Multi-Algorithmic Retinal Genetic Disease Detection using pupillometry

A Project Report Submitted in the
Partial Fulfillment of the Requirements
for the Award of the Degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

Shruti Gajre	**20881A0455**
B. Akhil	**20881A0412**
N. Koushik Reddy	**20881A0440**

SUPERVISOR
Dr. A. Jaya Lakshmi
Assistant Professor

Department of Electronics and Communication Engineering

VARDHAMAN COLLEGE OF ENGINEERING
(AUTONOMOUS)
Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC with A++ Grade, ISO 9001:2015 Certified
Kacharam, Shamshabad, Hyderabad – 501218, Telangana, India

April, 2024

Department of Electronics and Communication Engineering

CERTIFICATE

This is to certify that the project titled **Multi-Algorithmic Retinal Genetic Disease Detection using pupillometry** is carried out by

Shruti Gajre	**20881A0455**
B. Akhil	**20881A0412**
N. Koushik Reddy	**20881A0440**

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Electronics and Communication Engineering** during the year 2023-24.

Signature of the Supervisor	**Signature of the HOD**
Dr. A. Jaya Lakshmi	**Dr. P. Nageswara Rao**
Assistant Professor	**Professor and Head, ECE**

Project Viva-Voce held on ________________________

Examiner

Acknowledgement

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

We wish to express our deep sense of gratitude to **Dr. A. Jaya Lakshmi**, Assistant Professor and Project Supervisor, Department of Electronics and Communication Engineering, Vardhaman College of Engineering, for her able guidance and useful suggestions, which helped us in completing the project in time.

We are particularly thankful to **Dr. P. Nageswara Rao**, the Head of the Department, Department of Electronics and Communication Engineering, his guidance, intense support and encouragement, which helped us to mould our project into a successful one.

We show gratitude to our honorable Principal **Dr. J.V.R. Ravindra**, for providing all facilities and support.

We avail this opportunity to express our deep sense of gratitude and heartful thanks to **Dr. Teegala Vijender Reddy**, Chairman and **Sri Teegala Upender Reddy**, Secretary of VCE, for providing a congenial atmosphere to complete this project successfully.

We also thank all the staff members of Electronics and Communication Engineering department for their valuable support and generous advice. Finally thanks to all our friends and family members for their continuous support and enthusiastic help.

Shruti Gajre

B. Akhil

N. Koushik Reddy

Abstract

Children with inherited retinal disorders suffer from significant visual deficits. These are categorized as anomalies of the outer and inner retinas usually cause blindness in youngsters. There are several different clinical and genetic aspects (about 200 causal genes) that make this type of illness difficult to diagnose. It is typically based on a complex series of clinical examinations, some of which are invasive and may not be appropriate for young children. Now, a fresh technique is necessary, one that employs Chromatic Pupillometry, a technology that is gaining popularity for assessing both inner and outer retina function. The innovative Clinical Decision Support System (CDSS) presented in this work uses chromatic pupillometry to increase the accuracy of the diagnosis of infantile reflux disease (IRD).A unique machine learning system that employs SVM, KNN, Random Forest, Gradient Boosting, LSTM, and Bi-LSTM is combined with a special pupillometer in the suggested method. The method offers a non-invasive, kid-friendly substitute for invasive diagnostics, thereby addressing its limitations. The CDSS demonstrated good diagnostic accuracy in a thorough evaluation, suggesting that it could be a valuable tool in pediatric ophthalmology. Early treatment of children with severe vision impairments is made possible by this comprehensive technique, which can detect IRDs more accurately and rapidly.

Keywords: Inherited Retinal Disorder, Retinitis Pigmentosa, Clinical Decision support system

Table of Contents

List of Tables

List of Figures

Abbreviations

Abbreviation	Description
CDSS	Clinical Decision Support System
IRD	Infantile Reflux Disease
SVM	Support Vector Machine
LSTM	Long Short Term Memory
Bi-LSTM	Bidirectional Long Short Term Memory

CHAPTER 1

Introduction

Especially in developed market nations, inherited retinal diseases (IRDs) pose a serious risk to children's vision and are a primary source of severe visual impairments. This class of diseases includes degenerations of both photoreceptors and retinal ganglion cells, leading to diseases including Leber hereditary optic neuropathy, Retinitis Pigmentosa, congenital glaucoma, dominant optic atrophy, and Stargardt disease. The startling 1 in 3000 people who are afflicted with IRDs in developed market economies underscores the critical need for thorough comprehension and efficient diagnostic techniques.

1.1 Motivation

The startling numbers pertaining to blindness in young people serve as a driving force behind exploring the complex field of inherited retinal diseases. One-third of these cases in developed market nations are caused by IRDs, highlighting the urgent need for targeted study and creative solutions. The effects that IRDs have on children's quality of life and physical health provide strong incentives to address the problems that these disorders present. IRDs require immediate attention and a concentrated effort to create efficient diagnostic and treatment methods since they have the potential to cause irreversible damage to both the outer and inner retina.

1.2 Problem Statement

With more than 200 causal genes found to far, the genetic variability of inherited retinal diseases is the main obstacle to treatment. This great genetic variety leads to diverse clinical symptoms originating from the same gene and complicates the diagnosis process. The fact that IRDs include both inner and outer retinal issues adds to the complexity and calls for a comprehensive approach to diagnosis and therapy. Although useful, the typical clinical evaluations are frequently intrusive and unsuitable for infants and young children, which makes it more difficult to diagnose these conditions in a timely manner. Furthermore, the requirement for anesthesia during pediatric electrophysiological assessments introduces an additional level of complexity that impairs the effectiveness of diagnostic processes. Therefore, it is imperative that these barriers be removed in order to enable early and precise diagnosis, improving the likelihood of successful therapeutic approaches.

1.3 Objective

This work aims to clarify the complex field of Inherited Retinal Diseases, concentrating on degenerations of the photoreceptors and retinal ganglion cells. This entails a thorough investigation of the genetic terrain with the goal of optimizing the diagnostic procedure through the identification of significant genetic markers and comprehension of their effects on clinical manifestations. Furthermore, by investigating non-invasive and kid-friendly diagnostic techniques, the research seeks to overcome the shortcomings of conventional clinical assessments, particularly in pediatric instances. Improving diagnosis speed and accuracy is just one goal; another is to open the door for therapeutic approaches that are most suited to the particular difficulties that children with IRDs present. In brief, this study aims to close the knowledge gap on the intricacy of inherited retinal diseases and the demand for effective diagnostic and treatment approaches, with a particular emphasis on the difficulties encountered in juvenile instances. The ultimate goal is to give a ray of hope to children with IRDs and their families by revealing the genetic foundations, refining diagnostic techniques, and enhancing therapy approaches, thereby reducing the negative effects of these disorders on the eyesight and general well-being of these children.

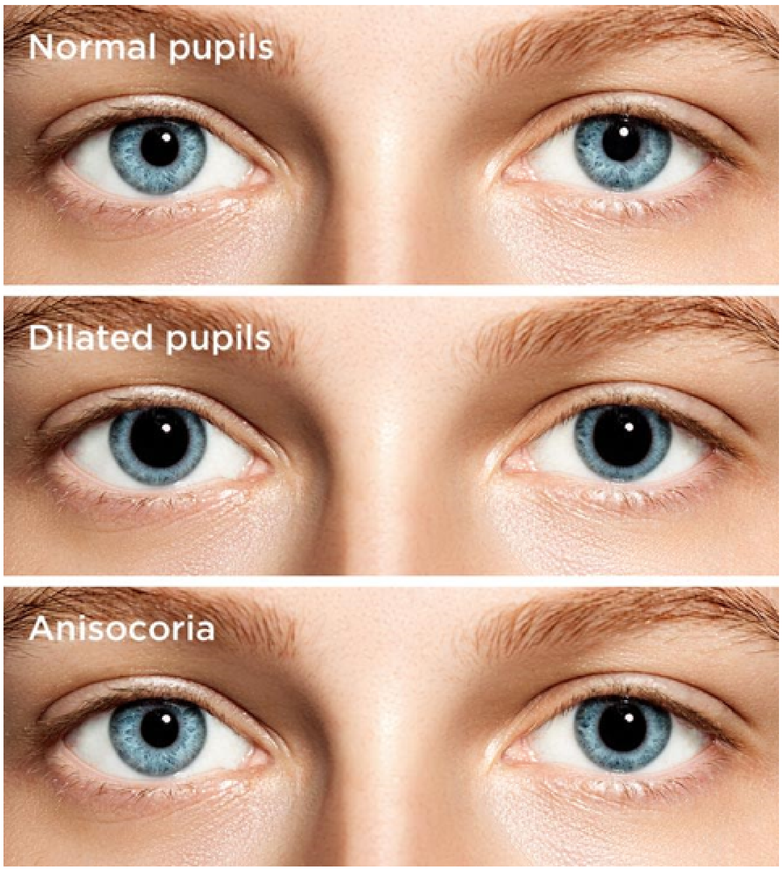


Figure 1.1: Variation of Pupil Size

1.4 Clinical Challenges in Diagnosis

Many difficulties arise in the diagnosis of IRDs, especially because these disorders often manifest in childhood. Newborns and early infants should not be exposed to the traditional clinical assessment methods, which are frequently intrusive and complex. This poses a significant obstacle to receiving prompt and correct diagnoses, which are essential for carrying out successful treatment plans. Furthermore, diverse and variable clinical manifestations resulting from the same causal gene can complicate diagnosis even more.

1.5 Pupillometery

Photoreceptor cells (rods and cones) have fast temporal kinetics and cause a rapid pupillary constriction in response to light, whereas inner retinal melanopsin-containing intrinsic photosensitive Retinal Ganglion Cells (ipRGCs) have slower temporal kinetics and elicit a sustained pupillary constriction to light stimuli that persists after light cessation.

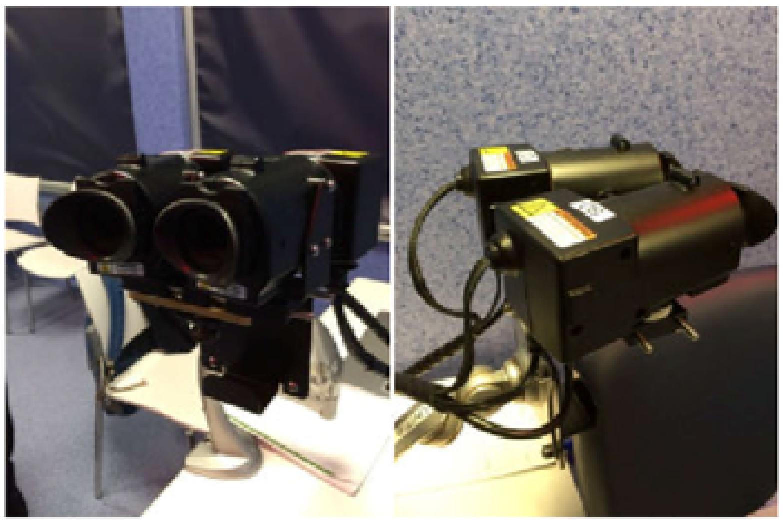


Figure 1.2: DP-2000 binocular pupillometer

The relative contributions of the three receptor types (rod, cone, and melanopsin photopigments) to the Pupillary Light Reflex (PLR) were investigated by varying the characteristics of large-field (90) flash stimuli and adaption conditions (light vs. dark).

1.6 The Need for Early Intervention

The creation of age and non-invasive diagnostic instruments is desperately needed, given the difficulties in identifying IRDs.Early intervention is critical for reducing the impact of these problems on a child's eyesight and overall development. The rapidly changing world of medical technology presents opportunities for novel diagnostic techniques that have the potential to completely transform the pediatric ophthalmology profession.

When it comes to pediatric visual impairments, inherited retinal diseases provide a significant difficulty due to their complex genetic makeup and wide range of clinical manifestations. These disorders are very common, which emphasizes the need for a concentrated effort to create precise, non-invasive, and age-appropriate diagnostic techniques. The objective is still the same as we work through the complexity of more than 200 causal genes and diverse phenotypes: to offer early and efficient therapies for young infants impacted by IRDs. By means of a cooperative strategy that incorporates developments in the fields of genetics, electrophysiology, and pediatric ophthalmology, we can endeavor to enhance the diagnostic environment and facilitate the development of optimum treatments for the youngest individuals affected by hereditary retinal disorders.

CHAPTER 2

Literature Survey

A significant increase in research has been conducted recently to utilize technology advancements to improve illness detection and diagnosis. This review of the literature looks at five different studies that showcase cutting-edge methods in medical technology, from sophisticated machine learning algorithms to cloud-based electronic medical records.

Huang's research delved into the genetic causes of blindness, examining 179 Chinese families with inherited retinal diseases (IRD) using advanced sequencing. They discovered 124 mutations in known retinal disease genes, with 55.3 percent being unique, revealing new insights into IRD complexity. The study identified novel phenotypic patterns and genotype-phenotype correlations, unveiling a potential new gene associated with IRD. These findings emphasize the significance of understanding genotype-phenotype relationships for tailored treatments and future therapies for IRD. [1]

Kardon et al. investigated polychromatic pupil responses, focusing on melanopsin-mediated activation over external photoreceptors. They explored how varying light parameters affect retinal ganglion cell activation in 43 healthy eyes and three cases of neuro-retinal vision loss. Using video cameras, they observed pupil movements during continuous Ganzfeld stimulation with red and blue light. Their findings suggest that pupil responses, especially to blue light, vary across individuals and can indicate phototransduction pathways involving rods, cones, or melanopsins.[2]

E. Iadanza et al.'s work presents a notable advancement in pediatric clinical decision support systems for Inherited Retinal diseases (IRDs), integrating polychromatic pupillometry with machine learning. By employing

Support Vector Machines (SVMs) to analyze pupillometric data, the method efficiently identifies features for diagnosing Retinitis Pigmentosa in pediatric patients. The ensemble model, combining two SVMs, demonstrates commendable performance metrics: specificity (0.786), sensitivity (0.846), and accuracy (0.937). This innovative approach marks the first utilization of machine learning techniques on pupillometric data for inherited retinal disease diagnosis in pediatric patients, promising significant advancements in pediatric ophthalmology.[3]

Addressing visual impairments in children with hereditary retinal disorders poses a significant challenge in pediatric ophthalmology. E.N. Vijaya Kumari underscores the necessity for advanced technological methods due to limitations in current clinical assessments. Chromatic Pupillometry stands out as a promising tool, assessing both internal and external retinal functions. Utilizing Support Vector Machine (SVM) technology and Clinical Decision Support System (CDSS) ensures a thorough and effective evaluation of visual impairments in these young patients.[4]

Recent advancements in deep literacy and image processing techniques have revolutionized early identification of eye diseases, enhancing medical diagnostics. Convolutional neural networks have notably improved performance in medical image processing, particularly in detecting eye complaints. This study delves into deep literacy strategies customized for eye disease detection, leveraging convolutional neural networks. Additionally, it evaluates existing datasets on color fundus retina, providing invaluable insights for researchers and highlighting gaps in current studies.[5]

In a study comparing individuals with inherited optic neuropathy (HON) to healthy controls, researchers found no significant differences in pupil responses to inner or external retinal photoreception. Despite marked visual impairment and optic atrophy in HON patients, no distinct disparities were observed in pupil reaction parameters indicating rod, cone, or melanopsin activity. However, connections were seen between the degree of visual field loss and the intensity of the cone response, indicating that pupil light responses deteriorated in the late stages of the illness.[6]

Retinal ganglion cells respond to light stimuli from rods, cones, and melanopsins, influencing pupil light response through wavelength, intensity, and duration variations. A study compared 3 individuals with neuroretinal vision loss to 43 with normal vision using continuous Ganzfeld stimulants with varying wavelengths and intensities. Pupillary responses were recorded before, during, and after stimulation, revealing differences in pupil contraction percentages between red and blue light stimuli across intensity levels. Normal eyes had stronger pupil responses to blue light at lower intensities than to red light, with persistent contractions frequently outperforming flash contractions, especially in low light situations.[7]

Comparison of pupil responses between insulated inheritable optic neuropathy (HON) patients and healthy controls revealed no significant differences in inner vs. external retinal photoreception. Three testing methods were employed, showing no disparity in reaction wind or pupil response parameters. HON patients exhibited symmetric bilateral optic atrophy and reduced visual field, but no significant variation in pupil responses. However, a correlation was found between visual field loss and cone response intensity, suggesting advanced stages of HON may affect pupil light responses differently.[8]

CHAPTER 3

Methodology

The study's goal is to create a comprehensive framework that combines multiple machine learning and statistical approaches to examine pupil data for the early identification and diagnosis of hereditary illnesses. [9]The study aims to increase illness diagnosis accuracy and efficiency by using modern algorithms and models, which will eventually lead to better patient outcomes and individualized treatment options.

