“Predicting Alzheimer’s Disease through Protein Structure”

A Industrial Internship Report
Submitted in Partial Fulfillment of the
Requirements for Award of the Degree of
BACHELOR OF TECHNOLOGY
IN
INFORMATION AND COMMUNICATION TECHNOLOGY

Submitted by
Ayush Zala
Roll No. 20BIT074

Under the Supervision and Guidence of
Dr. Vivek Kumar Pandit
Assistant Professor, Department of ICT,
Pandit Deendayal Energy Univeristy

Submitted to
Department of Information and Communication Technology
School of Technology
Pandit Deendayal Energy University (PDEU)
Gandhinagar, INDIA, 382007
May 2024

Certificate of Originality of Work

I hereby declare that the B.Tech. Project entitled "Predicting Alzheimer's Disease through Protein Structure" submitted by me for the partial fulfillment of the degree of Bachelor of Technology to the Dept. of Information and Communication Technology at the School of Technology, Pandit Deendayal Energy University, Gandhinagar, is the original record of the project work carried out by me under the supervision of Dr. Vivek Kumar Pandit.

I also declare that this written submission adheres to University guidelines for its originality, and proper citations and references have been included wherever required.

I also declare that I have maintained high academic honesty and integrity and have not falsified any data in my submission.

I also understand that violation of any guidelines in this regard will attract disciplinary action by the institute.

Name of the Student: Ayush Zala

Roll Number of the Student: 20BIT074

Signature of the Student:

Name of the Supervisor: Dr. Vivek Kumar Pandit

Designation of the Supervisor: Assistant Professor, Department of ICT, PDEU

Signature of the Supervisor:

Place:

Date: 20-05-2024

Certificate from the Project Supervisor/Head

This is to certify that the Comprehensive Project Report entitled **"Predicting Alzheimer's Disease through Protein Structure"** submitted by **Ayush Zala**, Roll No. **20BIT074** towards fractional compliance of the requirements for the award of a degree in "Bachelor of Technology" in the field of **"Information and Communication Technology"**Engineering from the **"School of Technology"**, Pandit Deendayal Energy University, Gandhinagar is a record of his efforts done under our supervision and assistance. The student's work has, in our judgment, attained a sufficient level to be accepted for assessment. To the best of our knowledge, the results of this significant project effort have not been submitted to any other University or Institution for the granting of any degree or certificate.

Name and Sign of the Supervisor　　　Name and Sign of the Industry Supervisor

Name and Sign of the HoD　　　Name and Sign of the Director

Place:

Date: 20-05-2024

Acknowledgement

I would like to express my sincere gratitude to my thesis advisor, Dr. Vivek Kumar Pandit, Assistant Professor in the Information and Communication Technology (ICT) department. His invaluable guidance, support and insightful feedback throughout this project were instrumental in its successful completion.

I am also thankful to Dr. Gangaprasad Pandey, Head of the ICT department at Pandit Deendayal Energy University, for providing the necessary facilities and resources to carry out this research.

I am grateful to Dr. Dhaval Pujara, Director of the School of Technology, for his encouragement of research opportunities within the university. This supportive environment has greatly benefited my academic journey. Additionally, I would like to thank Dr. Anirudh Kulkarni for his insightful seminar on thesis creation and defense. The knowledge gained from this seminar proved to be very helpful during the writing of my thesis.

I am immensely grateful to Mr. Ravi Jadhav, my industry supervisor from Horizontal, for providing me with the opportunity to work on this project and for his valuable insights and guidance. His industry knowledge and mentorship have been invaluable assets, enriching my learning experience.

Furthermore, I would like to acknowledge the valuable knowledge and expertise imparted by all the faculty members of the ICT department, which proved to be crucial in this project.

This project would not have been possible without the collective efforts and support of all those mentioned above, and for that, I am truly grateful.

Abstract

Alzheimer's disease (AD) poses a significant challenge to public health and is characterized by the progressive decline of cognitive functions, primarily attributable to the pathological aggregation of amyloid-beta peptides and tau proteins. Predicting Alzheimer's Disease through analysis of protein structures is a crucial step in understanding its pathogenesis and finding potential therapeutic targets. This report evaluates the efficacy of various machine learning models to classify and predict Alzheimer's Disease based on protein structural data. We compared six different models: Logistic Regression,K-Nearest Neighbors (KNN), XGBoost, Random Forest, Bidirectional Long Short-Term Memory (BiLSTM) and Convolutional Neural Network (CNN). Each model was assessed based on its accuracy, evaluation metrices and ability to handle complex data patterns. Logistic Regression provided a moderate accuracy of 52%, serving as a baseline for comparison. KNN emerged as a top performer with an accuracy of 98%, indicating its efficacy in scenarios where proximity-based inference is paramount. However, its precision and recall for the positive class (0.82 and 0.63 respectively) suggest areas for potential improvement in precision and detection rates. XGBoost and Random Forest demonstrated superior performance with accuracies of 96% and 98% respectively, benefiting from their robust ensemble learning frameworks which excel in capturing complex high-dimensional data. The BiLSTM model, with an accuracy of 95%, excelled in processing sequential data, highlighting its potential for analyzing protein dynamics. This model also showed impressive class-specific performance with precisions of 0.94 for both classes, and recalls of 0.94 and 0.97 for negative and positive classes respectively. CNNs proved their adaptability beyond image processing by achieving an accuracy of 91%, effectively capturing spatial and structural patterns in sequential datasets. The results indicate that while some models excel in overall accuracy, others provide valuable insights into different aspects of the data. The report also explores potential improvements for each model, suggesting avenues for further research such as parameter tuning, handling data imbalance, and enhancing feature extraction techniques. This study contributes to the ongoing efforts in computational biology to develop reliable predictive tools for Alzheimer's Disease, leveraging the intricate patterns found in protein structures.

Contents

List of Figures

List of Abbreviations

AD	Alzheimer's disease
ADNI	Alzheimer's Disease Neuroimaging Initiative
BiLSTM	Bidirectional Long Short-Term Memory
CNN	Convolutional Neural Network
FN	False Negatives
FP	False Positives
GPU	Graphics Processing Unit
KNN	K-Nearest Neighbors
LSTM	Long Short-Term Memory
PDB	Protein Data Bank
ROC	Receiver Operating Characteristic
TN	True Negatives
TP	True Positives
TPU	Tensor Processing Unit
UniProt	Universal Protein Resource
XGBoost	eXtreme Gradient Boosting

Chapter 1

Introduction

Alzheimer's disease (AD) stands as a principal neurodegenerative disorder that affects millions globally, manifesting in severe cognitive decline and behavioral changes. This disease is characterized by the formation of amyloid-beta (Aβ) plaques and neurofibrillary tangles composed of tau protein, which together impair neural functions and lead to neuronal death. Understanding the precise structure and dynamics of these proteins is critical, as it holds the potential to unravel the complex interactions and mechanisms at play within the brain, potentially paving the way for groundbreaking therapeutic developments.
AD prevalence continues to rise alongside increases in global life expectancy, underlining its strong association with aging. The complex pathophysiology of AD encompasses multifaceted genetic factors and significant protein dysfunctions, contributing to the disease's challenging landscape, with no definitive cure currently available. Advances in computational biology have significantly impacted this field, leading to the development of sophisticated models that not only predict AD-related biomarkers but also identify potential therapeutic targets based on comprehensive genetic and proteomic data. These models enhance our predictive accuracy and deepen our understanding of the disease's mechanisms. By integrating these advanced computational techniques, researchers are able to conduct detailed analyses and simulations that can accelerate the discovery of effective treatments and preventive measures [6] [10].
In-depth studies have further elucidated that modifications in tau protein, particu-

larly through abnormal phosphorylation processes, are central to AD pathology [8]. Likewise, the aggregation dynamics of amyloid-beta peptide, forming toxic plaques, are now recognized as critical in AD's development [12]. These insights are crucial for developing targeted therapies that can potentially alter the course of the disease. Leveraging network topology and advanced machine learning techniques, researchers have successfully identified novel candidate genes linked to AD. These methods analyze complex protein interactions and genetic networks, offering new avenues for understanding and interrupting the disease process [4]. Concurrently, the utilization of big data analytics to explore protein misfolding has shed light on the molecular mechanisms underpinning AD, suggesting novel therapeutic targets[9].

Recent technological advancements in the field of Alzheimer's research underscore the necessity for precise prediction and deeper understanding of critical biomolecules like amyloid proteins and tau. Innovative computational methods, such as the Segmented-PSSM technique combined with secondary structure-based alignments, have substantially improved the prediction accuracy of amyloid proteins. These methods transform variable-length protein sequence profiles into high-dimensional feature vectors, capturing vital evolutionary information for enhanced predictive power [14].

Further, Bayesian models have been adapted to refine β-sheet prediction in proteins [1]. These models use probabilistic frameworks merged with dynamic programming techniques to enhance the modeling of β-strand interactions and alignments, crucial for understanding protein stability and function.

In addition, advancements in non-invasive brain imaging and machine learning have enabled more effective prediction of tau protein accumulation in the brain, a key marker of AD progression. Techniques such as the MLP-att model leverage resting-state functional MRI (rs-fMRI) data to analyze brain networks and predict tau content, offering a promising diagnostic tool that avoids the drawbacks of traditional PET scans. [15]

Finally, the use of convolutional neural networks (CNN) to predict protein disorder from amino acid sequences represents a significant leap forward in the field of Alzheimer's research. These models, employing deep learning algorithms and sophisticated feature extraction techniques like OneHotEncoding, achieve high accuracy in classifying proteins associated with AD. This approach is crucial for the

identification of new biomarkers that could lead to earlier detection and more precise monitoring of the disease's progression. By leveraging the power of CNNs, researchers are able to analyze vast datasets of amino acid sequences quickly and efficiently, making it a potent tool for advancing our understanding of Alzheimer's at a molecular level. This method not only enhances the accuracy of predictions but also paves the way for personalized medicine strategies in combating this debilitating condition. [13]
Adding to these advancements, a new CNN model has been developed for early AD diagnosis and classification using MRI images [3]. This model leverages the Alzheimer's Disease Neuroimaging Initiative (ADNI) dataset, which includes a vast array of MRI images across various stages of AD. It achieved a remarkable 99% accuracy, significantly enhancing diagnostic capabilities and providing valuable assistance to medical professionals by enabling quicker, more accurate assessments and potentially reducing the costs associated with conventional diagnostic methods.

1.1 Brief history of Predicting Alzheimer's Disease through Protein Structure

The history of protein structure prediction in Alzheimer's disease (AD) research is a tale of gradual progress amid complex challenges. Over the decades, the quest to understand the structural nuances of proteins like amyloid-beta and tau, which are central to AD pathology, has evolved significantly, driven by advancements in computational methods and biophysical understanding.
Initially, the focus was primarily on amyloid-beta, known for its role in forming plaques in the brains of AD patients. Early computational models aimed to elucidate the structural transitions of Aβ from soluble monomers to insoluble fibrillary aggregates. These studies were fundamental in identifying potential therapeutic targets aimed at disrupting these aggregations. Techniques such as molecular dynamics simulations and nuclear magnetic resonance spectroscopy were pivotal in these early days, providing insights into the transient structures of amyloid fibrils [5]. The prediction and understanding of tau protein's structure also gained momentum as its role in forming neurofibrillary tangles became clearer. Unlike Aβ,

tau is a natively unfolded protein that undergoes misfolding and aggregation in AD. Computational modeling, including the use of Monte Carlo simulations and later machine learning techniques, helped to predict the aggregation-prone regions within tau and hypothesize how pathological phosphorylation could affect tau's structure and aggregation [2].

In the 21st century, the advent of more powerful computational tools and algorithms, such as those based on deep learning, has further advanced the field. These tools have been used to predict the complex, multi-scale structural dynamics of both Aβ and tau proteins, including their interactions with other biomolecules in the brain.

Furthermore, the integration of computational biology with experimental methodologies has led to a more holistic approach to studying AD. For example, cryo-electron microscopy (cryo-EM) has recently provided high-resolution structures of tau filaments from AD patients, which were previously impossible to obtain. These structures have validated some of the predictions made by earlier computational models and have opened new avenues for drug design by revealing novel therapeutic targets[2].

Overall, the journey of protein structure prediction in AD research reflects a broader shift in the field from basic structural elucidation to a more dynamic understanding of the molecular mechanisms underlying Alzheimer's disease. This progress continues to drive the search for effective therapies, with the hope that understanding protein structures at an atomic level will ultimately translate into clinical benefits.

1.2 Title of the Project

The title of this project is **"Predicting Alzheimer's Disease through Protein Structure"**.This project focuses on advancing the predictive understanding of protein misfolding in Alzheimer's Disease (AD) by leveraging cutting-edge computational models. The primary aim is to map the intricate structures of key proteins, such as amyloid-beta and tau, whose misfolding and subsequent aggregation are pivotal in the pathogenesis of AD. This approach integrates computational biology, bioinformatics, and machine learning techniques to develop predictive models that offer novel insights into the molecular mechanisms underlying AD.

1.3 Problem statement

The project aims to bridge a significant gap in understanding the neurodegenerative mechanisms of Alzheimer's disease (AD). At the core of AD pathogenesis is the misfolding and aggregation of proteins such as amyloid-beta and tau, which disrupt brain cell function. Traditional methods like X-ray crystallography, used to study these proteins, are costly and time-intensive, limiting their practicality for extensive research or dynamic analysis. To overcome these limitations, we employ machine learning models, including Logisitic Regression, KNN, XGBoost, Random Forest, BiLSTM and CNN. These models are specifically tailored to analyze and predict the complex patterns of protein misfolding, offering a faster and cost-effective alternative for research and potential therapeutic development. To evaluate these models, we utilize metrics such as precision, recall, and the F1 score, alongside various plots like receiver operating characteristic (ROC) curve, precision-recall curve, confusion matrix plot, feature importance plot, model accuracy plot, model loss plot, as well as comprehensive model summary and classification report. The goal is to improve the models' predictive accuracy and adaptability in clinical settings, thereby supporting broader Alzheimer's research and therapeutic development.

