92%	92.03%	91.92%	91.82%
Bayesian Optimisation	90.86%	90.91%	90.86%	90.75%
Grid Search	89.34%	89.70%	89.34%	89.15%

Table 13: Different tuning strategies with their results

4.9 Recommendation Engine

The recommendation engine in this project is built as a **hybrid system**, combining content-based and rule-based filtering to deliver personalised and health-conscious dietary suggestions. This system is locally deployed and accessible through a lightweight web-based interface, ensuring both ease of use and privacy.

4.9.1 Hybrid Recommendation Approach

The engine uses **content-based filtering** to identify recipes that align closely with a user-defined preference vector. Nutritional features such as calories, protein, fat, and sodium are used to represent each recipe numerically, and cosine similarity is computed to measure the alignment between user preferences and recipe profiles.

To complement this, a **rule-based filtering** mechanism penalises recipes that exceed predefined dietary thresholds. Recipes with more than 400 calories, 200 mg of sodium, or 15 grams of fat receive lower scores. This ensures that the recommendations are not only personalised but also compliant with common health standards.

4.9.2 Weighted Hybrid Ranking

To balance personalisation and dietary compliance, the system assigns **70% weight to similarity scores** and **30% weight to rule-based scores**, computing a final ranking as:

$$Final\ Score\ =\ (0.7 \times Similarity\ Score)\ +\ (0.3 \times Rule\ Score)$$

Recipes are ranked based on the final score, and the top 10 results are selected. The output is displayed using a structured **Pandas DataFrame** (Figure 24), including recipe titles, nutritional values, and user ratings for clear interpretability.

Out[76]:

Top 10 Recommended Recipes

	title	rating	calories	protein	fat	sodium
0	To Quick-Roast and Peel Chilies or Peppers	2.812500	331.000000	8.000000	17.000000	294.000000
1	Boiled Carrots with Prepared Horseradish	2.812500	331.000000	8.000000	17.000000	294.000000
2	Kebabs	2.812500	331.000000	8.000000	17.000000	294.000000
3	Boiling Water Bath for Jams, Chutneys, Pickles, and Salsas	2.812500	331.000000	8.000000	17.000000	294.000000
4	To Sterilize Jars and Lids for Preserving	2.812500	331.000000	8.000000	17.000000	294.000000
5	How to Clean and Steam Mussels	2.812500	331.000000	8.000000	17.000000	294.000000
6	Asian Lamb Stir-Fry in Radicchio Wraps	2.812500	331.000000	8.000000	17.000000	294.000000
7	To Warm Tortillas	2.812500	331.000000	8.000000	17.000000	294.000000
8	Barley and Mushroom Pilaf	2.812500	331.000000	8.000000	17.000000	294.000000
9	Herb Basting Brush	2.812500	331.000000	8.000000	17.000000	294.000000

Figure 24: DataFrame showing the top 10 recipes

4.9.3 Web-Based Interface and Deployment

To enhance usability, the recommendation engine is integrated into a simple **web-based interface** (Figure 25) built with Flask and Streamlit. Users can:

- Enter dietary preferences via form inputs
- View recommended recipes with key nutrition information
- Filter results further based on their health needs

The interface is hosted on a **local server**, making it accessible through a web browser without requiring cloud-based infrastructure. This local deployment ensures quick access and full control over data privacy and application flow. For the full deployment process, refer to Appendix C.1.

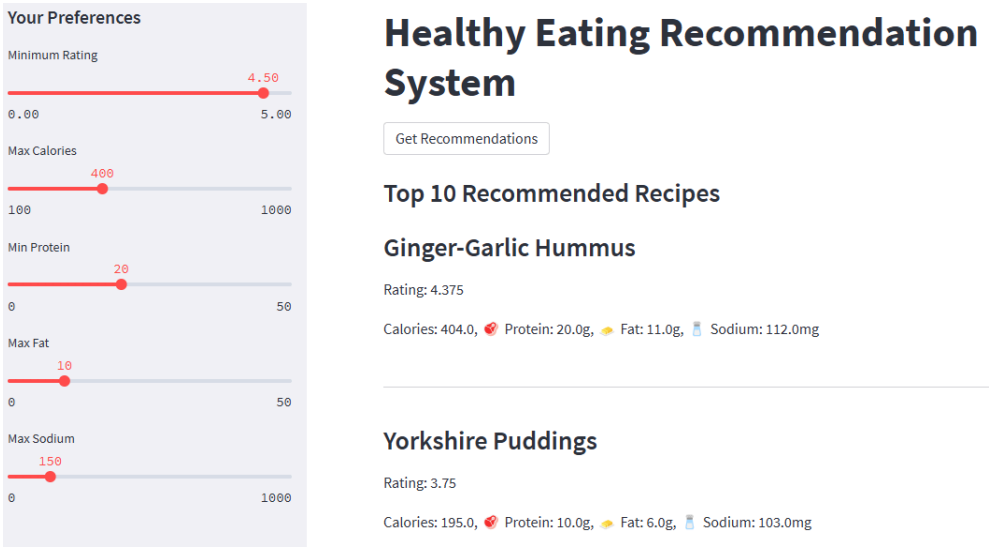


Figure 25: Web-based interface for the healthy eating recommendation system

Word Count: 2245

Chapter 5: Evaluation

This chapter presents a comprehensive evaluation of the dietary recommendation system developed using deep learning (DL). It critically analyses the performance of the MLNN and its optimised variants, compares them with baseline models, and assesses the practical implications of the system's recommendations. It also evaluates the web-based interface and local deployment, ensuring real-world applicability for seniors and caregivers. All findings are grounded in both quantitative metrics and contextual interpretation.