3.1 Block Diagram

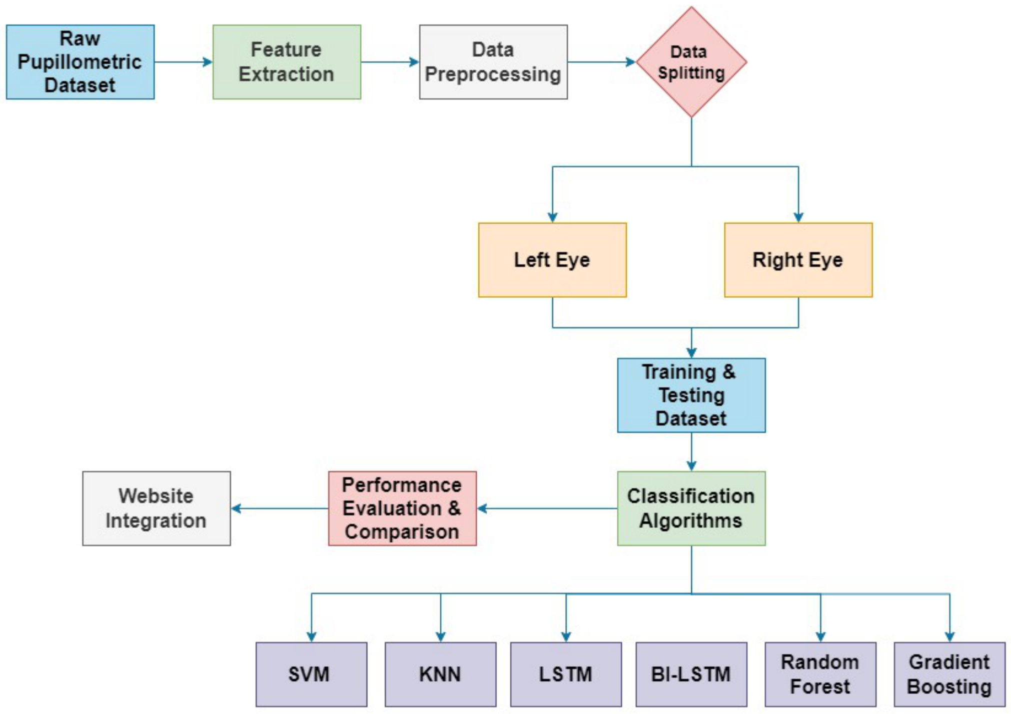


Figure 3.1: Block Diagram of genetic disease detection system

Data Set: The dataset comprises measurements obtained from a pupillometer, capturing the pupillary responses of individuals. These reactions give important insights into the inner and outer retina, which aids in the identification of retinal hereditary illnesses in youngsters. Each entry in the dataset likely includes attributes such as pupil size, response time, and possibly environmental factors. The dataset serves as the foundation for subsequent analysis and model development, enabling the extraction of meaningful patterns and features essential for accurate disease detection.

Data Preprocessing and Feature Extraction: Data preprocessing is the cleaning and transformation of raw pupillometer data to guarantee consistency and dependability. This may include handling missing values, normalizing data, and removing outliers. Feature extraction focuses on identifying relevant information from the preprocessed data that can distinguish between different retinal disorders. Features such as pupil constriction amplitude, latency, and waveform characteristics are extracted to capture distinctive patterns indicative of specific diseases. The quality of feature extraction greatly influences the performance of subsequent modeling steps, making it a critical stage in the analysis pipeline.

Data Splitting: Data splitting entails dividing the preprocessed dataset into distinct subgroups for training, validation, and testing. This guarantees that the model's performance is assessed using previously unknown data, resulting in a more accurate assessment of its generalization capabilities. The dataset is often separated into three parts: training data for training the model, validation data for fine-tuning model hyperparameters, and testing data for objectively evaluating the model's performance. To avoid bias and provide robustness, the distribution of classes or conditions is carefully maintained throughout the split datasets.

Model Training: Model training is the act of applying machine learning algorithms to training data to discover patterns and correlations between input features and desired results. The collected features are used to train a variety of methods, including Support Vector Machines (SVM), k-Nearest Neighbors (kNN), Random Forest, Gradient Boosting, and recurrent neural networks such as Long Short-Term Memory (LSTM) and Bidirectional LSTM (BILSTM). The goal is to optimize model parameters to reduce prediction errors and increase the model's capacity to distinguish between various retinal genetic abnormalities in children.

Model Evaluation: Model assessment is examining the performance of trained models against validation and testing datasets. Accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC) are examples of commonly used assessment measures. These metrics give information on the model's capacity to properly categorize cases of various illnesses while minimizing false positives and false negatives. Model performance is compared to several algorithms to determine the best effective technique for illness identification in youngsters.

Web Integration : Web integration involves incorporating the trained machine learning models into a user-friendly web application interface. Clinicians and healthcare professionals may now effortlessly engage with the algorithms, enter patient data, and obtain real-time illness likelihood forecasts. The backend infrastructure is built with technologies such as Flask, while the frontend interface is designed and implemented using HTML, CSS, JavaScript, and Bootstrap4. The web application provides an accessible and intuitive platform for leveraging the predictive capabilities of the machine learning models in clinical practice.

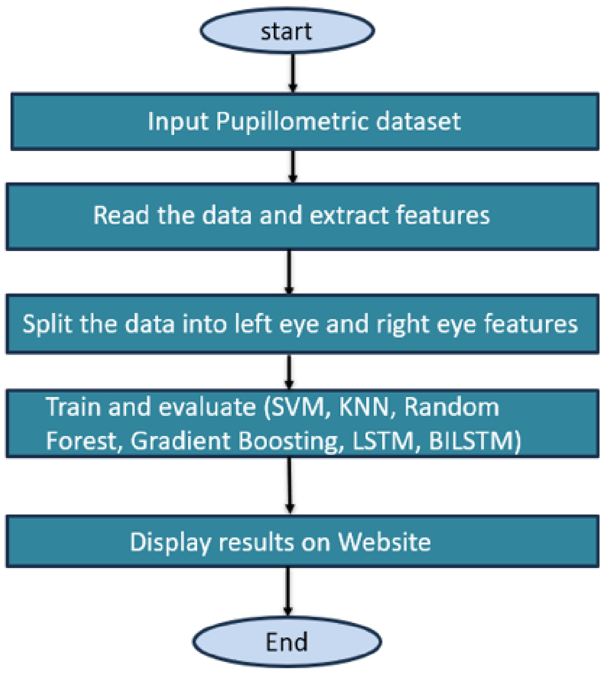

start
Input Pupillometric dataset
Read the data and extract features
Split the data into left eye and right eye features
Train and evaluate (SVM, KNN, Random Forest, Gradient Boosting, LSTM, BILSTM)
Display results on Website
End

Figure 3.2: Flow Chart of genetic disease detection system

The system for detecting retinal genetic diseases in children leverages machine learning and progresses through distinct stages. First, raw data from pupillometer is acquired and preprocessed to ensure consistency. Then, relevant features are extracted from the data, capturing the essence of the data informative for disease classification.

The preprocessed data and extracted features are then used to train a variety of machine learning algorithms, including Random Forest, Gradient Boosting, SVM, KNN, LSTM, and BiLSTM. During this training phase, the models learn to identify patterns in the features associated with the presence or absence of retinal diseases.

To assess the models' effectiveness, the data is split into training and testing sets. The training set is used to train the models, while the testing set evaluates their performance on unseen data. This helps determine the models' ability to generalize to new cases. By evaluating performance criteria such as accuracy, precision, recall, and F1 score, the best effective model for identifying retinal disorders may be determined.

Finally, a web application is integrated into the system. Users can upload parameters through this interface. In the backend, the chosen model analyzes the uploaded data and predicts the probability of a retinal disease being present. The results, including the disease probability prediction, are then presented to the user through the web application.

This system architecture provides a framework for developing a robust and automated system to aid in the early diagnosis of retinal genetic diseases in children.

3.1.1 Data Collection

The first fundamental phase of the CDSS is to analyze the raw files created by the binocular pupillometer after each measurement session in order to export the required data, which is given below.

- Patient ID
- Bilateral pupillary diameter signals for each step of the procedure

- A clinical specialist's diagnosis of 'Pathologic' or 'Healthy'

3.1.2 Feature Extraction:

After pre-processing, each pupillometric signal yields the 8-element feature vector shown below:[10]

- MAX: maximum pupil diameter at baseline
- MIN: minimum diameter at peak constriction
- DELTA: absolute difference between the aforementioned values
- CH: percentage maximum constriction (relative to pupillary diameter at rest)
- LATENCY: delay between the light stimulus and the onset of the pupillary constriction
- MCV: mean constriction velocity
- MDV: mean dilation velocity
- CVmax: maximum constriction velocity

The eight features given above, based on the filtered signal, were selected in accordance with the literature on pupillometry in various disorders and biometric verification. The physicians in this study use the same features on a regular basis, and the equipment gives them in its output files[11]. The time period used to create the aforementioned characteristics was rigorously confined to avoid the possibility of inaccurate values: MAX and LATENCY are calculated in the first second, while the others are obtained during a 5-second frame.

The concept of pupillary LATENCY is justified by the fact that contraction occurs a few milliseconds after the light stimulus is applied. Specifically, this parameter is determined as follows: The pupillometric signal's first derivative, dx(t), is computed; then, beginning with its absolute minimum, the array of values is validated backwards to determine the time instant at which dx(t)=0. The identification of an inflection point is avoided because, after initial SG-smoothing, dx(t) signals include significant noise, and zero-crossings are less sensitive to flickering signals. Although this looks to be the most straightforward method, the inflection point was not discovered since a properly smoothed signal cannot forecast a noisy derivative graph. In contrast, flickering signals have minimal effect on zero crossing detection.

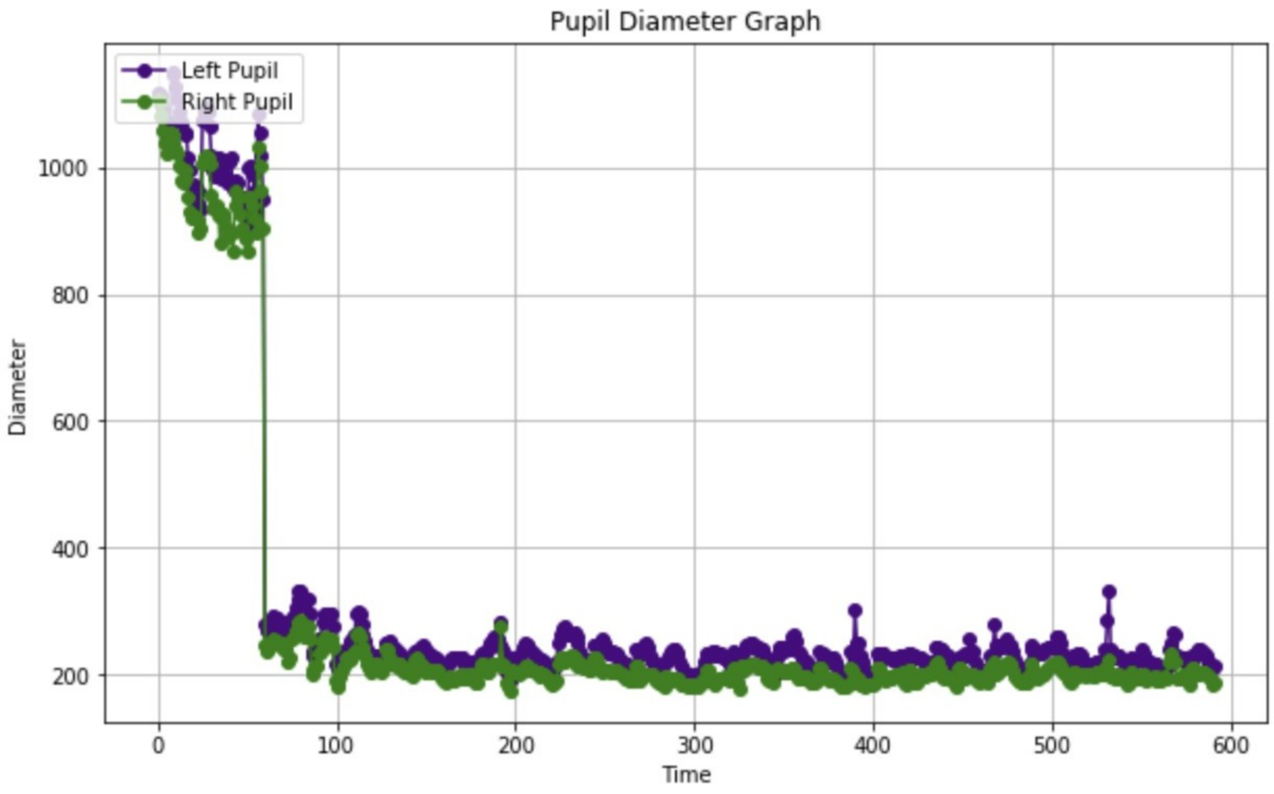


Figure 3.3: Pupil Diameter Graph

3.1.3 Data Preprocessing

The data preprocessing module serves as a pivotal stage in our work, enabling us to delve into the intricacies of our dataset. Here, we embark on a comprehensive exploration journey, unraveling the underlying patterns, distributions, and characteristics of the data. We learn about our dataset's nature using statistical analysis, visualizations, and data profiling approaches.

Furthermore, this module includes critical pretreatment tasks such as resolving missing values, outliers, and data transformations to assure the data's quality and usefulness for future analysis. By methodically preparing and improving the dataset at this point, we provide the groundwork for robust and useful analysis and model building in the next stages of our planned work.

3.1.4 Splitting Data into Train and Test

The data splitting module plays a pivotal role in our project by dividing our dataset into distinct training and testing subsets. Through this essential step, we ensure that our machine learning models are trained on a portion of the data while being evaluated on an independent set, thus enabling us to assess their generalization performance. By stratifying the data based on predefined criteria and maintaining the distribution of target variables across the train and test sets, we reduce the danger of bias and assure the dependability of our model evaluations[12]. This module establishes a crucial foundation for model training, validation, and ultimately, deployment in real-world scenarios.

3.1.5 Model Generation

The model generation module is critical to our workflow since it creates and evaluates machine learning algorithms for categorization jobs. We carefully assess various algorithms, considering factors such as their complexity, computational efficiency, and suitability for our dataset's characteristics.Using techniques such as random forest, support vector machines, and neural networks, we investigate several modeling approaches to see which one best reflects the underlying patterns in our dataset. Following algorithm selection, we instantiate and train the chosen classifier using the training dataset, fine-tuning its parameters to optimize performance. Finally, we evaluate the model's accuracy through rigorous testing on unseen data, ensuring its robustness and reliability in real-world applications. This module serves as the cornerstone of our work, laying the groundwork for the development of effective predictive models tailored to our specific objectives.