1.4 Motivation

The motivation for this project stems from the urgent need to better understand and treat Alzheimer's disease (AD), a leading cause of dementia that impacts millions globally. AD's complexity, characterized by abnormal protein accumulations such as amyloid-beta and tau, demands more effective approaches to study their structures. This project aims to leverage advanced computational models, particularly machine learning and deep learning, to predict protein structures more accurately, efficiently, and cost-effectively than traditional methods. These efforts are anticipated to enhance our understanding of AD's pathophysiology, aiding early diagnosis and development of new treatments, thereby potentially slowing or reversing disease progression. This project aligns with global scientific endeavors to combat neurodegenerative diseases and holds promise for significantly improving the lives of those affected.

1.5 Objective

- Employ data preprocessing techniques to isolate key protein data, such as amyloid-beta (Aβ) and tau, improving the accuracy and specificity of Alzheimer's disease predictions.
- Build and refine various machine learning and deep learning models, such as Logistic Regression, Random Forest, KNN, XGBoost, BiLSTM and CNN for accurate prediction of Alzheimer's disease-related protein structures.
- Focus on enhancing the accuracy, precision, and recall of these models to minimize incorrect predictions in Alzheimer's disease protein structure analysis.
- Conduct systematic tuning of model hyperparameters to optimize performance and ensure reliable detection of protein anomalies related to Alzheimer's disease.
- Assess and compare the effectiveness of each model on complex Alzheimer's datasets to determine the most accurate and reliable approach.

1.6 Organization of the Rest of the Report

Chapter 2
This chapter reviews existing research on the molecular biology of Alzheimer's disease, traditional and current methods in protein structure prediction, and the role of machine learning in biomedicine.
Chapter 3
In this chapter, a brief explanation of the proposed methodology and its implementation are described with the help of flowcharts and figures.
Chapter 4
This chapter contains results, outcomes, and analysis of our method and implementation.It discusses the performance and comparative analysis of the computational models.

Chapter 5
In this chapter, we discuss the conclusions of our project and future work that can be carried out.

Chapter 2

Literature Review & Problem Identification

2.1 Literature review

Alzheimer's disease (AD) remains a major health challenge globally, and it's crucial to diagnose it early and accurately for effective treatment. By using advanced computer techniques like machine learning and deep learning, along with new imaging methods, doctors can better diagnose AD. These technologies help analyze complex data and identify subtle signs and markers that are difficult to detect without technology. Moreover, using deep learning on brain images helps understand how AD affects the brain over time. This combination of advanced technology and smart algorithms is improving the way we diagnose AD, potentially allowing for earlier treatment and more personalized care plans. This progress highlights how important continued innovation is in tackling the challenges of AD.

2.1.1 Imaging Techniques and Machine Learning Applications

Recent advancements in imaging technology combined with machine learning algorithms have shown promising results in improving Alzheimer's diagnosis. The adoption of Convolutional Neural Networks (CNNs) to process 3D brain MRI images has been a significant development. In Paper [15], the application of deep 3D-CNNs is explored, demonstrating the technology's ability to maintain the spatial integrity of the brain images, an essential factor for accurate AD recognition. This method provides a comprehensive view of the brain's structure, distinguishing it from older methods that rely on 2D slices or specific regions of interest (ROI). These innovative 3D-CNN models capture the full dimensionality of the brain, offering a more detailed and holistic analysis than previously possible. This approach ensures that all relevant anatomical features are considered, enhancing the accuracy of detecting and staging Alzheimer's disease. The method also enables automated, precise assessments that are less prone to human error and variability, promising a shift towards more consistent and reliable diagnostic practices in the clinical setting.

2.1.2 Hybrid Models and Enhanced Diagnostic Accuracy

Hybrid models combining various machine learning architectures have shown to enhance diagnostic precision significantly. Paper [17] describes a hybrid CNN model that integrates Resnet50 with other prominent CNN architectures like Alexnet and Vgg16. This innovative model classifies the stages of Alzheimer's disease with an impressive accuracy, outperforming traditional models. The multi-architecture approach allows for robust feature extraction and classification, which is critical in diagnosing and staging Alzheimer's disease. By leveraging the strengths of multiple CNN frameworks, the hybrid model optimizes the interpretation of complex imaging data, enhancing the ability to distinguish between the nuanced stages of the disease. Furthermore, the use of a hybrid approach facilitates a more comprehensive analysis of MRI scans, combining the detailed textural and structural insights provided by different architectures. This results in a significantly improved diagnostic tool that offers higher sensitivity and specificity.

2.1.3 Computational Techniques for Protein Disorder Prediction

The prediction of protein structure and disorder using machine learning algorithms, especially CNNs, offers a cutting-edge approach to understanding the biochemical foundations of Alzheimer's disease. Protein disorders, particularly misfolded proteins and abnormal protein aggregations, are hallmarks of AD and many other neurodegenerative diseases. Misfolded amyloid-beta (Aβ) peptides and tau proteins, for instance, are critical biomarkers and pathological features of AD.

Paper [15] presents a model that utilizes CNNs to predict intrinsic disorder within protein sequences. This disorder can indicate a propensity for a protein to fail to adopt a stable 3D structure under physiological conditions, which is a precursor to the aggregation phenomena observed in AD pathology. The CNN model in this study effectively captures the complex patterns within amino acid sequences that could indicate disorder, using a methodology that relies on transforming these sequences into numerical data through techniques like OneHotEncoding. The model then processes these sequences through multiple convolutional layers designed to detect and learn from the underlying patterns in the data [15].

2.1.4 Advantages of CNNs in Protein Prediction

The use of CNNs in this context is particularly advantageous because of their ability to maintain spatial hierarchy in data processing, which is vital when dealing with high-dimensional biological data. This capability allows CNNs to perform feature extraction at multiple scales, identifying subtle cues that might indicate abnormal protein structures. Paper 7 highlights a similar approach, employing CNNs to examine amino acid sequences for disorder, with the network achieving a high performance metric (AUC-ROC score of 92 percentage), suggesting its efficacy in binary classification tasks pertinent to medical diagnostics [7].

2.1.5 Implications for Alzheimer's Disease Research and Treatment

The successful prediction of protein disorders using CNNs has significant implications for Alzheimer's research. By identifying proteins that are likely to be disordered, researchers can better understand the pathological processes at the molecular level in AD. This understanding is crucial for developing targeted therapies aimed at preventing or reducing the misfolding and aggregation of specific proteins.

Moreover, these computational techniques offer a non-invasive and efficient method to screen for potential biomarkers in AD, supporting early diagnosis and potentially guiding therapeutic development. The ability to predict protein structure and disorder from amino acid sequences could also lead to breakthroughs in personalized medicine, where treatments can be tailored based on an individual's protein profile.

2.1.6 Early Diagnosis and Biomarker Identification

The role of biomarkers in early diagnosis of Alzheimer's cannot be understated. Paper [5] discusses current approaches to AD diagnosis and highlights the significance of biomarkers in detecting the disease at its nascent stages. Machine learning and deep learning techniques have been particularly effective in identifying and validating these biomarkers, which include genetic markers and protein deposits.

2.1.7 Computational Models and Future Directions

The exploration of quantum computing as a means to overcome the limitations of traditional computational models presents a new frontier in AD research, as noted in Paper [5]. Quantum computing promises faster and more complex problem-solving capabilities, which could revolutionize how AD is diagnosed and understood. The potential for quantum computers to handle vast datasets and perform computations at unprecedented speeds could allow for the analysis of complex biological data at a molecular level, facilitating deeper insights into the pathological mechanisms of Alzheimer's disease. Additionally, quantum computing could significantly enhance the accuracy and efficiency of existing machine learning algorithms, resulting in advancements in early diagnosis and tailored treatment approaches.

In conclusion, the integration of machine learning techniques with advanced imaging and computational models is paving the way for major progress in the diagnosis and understanding of Alzheimer's disease. These technologies not only enhance diagnostic accuracy but also offer new insights into the molecular and genetic underpinnings of AD, guiding the development of future therapeutic strategies.

2.2 Problem Identification

1. **Complexity of Protein Structures:** Proteins linked to Alzheimer's disease, such as amyloid-beta and tau, are complex and exhibit intricate folding patterns that are challenging to predict using conventional methods.

2. **Handling of Missing Data:** Incomplete data is a common issue in medical datasets. Developing methods to handle missing data effectively without compromising the integrity and accuracy of model predictions is crucial.

3. **Encoding of Protein Sequences and Structures:** The accurate encoding of protein sequences and structures is crucial for training predictive models effectively. However, due to the lengthy nature of these sequences and the complexity of their structures, significant computational power and memory are required. Handling these large datasets often leads to challenges in data processing and model training.

4. **Data Imbalance:** Many machine learning models struggle with imbalanced data, where instances of one class (e.g., non-Alzheimer's cases) vastly outnumber instances of another (e.g., Alzheimer's cases). This can skew prediction accuracy and reduce the model's practical effectiveness.

5. **Accuracy of Prediction Models:** Existing computational models often struggle with accuracy, especially when predicting the behavior of proteins that do not conform to typical folding patterns. Enhancing the accuracy of these models is crucial for reliable identification of disease markers.

6. **Dynamic Nature of Disease Progression:** Alzheimer's disease progression can vary significantly among individuals, influenced by a host of genetic, lifestyle, and environmental factors. Predicting how the disease will progress in different individuals poses a substantial challenge for current models.

7. **Cross-disciplinary Integration:** Effective prediction models for Alzheimer's disease require the integration of insights from various fields such as neurology, biochemistry, genetics, and computer science. Bridging these disciplines

to create a unified approach presents both a logistical and a methodological challenge.

8. **Computation Cost:** While deep learning offers accurate and real-time detection capabilities, training and testing such models require high-powered GPUs or TPUs, significantly increasing computational expense.

In summary, predicting Alzheimer's disease through protein structure poses several key challenges. Complex protein folding patterns, like those in amyloid-beta and tau, are difficult to predict with conventional methods. Incomplete and imbalanced data further complicate model training and accuracy. Moreover, accurately encoding protein sequences demands substantial computational resources, which are also needed to manage large datasets effectively. Current models often fall short in precision, especially for atypical protein behaviors, making it essential to improve prediction accuracy for reliable disease marker identification. The variability in disease progression and the need for cross-disciplinary integration add further complexity. Additionally, the high computational costs associated with deep learning models, which require advanced hardware like GPUs or TPUs , underscore the need for more efficient computational strategies.

Chapter 3

Proposed Methodology and Implementation

3.1 Methodology

In this chapter, we outline the methodologies adopted to predict protein sequence functions using a variety of machine learning and deep learning models. Each model is selected for its specific strengths in pattern recognition and prediction within bioinformatic datasets.

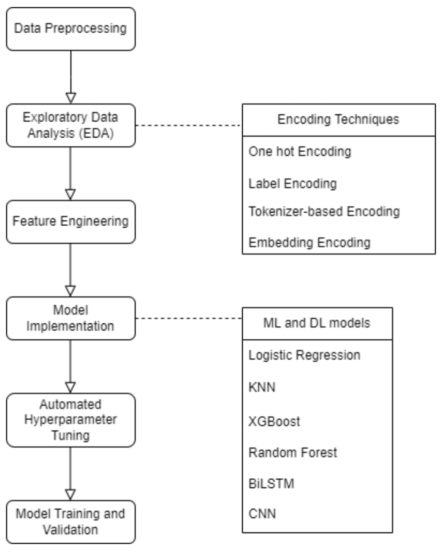


Figure 3.1: Methodology Flowchart

3.1.1 Data Preprocessing

The preprocessing of protein sequences is a critical step to ensure that the input data is suitable for training machine learning models effectively. The data preprocessing workflow adopted in this study consists of several detailed steps, each tailored to refine the dataset for optimal model performance:

Data Collection and Filtration

1. **Source Identification:**

   - Protein Data Bank (PDB) : The Protein Data Bank[11] is an open-access repository that stores 3D structural information of biological molecules, including proteins and nucleic acids, from crystallography, NMR spectroscopy, and cryo-electron microscopy. It serves as a critical resource for structural bioinformatics.
   - Universal Protein Resource (UniProt) : UniProt[16] is an extensive database for protein sequence and annotation data, combining information from sequencing projects, scientific literature, and expert analysis. It offers comprehensive insights into protein functions and biological roles.

2. **Initial Filtration:** The first and crucial step in creating the dataset was to accurately identify and segregate proteins known to be directly involved in the pathogenesis of Alzheimer's disease (AD). The filtration involved querying the PDB and UniProt databases for entries specifically linked to amyloid-beta and tau. This was primarily done through PBD IDs associated with these proteins.

   - Identifying AD proteins (Class 1): Proteins that matched these criteria were then categorized as Class 1. This classification was based on their proven and direct link to AD pathology, ensuring that these proteins are the primary focus of the study for their roles in disease mechanisms.
   - Identifying Non-AD Proteins (Class 0): All other proteins in the databases that did not meet the specific criteria for amyloid-beta or tau were clas-

sified as Class 0. This includes proteins without annotations or keywords linking them to AD.

.

Cleaning and Further Refinement

1. **Removal of Blank Entries:** During the data cleaning phase, entries with missing or incomplete data were removed. This step is essential to prevent any errors during the model training phase due to null or corrupt data.

