Smart Parenting Assistant

Session: 2022 - 2026

Submitted By:

Fasi-ur-Rehman	2022-CS-71
Ch. Noman Ahmad	2022-CS-91

Submitted to:

Dr. Samyan Qayyum Wahla

Department of Computer Science
University of Engineering and Technology
Lahore Pakistan

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Parenting toddlers is very fun and challenging task at the same time. There is rapid changes in growth of children during early years, this requires parents to be careful about monitoring their child's nutrition, health, and milestones. However, majority of parents lack the knowledge, time, or resources to make well-informed decisions about their child's care. This can lead to missed opportunities for early detection of stunned growth or other growth risks. There is growing demand for accessible and personalized parenting solutions. This made us think about the using Ai development as tool for bringing easiness in the life of a parent.

1.2 Problem Statement

For parents it is very difficult to keep track of their child's growth they often think about whether their infant or toddler is growing normally or not. They often think about what they should feed their children over the growing years. This is due to a lack of reliable and personalized guidance over the internet. Existing apps provide information but that information is static, they fails to provide a personalized growth evaluator and diet recommendations for your child. There is a need for software that can help them with their confusion and suggest plans that are personalized for their children, which can help them ease their minds and also help the children to have better growth and mood.

1.3 Solution

To address this problem of lack of reliable and personalized growth tracking and helping app we developed an app that can be a great help for the parents to feel easy about their child upbringing. Using this app they can easily track their child's growth using this app. They can also use this app to get a personalized diet recommendation, also they can detect any abnormality in their child's growth using this app. This app can bring parents to ease by providing answers to their confusion.

1.4 Objectives

The primary objective of this project is to develop an AI-powered application that assists parents in their child's upbringing by providing:

- **Growth Tracking:** A system to monitor and evaluate the child's growth over time.
- **Personalized Recommendations:** Tailored diet plans and health advice specific to the child's age, weight, and other factors.
- **Abnormality Detection:** Early identification of growth issues to ensure timely intervention.

1.5 Project Scope

This project focuses on developing an AI-powered solution for monitoring and improving child growth and development. The scope includes tracking milestones, offering dietary recommendations, and detecting growth abnormalities for children aged 0 month to 5 years. It does not include real-time medical diagnosis or comprehensive health management.

2 Project Features

The features and the functionalities of those features of Smart Parenting Assistants are given below:

2.1 User Registration and Authentication

This feature of the system allows users to access the application by creating a new account or logging in to the previous one.

2.1.1 User Registration

A parent can navigate to the app provide their email and set the password to create an account. For using the features of the application user will have to create their account first.

2.1.2 Login

After creating an account users can log in using the credentials they set during the signup. After successfully logging in to the account the user can get access to the dashboard where they can enjoy different functionalities.

2.1.3 Password Recovery

In case parents forget their credentials. This app provides them functionality to reset their password. So, they can use new credentials to log in to the app.

2.2 Child Profile Management

This feature allows user to create multiple profiles for their children, manage those profiles, and analyze their growth.

2.2.1 Create Child Profile

Once the user is logged into their account they can create a profile for their child, they can add their current weight, height, allergies, and other such information. Which will be used in other functionalities. Users can create multiple child profiles.

2.2.2 View All Children

Users can add multiple profiles. So it will be hard to remember which data is inserted so for that reason there is a page dedicated for viewing all the profiles added by the parent.

2.2.3 Edit Child Details

Data entered during the child profile creation is automatically fetched and used in other functionalities. So, that data is very important if there is a mistake in entering the data it can alter the other important functionalities. This can lead to wrong detection or diet recommendations. So, there is a functionality in this app to edit the data of children in the app.

2.2.4 Delete Profile

Users also have an option to delete the child profile if they ever decide to remove it from the history. So, they can delete it and the app will make sure to delete it completely so that confidentiality is maintained.