3.1.6 Web Application

The website facilitates user interaction through the "User signup and login" module, allowing individuals to register and access the platform. Once logged in, users can input specific parameters such as "ch," "latency," "mcv," and "id" for disease prediction. These parameters are then processed using machine learning algorithms tailored for retinal disease prediction. The system generates predictions based on the provided parameters, indicating the likelihood of retinal disease occurrence.

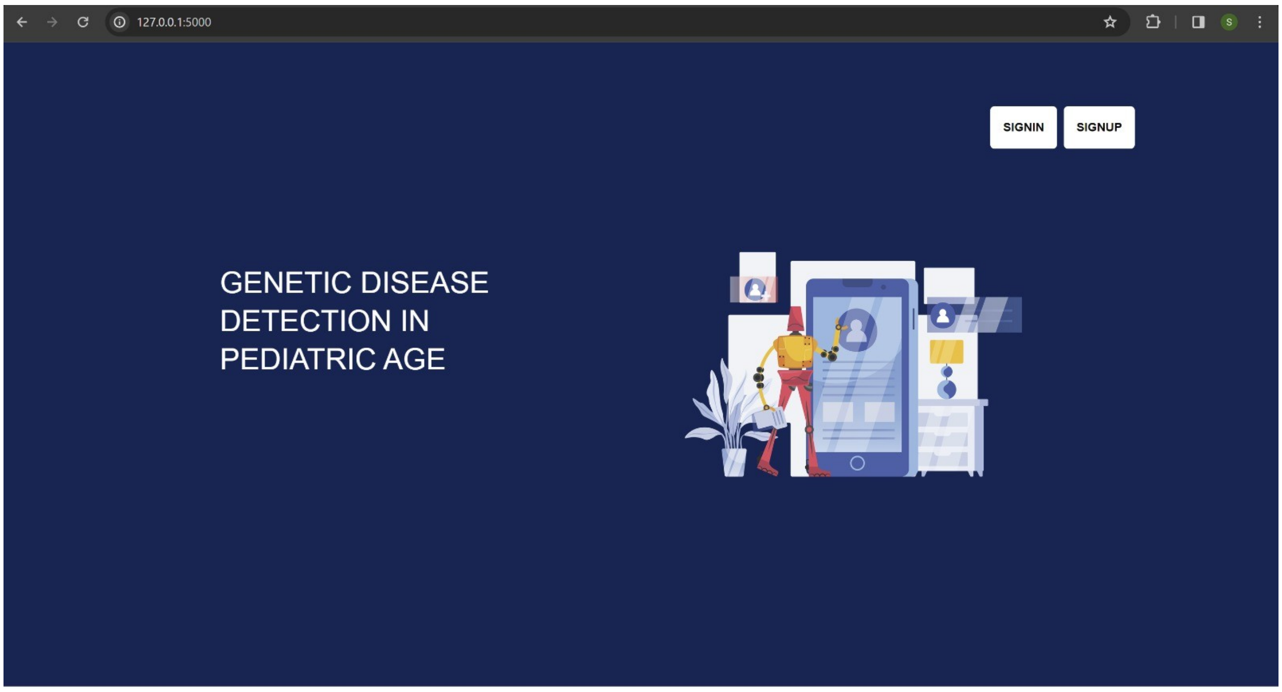


Figure 3.4: Graphical User Interface of Proposed Work

To facilitate model training and assessment, the dataset was divided into two sets: training and testing. Classification performance was examined using classic machine learning models such as Support Vector Machines (SVM) and K-Nearest Neighbours (KNN). Two SVM classifiers were trained individually for the left and right pupils, with the Radial Basis Function (RBF) kernel boosting classification accuracy. [13]

The left and right pupils' SVM classifier predictions were pooled to form an ensemble model. This strategy attempted to enhance overall accuracy by using each classifier's distinct capabilities. Deep learning models such as

Long Short-Term Memory (LSTM) and Bidirectional LSTM (Bi-LSTM) were employed to extract temporal correlations from the dataset. The Bi-LSTM model included dropout layers for regularization and bidirectional temporal dependencies, whereas the LSTM model was a single layer trained across 10 epochs.

Confusion matrices, accuracy, sensitivity, and specificity were the performance evaluation metrics. One could not test the model reliably because of the introduced noise, which consists of flipping 10 percent of the labels at random. Additionally, the study examined the application of Gradient Boosting and Random Forest classifiers, with parameters carefully adjusted for best results. The Gradient Boosting model was cross-validated.

3.2 Algorithms

3.2.1 SVM

Support vector machine (SVM): SVM is a strong machine learning method that may be used for regression, outlier detection, and linear and nonlinear classification. SVMs can be useful in several applications, such as text classification, image classification, handwriting recognition, spam detection, gene expression analysis, face identification, and anomaly detection [14]. Because they can handle high-dimensional data and nonlinear connections, SVMs are versatile and useful in a wide range of applications.

The SVM method seeks to determine the optimum line or decision boundary for partitioning an n-dimensional space into classes so that fresh data points may be conveniently added to the appropriate category later on. Hyperplanes represent the boundaries of optimal choice.

The extreme points and vectors that contribute to the hyperplane are picked using SVM. The technology is termed Support Vector Machine, and the extreme examples are referred to as support vectors. Examine the picture below, which employs a decision boundary or hyperplane to categorize two separate groups.

Linear SVM : Linearly separable data is defined as data that can be

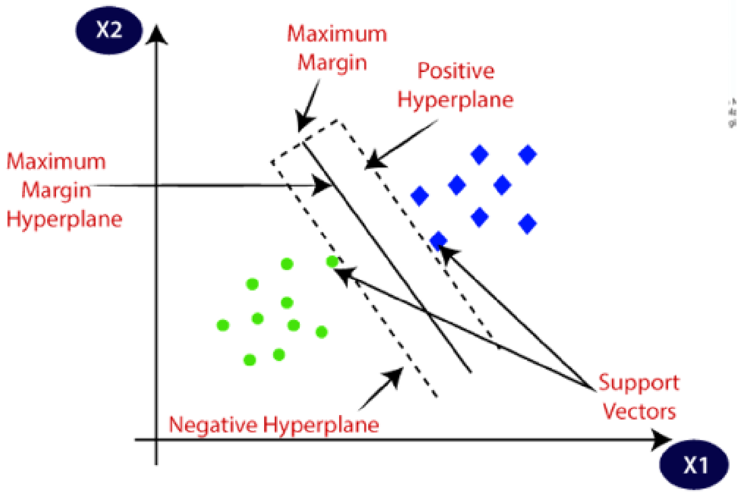


Figure 3.5: Illustration of Linear SVM Classifier

separated into two classes using a single straight line, with the Linear SVM classifier being used. This is a broad definition of linearly separable data.

The decision boundary equation for a linear SVM is as follows:

$$f(x) = w^T(x) + b \tag{3.1}$$

where,

w is the weight vector

x is the input feature vector

b is the bias term

Non Linear SVM: When a dataset cannot be categorized using a straight line, it is considered non-linear data, and the classifier employed is known as the Non-linear SVM. This indicates that non-linear SVM is utilized with data that is not linearly segregated.

A voting classifier is a machine learning model that is trained on a large number of models and makes predictions about an output (class) based on which class is most likely to be the result. It predicts the output class based on the greatest number of votes by simply aggregating the results of all classifiers fed into the Voting Classifier.[15]

For non-linearly separable data, a kernel function $K(x, x')$ is employed to translate input characteristics to a higher-dimensional space. Common kernel functions include Polynomial Kernel

$$(K(x, x') = (x \cdot x' + c)^d) \tag{3.2}$$

Gaussian (RBF) Kernel

$$(K(x, x') = exp(-||x - x'||^2)) \tag{3.3}$$

where,

c and d are hyperparameters to be tuned

x and x' represent data points

A voting classifier is a machine learning model that predicts the most likely result (class) after learning from a variety of models. It predicts the output class with the most votes by averaging the outputs of all classifiers that feed into the Voting Classifier. Rather than creating and assessing separate specialised models, we create a single model that learns from several models and predicts the result based on the overall number of votes for each output class. The Voting Classifier supports two voting mechanisms.

1. **Hard Voting:** In hard voting, the projected output class is the class that earned the most votes or had the best chance of being predicted by each classifier. Assume three classifiers predicted the output class (A, A, and B), with the majority expecting A. Hence, A will be the ultimate forecast.

2. **Soft Voting:** In soft voting, the output class is the predicted class based on the average probability ascribed to it. Assume that given a specific input to three models, the prediction probability for class A = (0.30, 0.47, 0.53) and B = (0.20, 0.32, 0.40). So, with an average of 0.4333 for class A and 0.3067 for class B, class A is clearly the winner because it had the highest probability averaged across all classifiers.

3. **Stacking Classifier:** Stacking is a method of assembling classification or regression models that consists of two-layer estimators. The first layer

includes all baseline models used to forecast outcomes on test datasets [16]. The second layer comprises of a Meta-Classifier or Regressor that accepts all of the baseline models' predictions as input and generates new predictions.

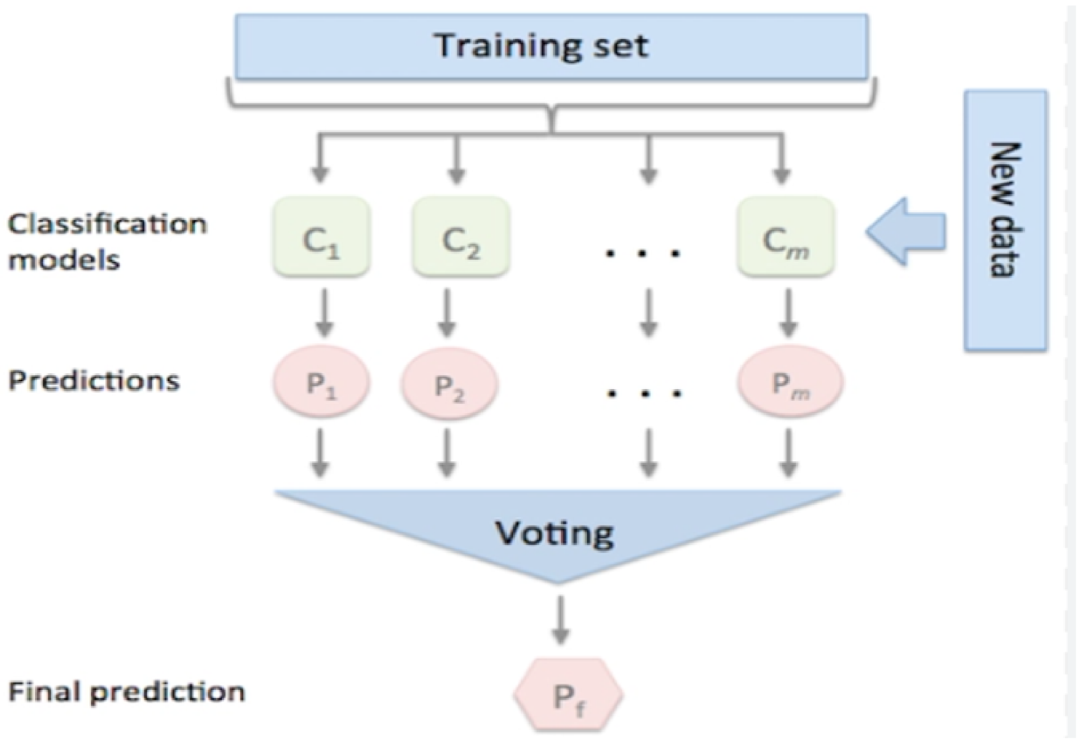


Figure 3.6: Architecture of Voting Classifier

A voting classifier is a machine learning ensemble approach that combines the predictions of several base models to get a single final prediction. This is a breakdown of the voting classifier architecture.

Training Set: The training set constitutes the initial block in the voting classifier architecture, serving as the foundation for training individual base models within the ensemble.This dataset contains labeled samples, often separated into features and matching target labels, which are used to train various machine learning models. The training set's quality and representativeness have a major impact on the ensemble's performance and generalization capabilities.

Classification Models: The categorization models block contains many machine learning models that were trained on the supplied dataset. These models might be of varying complexity and kind, ranging from basic algorithms like Support Vector Machines (SVM) and Random Forests to more modern techniques like neural networks. Each model learns patterns and correlations in the data on its own, giving the ensemble distinct insights.

Predictions: Each base model within the ensemble independently generates predictions on the input data. Represented by the ”Pi” blocks in the architecture, where ”i” represents the individual model; these predictions reflect the model’s output based on the patterns learnt from the training set. The diversity in predictions among the base models facilitates robust decision-making during the aggregation process.

Voting: The predictions generated by all base models are aggregated using a predefined voting rule, such as majority voting or weighted voting. This aggregation process combines the individual predictions to arrive at a final consensual decision.By using the collective knowledge of several models, the voting process improves the ensemble’s overall forecast accuracy and stability.

Final Prediction: The final prediction block represents the ultimate output of the voting classifier ensemble. Based on the combined output of the base models and the voting mechanism, this block produces a single prediction that reflects the consensus among the individual models. This final prediction serves as the actionable insight or decision generated by the ensemble, aiding in various classification tasks with improved reliability and performance.

Training Data: This foundational block represents the dataset utilized to train multiple base models within the stacking ensemble. It comprises labeled examples consisting of features and corresponding target variables, essential for model training and evaluation.

Base Learner Estimators (Model 1, Model 2, ..., Model N): These blocks symbolize the individual machine learning models trained on the training data. While the specific types of models are not specified in the image, they might include a wide variety of models, such as Support Vector Machines (SVM), Random Forests, and Neural Networks.

Stacking Architecture: Stacking is a two-layer ensemble machine learning technique that utilizes a set of base models to make predictions on a hold-out dataset, then trains a final model, called a meta-model, to combine those predictions and generate a final prediction. Here's a breakdown of the architecture:

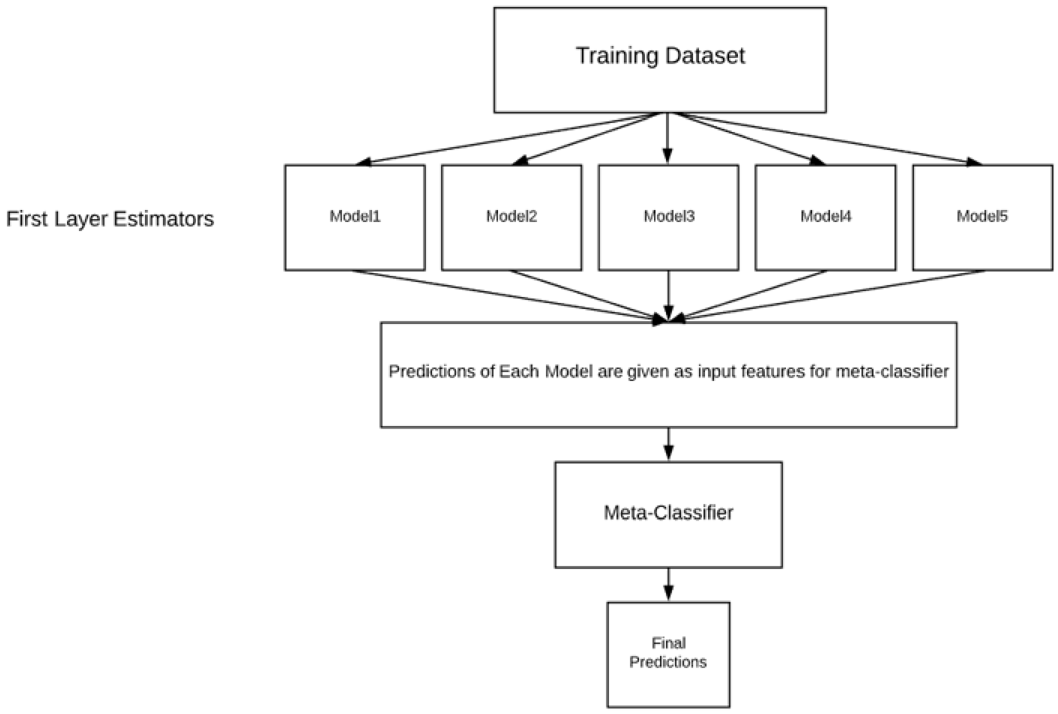


Figure 3.7: Stacking Architecture

Predictions of Each Model: These blocks denote the outputs or predictions generated by each base model on a distinct hold-out dataset, separate from the training data used to train the base models. This ensures unbiased assessments of model performance and facilitates robust ensemble learning.