2. **Sequence Verification:** Each sequence was checked for accuracy against known databases to ensure that only correct and verified sequences were included in the final dataset. This verification helps in maintaining the quality and reliability of the input data.

3.1.2 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is an essential first step in the process of analyzing data, particularly in complex fields like biological and medical research. It involves a systematic approach to uncovering the underlying patterns, spotting anomalies, testing assumptions, and checking the robustness of the data. In biological datasets, such as those involving protein sequences linked to diseases, EDA helps in understanding the typical characteristics of the data, such as sequence lengths and amino acid distributions. This process not only prepares the data for further statistical or machine learning analysis but also provides valuable insights that guide future research directions. Moreover, the visual and quantitative methods used in EDA enhance the communication of findings, ensuring that both technical and non-technical stakeholders can grasp the implications of the data effectively.

1. **Distribution of Sequence Lengths**
   The first plot examines the distribution of sequence lengths within the dataset. The bin sizes are set to 25 amino acids and truncated the display at 1000 amino acids because few sequences exceeded this length. The histogram reveals a range of sequence lengths, highlighting the diversity in size among the

protein sequences analyzed. This visualization helps in understanding the typical protein sizes and identifying any outliers or unusual patterns in sequence length distribution.

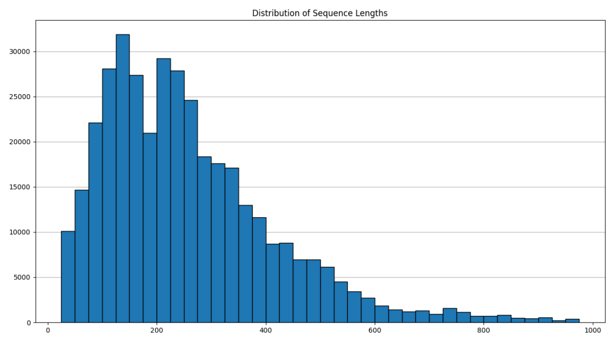


Figure 3.2: Distribution of Sequence Lengths

2. **Secondary Structure Distributions**
   This plot provide insights into the distribution of secondary structures in two formats: SST-8 and SST-3. In the SST-8 format, secondary structures are categorized into eight types (B, E, G, H, I, C, S, T), and in the SST-3 format, they are simplified into three types (E, H, C). The color coding (gold for extended structures, crimson for helical structures, and navy for turns and loops) aids in quick interpretation of the data.

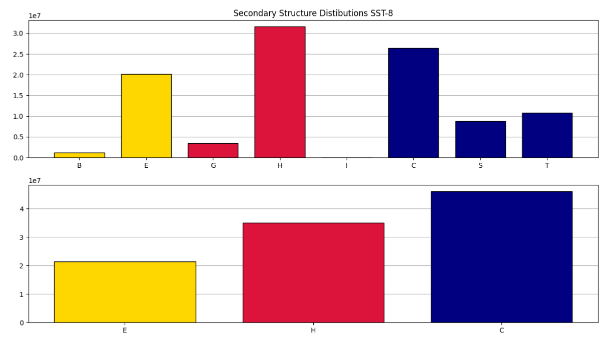


Figure 3.3: SST-8 and SST-3 Distributions

3. **Amino Acid Representation**
The frequency of each amino acid within the dataset is illustrated through a bar plot. This analysis is crucial as it reveals the most common amino acids and their relative proportions, providing insights into the chemical characteristics of the sequences.

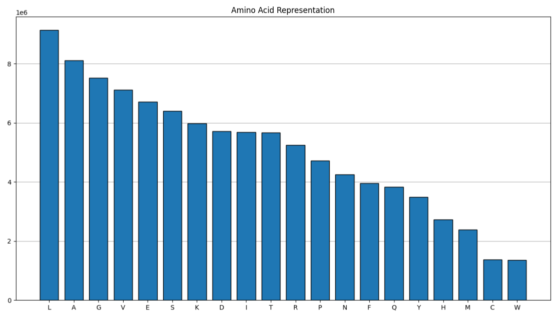


Figure 3.4: Amino Acid Representation

4. **Sequence Length by Neurodegenerative Protein Presence** A boxplot compares the sequence lengths between proteins associated with neurodegenerative diseases and those that are not. This plot is instrumental in determining if neurodegenerative proteins tend to have longer or shorter sequences compared to other proteins, which might suggest differences in structural complexity or functional roles.

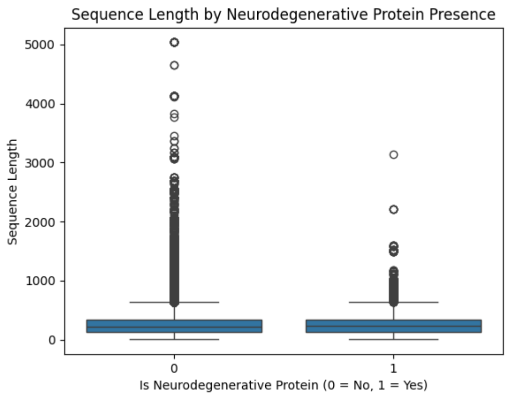
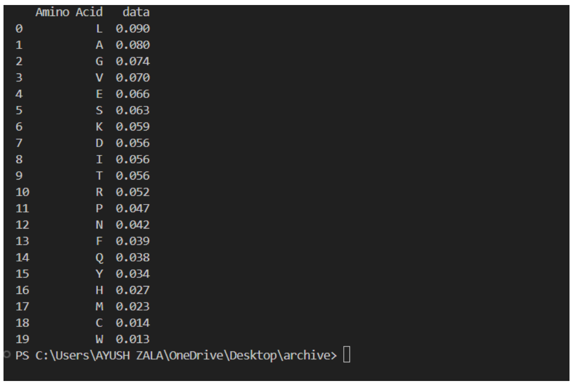


Figure 3.5: Boxplot of Sequence Length by Protein Presence

5. **Proportion of Amino Acids** In addition to visual plots, a table displaying the proportion of each amino acid relative to the total count provides a quantitative perspective. This table complements the amino acid frequency plot by giving exact figures, thereby enabling more detailed analysis.



```
   Amino Acid   data
0           L  0.090
1           A  0.080
2           G  0.074
3           V  0.070
4           E  0.066
5           S  0.063
6           K  0.059
7           D  0.056
8           I  0.056
9           T  0.056
10          R  0.052
11          P  0.047
12          N  0.042
13          F  0.039
14          Q  0.038
15          Y  0.034
16          H  0.027
17          M  0.023
18          C  0.014
19          W  0.013
PS C:\Users\AYUSH ZALA\OneDrive\Desktop\archive>
```

Figure 3.6: Proportion of Amino Acids

3.1.3 Feature Engineering

Feature engineering is a vital part of the data preprocessing stage when constructing machine learning models, where raw data is transformed and manipulated into formats that are better suited for these models. This process entails developing new features or altering existing ones to enhance the predictive capabilities of machine learning algorithms. Effective feature engineering enhances the accuracy and efficiency of the models by providing them with meaningful data that captures the underlying patterns better.

Feature Transformation

Feature transformation is a crucial step in handling and processing raw data to make it suitable for modeling, especially when dealing with categorical data like protein sequences. Here are the types of feature transformations implemented in this project:

1. **One-hot Encoding:** Each amino acid in a protein sequence is encoded as a 20-dimensional binary vector, corresponding to the 20 standard amino acids.

In this vector, the position associated with a specific amino acid is marked as 1, whereas all other positions are marked as 0. This encoding transforms the categorical data into a numerical format that can be efficiently processed by machine learning models. This encoding method transforms categorical data (amino acids) into a numerical format without imposing any ordinal relationship among them. This is crucial for models that cannot inherently handle categorical data and ensures each amino acid is treated as distinct without any implied order or proximity.

2. **Label Encoding:** Each unique amino acid is assigned a unique integer. This is a simpler encoding that converts categorical labels into a compact integer form.While label encoding efficiently reduces the dimensionality of the data by replacing categories with integers, it introduces an arbitrary ordinal relationship among categories, which might not be appropriate for all types of models. This method is often used when the categorical features are ordinal or when using algorithms that can handle categorical variables directly and interpret the numerical values meaningfully.

3. **Tokenizer-based Encoding:**A tokenizer converts text data into sequences of integers, where each integer corresponds to a unique token (or amino acid) based on the vocabulary learned from the dataset.Tokenizer-based encoding is used to prepare text for embedding layers within neural networks, mapping each unique word or character to a unique integer. This transformation is necessary for further processing like embedding or LSTM processing. In protein sequence analysis, tokenizer-based encoding transforms amino acid sequences into a format suitable for neural network models that learn sequence dependencies, such as LSTMs or models with embedding layers.

4. **Embedding Encoding:**An embedding layer takes sequences of integers (from tokenizer-based encoding) and maps them to dense vectors of a specified size. This layer is learnable and adjusts during model training to capture and represent more abstract and useful features of the input data.Embeddings provide a way to reduce the dimensionality of the input data and to capture hidden relationships between different tokens (amino acids), facilitating the learning

of complex patterns in data such as protein sequences. Embeddings are particularly useful in deep learning models where the relationships and patterns among amino acids can be intricately modeled. They allow models to capture contextual relationships within sequences, which is essential for tasks like protein function prediction or structure determination.

3.1.4 Models Implementation

K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a versatile machine learning algorithm primarily used for classification. It operates on the principle that similar instances are often located near each other in the feature space. In the KNN approach, the label of a new instance is determined based on a majority vote among its 'k' closest neighbors in the training dataset. This proximity is typically measured using Euclidean distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{3.1}$$

where x and y represent data points,and x_i and y_i are their respective coordinates.

In our research focused on Alzheimer's proteins, we selected KNN due to its effectiveness in handling complex, non-linear decision boundaries, which is crucial given the intricate nature of protein interactions. The simplicity of KNN also makes it an excellent baseline for comparing the performance of more complex models in early analysis stages.

XGBoost

XGBoost, or eXtreme Gradient Boosting, is a robust machine learning algorithm known for its speed and high performance. It utilizes a series of decision trees built sequentially, with each tree improving on the previous ones by correcting their errors. This technique helps XGBoost deliver accurate predictions on complex datasets.

In this project, XGBoost was chosen for its proficiency in handling structured, complex data like protein sequences, its ability to manage large datasets, and its balance of speed and accuracy, which is crucial for the rapid testing needed in medical research. During training, XGBoost minimizes an objective function that includes a loss function ℓ measuring prediction accuracy and a regularization term Ω that controls model complexity:

$$Obj(\Theta) = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \Omega(\Theta) \tag{3.2}$$

Here, y_i represents the true output, $\hat{y}_i$ represents the predicted output from the model, and Θ represents the parameters of the model. The regularization term Ω typically includes components like the sum of the squares of the weights (to control the model's complexity and prevent overfitting), defined as:

$$\Omega(\Theta) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{3.3}$$

where T is the number of trees, w_j are the leaf weights of the trees, γ is a parameter that controls the cost of adding new trees, and λ is a parameter that provides L2 regularization on the weights.

Logistic Regression

Logistic Regression is employed in this project as a baseline model for binary classification. It uses the logistic function to predict binary outcomes, such as the presence or absence of characteristics in disease-related proteins. This model is chosen for its simplicity and interpretability, which are crucial in medical research for understanding how specific features like amino acids or physical properties of protein sequences influence outcomes.

Additionally, Logistic Regression is computationally less intensive, making it ideal for initial exploratory analyses to rapidly test hypotheses concerning features that may predict disease-related proteins.

The logistic function in Logistic Regression is represented as:

$$p(y = 1|x) = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + \ldots + b_n x_n)}} \tag{3.4}$$

Here, $p(y = 1|x)$ is the probability of the binary outcome being 1 given predictors x, b_0, b_1,. . . , b_n are the model coefficients.

Random Forest

Random Forest is a robust ensemble learning technique well-suited for handling complex, high-dimensional datasets. This method constructs multiple decision trees during training, with each tree providing a prediction. In classification tasks, the final output is determined by the mode of the classes predicted by each tree, while in regression tasks, it is calculated as the average of the predictions.

In this project, Random Forest is selected for its effectiveness in managing complex, high-dimensional datasets, such as those containing protein sequences. This ensemble learning technique builds multiple decision trees during training and delivers the mode of the classes for classification tasks. It provides robustness and accuracy, handling the intricacies and potential noise in biological data efficiently.

The general formula for prediction in Random Forest, averaging the predictions of N trees, is given by:

$$RandomForestPrediction = \frac{1}{N}\sum_{i=1}^{N} T_i(x) \tag{3.5}$$

where $T_i(x)$ represents the prediction of the i-th decision tree for the input x, and N is the total number of trees in the forest.

BiLSTM (Bidirectional Long Short-Term Memory)

BiLSTM (Bidirectional Long Short-Term Memory) networks, a form of recurrent neural network, are exceptionally effective for analyzing sequential data, including complex biological sequences like proteins. The advantage of BiLSTMs over traditional LSTMs lies in their ability to capture long-range dependencies from both past and future contexts within the sequence, which is essential when the relationship between distant amino acids influences the protein's structure and function.

The computational model of BiLSTMs allows them to retain information over long periods from both directions of the sequence, which is crucial in modeling the intricate dependencies that exist in protein sequences. This capability is particularly important in projects focusing on the prediction and analysis of protein behaviors linked to neurodegenerative diseases such as Alzheimer's, where understanding the full sequence context in both temporal directions is vital to understanding protein functionality.