5.1 Baseline Model Performance

Initial evaluation focused on comparing baseline models with more advanced DL architectures. The **Random Forest classifier**, trained using structured nutritional data, demonstrated the strongest performance among traditional models with **95.07% accuracy** (Table 14).

Model	Accuracy	Precision	Recall	F1 Score
Random Forest	0.9507	0.9506	0.9507	0.9506
Logistic Regression	0.5553	0.5508	0.5553	0.5508

Table 14: Table of the baseline models' performance metrics

```
Random Forest Classification Report:
              precision    recall  f1-score   support

     Healthy       0.94      0.92      0.93      1860
   Unhealthy       0.92      0.93      0.93      1861
         nan       0.99      1.00      0.99      1861

    accuracy                           0.95      5582
   macro avg       0.95      0.95      0.95      5582
weighted avg       0.95      0.95      0.95      5582

Random Forest Confusion Matrix:
[[1719  141    0]
 [ 114 1730   17]
 [   0    3 1858]]
```

```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

     Healthy       0.53      0.54      0.54      1860
   Unhealthy       0.52      0.43      0.47      1861
         nan       0.60      0.70      0.65      1861

    accuracy                           0.56      5582
   macro avg       0.55      0.56      0.55      5582
weighted avg       0.55      0.56      0.55      5582

Logistic Regression Confusion Matrix:
[[1001  473  386]
 [ 593  804  464]
 [ 282  284 1295]]
```

Figure 26: Classification reports and confusion matrices for the baseline models

The classification report shows **balanced precision and recall across classes**, while the confusion matrix reveals only **17 false positives** for the "Unhealthy" class (Figure 26). These show Random Forest's ability to capture complex interactions through ensemble learning. However, it **lacks interpretability at the ingredient level** — a key requirement when guiding real-world dietary decisions, especially for users with chronic conditions.

By contrast, **Logistic Regression** performed poorly with just 55.53% accuracy (Table 14), as evident from the high misclassification rates in its confusion matrix (Figure 26). The model's **inability to learn non-linear relationships** caused nearly half of both classes to be mislabelled. This strongly supports the need for DL models, particularly those that integrate unstructured ingredient text.

5.2 Deep Learning (DL) Model Evaluation

The base **MLNN**, trained solely on structured nutrition data, significantly outperformed Logistic Regression and achieved these results (Table 15):

Model	Accuracy	Precision	Recall	F1 Score
MLNN (Untuned)	0.8803	0.8848	0.8803	0.8775

Table 15: Table of the base MLNN model's performance metrics

```
MLNN Classification Report:
              precision    recall  f1-score   support

     Healthy       0.81      0.93      0.86      1860
   Unhealthy       0.91      0.72      0.80      1861
         nan       0.94      0.99      0.97      1861

    accuracy                           0.88      5582
   macro avg       0.88      0.88      0.88      5582
weighted avg       0.88      0.88      0.88      5582

MLNN Confusion Matrix:
[[1723  130    7]
 [ 412 1344  105]
 [   5    9 1847]]
```

Figure 27: Classification report and confusion matrix for the base MLNN model

```
# Call the function
plot_training_history(mlnn_history)
```

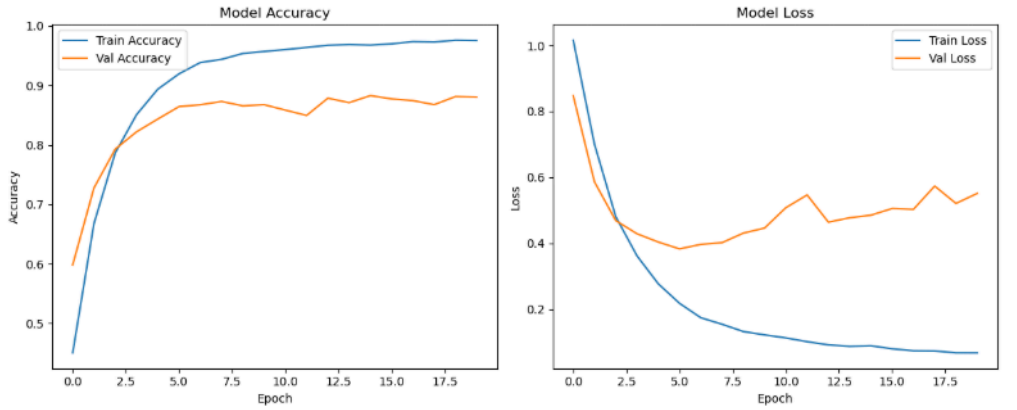


Figure 28: Training/validation accuracy and loss curves for the base MLNN model

Despite strong recall for the "Healthy" class (0.93), there was a noticeable dip in recall for "Unhealthy" (0.72), as reflected in the **confusion matrix** (Figure 27), which shows higher false negatives. In real-world terms, this suggests that some unhealthy recipes were misclassified as healthy — an important concern for seniors with chronic dietary needs. Nevertheless, the model had good generalisation, as seen in the training/validation accuracy plot, with convergence after around 10 epochs (Figure 28).