Meta-Classifier: This pivotal block represents the final model, also referred to as the meta-model or stacking classifier. Trained on the aggregated outputs from the base models and the original training data's target variables, the meta-classifier aims to learn optimal weighting and combination strategies to generate more accurate final predictions.

Final Predictions: This block represents the stacking ensemble's final output, resulting from the combined contributions of the base models and the meta-model. The stacking ensemble provides a refined and possibly superior final prediction by combining the combined insights of varied base models and the meta-classifier's learnt fusion technique, hence improving overall predictive performance and resilience.

SVC Classifier: The Support Vector Classifier (SVC) is a key component of the machine learning system used to diagnose retinal genetic abnormalities using pupillometry data. SVC is a supervised learning technique that is commonly used for classification tasks, particularly in scenarios involving two or more classes. It works by determining the ideal hyperplane for separating distinct classes in the feature space. One of SVC's primary characteristics is its ability to handle both linearly and non-linearly separable data using kernel functions such as linear, polynomial, radial basis function (RBF), and sigmoid kernels. This adaptability allows SVC to successfully capture complicated correlations among data, making it suited for the wide range of datasets encountered in medical diagnostics.

Furthermore, SVC provides resistance to overfitting, which is very useful when dealing with high-dimensional data collected from pupillometry readings. By controlling the margin width and regularization parameters, SVC strikes a balance between maximizing the margin of separation between classes and minimizing classification errors, thereby enhancing its generalization performance on unseen data. Moreover, the implementation of class weights in SVC allows for handling imbalanced datasets commonly encountered in medical diagnosis, where certain disease classes may be underrepresented.

The SVC classifier undergoes rigorous training and optimization processes to learn the underlying patterns from the extracted features of pupillometry data indicative of retinal genetic diseases. Following training, performance parameters such as accuracy, sensitivity, and specificity are used to assess the classifier's diagnostic effectiveness. The utilization of SVC within the machine learning framework underscores its significance as a powerful tool for discriminating between different disease classes based on pupillometry-derived features, thereby contributing to the development of an accurate and reliable clinical decision support system for pediatric ophthalmology.

3.2.2 K-Nearest Neighbors

This study used the non-parametric, instance-based K-Nearest Neighbours (KNN) algorithm to classify cognitive load levels based on pupil diameter measurements. KNN gauges an unclassified data point's proximity to its k-nearest neighbours in the feature space when used to assess cognitive burden. The approach selects the class that appears most frequently among the k-nearest data points to assign the class of the unclassified point based on a majority vote of its neighbors. Determining the proximity of instances requires careful consideration of the chosen distance metric, such as Manhattan distance or Euclidean distance. The approach is perfect for situations when the underlying data distributions are not explicitly known because of its clarity and simplicity.

$$d(xi, xj) = \sqrt{\sum_{k=1}^{n}(xi, k - xj, k)^2} \tag{3.4}$$

$d(x_i, x_j)$ represents the distance between data points x_i and x_j
'n' represents the number of features
x_i^k and x_j^k represent the kth feature value for data points x_i and x_j respectively

When using KNN to identify pupil diameters, the dataset is separated into training and testing sets, with the training sets being used to train the model. Each unclassified data point's proximity to its k-nearest neighbours is determined during prediction, and the point's majority class is determined by that calculation. The number of neighbours considered, or hyperparameter k, influences how sensitive the algorithm is to local fluctuations. Finding a suitable value for k that balances between overfitting and underfitting is crucial. KNN is a crucial part of the comparative analysis of cognitive load classification algorithms because of its interpretability, simplicity of use, and capacity to identify intricate connections within data.

K-Neighbor Classifier: The K-Nearest Neighbors (KNN) classifier is a key machine learning approach for illness identification. The KNN algorithm works

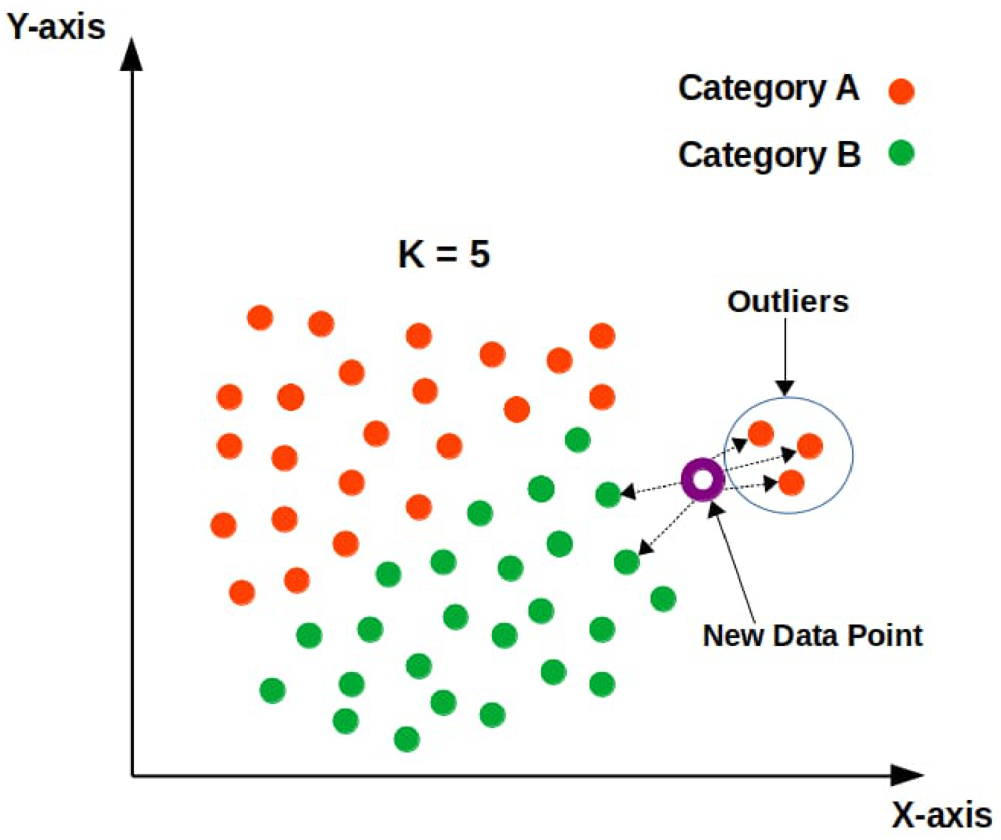


Figure 3.8: Illustrates the KNN algorithm

on the idea of categorizing new data points based on the majority class of their neighbors. It does not use an explicit training step and instead stores the full dataset in memory. When provided with a new data point, KNN determines the K nearest neighbors to it using a given distance metric, usually Euclidean distance, and assigns the class label that is most common among those neighbors.

The KNN classifier utilizes the features extracted from pupillometry data to classify whether a patient is affected by a retinal genetic disease. By considering the characteristics of the patient's pupillary responses and comparing them to those of known cases in the dataset, the KNN algorithm can make predictions regarding the presence or absence of the disease.

One of the most significant benefits of the KNN classifier is its simplicity and ease of implementation. It is a non-parametric method, which means it makes no assumptions about the data's underlying distribution, making it applicable to a broad range of applications. Furthermore, KNN is adaptable and can handle binary and multiclass classification problems successfully.

However, the KNN technique may be computationally inefficient, particularly with big datasets, because it must calculate distances between the new data point and all previous data points in the dataset. Furthermore, the selection of the hyperparameter K, which represents the number of neighbors to consider, can have a substantial influence on the algorithm's performance and may necessitate tuning using techniques such as cross validation.

Overall, the KNN classifier serves as a valuable component in the multi-algorithmic approach to disease detection, offering a straightforward yet effective method for leveraging pupillometry data in diagnosing retinal genetic diseases. Its inclusion in the project underscores the importance of exploring various machine learning techniques to achieve accurate and reliable diagnostic outcomes.

3.2.3 Long Short-Term Memory

This study uses pupil diameter data to categorise cognitive stress. A key component of this process is the Long Short-Term Memory (LSTM) approach. The temporal correlations between dynamic sequences of pupil diameter data can be effectively captured by the LSTM architecture. The network's single LSTM layer is especially intended to learn and retain data over long periods of time, allowing it to account for the unique temporal properties of physiological signals. By randomly deactivating a portion of the network's nodes after each iteration, the dropout regularisation technique improves the LSTM layer and introduces a stochastic element to the training process[17]. By using this dropout technique, overfitting is avoided and the LSTM model's ability to generalise to fresh data is guaranteed.

The three-dimensional input data that the LSTM model processes in the training phase consists of samples, features, and a time dimension that symbolises the consecutive nature of the pupil diameter measurements. The difference between expected and actual class labels is computed using the cross-entropy loss function, which helps optimise model parameters during backpropagation. The ability of the LSTM to capture long-range correlations is particularly helpful for assessing cognitive load since minute variations in pupil diameter over time are important indicators of changing levels of mental stress. After training, the LSTM model can accurately predict previously unobserved data, providing information about the cognitive load levels associated with certain temporal patterns in pupil diameter records.

An LSTM cell is made up of three gates (forget, input, and output) and one cell state. Here's a breakdown of the formulae for one timestep:

Forget Gate (f_t):

$$f_t = \sigma(W_f * [h(t-1), x_t] + b_f) \tag{3.5}$$

where,

σ : sigmoid activation function

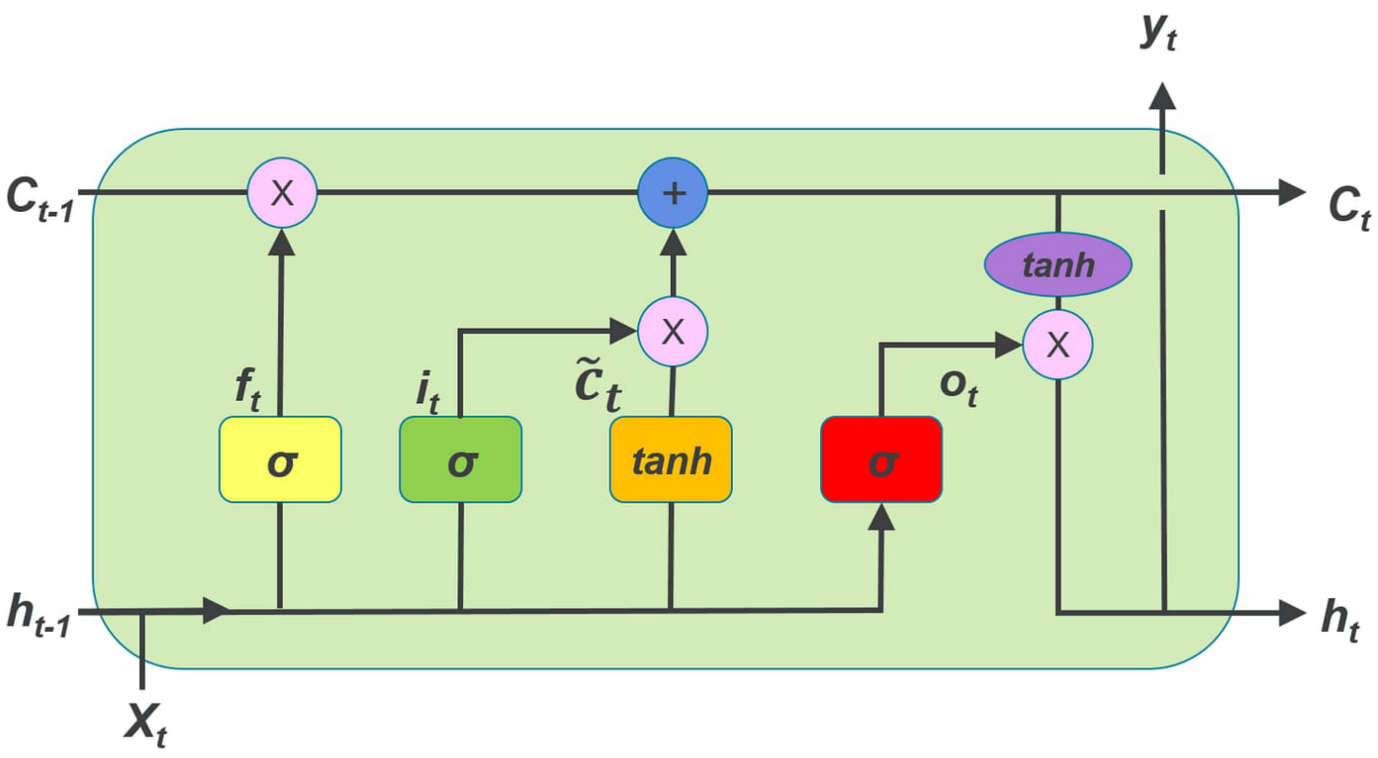


Figure 3.9: Architecture of LTSM Neural Network

W_f : forget gate weight matrix

b_f : forget gate bias vector

$h(t-1)$: hidden state from previous time step

x_t : current input vector

Input Gate (i_t):

$$i_t = \sigma(W_i * [h_(t-1), x_t] + b_i) \tag{3.6}$$

Similar to forget gate, but with different weights (W_i) and bias (b_i).

Candidate Cell State (C'_t):

$$C'_t = tanh(W_c * [h_(t-1), x_t] + b_c) \tag{3.7}$$

$tanh$: hyperbolic tangent activation function

W_c: candidate cell state weight matrix

b_c: candidate cell state bias vector

Cell State (C_t):

$$C_t = f_t * C_(t-1) + i_t * C'_t \tag{3.8}$$

Combines the previous cell state ($C_(t-1)$) with the forget gate and candidate cell state.

Output Gate (o_t):

$$o_t = \sigma(W_o * [h_(t-1), x_t] + b_o) \tag{3.9}$$

Similar to forget gate, but with different weights (W_o) and bias (b_o).

Hidden State (h_t):

$$h_t = o_t * tanh(C_t) \tag{3.10}$$

Combines the cell state with the output gate to create the hidden state for the current timestep.

3.2.4 Bidirectional Long Short-Term Memory

Using temporal connections in pupil diameter data, Bidirectional Long Short-Term Memory (Bi-LSTM) is a sophisticated deep learning architecture that classifies cognitive strain. Two sets of LSTM units—one forward and the other backward—process the input sequence in parallel in the first layer of the Bi-LSTM model[18]. The network's ability to recognise complex temporal correlations within time series data is enhanced by this bidirectional processing, which enables it to detect subtle patterns in both past and future contexts. The Bi-LSTM design effectively addresses the vanishing gradient issue that is common to LSTMs by mixing input from both sides. This improves model performance when it comes to long-term dependencies, which is crucial for the assessment of cognitive load.

Extra layers in the Bi-LSTM architecture are used for prediction and information refinement. The output is converted to a one-dimensional array using a Flatten layer after the bidirectional LSTM layers. Resolved linear unit (ReLU) activation and dropout regularisation are used in a Dense layer to enhance model durability and discourage overfitting. The model can identify pupil diameter data based on cognitive load levels because the final output layer uses the softmax activation function to generate probability distributions across the target classes. The Bi-LSTM design effectively addresses the vanishing gradient issue that is common to LSTMs by mixing input from both sides. This improves model performance when it comes to long-term dependencies, which is crucial for the assessment of cognitive load. With its extensive architecture that incorporates regularisation methods, bidirectional processing, and a well-designed output layer, Bi-LSTM is proven to be a useful tool for accurately classifying cognitive load by capturing intricate temporal dynamics in physiological signals.