2.3 Growth Monitoring

This application allows users to monitor their growth in the app. They can add their child's growth data like their height, weight, their milestones. They will be stored and they will be used to monitor the growth. Out of their growth data graph will be built which will show details about the speed of growth. Parents can show that to any pediatrician and they can have a good idea about child growth. Other than that when they enter the data in the app. The child's weight and height will be updated and the milestones will be added this data will be used to evaluate the growth and detect abnormalities and parents can also get a diet recommendation from it. So, there is a page dedicated to child growth monitoring where child growth can be added and monitored.

2.4 Reminder Management

This feature allows users to set reminders about essential things like Feeding time, and checkups. They can set up reminders so that they don't have to forget about important checkups ever again.

2.4.1 Set Reminder

User can log in to their accounts and add the reminder details like the reminder description and its date and time. In this way, they can use this app to add important reminders in this app so they don't have to forget about their important schedule anymore.

2.4.2 View Reminder

As user can add multiple reminders so they can keep track of all the reminders they created so that they can update anything if they find any anomaly.

2.4.3 Edit Reminder

If there is any change in plan or any mistake in reminder. This app provides users with an option to update the reminder information.

2.4.4 Delete Reminder

If the reminder is of no use anymore and the user decides to delete the reminder then they can do that in this application.

2.5 AI-powered Abnormal Growth detection

Parents are often worried about the growth of their children. They think whether the height of my child is normal or not according to the age. So, this app is here to ease their worries about their children's growth. This app provides them with an AI-powered feature they have to select the child and based on the child's data stored in the database app will tell whether the child's growth is normal or not. So, this app can help them to ease their confusion and if the growth is not normal they can contact their child's doctor.

2.6 AI-Powered Personalized Diet Recommendation

Parents often feel worried about what can they feed their children and what they can not according to their child's age, especially when their child has allergies. So, this has an AI-powered personalized diet recommendation system. This feature provides diet suggestions according to each child's age, growth, and dietary needs. Also, it considers the allergies of the child while suggesting the diet.

3 Work Flow for AI-Powered Abnormal Growth Detection

3.1 Methodology

To implement the functionality of AI-Powered Abnormal Growth Detection we had to prepare a model which could detect the abnormal growth by automatically analyzing the data entered by the parents in the child profile. So, the challenge was to get a dataset that could train our model to make this functionality possible. After data collection there was a complete flow of work which is explained in this section.

3.1.1 Data Collection

The dataset used in this feature was sourced from Kaggle [1]. The specific dataset, titled "Stunting Toddler (Balita) Detection (121K rows)", contains a large collection of records related to stunting detection in toddlers.

3.1.2 Translating Dataset

The dataset that we sourced from Kaggle was originally in the Indonesian language. So for our better understanding, we first translated the dataset into English by mapping the Indonesian words to English using pandas.

3.1.3 Data Features

The dataset used has 4 features, Which include age(months), height(cm), gender, and Nutrition Status. These are explained in table 1 with their datatypes and possible values. In these features, Age, Gender, and height are input features and Nutrition Status is the Target feature.

Table 1: Features of data set used in Abnormal Growth Detection with Data Types

No.	Feature	Data Type
1	Age (month)	Integer (0-60)
2	Gender	Categorical (Male/Female)
3	Height (cm)	Float
4	Nutrition Status	Categorical (Severly Stunned, Stunned, Normal, Tall)

3.1.4 Class Imbalance

Data was imbalanced, there were a lot more of the normal cases than the abnormal cases which made the data imbalanced. This can be visually viewed in count plot 1. As we can clearly see there were 67750 Normal cases and others' classes' accumulative sum was even less than the normal cases count. So, data was imbalanced which can cause bias towards the normal cases.