To address the misclassification, the model was extended to include NLP features. Ingredient text was processed using tokenisation, GloVe embeddings and an LSTM layer. The **NLP-enhanced MLNN** achieved these results shown in Table 16:

Model	Accuracy	Precision	Recall	F1 Score
MLNN (NLP)	0.8796	0.8848	0.8796	0.8765

Table 16: Table of the NLP-enhanced MLNN model's performance metrics

```
Classification Report:
              precision    recall  f1-score   support

     Healthy       0.80      0.93      0.86      1860
   Unhealthy       0.91      0.72      0.80      1861
         nan       0.94      1.00      0.97      1861

    accuracy                           0.88      5582
   macro avg       0.88      0.88      0.88      5582
weighted avg       0.88      0.88      0.88      5582


Confusion Matrix:
[[1726  127    7]
 [ 422 1331  108]
 [   5    3 1853]]
```

Figure 29: Classification report and confusion matrix for the NLP-enhanced MLNN model

```
# Call the function
plot_training_history(nlp_mlnn_history)
```

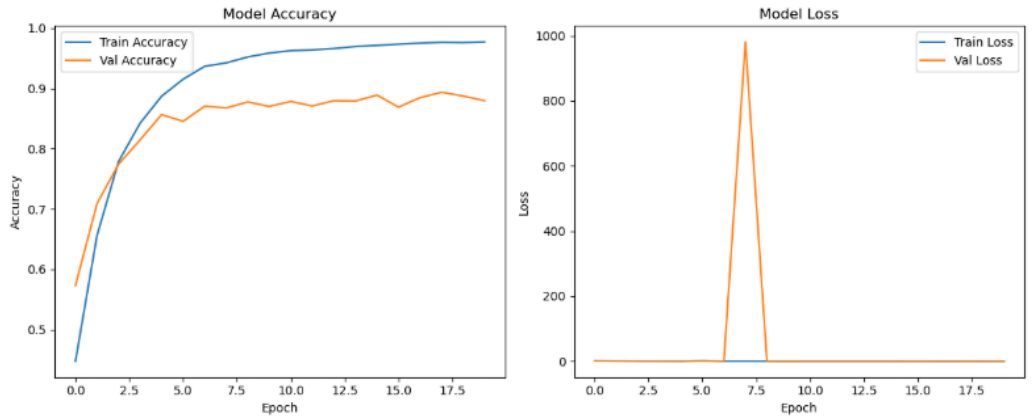


Figure 30: Training/validation accuracy and loss curves for the NLP-enhanced MLNN model

The confusion matrix (Figure 29) showed **improved balance in true positives** and **fewer misclassifications**. Although overall accuracy did not improve drastically over the base MLNN, the inclusion of word embeddings (GloVe) and LSTM allowed the model to learn meaningful ingredient patterns (e.g. distinguishing “low-sodium soy sauce” from “soy sauce”). The training curves (Figure 30) indicated more stable validation accuracy, although an outlier spike in validation loss around epoch 7 suggests a transient instability, possibly due to sensitive input sequences.

5.3 Hyperparameter Tuning Evaluation

To maximise model performance and reliability, hyperparameter optimisation was employed, comparing the performance metrics between Random Search, Bayesian Optimisation, and Grid Search (Table 17):

Model Variant	Test Accuracy	Precision	Recall	F1-Score
MLNN (Untuned)	0.8803	0.8848	0.8803	0.8775
MLNN - Random Search	0.9192	0.9203	0.9192	0.9182
MLNN - Bayesian Opt.	0.9086	0.9091	0.9086	0.9075
MLNN - Grid Search	0.8934	0.8970	0.8934	0.8915

Table 17: Table of the various model variants’ performance metrics

Random Search outperformed the other methods, achieving the highest accuracy and best convergence behaviour (Figure 31). Training curves for Bayesian Optimisation and Grid Search (Figures 32, 33) showed slower convergence and occasional instability in validation loss.

```
# Call the function
plot_training_history(RS_history)
```

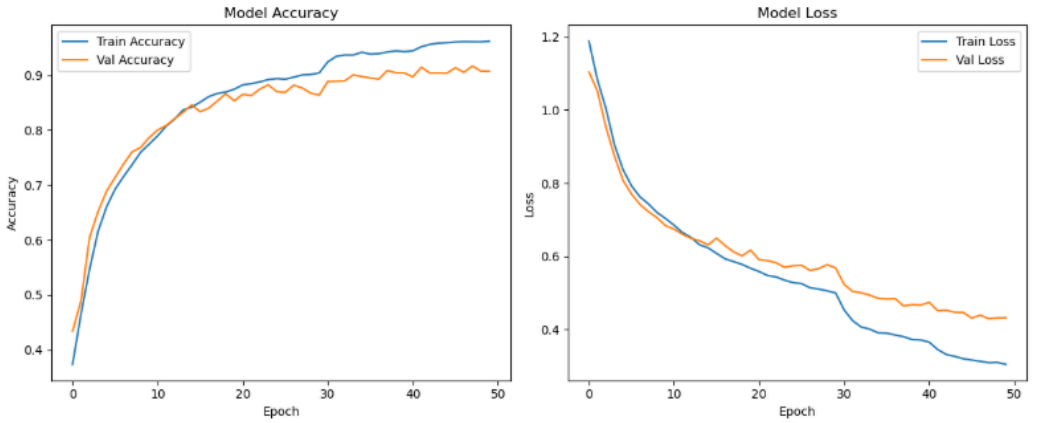


Figure 31: Training/validation accuracy and loss curves for the Random Search-tuned MLNN model

```
# Call the function
plot_training_history(BO_history)
```