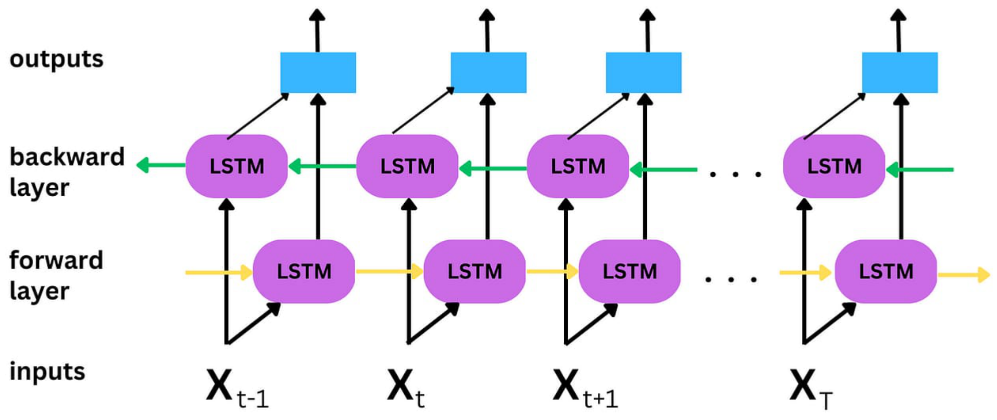


Figure 3.10: Architecture of Bi-LTSM Neural Network

A breakdown of each component of the Bidirectional Long Short-Term Memory (BiLSTM) architecture:

Input: The input layer of the BiLSTM architecture is the entry point for sequential data, such as time series or natural language sequences. It accepts input data in the form of feature vectors or embeddings, which represent the information to be processed by the network. Each element in the input sequence is often represented as a vector of numerical values whose dimensions match to the data's characteristics or attributes. The input layer ensures that the sequential data is appropriately structured before passing it to the succeeding levels for further processing. It is responsible for shaping the initial representation of the input sequence, which is subsequently propagated via the BiLSTM network's forward and backward layers.

Forward Layer: The BiLSTM architecture's forward layer is made up of a sequence of Long Short-Term Memory (LSTM) units that are coupled along the forward path. Each LSTM unit is responsible for forward processing the input sequence, recording temporal relationships, and identifying meaningful patterns from the data. The forward layer allows the network to learn sequential dependencies and temporal dynamics from the input sequence. By iteratively processing the input data from start to finish, the forward layer creates a representation of the input sequence that contains forward context information. This representation serves as the foundation for following layers, which will evaluate and extract higher-level information from the input sequence.

Backward Layer: In contrast to the forward layer, the BiLSTM architecture's backward layer is made up of a sequence of LSTM units connected along the backward route. The backward layer's LSTM units process the input sequence in reverse, collecting backward temporal relationships and identifying complementary patterns from the data. The backward layer acquires information about the context of the input sequence from end to beginning by processing it in reverse. This enables the network to capture backward context information, which may be critical for comprehending the full input sequence thoroughly. The backward layer complements the front layer by offering extra insights into the temporal dynamics and sequential relationships in the input data, increasing the BiLSTM network's total representational ability.

Output: The output layer of the BiLSTM architecture combines the representations created by the forward and backward layers to create the network's final output. It integrates the forward and backward background information gathered by the individual layers to provide a full representation of the input sequence. The output layer is often made up of one or more fully connected layers followed by an activation function, such as softmax or sigmoid, depending on the job (for example, classification or regression). The output layer transforms the combined representations into meaningful predictions or outputs, which may include class probabilities, continuous values, or categorical labels, depending on the specific application of the BiLSTM network. Overall, the output layer synthesizes the information learned from both the forward and backward paths to produce accurate and informative predictions or representations of the input sequence.

3.2.5 Random Forest

Using pupil diameter data, a sophisticated ensemble learning algorithm called Random Forest is frequently used to investigate cognitive load classification. This approach generates a large number of decision trees during training and returns the mean prediction (regression) of the individual trees or the mode of the classes (classification). Each decision tree is trained on a fraction of the original dataset through a process known as bootstrapping, which increases unpredictability. Furthermore, at each split in a tree, a random selection of characteristics is analyzed, increasing the model's generalizability by preventing a single feature from dominating. This strategy reduces overfitting while producing a strong classifier capable of managing complicated interactions between pupil diameter data.

The most discriminative pupil diameter properties can also be found thanks to Random Forest's helpful insights on feature relevance. Because it can effectively handle noise and imbalanced datasets—two crucial aspects when working with physiological data from real-world scenarios—the algorithm is very adaptive. By merging predictions from different decision trees, Random Forest leverages community expertise to produce a reliable classifier for measuring cognitive strain. The model is a good fit for this research because of its adaptability to various settings and resilience to outliers, which show how effective it is at the difficult task of estimating cognitive load levels using pupil diameter features.

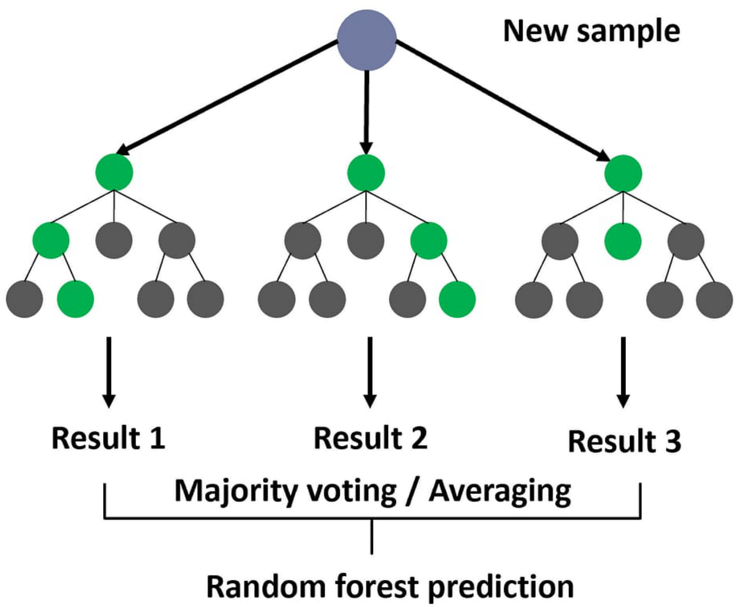


Figure 3.11: Architecture of Random Forest

Random forest prediction is an ensemble approach used in machine learning to solve classification and regression problems. It works by training several decision trees on data samples via replacement (bagging). Each tree in the forest contrasts a randomly chosen group of traits. When a new sample is provided to the forest, each tree votes on which class it belongs to, with the most popular class chosen as the forest's prediction.

The advantage of utilizing a random forest is that it minimizes the likelihood of the model overfitting to the training data. Overfitting happens when a model becomes overly tuned to the unique aspects of the training data and is unable to generalize well to new, unknown data. Random forests can capture essential data features without being unduly specialized to any single data point by training on a range of random feature subsets.

Random forests are also resistant to outliers in data. Outliers are data points that are distant from the average of the other data points. They can bias the outcomes of a machine learning model. Because random forests average the predictions of numerous trees, they are less susceptible to outliers than a single decision tree.

Random forests are an effective and adaptable machine learning approach that may be used to a wide range of jobs. They are a common solution for categorization tasks like spam filtering and fraud detection. They may also be applied to regression issues, such as forecasting stock prices or real estate values.

3.2.6 Gradient Boosting

A powerful ensemble learning technique called gradient boosting is essential to the careful analysis of pupil diameter data for the purpose of assessing cognitive stress. In order to create a strong model[19], the method generates a lot of weak learners—often decision trees—iteratively. The method focuses on the incorrectly categorised examples from the previous iteration in each new iteration, assigning these examples higher weights to emphasise their significance. Gradient Boosting regularly improves the model, [20] which effectively lowers overall prediction error. This iterative process ensures that weak learners make up for the shortcomings of their stronger counterparts, culminating in an ensemble model that can identify minute patterns in pupil diameter data.

Gradient Boosting's effectiveness also lies in its ability to adjust to the unique characteristics of the dataset, effectively handling complex feature interactions and non-linear correlations. Using a weighted additive approach, the system assigns a weight based on the performance of each weak learner. Gradient Boosting is perfect for the challenging task of cognitive load categorization because of its adaptability, which enables it to thrive in circumstances where other algorithms could falter. A final strong learner is produced with exceptional accuracy and robustness in predicting cognitive load levels from pupil diameter metrics. This is achieved by the integration of many decision trees, each of which improves the model based on the errors of its predecessors.

Gradient boosting is a strong machine learning approach that may be applied to both classification and regression applications. It works by

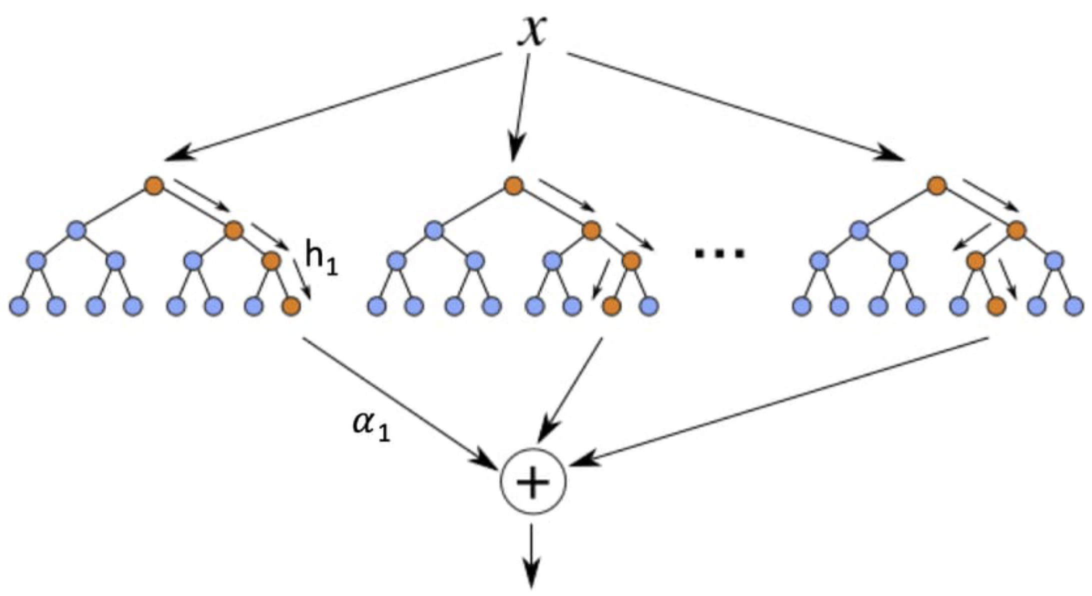


Figure 3.12: Architecture of Gradient Boosting

progressively fitting a sequence of models to the data, with each successive model attempting to repair the prior model's faults. This is accomplished by using a process known as gradient descent, which iteratively modifies the model in the direction that minimizes the loss function.

The process begins with fitting a base model, which can be a simple decision tree or another type of model, to the training data. This initial model provides a starting point for the boosting process.The errors (differences between the actual values and the initial model's predictions) for each data point are calculated. These errors are then converted into pseudo-residuals, which represent the direction and magnitude by which the model's predictions need to be adjusted.

A new model is fitted to the training data, with the pseudo-residuals as the goal variable. This following model, frequently another decision tree, focuses on identifying patterns in the original model's faults. The predictions from the new model are scaled by a learning rate (a hyperparameter that controls the impact of each subsequent model) and then added to the predictions from the previous model. This creates an ensemble model that incorporates the strengths of both the initial and subsequent models.

The steps are repeated iteratively. In each iteration, a new model is fitted to correct the preceding ensemble's mistakes, and the predictions are merged. This procedure continues until a stopping requirement is reached, such as completing a specified number of iterations or attaining a certain degree of accuracy.

By sequentially fitting models to address the errors of prior models, gradient boosting can achieve higher accuracy on complex tasks compared to individual models like decision trees.* Gradient boosting can be used with various base models, allowing it to adapt to different types of data and learning problems.

Gradient boosting can inherently handle missing values in the data, making it robust to data imperfections.While not as interpretable as some models, feature importance scores can be extracted from gradient boosting models to understand which features contribute most to the predictions.

However, there are several things to bear in mind. Training gradient boosting models may be computationally costly, particularly for big datasets with several iterations. Gradient boosting has various hyperparameters, such as the number of trees, learning rate, and maximum tree depth, that must be carefully tuned for best results.* Gradient boosting, like other ensemble approaches, can lead to overfitting if not properly regularized.

To summarize, gradient boosting is a flexible and effective machine learning approach that uses sequential model fitting and gradient descent to obtain high accuracy and handle complicated data. By understanding its strengths, limitations, and considerations, data scientists can effectively use gradient boosting for various tasks in classification, regression, and other machine learning applications.

CHAPTER 4

System Requirements and Design

We list the essential hardware and software requirements needed for the project's successful completion in the introduction to system requirements. In order to ensure the effectiveness and adherence to the intended use cases, this section outlines the specific software tools, programming languages, frameworks, and hardware components needed to support the functionality of the envisaged system.

4.1 Software requirements

Software requirements outline the hardware and setup requirements needed for a computer to perform a programme as best it can. Certain needs, also known as prerequisites, must be loaded individually before the product can be installed since they are commonly overlooked in the software installation package.

Platform - A platform in computing is a framework, either hardware or software, that allows software to execute. Platforms often include a computer's architecture, operating system, runtime libraries, and programming languages. When determining system requirements, one of the first items on the list is the operating system (software). Although software may not operate with multiple versions of the same operating system, backward compatibility is sometimes maintained.

APIs and drivers - Software that heavily utilises specialised hardware—like expensive display adapters—needs an updated device driver or a customised API. One such example is DirectX, a set of APIs used to control multimedia activities, particularly game development on Microsoft platforms.

Web browser - Most apps and online programs that rely significantly on the Internet use the device's default browser. Despite the concerns, Microsoft

Internet Explorer is a widely used software choice for Microsoft Windows that makes use of ActiveX components.

1) Software: Anaconda

2) Primary Language: Python

3) Frontend Framework: Flask

4) Back-end Framework: Jupyter Notebook

5) Database: Sqlite3

6) Front-End Technologies: HTML, CSS, JavaScript and Bootstrap4

4.2 Hardware requirements

Any operating system or software programme will most likely list hardware as one of its most common requirements. A hardware compatibility list (HCL), particularly for operating systems, is frequently given with a hardware requirements list. An HCL for a specific operating system or application contains a list of tested, compatible, and occasionally incompatible hardware components. The following subsections address various elements of the hardware requirements.

Architecture - Each operating system for a computer is intended to function with a certain architecture. Most software programs are only compatible with specific operating systems and architectural settings. While some operating systems and programs are architecture-independent, the majority require recompilation to operate on a new architecture. A list of popular operating systems and their supporting architectures can be found here.

Processing power - The power of the central processing unit (CPU) is a basic system requirement for any software. The bulk of software written for the x86 architecture identifies processing capabilities as the CPU model and clock speed. Many additional CPU variables, like as bus speed, cache, and MIPS, are often overlooked despite their major influence on power and performance. Because AMD Athlons and Intel Pentium CPUs with comparable clock rates typically have varying throughput speeds, this power estimate is not always correct. Due to their ubiquity, Intel Pentium CPUs are sometimes placed in this category.

Memory - All software is stored in a computer's random-access memory (RAM) until it is needed. Memory needs are determined by taking into account the operating system, supporting files and applications, the program, and other ongoing processes. This criterion takes into account how well unrelated software works when executed on a computer system that can multitask.

Secondary storage -The size of the program installation, the temporary files produced and saved during the installation or usage of the software, and the potential requirement for swap space (if RAM is insufficient) all determine how much space is required on the hard drive.

Display adapter - Software that requires a higher-than-average computer graphics display, such as graphics editors and complex games, may specify high-end display adapters as a system need.

Peripherals - Certain peripherals must be utilized often and/or explicitly by certain software programs, necessitating increased performance or capabilities. This category contains CD-ROM drives, keyboards, pointing devices, network devices, and others.