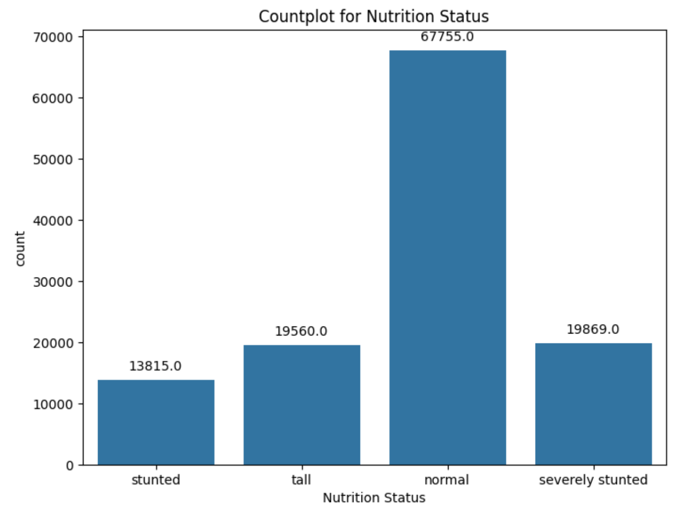


Figure 1: Count plot for showing the data imbalance among the classes

3.1.5 Feature Engineering

In feature engineering, we used one hot encoding and label encoding.

- **One hot encoding** was used on gender features. This is divided into 2 features Gender_male and Gender_female both containing binary values i.e. if the gender is male then the values will be 1 in Gender_male and 0 in Gender_female. It is done to improve the model working. One hot encoding was used instead of a label encoder which would assign 1 and 2 to the gender values. This can cause bias as 2 will get more weight in the training.
- **Label Encoder** was used on the target feature i.e. Nutrition status. We used a label encoder here instead of one hot encoding because the nutrition status was not part of the training so having a higher value in one category can not cause bias in this case so here we used a label encoder.

3.1.6 Handling Data Imbalance

As we discussed, the data was imbalanced between the classes of Nutrition Status. Normal class had a lot more values than other classes which can cause bias. To eliminate these chances of bias we decided to balance the classes. For balancing we choose SMOTE which is an oversampling technique.

We choose to oversample because it increases the instances of minority classes and keeps the majority class as it is. While in Under sampling the majority class has to lose some instances. To prevent this data loss and increase the model's recall and F1 score we choose oversampling and in oversampling. After balancing data the count plot can be visually viewed in 2

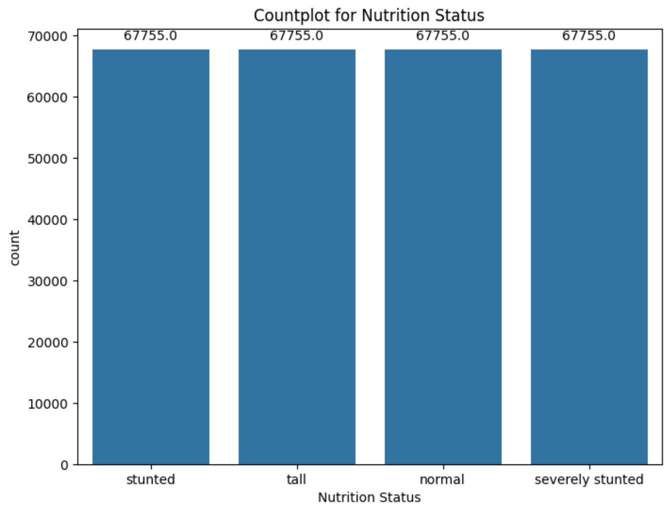


Figure 2: Count plot of Nutrition Value after using the oversampling Technique (SMOTE)

3.1.7 Model Selection

For selecting the final model for our application we used different models like K-Nearest Neighbor (KNN), Random Forest, and Gradient boosted tree using XGBoost.

- **Random Forest:** Random Forest is one of the machine learning algorithms that use multiple decision trees to reach the output. Random forest first creates multiple decision trees and then trains each decision tree with some random noise. After that, it uses the output of multiple decision trees to reach the final output.
- **XGBoost:** XGboost is an open-source library that is used to train the gradient-boosted tree and optimize the performance on large datasets. It is a supervised learning algorithm. It uses multiple decision trees sequentially. In this error, one decision tree goes into the input of the next tree and the next tree learns from that error in this way it reaches the output using multiple decision trees sequentially.
- **KNN:** The k-nearest neighbors (KNN) algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions. It works by assuming that simple points are close to each other.

These models were selected initially for the training on the dataset.

3.1.8 Test Train data split

The dataset was split into two parts. One was a training dataset and the other was a validation or testing dataset. The dataset was split using the library of Scikit Learn to make sure that the data was split randomly. About 80% of the dataset was for training purposes and the rest of 20% was dedicated to the training of the model.