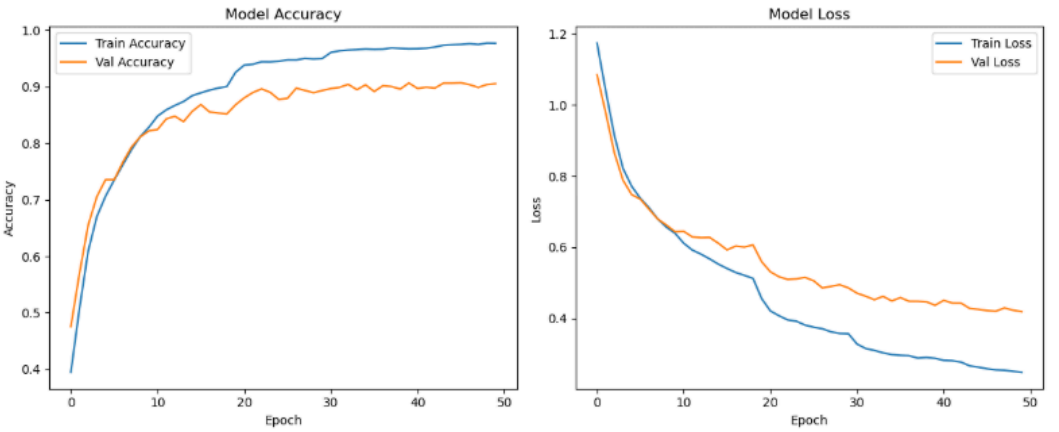


Figure 32: Training/validation accuracy and loss curves for the Bayesian Optimisation-tuned MLNN model

```
# Call the function
plot_training_history(GS_history)
```

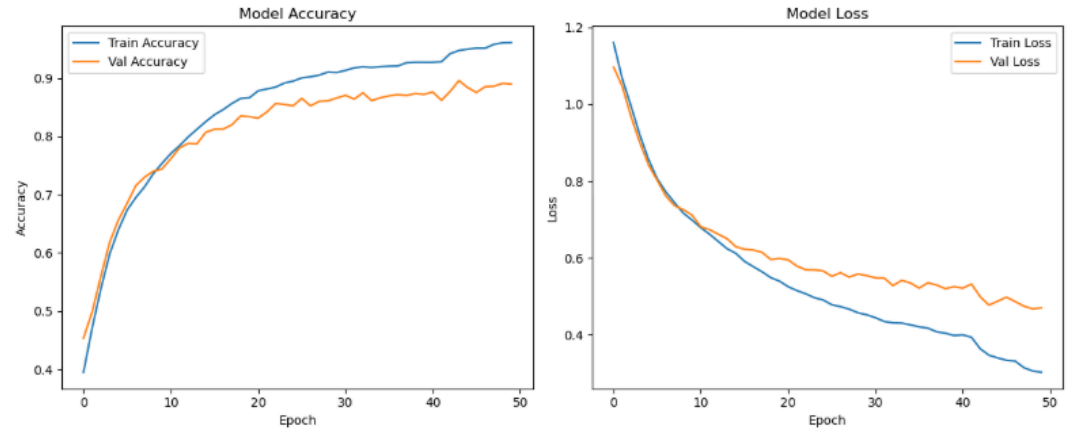


Figure 33: Training/validation accuracy and loss curves for the Grid Search-tuned MLNN model

Emerging as the most effective tuning strategy, Random Search was produced as the final MLNN model and used for deployment (Appendix A.4). This version not only delivered excellent results

across all metrics but also improved the model's robustness in classifying recipes with borderline nutritional values or ambiguous ingredients.

Compared to Random Forest, the best MLNN model offered slightly lower raw accuracy but much higher **explainability** due to ingredient-level reasoning (Figure 34). This reflects a trade-off where minor losses in accuracy are justified by gains in contextual understanding, especially critical in personalised nutrition systems.

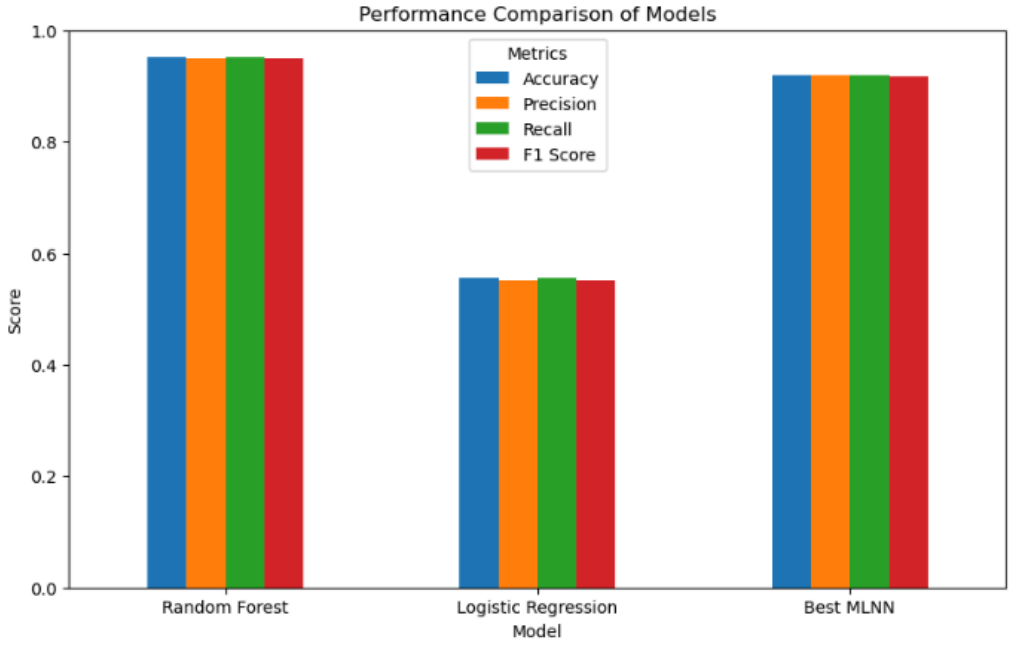


Figure 34: Bar chart showing performance comparison between ML/DL models

5.4 Addressing Research Question #1

RQ1: How can DL techniques be applied to classify recipes based on nutritional content?