1)Operating System: Windows Only
2)Processor: i7
3)Ram: 32 GB
4)Hard Disk: 256 GB

4.3 Software Environment

Anaconda for Python

1. Anaconda Environment Setup: Anaconda for Python offers a versatile environment accommodating various Python versions and package versions, facilitating package management and project deployment.

2. Installation Process: To install Anaconda, visit the Anaconda Documentation website and download the installer for your operating system.To begin installation, simply double-click the installer.

3. Path Configuration: During installation, choose "add Anaconda to my PATH environment variable" to provide system-wide access to Anaconda and its features.

4. Default Python Interpreter: It is recommended to select "enable Anaconda as my default Python" to set Anaconda as the default Python interpreter for the system.

5. Download Options: Anaconda can be obtained for Windows, Mac, or Linux, supporting both Python 3.7 and Python 2.7 versions for 32-bit or 64-bit systems.

6. Installation Configuration: The installer presents options to agree to the licensing agreement, choose installation for all users or just oneself, and specify the installation directory.

7. Advanced Settings: Advanced options include adding Anaconda to the system's PATH environment variable and making it the primary Python 3.7 installation.

8. Installation Progress: The installation procedure includes unpacking and extracting packages and data, which might take several minutes to finish.

9. Completion: Once installation finishes, users are presented with information about PyCharm and Anaconda Cloud, providing resources for further learning and utilization.

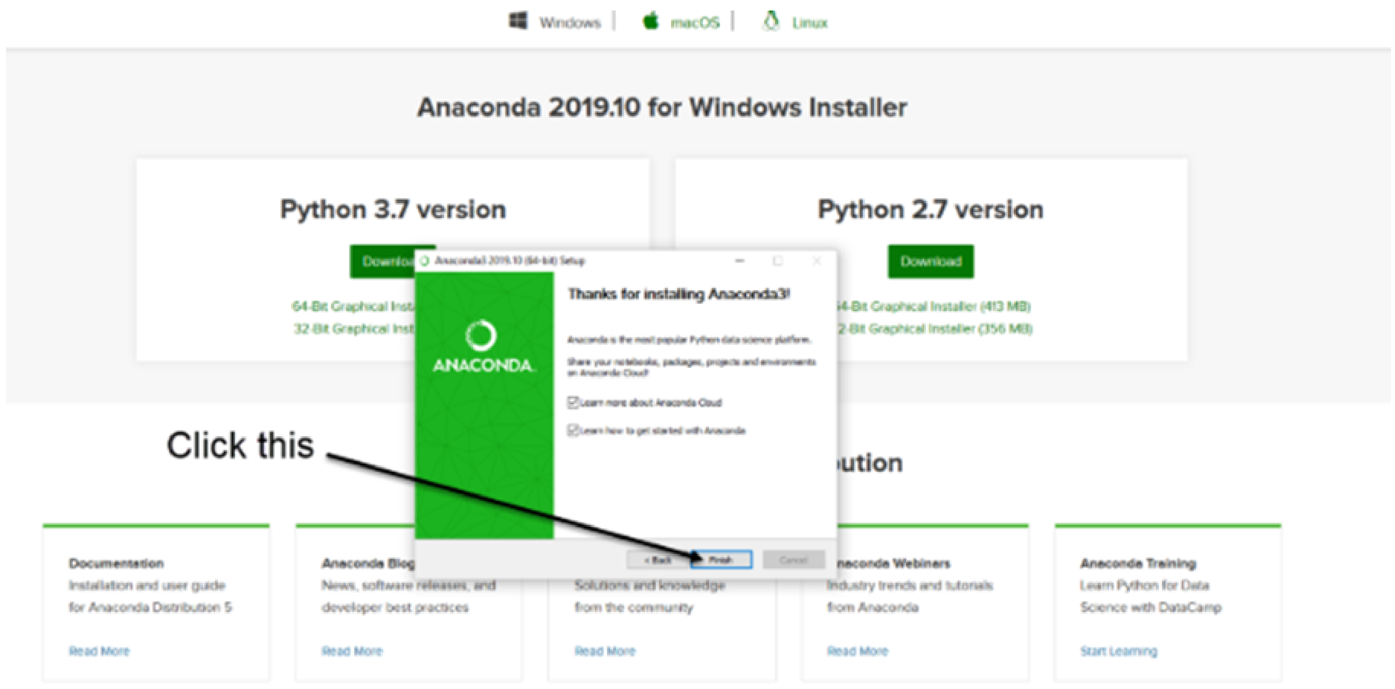
10. Verification: After installation, users can verify the successful installation by searching for Anaconda and accessing the installed options and functionalities.

4.4 Python Language

1. Interpreted and Dynamic: Python is an interpreted language with dynamic semantics, allowing rapid development and easy debugging due to its line-by-line execution and dynamic typing.

2. Object-Oriented and Procedural: Python supports both object-oriented and procedural programming paradigms, facilitating code organization and reuse through concepts like classes and encapsulation.

3. Simple Syntax: Python's simple and readable syntax, characterized



Thanks for installing Anaconda3!

Anaconda is the most popular Python data science platform.

Share your notebooks, packages, projects and environments on Anaconda Cloud!

☑ Learn more about Anaconda Cloud

☑ Learn how to get started with Anaconda

Installation and user guide for Anaconda Distribution 5

News, software releases, and developer best practices

Solutions and knowledge from the community

Industry trends and tutorials from Anaconda

Learn Python for Data Science with DataCamp

Figure 4.1: Installation of Anaconda

by indentations rather than brackets or semicolons, makes it easy to learn and understand for beginners.

4. Extensive Standard Library: Python has a huge standard library with several modules and functions, which reduces the need for developers to write code from scratch and increases productivity.

5. Cross-Platform and Portable: Python code can run on various platforms without modification, promoting portability and enabling developers to write code that works seamlessly across different operating systems.

6. Integrated and Extensible: Python is readily connected with other languages such as C and C++, which expands its possibilities and allows developers to use existing codebases and libraries.

7. GUI Support: Python provides support for GUI programming through frameworks like PyQt5 and Tkinter, enabling developers to create graphical user interfaces for their applications.

8. Memory Management: Python handles memory allocation dynamically, eliminating the need for explicit type declarations and allowing developers to focus on writing code without worrying about memory management.

9. Excellent for Web Development: Python is extensively used for both frontend and backend web development, and frameworks such as Django and Flask provide powerful tools for creating online applications.

10. High-Level and Developer-Friendly: Python's high-level design and developer-friendly features make it an excellent choice for both new and experienced programmers, promoting speedy development and code readability.

4.4.1 Machine Learning

Terminologies of Machine Learning: Machine learning employs a variety of terms that are essential for understanding its concepts and applications. A model, also known as a hypothesis, represents a specific representation derived from data using a machine learning method. Features, on the other hand, are quantifiable attributes of data used to define feature vectors, such as color or taste in the case of fruit classification. The target variable, often referred to as a label, is the value that the model will predict, such as the name of the fruit in a classification task. Training involves providing a collection of inputs (features) and predicted outputs (labels) to create a model that maps new data to trained categories. Finally, prediction occurs when the trained model is given input data to generate a predicted output.

Types of Machine Learning: Machine learning encompasses various types of learning paradigms, each suited for different tasks and data characteristics. Supervised learning involves learning from a labeled dataset to predict labels for new, unseen data instances.It consists of two types of tasks: classification, which uses discrete categories, and regression, which uses continuous values. Unsupervised learning, on the other hand, involves modeling the characteristics of a dataset without using labels, focusing on tasks like clustering and dimensionality reduction. Semi-supervised learning combines unlabeled and labeled data, leveraging the benefits of both approaches. Reinforcement learning revolves around learning optimal actions through trial and error, maximizing rewards based on the current situation.

Deep Learning: Deep learning is a kind of machine learning that trains neural networks to learn from vast volumes of data without explicit programming. Neural networks, modeled after the structure and function of the human brain, are made up of linked nodes grouped into layers. Convolutional Neural Networks (CNNs) excel in image recognition by using filters to extract features from pictures. Recurrent Neural Networks (RNNs) are designed for sequential data processing and retain recollection of previous inputs. Long Short-Term Memory (LSTM) networks solve the vanishing gradient problem in RNNs, making them ideal for jobs with long-range dependencies.

Training and Optimization: Deep learning models are trained by optimizing their parameters to minimize a loss function, which assesses the difference between expected and actual output. Stochastic gradient descent (SGD) is an optimization process that iteratively adjusts the model's parameters to enhance performance. Hyperparameters like as learning rate and batch size have a substantial influence on the training process and must be carefully regulated to get optimal results. Furthermore, network architecture influences the model's ability to learn complicated patterns from input. Our study intends to obtain state-of-the-art outcomes in multiple areas, including image identification, natural language processing, and time series analysis, through effective deep learning model training and optimization.

Challenges in Machine Learning: Despite the rapid progress in machine learning, several challenges persist, hindering its widespread adoption and deployment. Quality of data remains a significant challenge, as obtaining high quality data for training machine learning models can be challenging. Time-consuming tasks, such as data acquisition and feature extraction, also pose challenges, delaying the development and deployment of ML solutions. Additionally, the shortage of qualified experts in machine learning hampers the development of innovative solutions and applications. Lack of clear objectives for formulating business problems and issues related to overfitting, underfitting, curse of dimensionality, and difficulty in deployment further complicate the machine learning landscape. Addressing these challenges requires interdisciplinary collaboration and innovative approaches to developing robust and scalable machine learning solutions.

4.4.2 Libraries/Packages

Tensorflow: TensorFlow is a free and open-source software framework that supports differentiable programming and dataflow in a wide range of applications. It serves as both a symbolic math library and a machine learning application, such as neural networks. At Google, it is used for both production and research. TensorFlow was designed by Google's Brain team for internal use. On November 9, 2015, it was released under the Apache 2.0 open-source license.

Numpy: Numpy is a general-purpose array processing tool. It includes tools for manipulating these arrays as well as a powerful multidimensional array object. This is the main Python package for scientific computing. It has several properties, some of which are significant:

1. An N-dimensional array object with high power and sophisticated (broadcasting) capabilities

2. Integrating C/C++ and Fortran code tools. Useful functions for linear algebra, Fourier transform, and random numbers.

3. Numpy has numerous scientific uses, but it is also handy for storing general data in multidimensional containers. Because Numpy can create any data types, it can readily communicate with a wide range of databases.

Pandas: Pandas is an open-source Python toolkit with strong data structures that allows for high-performance data manipulation and analysis. Python was mostly used for preprocessing and data munging. It did not make a significant contribution to data analysis. Pandas found out the solution. Regardless of the data source, Pandas may be used to complete five standard data processing and analysis phases: load, prepare, modify, model, and analyze. Python and Pandas are used in a wide range of academic and professional disciplines, including finance, economics, statistics, analytics, and others.

Matplotlib: Matplotlib is a Python 2D plotting program that produces publication-quality figures in a variety of hardcopy and cross-platform interactive formats. Matplotlib supports four graphical user interface toolkits, web application servers, Jupyter Notebooks, Python scripts, and IPython shells. Matplotlib seeks to make complex tasks possible while making simple tasks simple. With a few lines of code, you may generate plots, histograms, power spectra, bar charts, error charts, scatter plots, and other graph types. See the thumbnail gallery and sample graphs for examples.

The pyplot package provides a MATLAB-like interface for simple charting, particularly when used with IPython. An object-oriented interface or a group of methods recognized by MATLAB users offer the power user with total control over line styles, font settings, axis characteristics, etc.

Scikit – learn: Scikit-learn provides a wide range of supervised and unsupervised learning methods via a common Python interface. It supports both commercial and academic use and is included with many Linux distributions under a liberal simplified BSD license.

Seaborn: Seaborn, a Python visualisation library, is developed on top of Matplotlib. It has a powerful drawing tool for producing eye-catching and informative data graphics. Seaborn is really useful when it comes to developing small code to produce complex visualisations. It includes tools for creating visually appealing color schemes and charting numerical, relational, and category data. Seaborn is a popular tool for displaying and analyzing data.

Protobuf: Google developed a technology called Protocol Buffers, or Protobuf, for serialising structured data. It is a language-neutral, platform-neutral, and extensible framework for serializing structured data, similar to XML and JSON. Protobuf provides several advantages over traditional serialization systems, including support for schema evolution, quicker serialization and deserialization, and smaller message sizes. It is widely used for data storage, network communication, and distributed systems.

Werkzeug: Werkzeug is a comprehensive Python utility package for the Web Server Gateway Interface (WSGI). It provides low-level APIs for URL routing, form processing, and handling HTTP requests and answers. Werkzeug is commonly used as a foundational component for Flask and other web frameworks and apps. It includes an interactive debugger toolbar for web development, a development server, and a powerful debugger.

Markupsafe: MarkupSafe is a Python library that handles XML/HTML/XHTML markup securely with strings. To prevent XSS (Cross-Site Scripting) attacks and other security issues, it includes features for properly escaping and displaying HTML/XML entities. In web development, markupsafe is commonly used to ensure that user-generated content is shown safely without jeopardizing the application's security.

Keras: Python-based Keras is an open-source deep learning library. The high-level neural network API allows developers to easily design and train deep learning models. Keras supports several backends, including Microsoft Cognitive Toolkit (CNTK), TensorFlow, and Theano. It has an easy-to-use interface for building neural network topologies, such as layers, optimizers, activation functions, and loss functions. Keras is widely used in applications such as image classification, natural language processing, and reinforcement learning.

CHAPTER 5

System Testing and Results

5.1 System Testing

The purpose of the system testing introduction is to verify that the created system operates as intended. This entails creating thorough testing procedures, creating test cases to confirm system behaviour, and analysing the results to make sure the project's goals are reached and any problems are found and fixed.

System testing is the process by which a quality assurance (QA) team assesses how the many components of an application interact inside the larger, integrated system or application. It is also known as system level testing or system integration testing. System testing ensures that a software performs its intended functions.

This method concentrates on an application's functionality and is akin to black box testing. For instance, system testing can guarantee that all user input formats result in the intended output for the entire programme. System testing is often performed by a QA team following the integration of each component and functional or user-story testing of individual modules.

After passing system testing, a software build moves on to acceptance testing so that users can test it before it is put into production. An app development team keeps track of every issue and decides what kinds and amounts of mistakes are reasonable.

Optimizing the approach is the most effective strategy to improve software engineering testing. A software testing plan specifies the procedures necessary to produce a high-quality final product, including what, when, and how. To achieve this crucial goal, the following software testing methodologies and combinations are commonly employed.

One early-stage testing method that avoids using the developing product in production is called static testing. In essence, desk-checking is necessary to find errors and issues that are present in the code itself. This kind of pre-deployment inspection is required to prevent issues brought on by software structural flaws and code errors.

