**CHAPTER THREE**  
**METHODOLOGY**

**3.1 Introduction**

This chapter describes the methodology employed in developing a mobile-based student performance prediction system using deep learning. The proposed approach integrates Long Short-Term Memory (LSTM) and MobileBERT, with the Genetic Algorithm (GA) utilized for feature selection, hyperparameter optimization, and overall system optimization. The methodology involves data collection, preprocessing, feature selection, model development, system implementation, and performance evaluation.

**3.2 System Analysis and Design**

The system is designed as a hybrid deep learning framework that leverages the strengths of LSTM and MobileBERT. The LSTM model captures temporal patterns in student performance data, while MobileBERT, a lightweight version of BERT optimized for mobile deployment, is used for textual data analysis and feature representation.

**3.2.1 System Flowchart**

The system flowchart outlines the sequence of processes involved in training and deploying the predictive model. It starts from data acquisition, preprocessing, feature selection via the Genetic Algorithm, model training, evaluation, and deployment into a mobile-based application.

**System Flowchart:**

Start → Data Collection → Data Preprocessing → Feature Selection (GA) → Model Training (LSTM & MobileBERT) → Evaluation → Deployment → End



Fig 3.1 System flowchart

**3.3 Data Collection**

The dataset for this study consists of student academic records, including demographic details, past academic performance, attendance records, and other relevant features. These data points are collected from academic institutions and open-source educational datasets.

**3.4 Data Preprocessing**

To enhance the quality of input data for the model, several preprocessing steps are applied:

1. **Handling Missing Values:** Missing entries are imputed using statistical techniques such as mean or median replacement.
2. **Feature Scaling:** Continuous variables are normalized using Min-Max Scaling to standardize the data.
3. **Categorical Encoding:** Categorical attributes are converted into numerical form using **One-Hot Encoding**.
4. **Data Balancing:** If necessary, the Synthetic Minority Oversampling Technique (SMOTE) is applied to balance the dataset.
5. **Splitting Dataset:** The dataset is split into **training (70%)**, **validation (15%)**, and **testing (15%)** subsets.

**3.5 Feature Selection and Optimization**

Feature selection plays a crucial role in improving model performance and efficiency. The **Genetic Algorithm (GA)** is used for:

* Selecting the most relevant features.
* Optimizing hyperparameters such as learning rate, batch size, and number of LSTM units.
* Reducing computational overhead while retaining predictive accuracy.

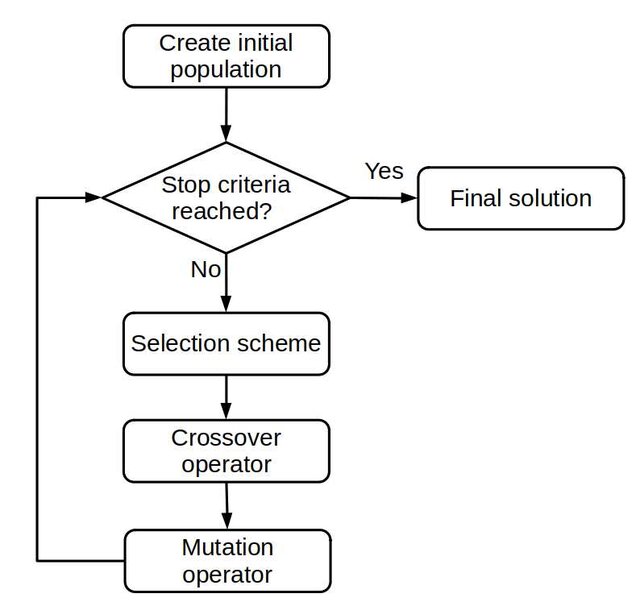


Fig 3.1 Genetic algorithm workflow

**3.6 Model Development**

The hybrid model integrates:

* **Long Short-Term Memory (LSTM):** Captures sequential dependencies in student performance data.