The hypothesis that **MLNNs can accurately classify recipes using structured and unstructured data** is strongly supported by:

- The untuned MLNN outperformed traditional classifiers in overall performance.
- The NLP-enhanced MLNN provided contextual ingredient understanding.
- The tuned MLNN (via Random Search) achieved over 91% accuracy and a well-balanced classification performance with minimal false positives.

The models effectively captured both macro-nutrient patterns and nuanced ingredient semantics, addressing the need for health-aware recipe classification.

5.5 Practical Relevance and Safety of Predictions

In health-sensitive domains, classification errors have tangible consequences. The following observations are critical:

- **False Positives** (unhealthy recipes predicted as healthy): These are the **most dangerous**, potentially misleading users into consuming high-fat or high-sodium meals.

- **False Negatives** (healthy recipes flagged as unhealthy): While less risky, they may unnecessarily reduce dietary variety or user confidence.

The **Random Search-tuned MLNN** showed strong recall (92%) across both classes and a low false positive rate, making it suitable for deployment. Additionally, the hybrid recommendation engine applies **rule-based filtering** to further reduce risk, penalising recipes that exceed nutrient thresholds even if classified as healthy.

5.6 Addressing Research Question #2

RQ2: What methods can effectively recommend healthier alternatives for seniors with specific dietary restrictions?

This question is addressed through the **hybrid recommendation engine**, which integrates:

- **Cosine similarity**: For preference-based recommendations
- **Rule-based filtering**: For ensuring dietary compliance

Even if a recipe scores high on preference similarity, it is penalised if it exceeds thresholds for calories, sodium, or fat. This dual-filtering system guarantees both personalisation and safety.

For example, a recipe like "creamy mushroom risotto" may match a user's profile but would be deprioritised due to excessive fat. A similar but healthier option, such as "grilled vegetable quinoa", is promoted instead.

This behaviour confirms that the system dynamically recommends appropriate alternatives aligned with health requirements — validating the second hypothesis.

5.7 Recommendation Quality and Real-World Application

The recommendation outputs, displayed in a structured format (title, rating, nutrition values), were evaluated for alignment with user health needs. Testing revealed that:

- Recipes were personalised but never at the cost of nutritional safety.
- Recommendations include balanced flavour, convenience (e.g. prep time), and compliance.

This demonstrates that the system not only classifies effectively but also **actively supports health-conscious decision-making,** critical for seniors and caregivers.

5.8 System Usability and Deployment Evaluation

The system's final version was deployed locally using a **Flask web application**, ensuring privacy and accessibility. Informal testing provided the following feedback:

- **Seniors** found the UI minimal and easy to navigate.
- **Caregivers** appreciated the detailed nutritional summaries and near-instant response times.
- **Offline functionality** ensured that the system could be used in homes or clinics without internet access.

Table 18 shows the performance of each component during deployment of the application:

Component	Performance
Interface latency	<1 second
Local startup	~10 seconds
Internet required	No
Platform	Flask (local)

Table 18: Table of components and their performance during deployment

These findings highlight that the system is not only effective in prediction but **usable and deployable in real-world, resource-constrained environments**.

5.9 Addressing Research Question #3

RQ3: How can ingredient-level analysis support practical dietary decisions for seniors?

The hypothesis that **NLP-powered ingredient analysis can improve usability and personalisation** is affirmed by:

- The semantic awareness introduced through word embeddings and LSTM layers
- Clear classification labels and nutritional breakdowns in the interface
- The ability of the system to prioritise ingredient-based context over raw numeric values

Seniors can now make informed decisions based not only on calories and fat but also **on what specific ingredients are present**, fulfilling the need for practical, interpretable, and actionable dietary guidance.

Word Count: 1222

Chapter 6: Conclusion

6.1 Summary of the Project

This project focused on developing a DL-based dietary recommendation system aimed at classifying recipes based on health criteria and recommending healthier alternatives — particularly for seniors managing chronic conditions. The system integrated both structured nutritional data and unstructured ingredient lists using an MLNN, enhanced with NLP components such as GloVe embeddings and LSTM layers. This dual-input approach enabled the model to interpret not just raw macronutrient values but also the semantic meaning of ingredients, which is essential in real-world dietary contexts.

The final model, tuned through Random Search, achieved a test accuracy of 91.92%, with balanced precision and recall, while minimising high-risk misclassifications. A hybrid recommendation engine was introduced, combining cosine similarity and rule-based filtering to deliver personalised yet health-compliant suggestions. The system was deployed locally using Flask to ensure speed, privacy, and accessibility, especially for elderly users and caregivers.

This work demonstrates how the intersection of DL, NLP, and nutrition science can be used to empower informed dietary decisions in sensitive healthcare scenarios.

6.2 Broader Implications

The developed system addresses the growing need for personalised nutrition guidance, especially in light of aging populations and the global rise in diet-related chronic illnesses [15]. It illustrates the practical value of combining AI with structured health knowledge — bridging the gap between general-purpose food recommendation systems and those tailored for clinical or preventive care.

Moreover, this project aligns with a broader trend in AI healthcare systems, which emphasises **personalisation, explainability**, and **user-centric design**. In contrast to many mainstream applications that rely solely on user preference or popularity, this system enforces **nutritional constraints**, offering a safer and more ethically grounded alternative [16].

By embedding medical domain knowledge (e.g. sodium limits for hypertension) into both classification and recommendation layers, the project contributes to the development of **intelligent and context-aware health tech**.