The key to BiLSTM's effectiveness is its internal structure, which includes memory cells that regulate the flow of information in both forward and backward directions. These cells decide what to retain or discard from the cell state, thus allowing the model to selectively remember or forget patterns from both past and future inputs. The mathematical representation of a BiLSTM unit involves two sets of LSTM equations, one for the forward pass and one for the backward pass, which are processed simultaneously:

Forward LSTM Pass:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (ForgetGate) \tag{3.6}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (InputGate) \tag{3.7}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (CellCandidate) \tag{3.8}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (CellStateUpdate) \tag{3.9}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (OutputGate) \tag{3.10}$$

$$h_t = o_t * \tanh(C_t) \quad (HiddenState) \tag{3.11}$$

In these equations for the forward pass:

- σ denotes the sigmoid activation function, which is used to scale the inputs to the gates between 0 and 1.
- tanh is the hyperbolic tangent function, providing outputs between -1 and 1,

used to regulate the information flowing through the network.

- W and b represent the weight matrices and bias vectors for the respective gates.
- x_t is the input vector at time step t.
- h_t is the output vector from the LSTM unit at time step t, also known as the hidden state.
- C_t is the cell state at time step t, which acts as an "internal memory" of the LSTM cell.

Backward LSTM Pass:

$$f'_t = \sigma(W'_f \cdot [h'_{t+1}, x_t] + b'_f) \quad (ForgetGate) \tag{3.12}$$

$$i'_t = \sigma(W'_i \cdot [h'_{t+1}, x_t] + b'_i) \quad (InputGate) \tag{3.13}$$

$$\tilde{C}'_t = \tanh(W'_C \cdot [h'_{t+1}, x_t] + b'_C) \quad (CellCandidate) \tag{3.14}$$

$$C'_t = f'_t * C'_{t+1} + i'_t * \tilde{C}'_t \quad (CellStateUpdate) \tag{3.15}$$

$$o'_t = \sigma(W'_o \cdot [h'_{t+1}, x_t] + b'_o) \quad (OutputGate) \tag{3.16}$$

$$h'_t = o'_t * \tanh(C'_t) \quad (HiddenState) \tag{3.17}$$

In these equations for the backward pass:

- f'_t, i'_t, $\tilde{C}'_t$, C'_t, o'_t, and h'_t represent the backward forget gate, input gate, cell state candidate, cell state, output gate, and hidden state at time step t, respectively.
- W'_f, W'_i, W'_c, and W'_o are the weight matrices for the backward pass corresponding to the forget gate, input gate, cell candidate, and output gate.

- b'_f, b'_i, b'_c, and b'_o are the bias vectors for the respective gates in the backward direction.

- h'_{t+1} is the backward hidden state from the next time step (as this is the backward pass), providing the linkage between sequential elements in the reverse sequence.

- x_t remains the same input vector at time step t as used in the forward pass, emphasizing that the same input sequence is processed in reverse.

Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is extremely effective for processing data with grid-like topology, such as images, or, in the context of our project, protein sequences. The strength of CNNs lies in their ability to detect and utilize spatial hierarchies and patterns within the data. In protein sequences, these patterns often manifest as motifs or structural features crucial for biological functions.

In this project, CNNs are employed to analyze protein sequence data, leveraging their ability to detect patterns akin to motifs which are vital for protein functionality. This approach allows us to harness the CNN's pattern recognition prowess to effectively study complex biological sequences, providing insights into protein functions associated with disease states.

The key operation in a CNN, the convolution process, can be mathematically described by the following equation:

$$S(i,j) = (K * X)(i,j) = \sum_m \sum_n K(m,n)X(i-m, j-n) \tag{3.18}$$

Here, S(i,j) is the output of the convolution operation at position (i,j), K is the kernel or filter matrix, X is the input matrix (e.g., encoded protein sequences), and $\star$ denotes the convolution operation. The indices m and n traverse the kernel's dimensions, applying the kernel across the input matrix to produce a feature map that highlights important patterns.

3.1.5 Automated Hyperparameter Tuning

Hyperparameter tuning is an essential step in developing machine learning models because it involves choosing the optimal set of parameters that control the training process of an algorithm. These parameters, often not learned directly from the training process itself, can significantly influence model performance. Automated hyperparameter tuning refers to the process of systematically searching for the ideal set of hyperparameters for a given model, using specific algorithms designed to explore the parameter space efficiently and effectively.

K-Nearest Neighbors (KNN)

For K-Nearest Neighbors (KNN), the choice of the number of neighbors (k) is crucial as it directly impacts the model's performance by influencing the bias-variance trade-off. In this project, we have optimized the k parameter along with other key hyperparameters using GridSearchCV, a powerful tool that automates the process of parameter tuning to find the most effective model settings.The specific parameters we explored in our GridSearch are as follows:

1. **Number of Neighbors:** We tested several values for k including 3, 5, 7, and 10 to determine the optimal number of neighbors.

2. **Weights:** We experimented with both 'uniform' and 'distance' weighting methods.

3. **Metric:** We considered two types of distance metrics - 'euclidean' and 'manhattan'.

XGBoost

In our project, we have optimized the XGBoost model using GridSearchCV to systematically find the best hyperparameter settings for our predictive tasks. The parameters included in our grid search were:

1. **Max Depth:** We tested tree depths of 6 and 10 to control the complexity of the model.

2. **Min Child Weight:** Values of 1 and 5 were evaluated to manage over-fitting by influencing the decision of making further partitions.

3. **Gamma:** Set at 0 and 0.1 to regulate the model's complexity.

4. **Number of Estimators:** We explored 100 and 150 to determine the optimal number of boosting stages.

5. **Learning Rate:** We experimented with rates of 0.1, 0.2, and 0.3 to manage the contribution of each tree in the ensemble.

Logistic Regression

For our Logistic Regression model, we utilized GridSearchCV to fine-tune the hyperparameters, ensuring optimal model performance. The parameters included in our search were:

1. **Regularization Strength (C):** We explored a range of values from 10^{-4} to 10^{4}, using np.logspace to evenly distribute the values on a logarithmic scale.

2. **Solver:** We tested 'liblinear' and 'lbfgs', which are algorithms suited for different types of optimization problems

3. **Class Weight:** Options included 'None' and 'balanced' to address class imbalance,

Random Forest

In our project, we focused on optimizing key hyperparameters of the Random Forest model using GridSearchCV to ensure the best possible predictions. The specific parameters we included in our search were:

1. **Number of Trees:** We tested with 100 and 200 trees. Increasing the number of trees can improve the model's accuracy by averaging more decision trees.

2. **Maximum Depth of Trees:** We explored depths of 10, 20, and 'None'.

BiLSTM (Bidirectional Long Short-Term Memory)

For BiLSTM models, essential hyperparameters encompass the number of LSTM units in each direction, the number of layers, and the learning rate. Given the increased complexity and computational demand of training BiLSTMs, which process data both forwards and backwards, Bayesian optimization is particularly advantageous. This method improves efficiency in exploring the hyperparameter space compared to grid search or randomized search by leveraging previous evaluations to predict which hyperparameters might deliver superior results.

Convolutional Neural Network (CNN)

CNNs involve hyperparameters like the number of convolutional layers, the size of the filters, the depth of each layer, and the dropout rate. Automated tuning for CNNs often uses techniques like Bayesian optimization, which can smartly manage the vast space of possible hyperparameter combinations more effectively than exhaustive searches. This is particularly beneficial given the deep architecture of CNNs and the intricate dependencies between layers.

3.1.6 Model Training and Validation

The process of model training and validation is fundamental to the development and assessment of machine learning algorithms. It involves two primary phases: training the model on a specific set of data and then validating its performance on a separate set of data that it has not encountered before. This approach is crucial to ensure that the model can effectively generalize to new, unseen data rather than merely performing well on the data it was trained on (a problem known as overfitting).

Model Training

During the training phase, the model learns to make predictions or decisions based on the data provided. This process entails tweaking the model's parameters (or weights, in the context of neural networks) to reduce a loss function, which measures the discrepancy between the model's predictions and the actual results. The specific algorithms and their corresponding parameters (which may have been optimized through hyperparameter tuning) are applied to the training data. For models like K-Nearest Neighbors, this phase may simply involve storing the training data, whereas for models like XGBoost or Neural Networks, a more complex iterative process of updates to model weights is involved.

Model Validation

Once the model has been trained, it is crucial to validate its performance to ensure it is both effective and robust. Validation typically involves testing the model on a set of data that was not used during the training phase. This helps in evaluating the model's ability to generalize. There are several methods of validation:

1. **Holdout Method:** Dividing the dataset into a training set and a testing set, where the model is trained on the training portion and validated on the testing portion.

2. **Cross-Validation:** Often used to better utilize the available data, cross-validation involves partitioning the data into several subsets and conducting multiple fitting sessions. Each subset is used as a testing set at turns, while the remaining serve as the training set.

3. **Bootstrapping:** This involves repeatedly sampling subsets of the dataset with replacement, training the model on each subset, and validating it on samples not included in the subset. This technique helps in assessing the variability of the model performance.

Evaluating the performance of the model:

To assess the performance of the proposed model we need some performance matrices that could tell how good our model is performing. The following section discusses the performance matrices that are used for the evaluation of the model:

- **Precision :** Precision measures the proportion of predicted positives (positive class label assigned by the model) that are actually correct (True Positives). It reflects how precise the model is in identifying the target class. It can be calculated by equation 3.19.

$$Precision = \frac{TP}{TP + FP} \tag{3.19}$$

- **Recall :** Recall measures the proportion of actual positive cases (belonging to the target class) that are correctly identified by the model (True Positives). It reflects how well the model captures all the relevant instances of the target class. It can be calculated by equation 3.20 .

$$Recall = \frac{TP}{TP + FN} \tag{3.20}$$

- **Accuracy :** Accuracy is the most basic metric, representing the overall proportion of correct predictions made by the model. It is calculated by dividing the sum of True Positives (TP) and True Negatives (TN) by the total number of predictions made. It can be calculated by equation 3.21.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{3.21}$$

- **F1-Score :** The F1-Score is the harmonic mean of precision and recall, integrating both metrics into a single value. It provides a balanced view of the model's performance, addressing the potential shortcomings of focusing solely on accuracy. It can be calculated by equation 3.22.

$$F1 - Score = \frac{2 \times Preci\ ion \times Recall}{Precision + Recall} \tag{3.22}$$

3.2 Simulation environment

3.2.1 Hardware Requirements

The computational demands of machine learning, particularly deep learning models, are substantial. To handle these demands, especially when processing large datasets or employing complex neural network architectures, robust hardware infrastructure is necessary.

1. **GPUs:** Training deep learning models is computationally intensive, and GPUs (Graphics Processing Units) are critical in this regard. They are designed to handle multiple calculations simultaneously, making them significantly faster than general-purpose CPUs for machine learning tasks. For this project, an Nvidia GeForce RTX 3050 GPU was utilized, which significantly reduced training time from days to hours, a vital factor in iterative model development and tuning processes.
2. **Memory:** Adequate RAM is necessary to load training datasets and support the operations of large models during training. Insufficient memory can lead to swapping data to disk, significantly slowing down the training process. In this case, 16GB of RAM was available, which provided a balance between performance and cost, accommodating the needs of most training scenarios without frequent data swapping.
3. **Storage:** Fast and ample storage is necessary to handle large datasets and the multiple model configurations that arise during experimental trials. Solid-state drives (SSD) are preferred over hard disk drives (HDD) for faster data access and processing speeds. The Samsung SSD 512GB used in this project ensured quick read/write speeds, which is crucial for managing large data volumes and complex model architectures.

3.2.2 Software Requirements

The choice of programming languages and libraries is critical in defining the efficiency and ease of model development, training, and validation.

Python 3.11.3:

Python remains the most popular language for machine learning due to its simplicity and the vast availability of libraries and frameworks that streamline the coding process. Python 3.11.3 version offer the latest features and improvements, making them suitable for modern machine learning tasks.

Libraries and Frameworks:

1. **NumPy and Pandas:** These libraries are essential for data manipulation and analysis. NumPy provides support for large, multi-dimensional arrays and matrices along with a large collection of high-level mathematical functions to operate on these arrays. Pandas offers powerful and flexible data structures like DataFrames, making it easy to manipulate structured data.

2. **Matplotlib:** This plotting library is essential for visualizing data, which is an integral part of exploratory data analysis. It helps in understanding the distributions and relationships in the data through graphical representations.

3. **TensorFlow and Keras:** TensorFlow is a comprehensive framework that allows the building and training of neural networks with high scalability and flexibility. Keras, which runs on top of TensorFlow, provides a simpler, high-level API for creating deep learning models. It simplifies many tasks and is highly suitable for fast prototyping.

4. **scikit-learn:** This library offers straightforward and effective tools for data mining and data analysis, built upon NumPy, SciPy, and Matplotlib. It is particularly useful for classical machine learning models like regression, clustering, and classification, offering numerous built-in methods for model selection and evaluation.

Environment Management:

Visual Studio Code (VSCode) is utilized as the development environment for managing and executing the simulation of models in this research. This lightweight and versatile IDE supports Python, offering robust features such as intelligent code completion, integrated Git control, and a variety of extensions tailored for efficient data science and machine learning workflows. Its adaptability and comprehensive toolset make VSCode an ideal choice for developing, testing, and maintaining complex machine learning models and datasets.