3.1.9 Model Training

Each selected machine learning model was trained firstly on the imbalanced data as it is mentioned that data was imbalanced then data was balanced using the SMOTE technique. We trained our model on the imbalanced and balanced data for the comparison.

3.1.10 Model Evaluation

After training each model on the dataset. We used the following matrices for the model evaluation.

- **Accuracy:** The proportion of correctly predicted instances among the total.

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

- **Precision:** The ratio of true positives to the sum of true positives and false positives is called precision. It tells us about the model's ability to differentiate between true positive and false positive.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **Recall:** The ratio of true positives to the sum of true positives and false negatives, indicates the model's sensitivity to true instances.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **F1-Score:** F1-score is combination of precision and recall.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

3.2 Result and discussion

Each model was evaluated on balanced and imbalanced data. Their results are discussed below:

3.2.1 Evaluation on Imbalanced data

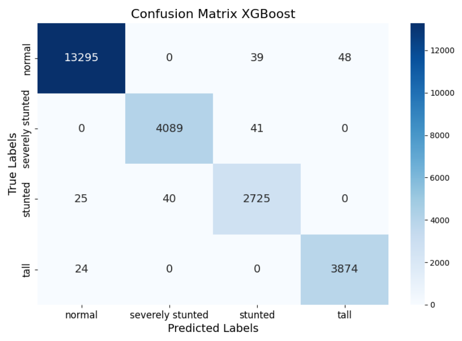
Models were evaluated on the imbalanced data and they showed the following result: Tabe 2 shows

Table 2: Performance of different models on imbalanced datasets, through various metrics.

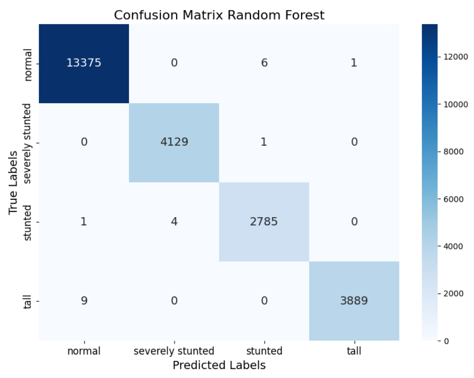
Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Test Accuracy**
XGBoost	0.99	0.99	0.99	0.99	99.10%
Random Forest	1.00	1.00	1.00	1.00	99.91%
KNN	1.00	1.00	1.00	1.00	99.66%

that the model showed high test accuracy for each model but Random Forest showed better test accuracy than the other models.

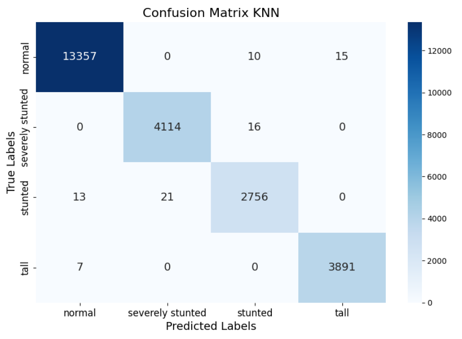
A confusion matrix was also created on the imbalance data for each model it can be visually represented using the heatmap. It is shown in figure 3 As we can clearly see in the figure the count for true positive and true negative was good even in the unbalanced data but we can see that most of cases were normal so there is better matrix for normal than the others. So, it can cause some level of bias in the final. So that's why we needed to balance the data.



(a) Confusion matrix for XGBoost on imbalanced data



(b) Confusion matrix for Random Forest on imbalanced data



(c) Confusion matrix for Random Forest on imbalanced data

Figure 3: Confusion Matrices for each model on the imbalance data

3.2.2 Evaluation on Balanced Data

To avoid bias toward normal we balanced the data discussed before and then evaluated our models on the balanced dataset. The performance on the balanced dataset can be evaluated through the metrics given in the table 3

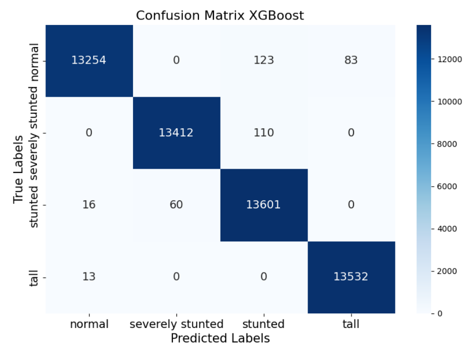
Table 3: Performance of different models on imbalanced datasets, through various metrics.