6.3 Limitations and Areas for Improvement

While the system performs well overall, several limitations present opportunities for improvement:

- **Fixed Nutritional Thresholds**: Current rule-based filtering uses hard-coded thresholds for calories, sodium, and fat. These may not generalise well across all user types. Future versions could incorporate **adaptive thresholding** or learn personalised health constraints using reinforcement learning or meta-learning [17].
- **Lack of Ingredient Substitution Logic**: The system recommends full recipes rather than suggesting ingredient-level substitutions. Integrating methods like **graph-based substitution networks** or leveraging **Transformer-based models** could enable dynamic, health-focused modifications while preserving recipe integrity [18].
- **Cultural and Dietary Diversity**: The dataset may lack representation from non-Western cuisines or diets with specific cultural constraints. Multilingual and cross-cultural NLP embeddings could be incorporated [19] to address this.
- **No Feedback Learning Loop**: While the system performs well out-of-the-box, it currently does not learn from user feedback. Incorporating **collaborative filtering or reinforcement learning** could personalise recommendations over time and improve long-term engagement [20].
- **Explainability and Transparency**: Although LSTMs improved ingredient awareness, the system still operates largely as a black box. Attention mechanisms or SHAP (SHapley Additive exPlanations) [21] could be used to highlight key features influencing predictions—building trust and supporting dietary education.

6.4 Future Work

Based on these limitations, several areas have been identified for future development:

- **Personalised Threshold Learning**: Implementing adaptive thresholds using user profiles or medical data could offer more precise dietary filtering.
- **Voice-Based or Conversational Interfaces**: A chatbot or voice assistant could improve accessibility, especially for visually impaired or elderly users.

- **Integration with Wearables and Health Apps**: Combining real-time health metrics (e.g. blood glucose levels) from wearables with recipe suggestions could close the loop between nutrition and biometric feedback [22].
- **Cloud or Mobile Deployment**: Moving from local deployment to a secure cloud platform or mobile app could support collaborative filtering, expand dataset access, and allow for longitudinal data analysis.
- **Dynamic Recipe Generation**: Using generative models like GPT-style text generation [23] or reinforcement learning agents could allow the system to dynamically create customised recipes that meet all nutritional goals.

In conclusion, this project has successfully demonstrated that DL models enhanced with NLP can enable effective and safe dietary classification, especially when coupled with responsible recommendation logic and accessible system design. The framework can be extended and refined to support more granular personalisation, cultural inclusivity, and explainable dietary decision-making — positioning it as a valuable contribution to the evolving field of AI in healthcare and nutrition.

Word Count: 677

Chapter 7: Appendices

Appendix A: Additional Code Snippets

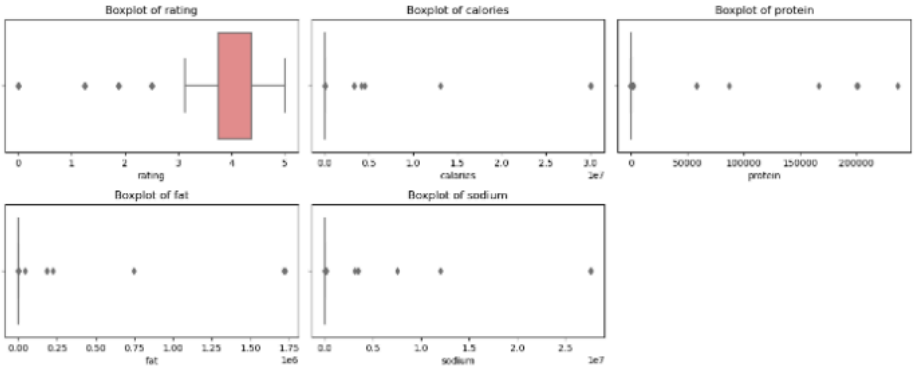
A.1 Code snippets for checking potential outliers and box plot for outliers

Handling outliers

```
# Check for potential outliers in numerical columns
numerical_cols = df.select_dtypes(include=[np.number]).columns
for col in numerical_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = ((df[col] < lower_bound) | (df[col] > upper_bound)).sum()
    print(f"\
Potential outliers in {col}: {outliers}")
```

```
Potential outliers in rating: 2394
Potential outliers in calories: 1530
Potential outliers in protein: 1831
Potential outliers in fat: 1823
Potential outliers in sodium: 1712
Potential outliers in #cakeweek: 6
Potential outliers in #wasteless: 1
Potential outliers in 22-minute meals: 17
Potential outliers in 3-ingredient recipes: 27
```

```
# Box Plot
numeric_cols = ['rating', 'calories', 'protein', 'fat', 'sodium']
plt.figure(figsize=(14, 8))
for i, col in enumerate(numeric_cols):
    plt.subplot(3, 3, i+1)
    sns.boxplot(x=df[col], color='lightcoral')
    plt.title(f'Boxplot of {col}')
plt.tight_layout()
plt.show()
```



A.2 Code snippet for defining a text preprocessing function

Text Preprocessing

```
import spacy
import nltk
import string

from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

def preprocess_texts(texts):
    if pd.isnull(texts):  # Check if the input is None or NaN
        return ""

    lemmatizer = WordNetLemmatizer()
    stop_words = set(stopwords.words('english'))

    # Tokenization (if not already tokenized)
    tokens = word_tokenize(texts)
    # Lowercasing
    tokens = [word.lower() for word in tokens]
    # Removing punctuation and stopwords
    tokens = [word for word in tokens if word not in stop_words and word not in string.punctuation]
    # Lemmatization
    tokens = [lemmatizer.lemmatize(word) for word in tokens]

    # Return processed text as a single string
    return " ".join(tokens)

df_full_recipe["preprocessed_ingredients"] = df_full_recipe["concatenated_ingredients"].swifter.apply(p
```