3.3 Dataset Description

The dataset used in this study, transformed from a raw file downloaded from RCSB PDB [11], is formatted into a structured table that lists peptide sequences with their corresponding secondary structures in both Q8 and Q3 formats, alongside additional relevant data:

- **pdb_id:** Identifier for the protein data bank entry. The dataset contains a total of 164,619 unique PDB identifiers out of 477,154 entries, highlighting different varieties of proteins included.
- **chain_code:** Identifier for specific chains within protein structures, as proteins can be composed of multiple peptide chains.
- **seq:** Amino acid sequence of the peptide. There are 112,700 unique sequences in the dataset, indicating a diverse range of peptide structures.
- **sst8:** Assigned eight-state secondary structure. The dataset includes 394,837 unique eight-state secondary structures, demonstrating the complexity and variability of protein conformations.
- **sst3:** Simplified three-state secondary structure, with 346,857 unique entries. This reflects the broader categorization of secondary structures into three primary types: helix, sheet, and coil.
- **len:** Length of the peptide.
- **has_nonstd_aa:** Indicates the presence of nonstandard amino acids, identified by special characters like B, O, U, X, or Z.

Training and Testing Sets:

The dataset is divided into training and testing sets, typically using an 80/20 ratio. This split strategy ensures that the majority of the data is used for training the models, while a substantial subset remains isolated for final evaluation. This method aids in evaluating the model's performance on unseen data, providing insights into its generalization capability.

Chapter 4

Results

4.1 Outcomes of the proposed model

4.1.1 Logistic Regression

Logistic regression is traditionally valued for its simplicity and interpretability. However, in our specific analysis, the performance of the model presents some challenges. It achieves an overall accuracy of 53%, which is considerably lower than one might expect from this model in simpler scenarios. This suggests limitations in handling complex patterns, which more sophisticated models might better capture.

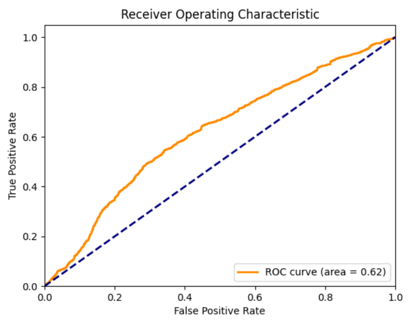


Figure 4.1: Logistic Regression ROC Curve

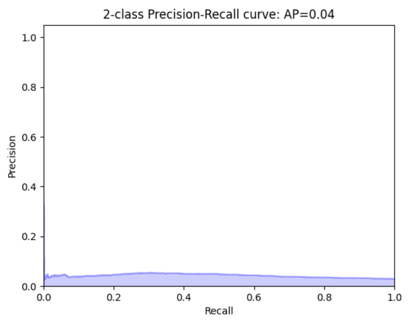


Figure 4.2: Logistic Regression Precision and Recall Curve

The evaluation using error metrics indicates significant predictive challenges:

1. **Mean Absolute Error (MAE):** 0.48, suggesting notable average error per case.

2. **Mean Squared Error (MSE):** 0.24, representing the average squared difference between the estimated values and the actual value.

3. **Root Mean Squared Error (RMSE):** 0.49, indicating the square root of the second sample moment of the differences between predicted values and observed values.

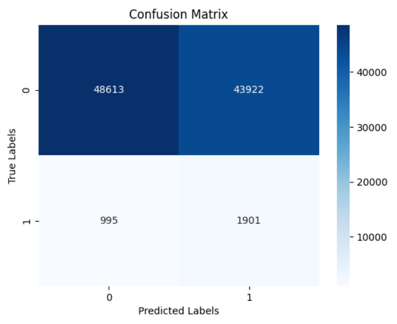


Figure 4.3: Logistic Regression Confusion Matrix

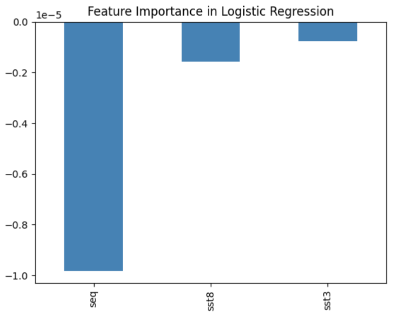
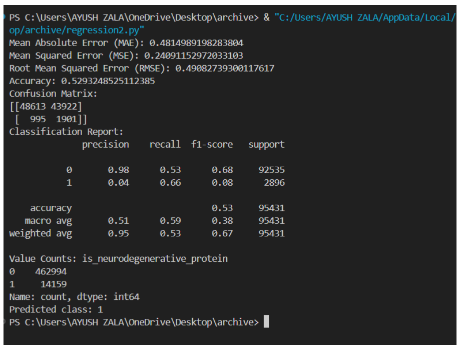


Figure 4.4: Logistic Regression Feature Importance

The precision and recall for each class further elucidate the model's specific areas of strength and weakness:

- **Class 0 (Negative Class):** High precision of 0.98 shows the model's effectiveness in identifying true negatives accurately. However, a recall of 0.53 points to the model identifying just over half of the actual negatives correctly.

- **Class 1 (Positive Class):** Precision significantly drops to 0.04, indicating that when the model predicts positives, it is correct only 4% of the time. Despite this, the recall of 0.66 is moderately high, suggesting that the model can identify about two-thirds of all actual positives.



```
PS C:\Users\AYUSH ZALA\OneDrive\Desktop\archive> & "C:/Users/AYUSH ZALA/AppData/Local/
op/archive/regression2.py"
Mean Absolute Error (MAE): 0.4814989198283804
Mean Squared Error (MSE): 0.24091152972033103
Root Mean Squared Error (RMSE): 0.49082739300117617
Accuracy: 0.5293248525112385
Confusion Matrix:
[[48613 43922]
 [  995  1901]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.53      0.68     92535
           1       0.04      0.66      0.08      2896

    accuracy                           0.53     95431
   macro avg       0.51      0.59      0.38     95431
weighted avg       0.95      0.53      0.67     95431

Value Counts: is_neurodegenerative_protein
0    462994
1     14159
Name: count, dtype: int64
Predicted class: 1
PS C:\Users\AYUSH ZALA\OneDrive\Desktop\archive> 
```

Figure 4.5: Logistic Regression Classification Report

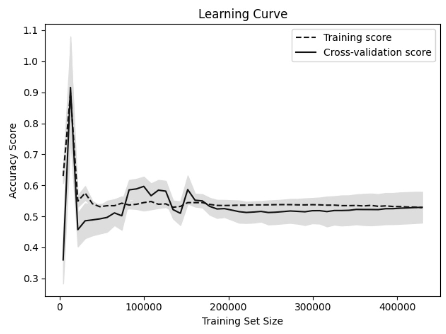


Figure 4.6: Logistic Regression Learning Curve

4.1.2 K-Nearest Neighbors (KNN)

The K-Nearest Neighbors (KNN) algorithm demonstrated remarkable accuracy of 98% in our evaluation, showcasing strong performance metrics particularly in the classification of the negative class. Here are the detailed results:

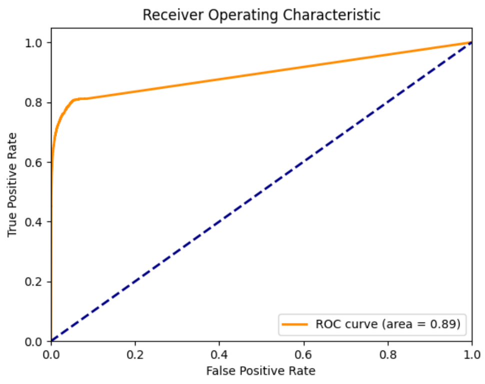


Figure 4.7: KNN ROC Curve

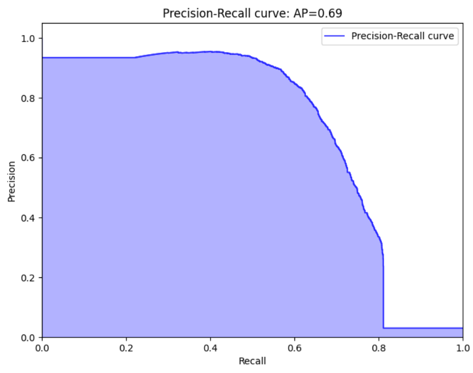


Figure 4.8: KNN Precision and Recall Curve

- **Accuracy:** 98.4334%, indicating high overall performance.
- **Parameter Optimization:** The model's performance was finely tuned using a 5-fold cross-validation approach across 16 candidates, resulting in 80 total

fits. This optimization strategy helped pinpoint the most effective parameters for maximizing the predictive accuracy of the model, which included:

1. **Distance Metric:** 'Manhattan', optimizing the approach to measure distances between points.
2. **Number of Neighbors:** 10, utilizing a moderate number to balance the model's generalization and specificity.
3. **Weighting Type:** 'Distance-based', emphasizing closer neighbors more significantly, which enhances the model's sensitivity to the local data structure.

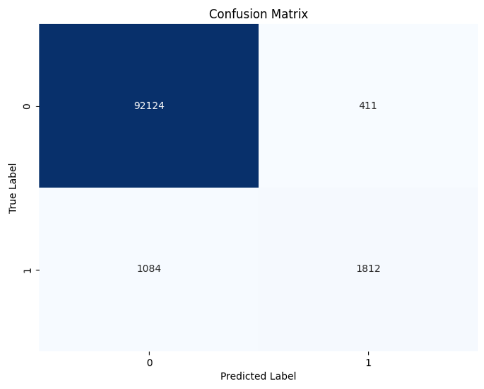


Figure 4.9: KNN Confusion Matrix

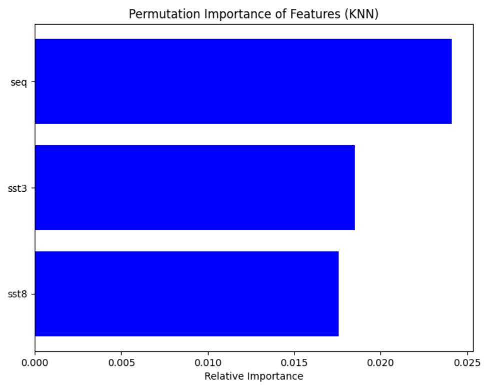
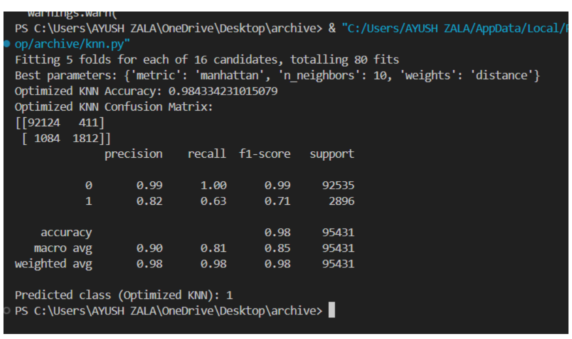


Figure 4.10: KNN Feature Importance

The Precision and Recall Analysis for K-Nearest Neighbors (KNN) includes:

- **Class 0 (Negative Class):** High precision of 0.99 indicates the model's effectiveness in accurately identifying true negatives, nearly without error. The recall of 1.00 demonstrates that the model identifies all actual negative instances correctly, confirming its exceptional capability in handling this class.
- **Class 1 (Positive Class):** The precision significantly decreases to 0.82, suggesting that when the model predicts an instance as positive, it is correct approximately 82% of the time, which is reasonably high but indicates some

room for improvement. However, the recall at 0.63 is moderately high, indicating that the model can identify about two-thirds of all actual positive cases, reflecting a potential area of weakness in detecting positive instances.



```
  warnings.warn(
PS C:\Users\AYUSH ZALA\OneDrive\Desktop\archive> & "C:/Users/AYUSH ZALA/AppData/Local/
op/archive/knn.py"
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Best parameters: {'metric': 'manhattan', 'n_neighbors': 10, 'weights': 'distance'}
Optimized KNN Accuracy: 0.984334231015079
Optimized KNN Confusion Matrix:
[[92124   411]
 [ 1084  1812]]
              precision    recall  f1-score   support

           0       0.99      1.00      0.99     92535
           1       0.82      0.63      0.71      2896

    accuracy                           0.98     95431
   macro avg       0.90      0.81      0.85     95431
weighted avg       0.98      0.98      0.98     95431

Predicted class (Optimized KNN): 1
PS C:\Users\AYUSH ZALA\OneDrive\Desktop\archive>
```

Figure 4.11: KNN Classification Report

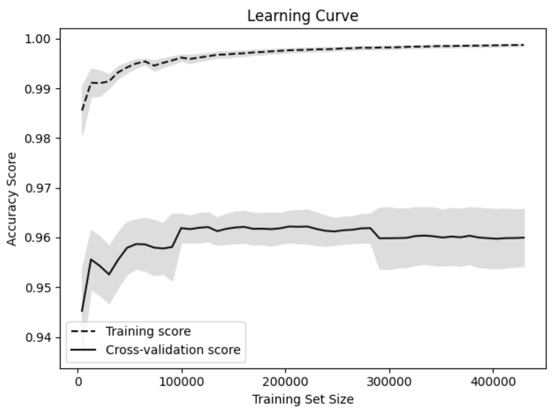


Figure 4.12: KNN Learning Curve

4.1.3 XGBoost

XGBoost, a leading machine learning algorithm for regression and classification problems, has shown excellent performance in our study with an accuracy of 96%. This performance is noteworthy, especially considering the model's ability to handle complex data patterns across varied datasets. Here are the detailed results:

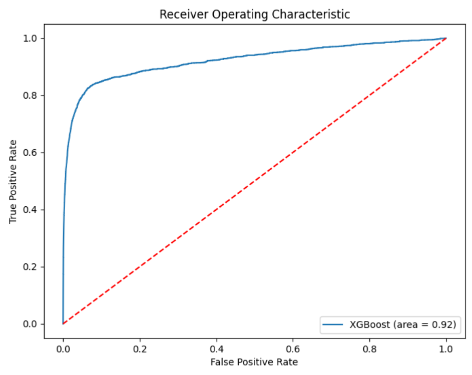


Figure 4.13: XGBoost ROC Curve

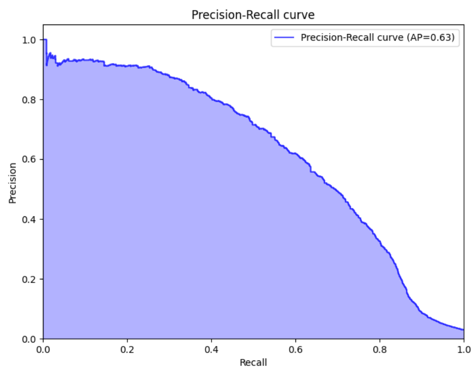


Figure 4.14: XGBoost Precision and Recall Curve

- **Accuracy:** 96.1433%, demonstrating robust overall performance and effective handling of the dataset.
- **Parameter Optimization:** The model underwent a rigorous tuning process with 3-fold cross-validation across 48 candidate configurations, totaling 144 fits. This thorough optimization helped identify the best parameters to maximize the model's performance, which were found to be:
  1. **Gamma:** 0, minimizing regularization and allowing for more complexity in the decision boundaries.
  2. **Learning Rate:** 0.3, offering a balance between speed of convergence and accuracy.
  3. **Max Depth:** 10, allowing the model to capture more complex patterns without becoming overly specific to the training data.
  4. **Min Child Weight:** 1, which represents the minimum sum of instance weight required in a child node to prevent overfitting.
  5. **N Estimators:** 150, defining the number of trees in the forest, optimized for both performance and computational efficiency.