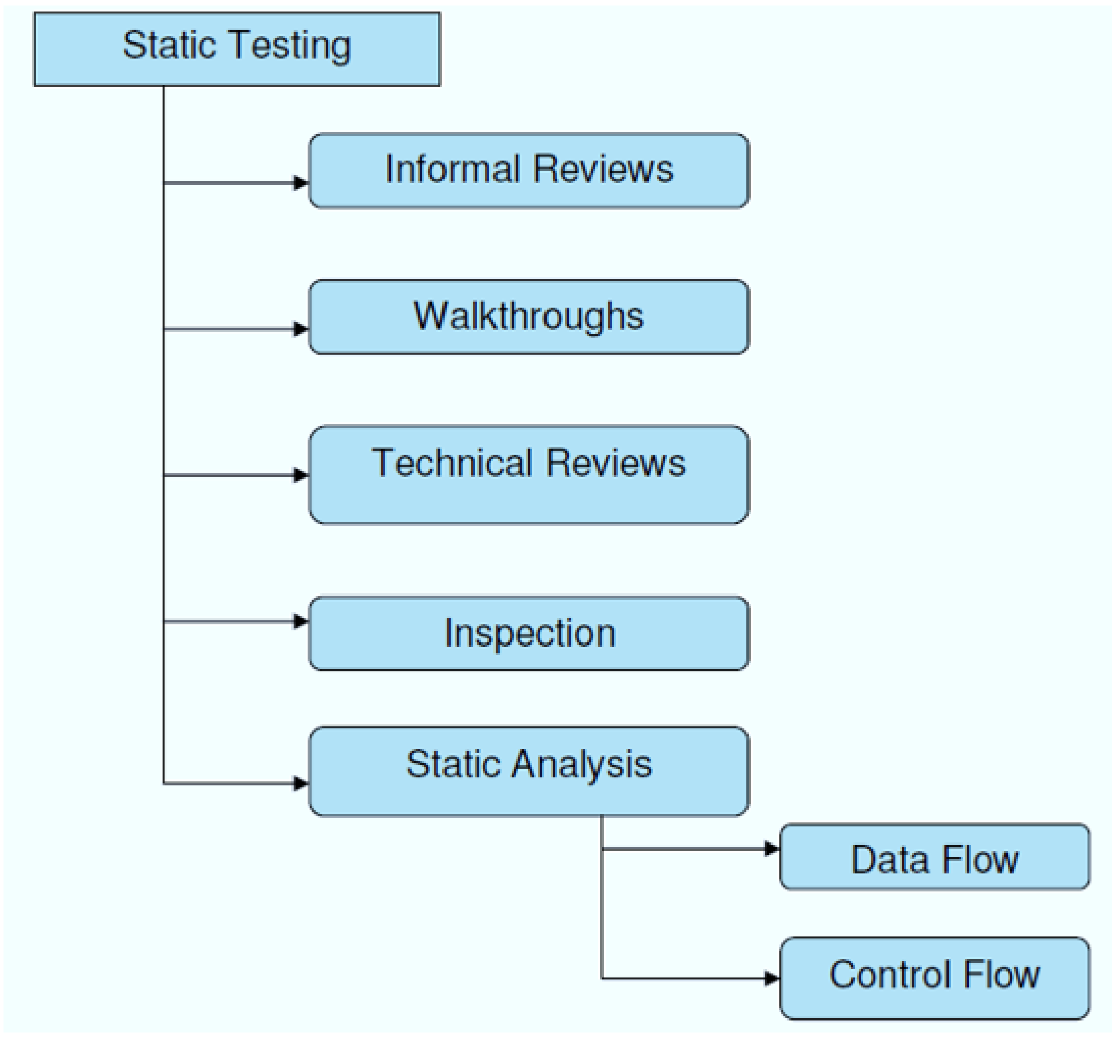


Figure 5.1: Flow Diagram of System Testing

Static testing is a type of software testing where the analysis is done on the code itself, as opposed to dynamic testing which involves running the code with test cases. Static testing is usually done in the early phases of the software development lifecycle.

The different steps involved in static testing as depicted in the image are:

Informal Reviews: This involves a casual examination of the code or documentation by a developer or team member to identify potential errors or areas for improvement.

Walkthroughs: A walkthrough is a more formal peer review process where a developer or a team of developers presents their work to others for feedback. This can help to identify defects in logic, coding style, or potential areas where mistakes could be made.

Technical Reviews: A technical review is a thorough evaluation of the code or documentation by a professional in a certain area, such as security or performance. This form of assessment can help discover flaws that a normal developer may miss.

Inspection: Inspections are the most formal type of static testing and involve a rigorous examination of the code or documentation by a team of reviewers according to a predefined checklist. Inspections can be very effective at identifying defects, but they can also be time-consuming and expensive to conduct.

Static Analysis: Static analysis refers to the use of automated tools to analyze the code for errors or potential problems. There are many different types of static analysis tools available, and they can be used to identify a wide range of issues, such as coding defects, security vulnerabilities, and performance bottlenecks.

5.1.1 Structural Testing

Without running the programme, testing it thoroughly is impossible. White-box testing, or structural testing, is a crucial part of software development that finds and fixes errors and issues that crop up in the pre-production phase. Unit testing on the programme structure is now done using regression testing. The test automation framework's automated approach is typically used to expedite the development process at this stage. The structure and data flows of the software are completely accessible to developers and QA engineers (data flows testing), which enables them to track any alterations (mutation testing) in the behaviour of the system by contrasting the test results with previous iterations (control flow testing).

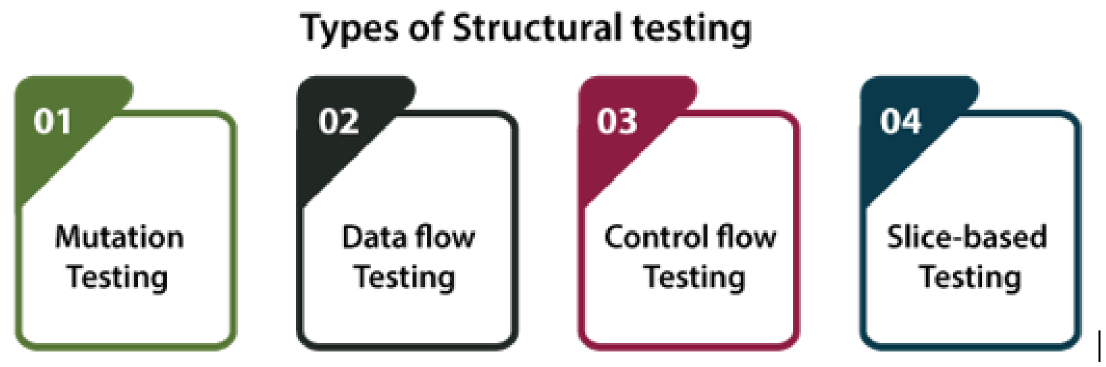


Figure 5.2: Types of Structural Testing

5.1.2 Behavioral Testing

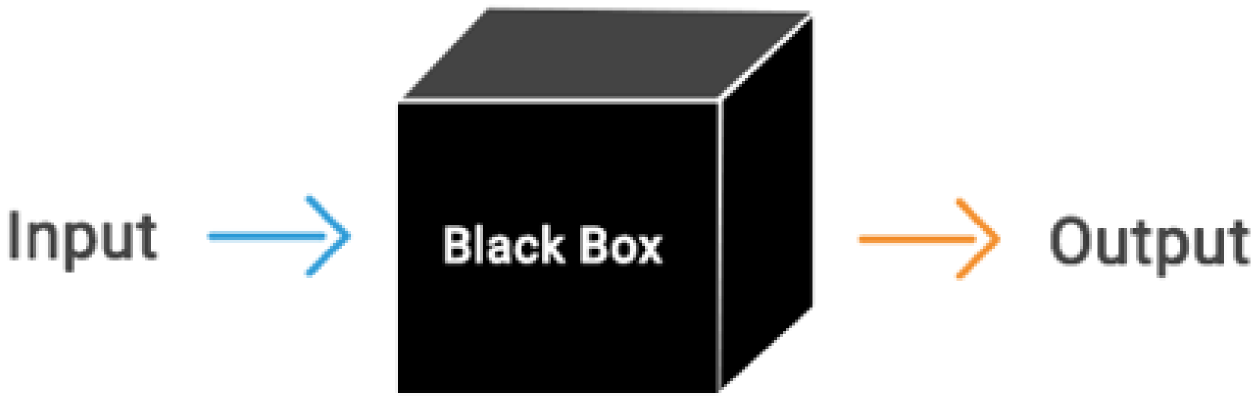


Figure 5.3: Figure of Black Box Testing

The last stage of testing focuses on how the software responds to various activities rather than the procedures that generate these reactions. In other words, behavioral testing, often known as black-box testing, involves running a large number of tests, usually manually, to observe the product from the user's perspective. QA engineers generally have specific knowledge about a firm or other software purposes ('the black box') in order to perform usability tests and respond to concerns in the same way that regular users of the product would. Automation (regression tests) can also be employed in behavioral testing to eliminate human errors in repetitive tasks. For example, you may need to complete 100 registration forms on the website to see how the product handles such an activity.

5.2 Results

Upon inputting parameter values and clicking ”submit,” the system promptly processes the data and delivers a prediction regarding the likelihood of disease occurrence. This streamlined approach ensures rapid access to vital health insights, facilitating early detection and intervention strategies.

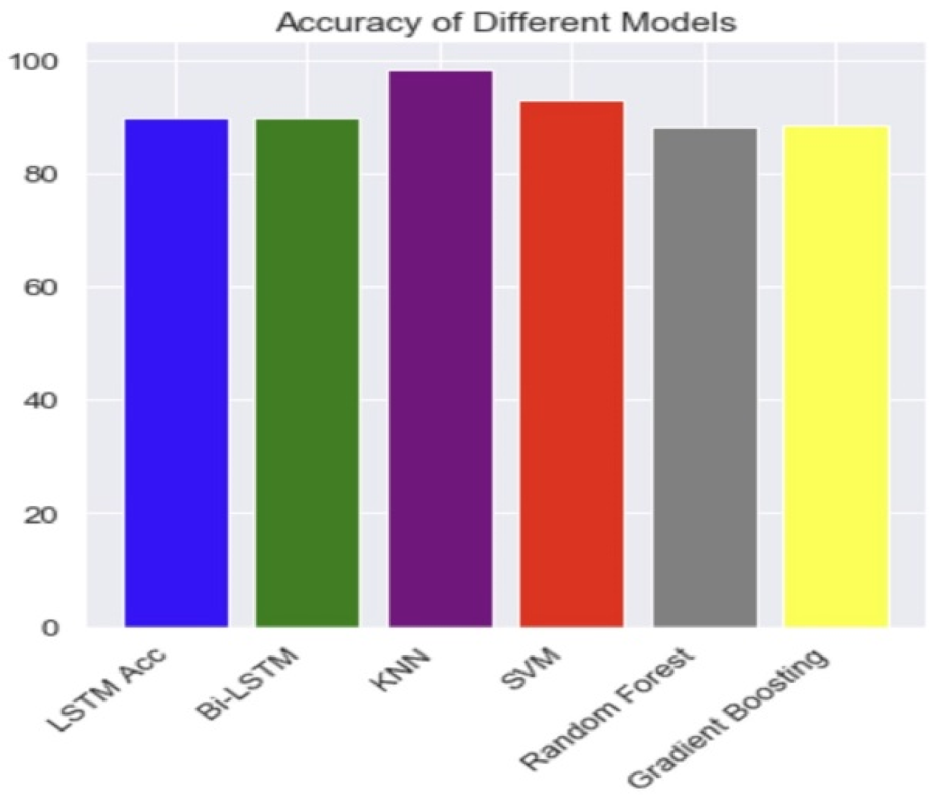


Figure 5.4: Accuracy of Different Models

We acheived high accuracy ratings across a variety of machine learning models. Support Vector Machine (SVM) exhibited a commendable accuracy of 92.85, while k-Nearest Neighbors (KNN) surpassed expectations with a remarkable accuracy of 98.31. Random Forest and Gradient Boosting algorithms demonstrated competitive accuracies of 88.18 and 88.43, respectively. Additionally, our recurrent neural network models, LSTM and Bi-LSTM, showcased strong performance with accuracies of 89.8 and 89.6, respectively.

Table 5.1: Accuracy of Different Models

Accuracy(in percentage)	Model
92.85	SVM
98.31	KNN
88.18	Random Forest
88.43	Gradient Boosting
89.8	LSTM
89.6	Bi-LSTM

In our proposed study, we rigorously analyzed the performance of numerous machine learning models, each of which shown unique skills in properly predicting retinal genetic illnesses in infants. Among these models, Support Vector Machine (SVM) showcased commendable accuracy, achieving an impressive rate of 92.85. Similarly, k-Nearest Neighbors (KNN) demonstrated robust predictive power, boasting an accuracy score of 98.31. Additionally, Random Forest, a popular ensemble learning algorithm, exhibited strong performance with an accuracy rate of 88.18, while Gradient Boosting, another ensemble method, closely followed with an accuracy of 88.43.

Furthermore, we investigated the effectiveness of recurrent neural networks (RNNs) in our research, with a particular emphasis on Long Short-Term Memory (LSTM) and Bidirectional LSTM (Bi-LSTM) architectures. The LSTM produced good findings, with an accuracy of 89.8, demonstrating its capacity to detect long-term relationships in sequential data. Meanwhile, Bi-LSTM, which integrates both forward and backward information flows, performed well with an accuracy rate of 89.6.

The web interface is developed using a combination of software and technologies to ensure a seamless user experience. Anaconda serves as the primary platform for managing dependencies and environments, ensuring

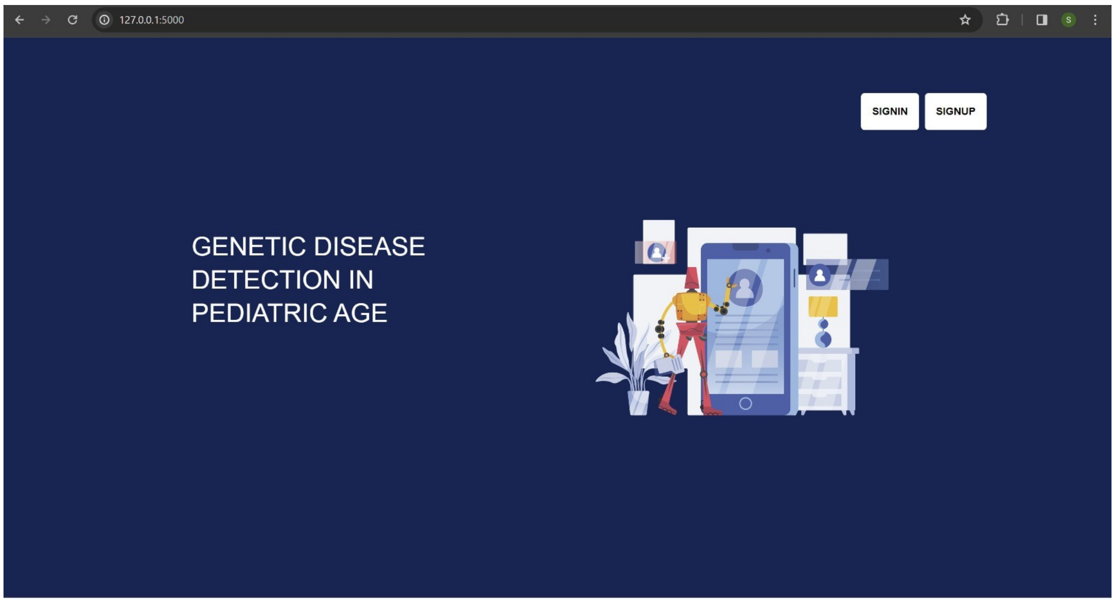
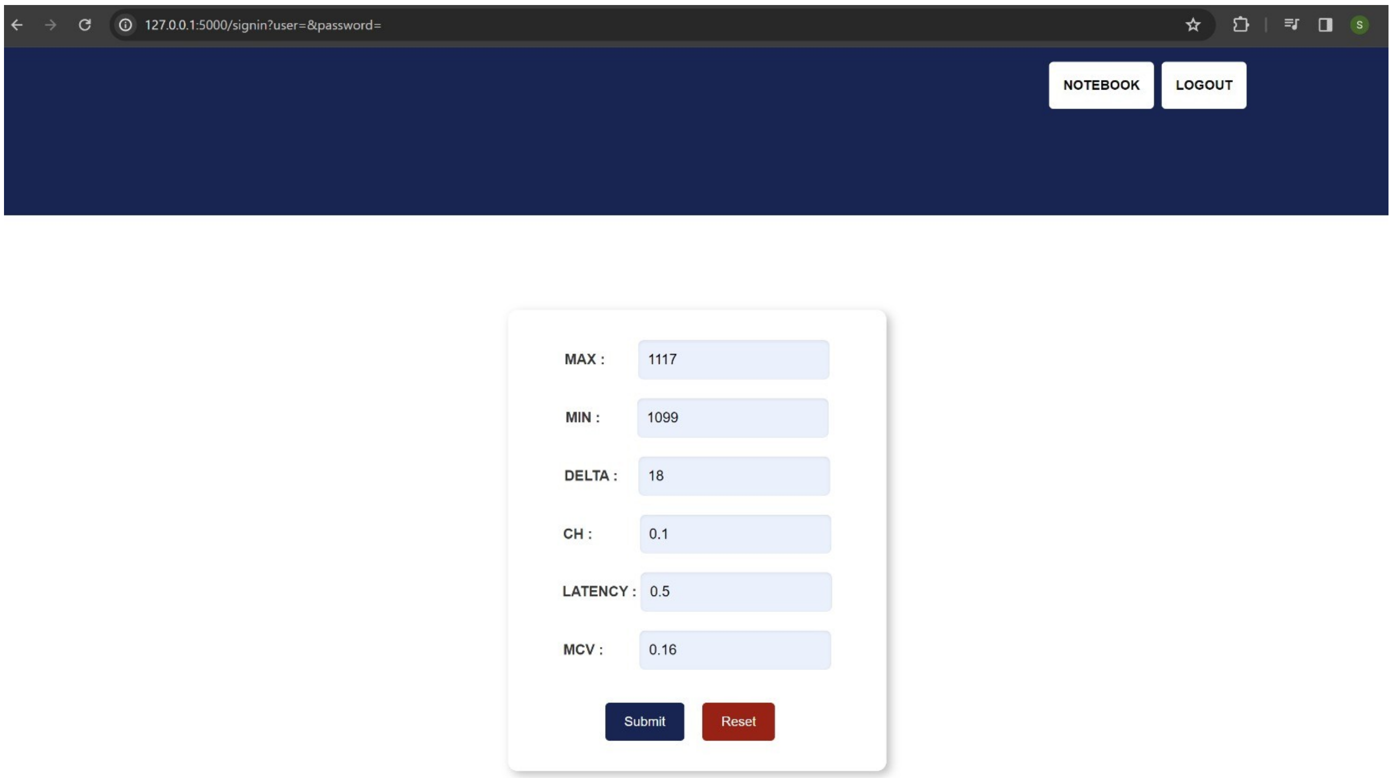