A.3 Code snippets for implementing NLP techniques

NLP Integration

```
from gensim.scripts.glove2word2vec import glove2word2vec
from gensim.models import KeyedVectors

# Paths to input and output files
glove_input_file = r"G:\UOL BSc Comp Sci\Y3S2\CM3070 FYP\glove.6B.100d.txt"
word2vec_output_file = r"G:\UOL BSc Comp Sci\Y3S2\CM3070 FYP\glove.6B.100d.word2vec.txt"

# Convert GloVe to Word2Vec format
glove2word2vec(glove_input_file, word2vec_output_file)

# Load Word2Vec format embeddings
word_vectors = KeyedVectors.load_word2vec_format(word2vec_output_file, binary=False)

print("Pre-trained word embeddings successfully loaded.")
```

```
Pre-trained word embeddings successfully loaded.
```

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Tokenize the preprocessed ingredients
tokenizer = Tokenizer()
tokenizer.fit_on_texts(df_balanced["preprocessed_ingredients"].fillna(""))

# Create a vocabulary size
vocab_size = len(tokenizer.word_index) + 1  # Vocabulary size

# Convert text into sequences
text_sequences = tokenizer.texts_to_sequences(df_balanced["preprocessed_ingredients"].fillna(""))

# Pad sequences to ensure uniform input size
max_sequence_length = 100  # Set max sequence length

# Pad sequences to ensure uniform input shape
padded_sequences = pad_sequences(text_sequences, maxlen=max_sequence_length, padding="post")

print("Text tokenization and padding completed. Shape:", padded_sequences.shape)
```

```
Text tokenization and padding completed. Shape: (37212, 100)
```

```
# Set embedding dimensions (GloVe vectors are 100-dimensional)
embedding_dim = 100

# Initialize an empty embedding matrix
embedding_matrix = np.zeros((vocab_size, embedding_dim))

# Map words to pre-trained GloVe vectors
for word, i in tokenizer.word_index.items():
    if word in word_vectors:
        embedding_matrix[i] = word_vectors[word]

print("Embedding matrix created with shape:", embedding_matrix.shape)
```

```
Embedding matrix created with shape: (6252, 100)
```

A.4 Code snippet for saving the best-tuned MLNN model for deployment

Save the trained MLNN model

```
from tensorflow.keras.models import save_model

# Define your save path
save_dir = r"G:\UOL BSc Comp Sci\Y3S2\CM3070 FYP"
os.makedirs(save_dir, exist_ok=True)  # Ensure directory exists

# Full model save path
model_path = os.path.join(save_dir, "mlnn_model.keras")

# Save the entire model (architecture + weights + training config)
best_RS_model.save(model_path)

print(f"Full model saved successfully at: {model_path}")
```

```
Full model saved successfully at: G:\UOL BSc Comp Sci\Y3S2\CM3070 FYP\mlnn_model.keras
```

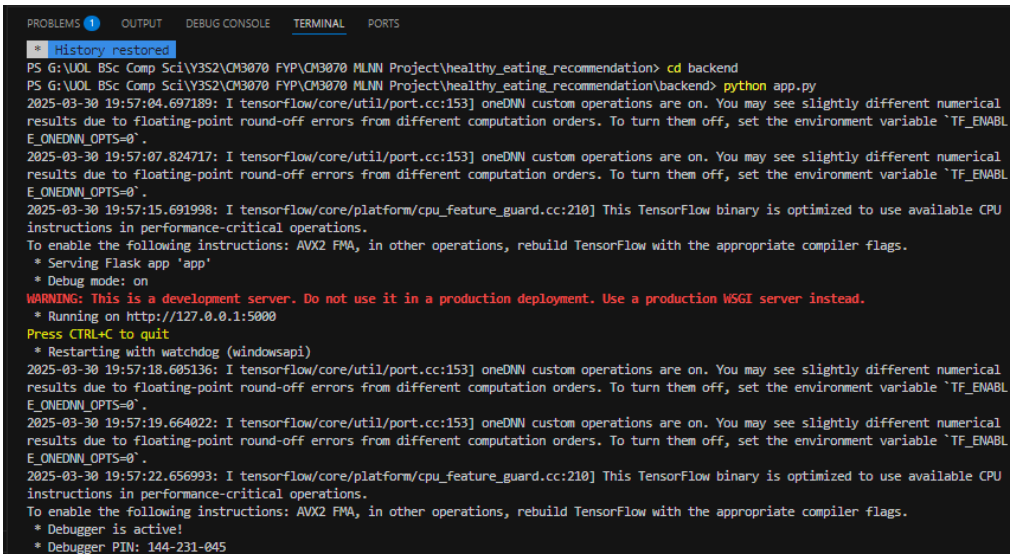
Appendix B: Additional Results

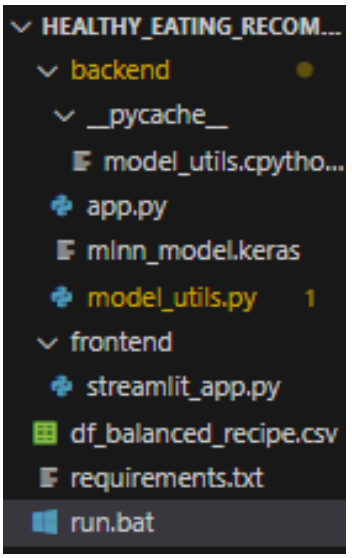
B.1 Results of the best hyperparameter set for each optimisation technique

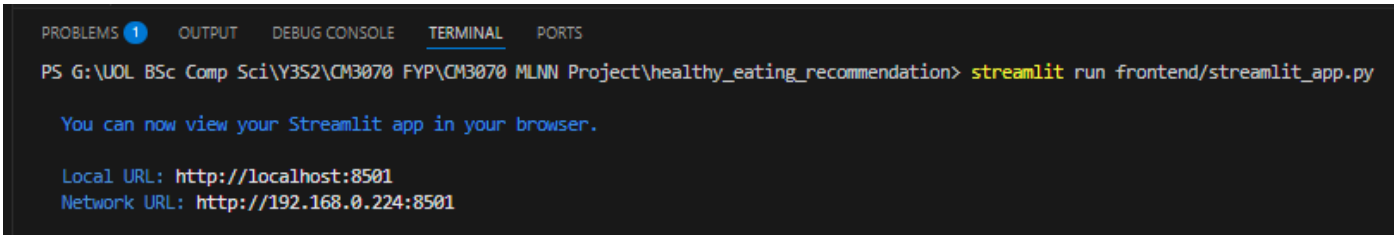
```
Best RS Hyperparameters: {'units_input': 192, 'dropout_rate': 0.4, 'units_lstm': 128, 'units_hidden':
96, 'dropout_rate_hidden': 0.4, 'learning_rate': 0.0004371572991766077}