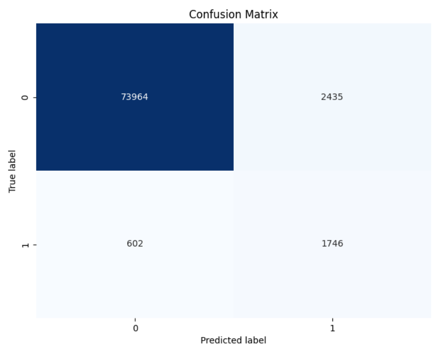


Figure 4.15: XGBoost Confusion Matrix

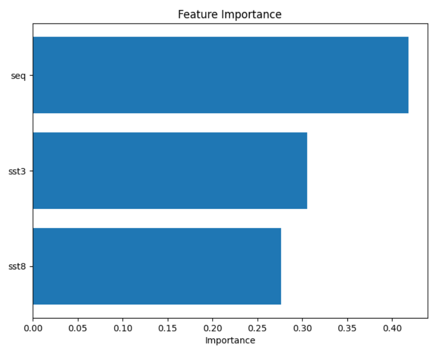
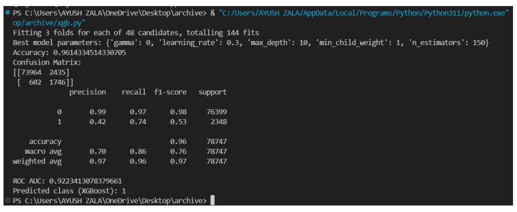


Figure 4.16: XGBoost Feature Importance

The Precision and Recall Analysis for XGBoost includes:

- **Class 0 (Negative Class):** A high precision of 0.99 demonstrates the model's exceptional accuracy in correctly identifying true negatives. This nearly perfect precision score means that there are almost no false positives. The recall of 0.97 indicates that the model successfully identifies 97% of all actual negative instances. This nearly perfect precision score means that there are almost no false positives

- **Class 1 (Positive Class):** The precision drops to 0.42, showing that when XGBoost predicts an instance as positive, it is only correct about 42% of the time. This lower precision suggests a susceptibility to false positives. With a recall of 0.74, the model is able to identify approximately 74% of all actual positive instances.



```
PS C:\Users\AYUSH ZALA\OneDrive\Desktop\archive> & "C:/Users/AYUSH ZALA/AppData/Local/Programs/Python/Python311/python.exe"
op/archive/xgb.py"
Fitting 3 folds for each of 48 candidates, totalling 144 fits
Best model parameters: {'gamma': 0, 'learning_rate': 0.3, 'max_depth': 10, 'min_child_weight': 1, 'n_estimators': 150}
Accuracy: 0.9614334514330705
Confusion Matrix:
[[73964  2435]
 [  602  1746]]
              precision    recall  f1-score   support

           0       0.99      0.97      0.98     76399
           1       0.42      0.74      0.53      2348

    accuracy                           0.96     78747
   macro avg       0.70      0.86      0.76     78747
weighted avg       0.97      0.96      0.97     78747

ROC AUC: 0.9223413078379661
Predicted class (XGBoost): 1
PS C:\Users\AYUSH ZALA\OneDrive\Desktop\archive>
```

Figure 4.17: XGBoost Classification Report

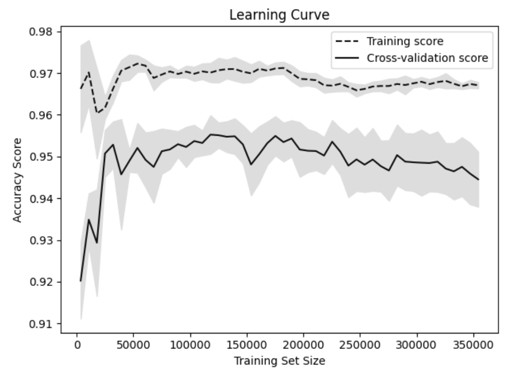


Figure 4.18: XGBoost Learning Curve

4.1.4 Random Forest

Random Forest, a powerful ensemble learning technique used for classification and regression tasks, has demonstrated exceptional performance in our analysis with an accuracy of 98%. This high level of accuracy underscores the model's efficacy in managing and analyzing large and diverse datasets. Here are the detailed results:

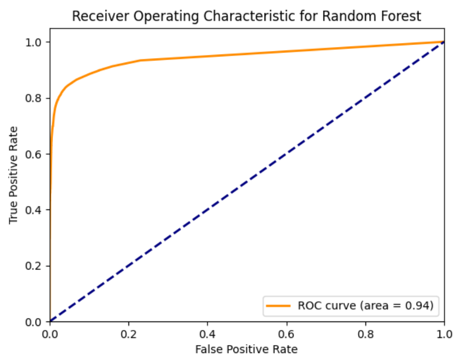


Figure 4.19: Random Forest ROC Curve

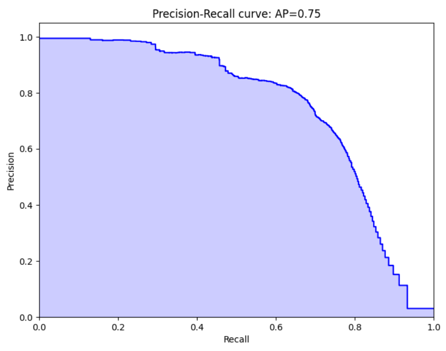


Figure 4.20: Random Forest Precision and Recall Curve

- **Accuracy:** 98.4334%, indicating superior overall performance, particularly in identifying and classifying instances accurately.

- **Parameter Optimization:** The Random Forest model's performance was optimized using a 3-fold cross-validation approach over 6 different parameter configurations, resulting in a total of 18 model fits. This optimization strategy helped pinpoint the most effective parameters for maximizing the predictive accuracy of the model, which included:

  1. **Max Depth:** 'None', permitting the trees within the forest to grow until all leaves are pure or until all leaves contain fewer than the minimum split samples.

  2. **N Estimators:** 200, indicating the number of trees in the forest. A greater number of estimators typically enhances performance but also raises computational demand and processing time.

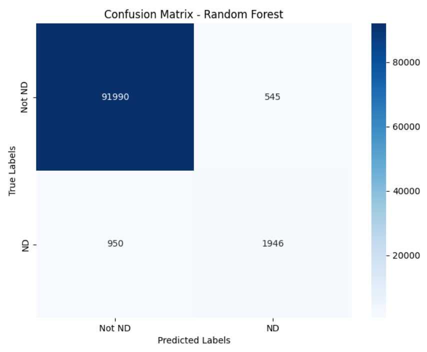


Figure 4.21: Random Forest Confusion Matrix

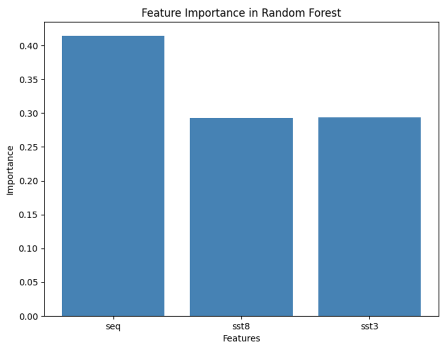
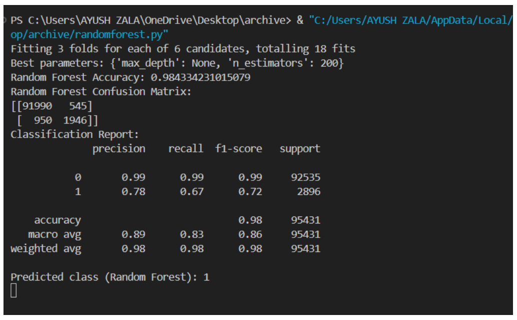


Figure 4.22: Random Forest Feature Importance

The Precision and Recall Analysis for Random Forest includes:

- **Class 0 (Negative Class):** A high precision of 0.99 demonstrates the model's exceptional accuracy in correctly identifying true negatives. The recall of 0.99 similarly indicates that the model successfully identifies 99% of all actual negative instances.
- **Class 1 (Positive Class):** The precision drops to 0.78, showing that when Random Forest predicts an instance as positive, it is correct approximately 78% of the time. With a recall of 0.67, the model is able to identify about 67% of all actual positive instances.



```
PS C:\Users\AYUSH ZALA\OneDrive\Desktop\archive> & "C:/Users/AYUSH ZALA/AppData/Local/
op/archive/randomforest.py"
Fitting 3 folds for each of 6 candidates, totalling 18 fits
Best parameters: {'max_depth': None, 'n_estimators': 200}
Random Forest Accuracy: 0.984334231015079
Random Forest Confusion Matrix:
[[91990   545]
 [  950  1946]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     92535
           1       0.78      0.67      0.72      2896

    accuracy                           0.98     95431
   macro avg       0.89      0.83      0.86     95431
weighted avg       0.98      0.98      0.98     95431

Predicted class (Random Forest): 1
```

Figure 4.23: Random Forest Classification Report

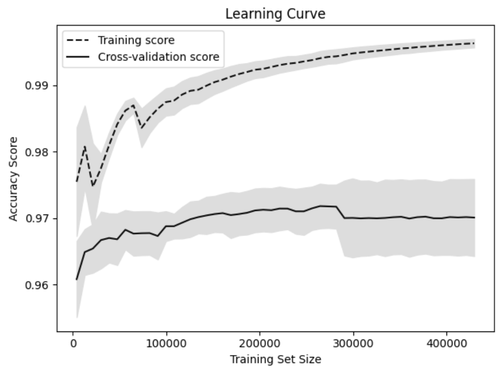


Figure 4.24: Random Forest Learning Curve

4.1.5 Bidirectional Long Short-Term Memory (BiLSTM)

Bidirectional Long Short-Term Memory (BiLSTM) network has demonstrated exceptional performance, achieving an accuracy of 95%. This accuracy underscores the model's robust ability to capture and leverage the temporal and sequential dependencies present within the dataset.

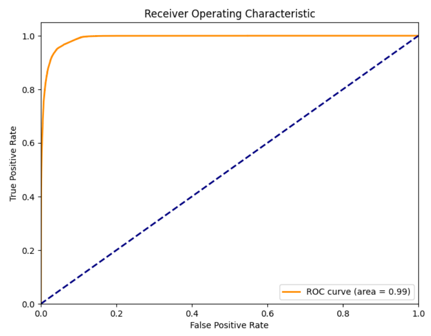


Figure 4.25: BiLSTM ROC Curve

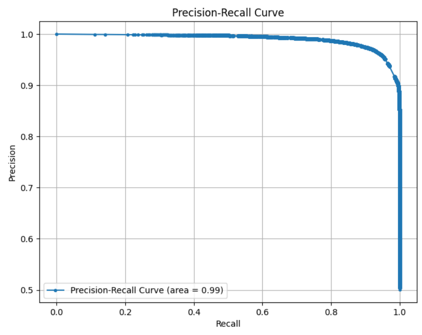


Figure 4.26: BiLSTM Precision and Recall Curve

The model configuration for this study included a bidirectional layer that processes data in both forward and reverse directions. This layer is followed by multiple dense layers, which contribute to the model's ability to learn non-linear relationships in the data effectively. The entire network was trained over 10 epochs, utilizing a total of 23,150 data points.

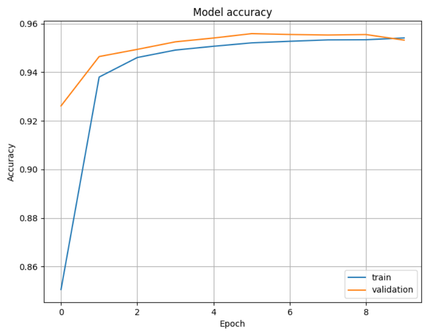


Figure 4.27: BiLSTM Model Accuracy

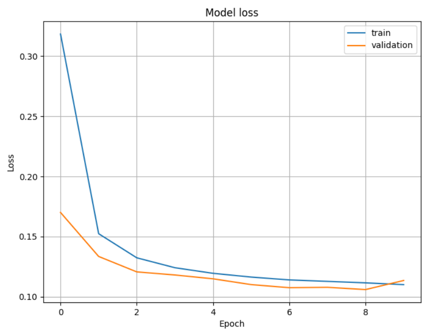


Figure 4.28: BiLSTM Model Loss

From a technical perspective, the BiLSTM network was composed of 84,800 trainable parameters. This considerable number of parameters indicates a deep and complex model capable of capturing intricate structures in the data. Importantly, all parameters in the model were trainable, which means the network did not rely on any non-trainable or pre-defined features, ensuring that all aspects of the learning process were derived from the input data itself.