Figure 5.5: Graphical User Interface of Proposed Work

compatibility and stability. Python, as the primary language, powers the backend functionality, with Flask acting as the frontend framework to handle user requests and render dynamic content. Jupyter Notebook complements Flask by providing a convenient environment for developing and testing backend functionalities, ensuring robustness and efficiency. The use of SQLite3 as the database management system enables efficient storage and retrieval of user data, ensuring data integrity and security.The frontend uses HTML, CSS, JavaScript, and Bootstrap4 to design and implement the user interface, resulting in a visually beautiful and responsive layout. When visitors visit the website, they are requested to check in, and then taken to a screen where they may provide pertinent information.

Upon logging in, users are directed to a new page featuring a parameter input box. Within this box, users are prompted to input values for parameters such as MAX, MIN, DELTA, CH, LATENCY, and MCV. These parameters likely pertain to specific thresholds or configurations relevant to the system or application being accessed. Users are required to input numerical values corresponding to each parameter, ensuring precision and accuracy in setting the desired parameters. Once all values are entered, users proceed by clicking the submit button, initiating the transmission of the provided parameter values for further processing or application within the system. This standardized

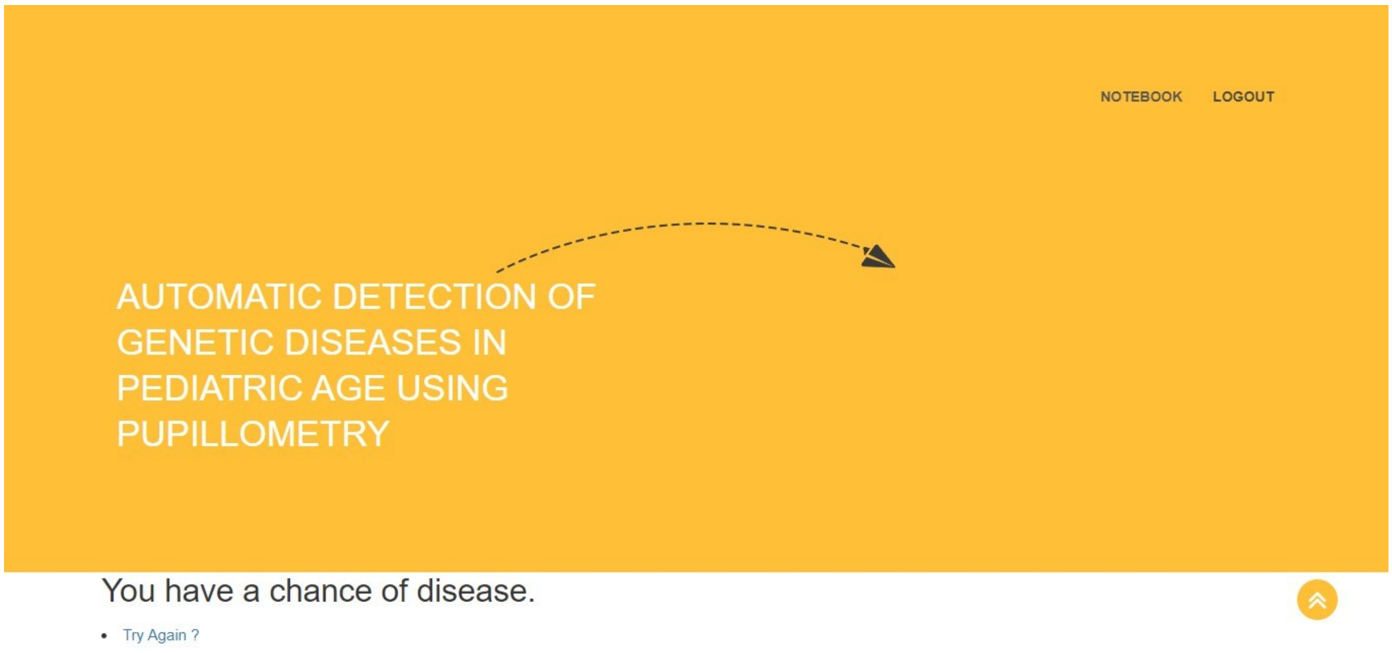


MAX :	1117
MIN :	1099
DELTA :	18
CH :	0.1
LATENCY :	0.5
MCV :	0.16

Figure 5.6: input Webpage

procedure ensures consistency and facilitates efficient interaction with the system, enabling users to tailor settings or configurations according to their specific requirements or preferences without the need for manual intervention or customization.

After providing the input values and clicking submit, a new webpage will open. On this page, the system will display the result indicating whether there is a chance of the disease or not. This feedback provides users with crucial information regarding the likelihood of the disease based on the input parameters provided. By presenting the outcome in a clear and accessible manner, users can make informed decisions regarding further actions or medical interventions. This seamless process streamlines the disease detection process, empowering users with timely insights and facilitating proactive measures for addressing potential health concerns.



AUTOMATIC DETECTION OF GENETIC DISEASES IN PEDIATRIC AGE USING PUPILLOMETRY

You have a chance of disease.

- Try Again ?

Figure 5.7: Output of The Proposed Work

CHAPTER 6

Conclusion and Future Scope

6.1 Conclusion

In conclusion, the findings of this work indicate the efficacy of a variety of deep learning and machine learning models for analyzing pupil data collected from eye-tracking trials. The Bidirectional LSTM (Bi-LSTM) and Long Short-Term Memory (LSTM) models achieve 89.6 percent and 89.8 percent accuracy, respectively. These results suggest that temporal correlations displayed in the dataset can be captured by recurrent neural network topologies. The K-Nearest Neighbours (KNN) method showed its exceptional performance in problem categorization with an accuracy of 98.31 percent. With an accuracy score of 92.85 percent, the Support Vector Machine (SVM) model proved its ability to identify patterns in student behaviour. The Random Forest and Gradient Boosting models, on the other hand, achieved interesting accuracy rates of 88.18 percent and 88.43 percent, respectively.

The trade-offs between computational complexity, interpretability, and classification performance need to be carefully considered in light of the observed variations in model accuracy. The KNN technique can be applied in situations where exact classification is necessary due to its high accuracy. As this is going on, the LSTM and Bi-LSTM models show how well recurrent neural networks can recognise sequential patterns in student data. These results have significant ramifications for further research and applications in behaviour analysis of students. They also assist us in understanding the benefits and drawbacks of different algorithms in this particular situation.

6.2 Future Scope

As machine learning advances, there is a great opportunity to include cutting-edge approaches and algorithms into your project. For example, advances in deep learning architectures like as convolutional neural networks (CNNs) and transformer models may improve the accuracy and efficiency of illness diagnosis in children with ocular genetic abnormalities. Furthermore, exploring ensemble learning methods beyond traditional stacking and voting classifiers, such as gradient boosting machines (GBMs) and neural network ensembles, could unlock new avenues for improving predictive performance and robustness. Additionally, the integration of reinforcement learning techniques could enable adaptive decision-making and model refinement over time, further enhancing the utility and effectiveness of your system in pediatric ophthalmology settings.

The future scope of your work also extends to the integration of emerging technologies and modalities for enhanced disease diagnosis and patient care. For instance, advancements in wearable devices and remote sensing technologies could enable continuous monitoring of retinal health in children, providing real-time feedback and early intervention opportunities. Moreover, the incorporation of augmented reality and virtual reality technologies into your web application interface could facilitate immersive visualization and interpretation of retinal data, improving clinician decision-making and patient engagement. Additionally, the integration of telemedicine platforms and cloud-based solutions could enable seamless collaboration and data sharing among healthcare providers, empowering remote diagnosis and treatment of retinal genetic disorders in children across geographical boundaries. Overall, the future scope of your project lies in leveraging emerging technologies to advance the diagnosis, monitoring, and management of retinal diseases in pediatric populations, ultimately improving patient outcomes and quality of life.

REFERENCES

[1] Xiu-Feng Huang, Fang Huang, Kun-Chao Wu, Juan J Wu, Jie Chen, Chi Pang, Fan Lu, Jia Qu, and Zi-Bing Jin. “Genotype-phenotype correlation and mutation spectrum in a large cohort of patients with inherited retinal dystrophy revealed by next-generation sequencing”. In: *Genetics in Medicine* (Nov. 2014).

[2] Randy Kardon, Susan Anderson, Tina Damarjian, Elizabeth Grace, Edwin Stone, and Aki Kawasaki. “Chromatic Pupil Responses Preferential Activation of the Melanopsin-mediated versus Outer Photoreceptor-mediated Pupil Light Reflex”. In: *Ophthalmology* 116 (July 2009), pp. 1564–73.

[3] Ernesto Iadanza, Francesco Goretti, Michele Sorelli, Paolo Melillo, Leandro Pecchia, Francesca Simonelli, and Monica Gherardelli. “Automatic Detection of Genetic Diseases in Pediatric Age Using Pupillometry”. In: *IEEE Access* PP (Feb. 2020), pp. 1–1.

[4] E.N. VijayaKumari, Kora. Swetha, Atul Kumar, and Rajesh Tulasi. “Automatic Diagnosis of Congenital Conditions in Pediatric Age Using Pupillometry with Machine Learning”. In: *2023 4th International Conference on Smart Electronics and Communication (ICOSEC)*. 2023, pp. 808–812.

[5] I P Keerthana and R Satheesh Kumar. “A Survey of Deep Learning Models to Detect and Classify Eye Disorders”. In: *2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS)*. 2023, pp. 30–36.

[6] Leandro Schwarz, Humberto Remígio Gamba, Fábio Pacheco, Rodrigo Belisário Ramos, and Miguel Ant_onio Sovierzoski. “Pupil and iris detection in dynamic pupillometry using the OpenCV library”. In: *2012 5th International Congress on Image and Signal Processing*. 2012, pp. 211–215.

[7] Muhammad Salman Haleem, Liangxiu Han, Jano van Hemert, Baihua Li, and Alan Fleming. “Retinal Area Detector From Scanning Laser Ophthalmoscope (SLO) Images for Diagnosing Retinal Diseases”. In: *IEEE journal of biomedical and health informatics* 19 (Aug. 2014).

[8] Radim Kolar, Ralf. P. Tornow, and Jan Odstrcilik. “Retinal image registration for eye movement estimation”. In: *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2015, pp. 5247–5250.

[9] Harshita Sharma, Lior Drukker, Aris T. Papageorghiou, and J. Alison Noble. “Multi-Modal Learning from Video, Eye Tracking, and Pupillometry for Operator Skill Characterization in Clinical Fetal Ultrasound”. In: *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*. 2021, pp. 1646–1649.

[10] Cong Xiao-yan, Shang Kai, Wang Chun-peng, Zhu Hong-xuan, and Liu Chun-yuan. "Analysis of pupil size amplitude signal in field fatigue detection". In: *2020 7th International Conference on Information Science and Control Engineering (ICISCE)*. 2020, pp. 302–305.

[11] Richa Gupta, Vikas Tripathi, and Amit Gupta. "An Efficient Model for Detection and Classification of Internal Eye Diseases using Deep Learning". In: *2021 International Conference on Computational Performance Evaluation (ComPE)*. 2021, pp. 045–053.

[12] Yu Ye, Jianbo Mao, Lei Liu, Shulin Zhang, Lijun Shen, and Mingzhai Sun. "Automatic Diagnosis of Familial Exudative Vitreoretinopathy Using a Fusion Neural Network for Wide-Angle Retinal Images". In: *IEEE Access* 8 (2020), pp. 162–173.

[13] Gauri Ramanathan, Diya Chakrabarti, Aarti Patil, Sakshi Rishipathak, and Shubhangi Kharche. "Eye Disease Detection Using Machine Learning". In: *2021 2nd Global Conference for Advancement in Technology (GCAT)*. 2021, pp. 1–5.

[14] Vedant Jolly, Yash Patel, Samkit Shah, and Jyoti Ramteke. "Eye Disease Detection using MobiNet". In: *2023 2nd International Conference on Paradigm Shifts in Communications Embedded Systems, Machine Learning and Signal Processing (PCEMS)*. 2023, pp. 1–6.

[15] Jason Orlosky, Yuta Itoh, Maud Ranchet, Kiyoshi Kiyokawa, John Morgan, and Hannes Devos. "Emulation of Physician Tasks in Eye-Tracked Virtual Reality for Remote Diagnosis of Neurodegenerative Disease". In: *IEEE Transactions on Visualization and Computer Graphics* 23.4 (2017), pp. 1302–1311.

[16] Raji Elsa Varghese and Immanuel Alex Pandian. "Inception-Resnet V2 Based Eye Disease Classification Using Retinal Images". In: *2023 3rd International Conference on Mobile Networks and Wireless Communications (ICMNWC)*. 2023, pp. 1–5.

[17] D. Selvathi and K. Suganya. "Support Vector Machine Based Method for Automatic Detection of Diabetic Eye Disease using Thermal Images". In: *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)*. 2019, pp. 1–6.

[18] Yuxing Mao, Yinghong He, Lumei Liu, and Xueshuo Chen. "Disease Classification Based on Synthesis of Multiple Long Short-Term Memory Classifiers Corresponding to Eye Movement Features". In: *IEEE Access* 8 (2020), pp. 151624–151633.

[19] Nikolas Papadopoulos, Nikos Melanitis, Antonio Lozano, Cristina Soto-Sanchez, Eduardo Fernandez, and Konstantina S. Nikita. "Machine Learning Method for Functional Assessment of Retinal Models". In: *2021 43rd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*. 2021, pp. 4293–4296.

[20] Archana Saini, Kalpna Guleria, and Shagun Sharma. “An Efficient Deep Learning Model for Eye Disease Classification”. In: *2023 International Research Conference on Smart Computing and Systems Engineering (SCSE)*. Vol. 6. 2023, pp. 1–6.