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Test Accuracy**
XGBoost	0.99	0.99	0.99	0.99	99.25%
Random Forest	1.00	1.00	1.00	1.00	99.98%
KNN	1.00	1.00	1.00	1.00	99.91%

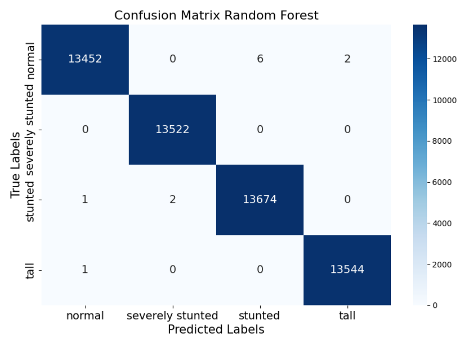
If we compare table 2 with table 3 we can clearly see that models performed a little better on the balanced data.

Other than that model had balanced classes this time so the chances of bias toward normal classes were low. The confusion matrix for the models' evaluation on the balanced data can be viewed in

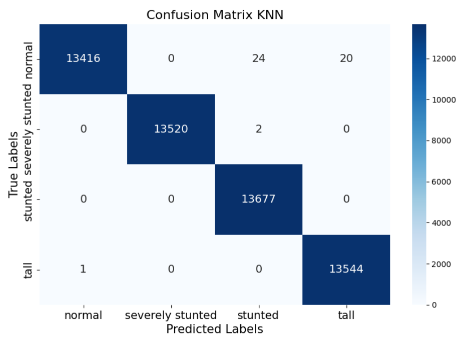
Figure 4. If we compare it with Figure 3, which show confusion matrixes for the imabalnce data, We can clearly see that now the diagonal values have almost equal density.



(a) Confusion matrix for XGBoost on imbalanced data



(b) Confusion matrix for Random Forest on balanced data



(c) Confusion matrix for Random Forest on balanced data

Figure 4: Confusion Matrices for each model on the balanced data

3.2.3 Final Model Selection

By evaluating the confusion matrices and other evaluation matrices, We can clearly see that Random Forest outperforms every other machine learning algorithm used. Random Forest outperformed others because it is very effective in the simple dataset. As our dataset was simple that's why Random Forest outperformed others. So Random Forest was our clear choice as the model to be integrated for the feature. The confusion matrix only for random forests is given in the following table 4.

Table 4: Confusion Matrix of Final Selected model (Random Forest)

	normal	**severely stunted**	**stunted**	**tall**
normal	13452	0	6	2
severely stunted	0	13522	0	0
stunted	1	2	13674	0
tall	1	0	0	13544

3.2.4 Making Sure there is no overfitting

As the dataset was very large and simple there were chances that this would cause overfitting of the model. Model Overfitting is when the model performs really well on the training dataset but performs badly in the testing dataset. To make sure that the model there is no overfitting we took the following steps:

- First, to avoid this overfitting we used the scikit learn library to divide the train and test randomly. And model gave good test accuracy even than so there are fewer chances that the model is overfitted.
- After that we did a cross-validation of our model to make sure that there is no overfitting. The cross-validation was done using the scikit learn library. The score for the Random Forest is given below:

  Table 5: Random Forest Cross-Validation Accuracy Scores

Fold 1	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Statistics**
99.99%	99.98%	99.99%	99.98%	99.99%	Mean: 99.98% Std. Dev.: 0.00%

  The cross-validation scores across different folds were consistent and gave a high mean accuracy of 99.98% and the standard deviation was 0.00%. This score tells us that the model was not overfitted.

These steps make sure to a very large extent, that there was no overfitting while model training.