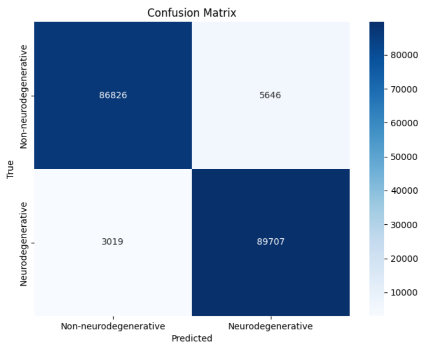
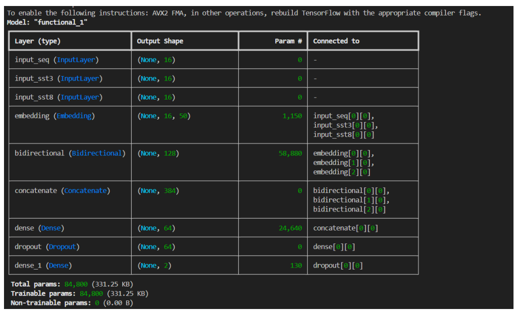


Figure 4.29: BiLSTM Confusion Matrix



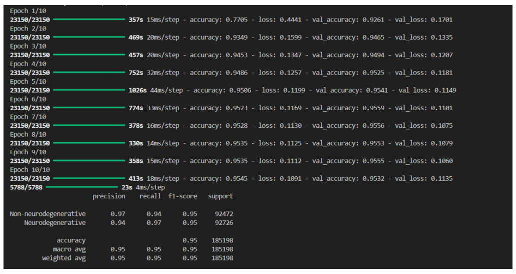
```
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
Model: "functional_1"
┌───────────────────────────────┬──────────────────┬───────────┬────────────────────────┐
│ Layer (type)                  │ Output Shape     │   Param # │ Connected to           │
├───────────────────────────────┼──────────────────┼───────────┼────────────────────────┤
│ input_seq (InputLayer)        │ (None, 16)       │         0 │ -                      │
├───────────────────────────────┼──────────────────┼───────────┼────────────────────────┤
│ input_sst3 (InputLayer)       │ (None, 16)       │         0 │ -                      │
├───────────────────────────────┼──────────────────┼───────────┼────────────────────────┤
│ input_sst8 (InputLayer)       │ (None, 16)       │         0 │ -                      │
├───────────────────────────────┼──────────────────┼───────────┼────────────────────────┤
│ embedding (Embedding)         │ (None, 16, 50)   │     1,150 │ input_seq[0][0],       │
│                               │                  │           │ input_sst3[0][0],      │
│                               │                  │           │ input_sst8[0][0]       │
├───────────────────────────────┼──────────────────┼───────────┼────────────────────────┤
│ bidirectional (Bidirectional) │ (None, 128)      │    58,880 │ embedding[0][0],       │
│                               │                  │           │ embedding[1][0],       │
│                               │                  │           │ embedding[2][0]        │
├───────────────────────────────┼──────────────────┼───────────┼────────────────────────┤
│ concatenate (Concatenate)     │ (None, 384)      │         0 │ bidirectional[0][0],   │
│                               │                  │           │ bidirectional[1][0],   │
│                               │                  │           │ bidirectional[2][0]    │
├───────────────────────────────┼──────────────────┼───────────┼────────────────────────┤
│ dense (Dense)                 │ (None, 64)       │    24,640 │ concatenate[0][0]      │
├───────────────────────────────┼──────────────────┼───────────┼────────────────────────┤
│ dropout (Dropout)             │ (None, 64)       │         0 │ dense[0][0]            │
├───────────────────────────────┼──────────────────┼───────────┼────────────────────────┤
│ dense_1 (Dense)               │ (None, 2)        │       130 │ dropout[0][0]          │
└───────────────────────────────┴──────────────────┴───────────┴────────────────────────┘
 Total params: 84,800 (331.25 KB)
 Trainable params: 84,800 (331.25 KB)
 Non-trainable params: 0 (0.00 B)
```

Figure 4.30: BiLSTM Model Summary

The BiLSTM model comprises an Embedding layer with 1,150 parameters, a Bidirectional LSTM layer with 58,880 parameters for capturing sequential patterns, a Dense layer with 24,640 parameters for feature interpretation, and a final Output Dense layer with 130 parameters for classification. The precision and recall metrics further illustrate the model's effectiveness:

- **Class 0 (Negative Class):** The model achieved a high precision of 0.97, indicating that it is highly accurate in identifying true negative instances, with a very low rate of false positives. The recall of 0.94 for this class demonstrates that the model is also effective in capturing the majority of actual negative cases, missing very few.

- **Class 1 (Positive Class):** Precision for the positive class was impressively high at 0.94, suggesting that when the model predicts an instance as positive, it is correct most of the time. The recall of 0.97 in this class indicates an excellent ability to identify almost all true positive instances.



```
Epoch 1/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 357s 15ms/step - accuracy: 0.7705 - loss: 0.4441 - val_accuracy: 0.9261 - val_loss: 0.1701
Epoch 2/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 469s 20ms/step - accuracy: 0.9349 - loss: 0.1599 - val_accuracy: 0.9465 - val_loss: 0.1335
Epoch 3/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 457s 20ms/step - accuracy: 0.9453 - loss: 0.1347 - val_accuracy: 0.9494 - val_loss: 0.1207
Epoch 4/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 752s 32ms/step - accuracy: 0.9486 - loss: 0.1257 - val_accuracy: 0.9525 - val_loss: 0.1181
Epoch 5/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 1026s 44ms/step - accuracy: 0.9506 - loss: 0.1199 - val_accuracy: 0.9541 - val_loss: 0.1149
Epoch 6/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 774s 33ms/step - accuracy: 0.9523 - loss: 0.1169 - val_accuracy: 0.9559 - val_loss: 0.1101
Epoch 7/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 378s 16ms/step - accuracy: 0.9528 - loss: 0.1130 - val_accuracy: 0.9556 - val_loss: 0.1075
Epoch 8/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 330s 14ms/step - accuracy: 0.9535 - loss: 0.1125 - val_accuracy: 0.9553 - val_loss: 0.1079
Epoch 9/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 358s 15ms/step - accuracy: 0.9535 - loss: 0.1112 - val_accuracy: 0.9555 - val_loss: 0.1060
Epoch 10/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 413s 18ms/step - accuracy: 0.9545 - loss: 0.1091 - val_accuracy: 0.9532 - val_loss: 0.1135
5788/5788 ━━━━━━━━━━━━━━━━━━━━ 23s 4ms/step
                      precision    recall  f1-score   support

Non-neurodegenerative       0.97      0.94      0.95     92472
    Neurodegenerative       0.94      0.97      0.95     92726

             accuracy                           0.95    185198
            macro avg       0.95      0.95      0.95    185198
         weighted avg       0.95      0.95      0.95    185198
```

Figure 4.31: BiLSTM Model Report

4.1.6 Convolutional Neural Network (CNN)

In the context of exploring the adaptability of Convolutional Neural Networks (CNNs) beyond their traditional domain of image processing, this study demonstrates their efficacy in handling complex pattern recognition tasks in sequential data. The CNN model achieved an accuracy of 91%, validating its capability to effectively capture spatial and structural patterns not just in images, but across diverse data types.

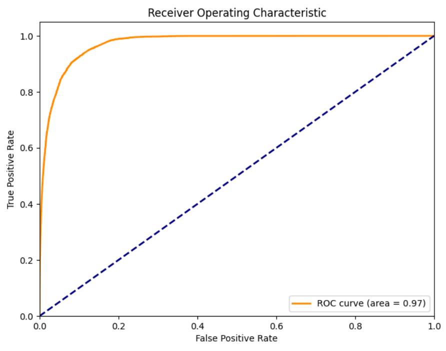


Figure 4.32: CNN ROC Curve

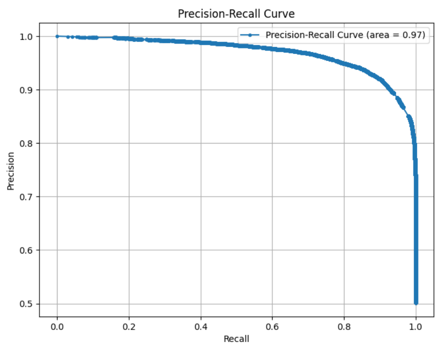


Figure 4.33: CNN Precision and Recall Curve

The network's architecture was designed to include a convolutional layer, which is essential for detecting local conjunctions of features and spatial hierarchies in data. This convolutional base was followed by a global max pooling layer, which helps to reduce the dimensionality of the data, thus minimizing the number of parameters and computation required. The entire model was subjected to training over 10 epochs using a dataset comprising 23,150 entries, providing a robust training environment that facilitated significant learning and model refinement.

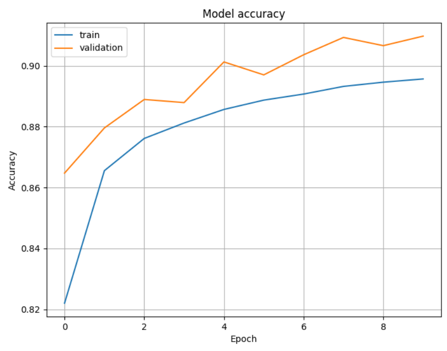


Figure 4.34: CNN Model Accuracy

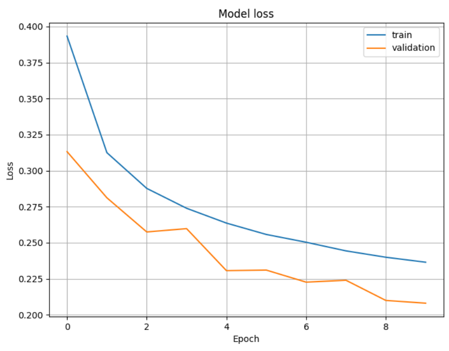


Figure 4.35: CNN Model Loss

From a parameter perspective, the model is relatively compact, consisting of 23,296 trainable parameters. This lower count of parameters, compared to more complex model like BiLSTM, indicates a streamlined model that balances performance with computational efficiency—important in scenarios where resource constraints are a consideration.

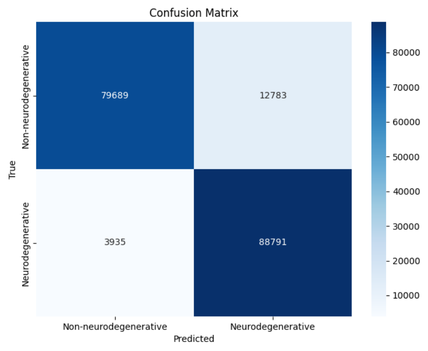
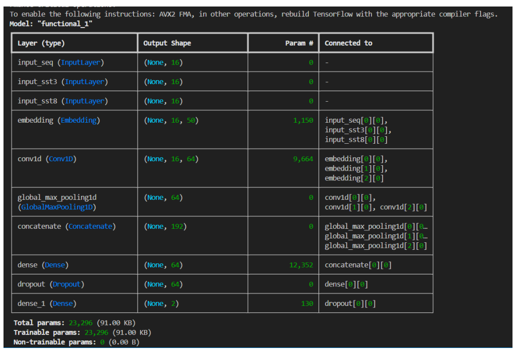


Figure 4.36: CNN Confusion Matrix

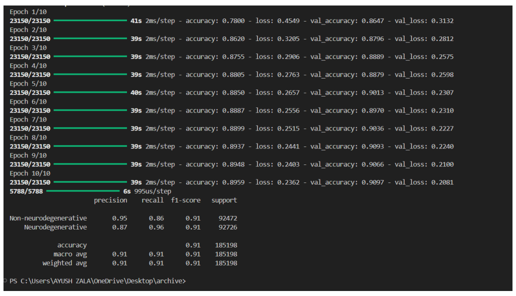


```
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
Model: "functional_1"
┌──────────────────────────────┬────────────────────┬───────────┬──────────────────────────────┐
│ Layer (type)                 │ Output Shape       │   Param # │ Connected to                 │
├──────────────────────────────┼────────────────────┼───────────┼──────────────────────────────┤
│ input_seq (InputLayer)       │ (None, 16)         │         0 │ -                            │
├──────────────────────────────┼────────────────────┼───────────┼──────────────────────────────┤
│ input_sst3 (InputLayer)      │ (None, 16)         │         0 │ -                            │
├──────────────────────────────┼────────────────────┼───────────┼──────────────────────────────┤
│ input_sst8 (InputLayer)      │ (None, 16)         │         0 │ -                            │
├──────────────────────────────┼────────────────────┼───────────┼──────────────────────────────┤
│ embedding (Embedding)        │ (None, 16, 50)     │     1,150 │ input_seq[0][0],             │
│                              │                    │           │ input_sst3[0][0],            │
│                              │                    │           │ input_sst8[0][0]             │
├──────────────────────────────┼────────────────────┼───────────┼──────────────────────────────┤
│ conv1d (Conv1D)              │ (None, 16, 64)     │     9,664 │ embedding[0][0],             │
│                              │                    │           │ embedding[1][0],             │
│                              │                    │           │ embedding[2][0]              │
├──────────────────────────────┼────────────────────┼───────────┼──────────────────────────────┤
│ global_max_pooling1d         │ (None, 64)         │         0 │ conv1d[0][0],                │
│ (GlobalMaxPooling1D)         │                    │           │ conv1d[1][0], conv1d[2][0]   │
├──────────────────────────────┼────────────────────┼───────────┼──────────────────────────────┤
│ concatenate (Concatenate)    │ (None, 192)        │         0 │ global_max_pooling1d[0][0…   │
│                              │                    │           │ global_max_pooling1d[1][0…   │
│                              │                    │           │ global_max_pooling1d[2][0]   │
├──────────────────────────────┼────────────────────┼───────────┼──────────────────────────────┤
│ dense (Dense)                │ (None, 64)         │    12,352 │ concatenate[0][0]            │
├──────────────────────────────┼────────────────────┼───────────┼──────────────────────────────┤
│ dropout (Dropout)            │ (None, 64)         │         0 │ dense[0][0]                  │
├──────────────────────────────┼────────────────────┼───────────┼──────────────────────────────┤
│ dense_1 (Dense)              │ (None, 2)          │       130 │ dropout[0][0]                │
└──────────────────────────────┴────────────────────┴───────────┴──────────────────────────────┘
 Total params: 23,296 (91.00 KB)
 Trainable params: 23,296 (91.00 KB)
 Non-trainable params: 0 (0.00 B)
```