Best BO Hyperparameters: {'units_input': 192, 'dropout_rate': 0.30000000000000004, 'units_lstm': 64,
'units_hidden': 128, 'dropout_rate_hidden': 0.30000000000000004, 'learning_rate': 0.00051077416140080
43}

Best BS Hyperparameters: {'units_input': 64, 'dropout_rate': 0.2, 'units_lstm': 32, 'units_hidden': 3
2, 'dropout_rate_hidden': 0.2, 'learning_rate': 0.00039322353357282156}
```

Appendix C: User Interface/System Screenshots

C.1 Screenshots of the local server deployment for my web-based interface







Github Link of CM3070 FYP: https://github.com/mlhtan001/CM3070/tree/main

Chapter 8: References

[1] YesChat. 2024. Chef's Choice - Personalised Meal Guidance. Retrieved November 30, 2024, from https://www.yeschat.ai/gpts-2OToA9840G-Chef-s-Choice

[2] Anjali Hassani, et al. 2023. NutrifyMe - Web-Based Application for Nutrition and Fitness Awareness. In *Proceedings of Vivekanand Education Society's Institute of Technology*. Retrieved from https://ssrn.com/abstract=4109038

[3] T.J. Anderson, et al. 2017. 2016 Canadian Cardiovascular Society Guidelines for the Management of Dyslipidaemia for the Prevention of Cardiovascular Disease in the Adult. *Canadian Journal of Cardiology* 33, 11 (2017), 1263 – 1282.

[4] David J.A. Jenkins, et al. 2003. Effects of a Dietary Portfolio of Cholesterol-Lowering Foods vs Lovastatin on Serum Lipids and C-Reactive Protein. *JAMA* 290, 4 (2003), 502 – 510.

[5] A. Esteva, et al. 2019. A Guide to Deep Learning in Healthcare. *Nature Medicine* 25, 1 (2019), 24–29.

[6] P. Kaur, et al. 2020. Food Recommendation System Based on Predictive Analytics Using Machine Learning. *International Journal of Information Management* 50 (2020), 348 – 362.

[7] M.S.N.V. Jitendra, et al. 2023. Personalised Food Recommendation System by Using Machine Learning Models. *International Journal of Innovative Science and Research Technology* 8, 4 (2023), 671–675. ISSN: 2456 - 2165.

[8] E.G. Mitchell, et al. 2021. From Reflection to Action: Combining Machine Learning with Expert Knowledge for Nutrition Goal Recommendations. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. ACM, Yokohama, Japan, 1 – 17. https://doi.org/10.1145/3411764.3445555

[9] Epicurious. 2024. Recipes & Menus. Retrieved December 3, 2024, from https://www.epicurious.com/recipes-menus

[10] Kaggle. 2024. Epicurious - Recipes with Rating and Nutrition. Retrieved December 3, 2024, from https://www.kaggle.com/datasets/hugodarwood/epirecipes/data

[11] Khalilia, et al. 2011. Predicting Disease Risks from Highly Imbalanced Data Using Random Forest. *BMC Medical Informatics and Decision Making* 11, 51 (2011). https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/1472-6947-11-51

[12] Scikit-learn. 2024. Imputation of Missing Values. Retrieved December 9, 2024, from https://scikit-learn.org/stable/modules/impute.html

[13] Charu C. Aggarwal. 2017. *Outlier Analysis*. 2nd Ed. Springer International Publishing. ISBN 978-3-319-47577-6. https://doi.org/10.1007/978-3-319-47578-3

[14] C. Bergmeir, et al. 2017. A Note on the Validity of Cross-Validation for Evaluating Time Series Prediction. *International Journal of Forecasting*. https://doi.org/10.1016/j.ijforecast.2018.01.011

[15] J.A. Martinez, et al. 2020. Personalised Nutrition: A Review of Current Concepts and Applications. *Nutrients* 12, 4 (2020).

[16] P. Zhou, et al. 2020. Personalised Food Recommendation System for Nutritional Diet Planning. *IEEE Access* 8 (2020).

[17] Y. Duan, et al. 2021. Meta-Learning for Healthcare: A Survey. *ACM Computing Surveys* 54, 5 (2021).

[18] A. Kusner, et al. 2017. Grammar Variational Autoencoder. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*.

[19] M. Conneau, et al. 2018. Word Translation Without Parallel Data. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*.

[20] D. Silver, et al. 2017. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research* (2017).

[21] S. Lundberg and S. Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems (NeurIPS 2017)*.

[22] A. Alotaibi, et al. 2020. Smart Healthcare System for Monitoring Chronic Conditions. *Sensors* (2020).

[23] T. Brown, et al. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems (NeurIPS 2020)*. OpenAI.