3.2.5 Integrating Model

After selecting a model out of selected models and making sure that there is no overfitting while training the model we integrated the model with the rest of the app. For that, we used the pickle library to make a .pkl file of the pre-trained model and label encoder. So, that we don't have to train our model each time the application runs. We had to make a .pkl file of the label encoder also so that when the pre-trained model gives a result it will be in labels and the same label encoder should be used for decoding those labels. That's why we had to make .pkl of label encoder also.

4 Work Flow for AI-Powered Diet Recommendation

4.1 Methodology

For AI-powered diet recommendations, we used an API of Gemini. But the handling was not as straightforward as it looked. The complete procedure is described in this section.

4.1.1 Model Selection

We chose this model because it is better than other available models in diet recommendation. It gives more precise recommendations compared to other general-purpose generative AI models. According to research, Gemini has a better benchmark than the openAI and other GenAI models [2]. That's why we chose Gemini as the genAI model for this feature so, that the best response is provided to parents.

4.1.2 Model Preparation

We had to refine the model according to our use. So, to do that we refined the prompt that will be given to the genAI so, it can give us concise answers. After the model's answers were very raw we had to do the preprocessing of the model's response so it could be displayed on the application for the use of parent.

4.2 Result and Discussion

After doing the above steps we were ready to integrate the model with the rest of our app so it can provide the facilities to parents. A parent can use this feature once they add their child's profile. After adding the child's provide the model will extract the data from there and it will automatically show the response to the parents.

5 Technology Stack

The technology stack used for this project is given in table 6.

Table 6: Technology Stack

Backend Language	Python
Platform	Android
Front-end Technology	Flutter, Dart
Database	MongoDB
API	Fast API

6 Conclusion

The Smart Parenting Assistant aims to provide parents with a supportive, AI-driven tool for tracking and enhancing child development. By offering features like personalized growth monitoring, Diet recommendations, abnormal growth detection, and reminder management, this application helps parents make informed decisions for their children's well-being.

With a user-centered design and alignment with WHO standards, the Smart Parenting Assistant ensures reliable, safe, and effective support tailored to each child's needs. The system's integration of both functional and non-functional requirements enables a practical solution that balances performance, security, and usability.

References

[1] Rendiputra. Stunting toddler (balita) detection (121k rows). `https://www.kaggle.com/datasets/rendiputra/stunting-balita-detection-121k-rows`, 2023. Retrieved from Kaggle.

[2] Khaled Saab, Tao Tu, Wei-Hung Weng, Ryutaro Tanno, David Stutz, Ellery Wulczyn, Fan Zhang, Tim Strother, Chunjong Park, Elahe Vedadi, Juanma Zambrano Chaves, Szu-Yeu Hu, Mike Schaekermann, Aishwarya Kamath, Yong Cheng, David G.T. Barrett, Cathy Cheung, Basil Mustafa, Anil Palepu, Daniel McDuff, Le Hou, Tomer Golany, Luyang Liu, Jean baptiste Alayrac, Neil Houlsby, Nenad Tomasev, Jan Freyberg, Charles Lau, Jonas Kemp, Jeremy Lai, Shekoofeh Azizi, Kimberly Kanada, SiWai Man, Kavita Kulkarni, Ruoxi Sun, Siamak Shakeri, Luheng He, Ben Caine, Albert Webson, Natasha Latysheva, Melvin Johnson, Philip Mansfield, Jian Lu, Ehud Rivlin, Jesper Anderson, Bradley Green, Renee Wong, Jonathan Krause, Jonathon Shlens, Ewa Dominowska, S. M. Ali Eslami, Katherine Chou, Claire Cui, Oriol Vinyals, Koray Kavukcuoglu, James Manyika, Jeff Dean, Demis Hassabis, Yossi Matias, Dale Webster, Joelle Barral, Greg Corrado, Christopher Semturs, S. Sara Mahdavi, Juraj Gottweis, Alan Karthikesalingam, and Vivek Natarajan. Capabilities of gemini models in medicine. 2024. URL: `https://doi.org/10.48550/arXiv.2404.18416`.