Figure 4.37: CNN Model Summary

The CNN model features an Embedding layer with 1,150 parameters, a Convolutional layer with 9,664 parameters for extracting spatial features, a Dense layer with 12,352 parameters for processing those features, and a final Output Dense layer with 130 parameters for classification. The performance of the model in terms of precision and recall is notably commendable:

- **Class 0 (Negative Class):** The precision of 0.95 indicates that the model is highly accurate when it predicts negative instances, with few false positives. However, the recall of 0.86 suggests that while it captures a majority of the negative cases, there is still room for improvement in identifying all actual negatives.
- **Class 1 (Positive Class):** For the positive class, the precision of 0.87 shows that there are some instances of false positives, though the figure remains high. The recall of 0.96 is particularly impressive, indicating that the model is highly effective at identifying almost all true positive instances.



```
Epoch 1/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 41s 2ms/step - accuracy: 0.7800 - loss: 0.4549 - val_accuracy: 0.8647 - val_loss: 0.3132
Epoch 2/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 39s 2ms/step - accuracy: 0.8620 - loss: 0.3205 - val_accuracy: 0.8796 - val_loss: 0.2812
Epoch 3/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 39s 2ms/step - accuracy: 0.8755 - loss: 0.2906 - val_accuracy: 0.8889 - val_loss: 0.2575
Epoch 4/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 39s 2ms/step - accuracy: 0.8805 - loss: 0.2763 - val_accuracy: 0.8879 - val_loss: 0.2598
Epoch 5/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 40s 2ms/step - accuracy: 0.8850 - loss: 0.2657 - val_accuracy: 0.9013 - val_loss: 0.2307
Epoch 6/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 39s 2ms/step - accuracy: 0.8887 - loss: 0.2556 - val_accuracy: 0.8970 - val_loss: 0.2310
Epoch 7/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 39s 2ms/step - accuracy: 0.8899 - loss: 0.2515 - val_accuracy: 0.9036 - val_loss: 0.2227
Epoch 8/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 39s 2ms/step - accuracy: 0.8937 - loss: 0.2441 - val_accuracy: 0.9093 - val_loss: 0.2240
Epoch 9/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 39s 2ms/step - accuracy: 0.8948 - loss: 0.2403 - val_accuracy: 0.9066 - val_loss: 0.2100
Epoch 10/10
23150/23150 ━━━━━━━━━━━━━━━━━━━━ 39s 2ms/step - accuracy: 0.8959 - loss: 0.2362 - val_accuracy: 0.9097 - val_loss: 0.2081
5788/5788 ━━━━━━━━━━━━━━━━━━━━ 6s 995us/step
                      precision    recall  f1-score   support

Non-neurodegenerative       0.95      0.86      0.91     92472
    Neurodegenerative       0.87      0.96      0.91     92726

             accuracy                           0.91    185198
            macro avg       0.91      0.91      0.91    185198
         weighted avg       0.91      0.91      0.91    185198

PS C:\Users\AYUSH ZALA\OneDrive\Desktop\archive>
```

Figure 4.38: CNN Model Report

4.2 Result Analysis

- **Logistic Regression:**
  - **Performance:** Logistic Regression showed relatively low accuracy, indicating that it struggles with this particular dataset or problem type.
  - **Improvements:** This model could benefit greatly from advanced feature engineering to better capture nonlinear relationships within the data. Applying regularization techniques like L1 or L2 might also help to improve overfitting and model generalization by penalizing overly complex models.

- **K-Nearest Neighbors (KNN):**
  - **Performance:** Achieved high accuracy, especially in classifying the negative class, but struggled with precision and recall in the positive class.
  - **Improvements:** Adjusting the number of neighbors based on validation performance could help, as well as exploring different distance metrics to better handle minority classes. Weighted distances might offer a method to give more importance to nearer neighbors.
- **XGBoost:**
  - **Performance:** Showed robust performance with notable precision in the negative class but lower precision for the positive class.
  - **Improvements:** Further tuning of hyperparameters like gamma and learning rate could be beneficial. Additionally, implementing cost-sensitive learning could address the imbalance towards the negative class.
- **Random Forest:**
  - **Performance:** Offered high accuracy overall with excellent control over overfitting, though it underperformed in detecting the positive class.
  - **Improvements:** Techniques like Synthetic Minority Over-sampling Technique (SMOTE) could be utilized to balance class distribution. Adjusting tree-specific parameters like max depth and minimum samples per leaf could also be beneficial.
- **Bidirectional Long Short-Term Memory (BiLSTM):**
  - **Performance:** Excels in capturing temporal and sequential dependencies, with exceptional precision and recall for both classes.
  - **Improvements:** Increasing the dataset size and running more training epochs could potentially boost its ability to learn even more complex patterns. Also, experimenting with different architectures and dropout rates might improve performance

- **Convolutional Neural Network (CNN):**
  - **Performance:** Though traditionally used for image data, it adapted well to sequential data, showing good precision and excellent recall for the positive class.
  - **Improvements:** Implementing different types of layers, such as additional convolutional or pooling layers, could enhance its ability to capture more complex spatial relationships. Experimenting with larger filter sizes or deeper networks might also help in capturing more detailed features.

Chapter 5

Conclusions And Future Work

5.1 Conclusion

The exploration of different machine learning models in this study has provided valuable insights into their performance and applicability across various metrics. Each model has demonstrated unique strengths and limitations depending on the complexity and nature of the data being analyzed.Logistic Regression demonstrated a basic performance with an accuracy of 52%, indicating it is better suited for simpler, linearly separable datasets. To enhance its utility, the application of more sophisticated feature engineering or regularization techniques could be considered to improve model accuracy and prevent overfitting.The K-Nearest Neighbors model demonstrated strong overall accuracy at 98%, excelling in one class but less so in another. Future improvements could include adjusting the k-value, exploring different distance weighting methods, and enhancing techniques for managing imbalanced datasets to boost its performance across various classes.XGBoost's accuracy of 96% underscores its effectiveness in managing complex data relationships. The precision in the positive class could be enhanced, possibly by revising the model complexity or further tuning hyperparameters to prevent potential overfitting. Additionally, more extensive use of cross-validation could help refine its performance evaluation.The Random Forest model excelled with a robust accuracy of 98%, especially proficient in classifying the negative class with very high precision and recall. The model underperformed for the positive class, suggesting a need for further tuning to balance its predictive capability across classes. Techniques such as adjusting class weights or exploring synthetic data generation methods like SMOTE for balancing classes could be useful.The BiLSTM network, with an accuracy of 95%, demonstrated its strength in capturing temporal sequences effectively, making it ideal for time-series analysis. The model showed excellent balance in precision and recall across classes. Future improvements could include increasing the model complexity slightly or experimenting with different network architectures to further boost its learning capacity.CNN achieved an accuracy of 91%, proving effective in recognizing spatial and structural patterns in a variety of data types. To improve the recall for the negative class, variations in network architecture, such as different pooling layers or convolutional configurations, could be explored. Employing techniques like transfer learning might also enhance performance, especially in data-constrained environments.

5.2 Future Work

To build on the current research, several avenues are proposed for future work:

1. **Improving Class Imbalance Handling:**
   Further research could investigate advanced strategies for addressing class imbalance, including synthetic data generation (SMOTE), sophisticated ensemble methods, or alternative loss functions that impose heavier penalties for misclassifying the minority class.

2. **Hybrid Models:**
   Developing hybrid models that combine the strengths of two or more of the studied models could potentially leverage the advantages of each. For instance, integrating BiLSTM with CNN could enhance feature extraction capabilities in complex datasets with both spatial and temporal dimensions.

3. **Deployment and Real-world Testing:**
   Future work should also include the deployment of these models in real-world environments to test their robustness, scalability, and adaptability. This includes monitoring model drift over time and updating models accordingly.

4. **Expanding Data Sources:**
   Incorporating more diverse datasets or increasing data dimensionality could provide a broader basis for testing the generalizability of the models. Additionally, multi-source data integration could be explored to enrich the models' training environments.

5. **Exploration of Newer Models and Techniques:**
   The field of machine learning is continually evolving. Future studies should consider evaluating newer models or emerging techniques that might offer improvements over the current state-of-the-art, particularly those that can automate parts of the model tuning and feature selection process.

Bibliography

[1] Z. Aydin, Y. Altunbasak, and H. Erdogan. Bayesian models and algorithms for protein -sheet prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(2):395–409, 2011.

[2] A. Delacourte. The natural and molecular history of alzheimer's disease. *Journal of Alzheimer's Disease*, 9(s3):187–194, 2006.

[3] A. Farooq, S. Anwar, M. Awais, and S. Rehman. A deep cnn based multi-class classification of alzheimer's disease using mri. In *2017 IEEE International Conference on Imaging systems and techniques (IST)*, pages 1–6. IEEE, 2017.

[4] N. S. Gnanadesigan, N. Dhanasegar, M. D. Ramasamy, S. Muthusamy, O. P. Mishra, G. K. Pugalendhi, S. C. M. Sundararajan, and A. Ravindaran. An integrated network topology and deep learning model for prediction of alzheimer disease candidate genes. *Soft Computing*, 27(19):14189–14203, 2023.

[5] G. G.S and N. S. A comprehensive review on early diagnosis of alzheimer's disease detection. In *2023 International Conference on Research Methodologies in Knowledge Management, Artificial Intelligence and Telecommunication Engineering (RMKMATE)*, pages 1–6, 2023.

[6] G. Gupta, N. Gupta, A. Gupta, P. Vaidya, G. K. Singh, and V. Jaiswal. Prediction of alzheimer associated proteins (paap): a perspective to understand alzheimer disease for therapeutic design. *International Journal of Bioinformatics Research and Applications*, 17(4):363–374, 2021.

[7] S. Iyer. Identify protein disorder from amino acid sequences with machine learning. In *2021 IEEE International Conference on Electro Information Technology (EIT)*, pages 429–436, 2021.

[8] M. Kolarova, F. García-Sierra, A. Bartos, J. Ricny, D. Ripova, et al. Structure and pathology of tau protein in alzheimer disease. *International journal of Alzheimer's disease*, 2012, 2012.

[9] M. G. Krokidis, G. N. Dimitrakopoulos, T. P. Exarchos, A. G. Vrahatis, and P. Vlamos. Advanced big data analysis for deciphering the role of protein misfolding and interactions in the pathogenesis of alzheimer's disease. In *2023 IEEE International Conference on Big Data (BigData)*, pages 4614–4617, 2023.

[10] J. Orpiszewski, N. Schormann, B. Kluve-Beckerman, J. J. Liepnieks, and M. D. Benson. Protein aging hypothesis of alzheimer disease. *The FASEB Journal*, 14(9):1255–1263, 2000.

[11] Research Collaboratory for Structural Bioinformatics (RCSB). Rcsb protein data bank. Available: https://www.rcsb.org/. Accessed: 2024-05-13.

[12] R. Roychaudhuri, M. Yang, M. M. Hoshi, and D. B. Teplow. Amyloid β-protein assembly and alzheimer disease. *Journal of Biological Chemistry*, 284(8):4749–4753, 2009.

[13] A. W. Salehi, P. Baglat, B. B. Sharma, G. Gupta, and A. Upadhya. A cnn model: earlier diagnosis and classification of alzheimer disease using mri. In *2020 International Conference on Smart Electronics and Communication (ICOSEC)*, pages 156–161. IEEE, 2020.

[14] M. A. Sofi and M. ArifWani. Improving prediction of amyloid proteins using secondary structure based alignments and segmented-pssm. In *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 87–92, 2021.

[15] L. Sun, Q. Peng, C. Qian, and J. Li. Tau content prediction based on brain network mlp-att model. In *2022 International Conference on Machine Learning, Cloud Computing and Intelligent Mining (MLCCIM)*, pages 353–358, 2022.

[16] UniProt Consortium. Uniprot: a worldwide hub of protein knowledge. Available: https://www.uniprot.org/. Accessed: 2024-05-13.

[17] M. Yildirim and A. Cinar. Classification of alzheimer's disease mri images with cnn based hybrid method. *Ingénierie des Systèmes d Inf.*, 25(4):413–418, 2020.