***Algorithm 3.1 Pseudo code for the implementation of LSTM***

**BEGIN**

**Step 1:** Load and Preprocess Data Load student performance dataset

Convert student records into a sequential format based on time (e.g., semesters, years)

Split dataset into Training (70%), Validation (15%), and Testing (15%)

**Step 2:** Define LSTM Model Architecture.

INPUT Sequential Student Performance Data (e.g., past grades, attendance records, study hours)

Sequential Student Performance Data (e.g., past grades, attendance records, study hours)

**Step 3:** Compile and Train Model

Train Model using Training Data

**Step 4:** Evaluate Model

Evaluate Model on Test Data

Compute Accuracy, Precision, Recall, and F1-score

Display Confusion Matrix

**Step 5:** Deploy Model

Convert Model to TensorFlow Lite (TFLite) for mobile deployment

Integrate into Android application for real-time student performance prediction

**END**

* **MobileBERT:** Processes textual and categorical inputs efficiently for mobile deployment.

***Algorithm 3.2 Pseudo code for the implementation of Mobile BERT***

**BEGIN**

**Step 1:** Load Pre-trained MobileBERT Model

**Step 2:** Data Preprocessing for MobileBERT

**Step 3:** MobileBERT Feature Extraction

**Step 4:** Feature Processing for Hybrid Model

**Step 5:** Optimize MobileBERT for Mobile Deployment

Convert Model to TensorFlow Lite (TFLite)

Apply Quantization to Reduce Model Size

Optimize Performance for Low-Latency Inference

**Step 6:** Deploy to Mobile Application

Integrate MobileBERT into an Android-based student performance prediction system.

Allow real-time predictions based on textual and categorical inputs

**END**

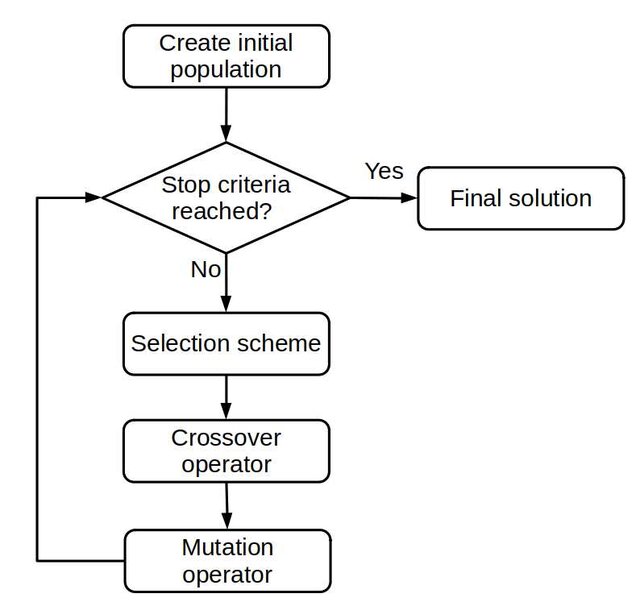


Fig 3.2 Genetic algorithm workflow

***Algorithm 3.3 Pseudo code for the Hybrid integration of LSTM and MobileBERT***

**BEGIN**

**Step 1:** Load and Preprocess Data. Preprocess data: Handle missing values, normalize numerical features, and encode categorical features

**Step 2:** Apply Genetic Algorithm for Feature Selection and Hyperparameter Optimization

Initialize population of feature subsets and hyperparameter sets

**Step 3:** Define Hybrid Model Architecture

**Step 4:** Compile and Train Model

Train Model on Training Data

**Step 5:** Evaluate Model

Evaluate Model on Test Data

Output performance metrics (Accuracy, Precision, Recall, F1-score)

**Step 6:** Deploy Model on Mobile Application

Convert Model to TensorFlow Lite (TFLite) for mobile deployment

Integrate into Android application

Allow real-time predictions for student performance

**END**

**Model Training Steps:**

1. Feed preprocessed data into LSTM for sequential learning.
2. Use MobileBERT for analyzing textual components of the data.
3. Combine the output of both models through a fully connected layer.
4. Apply the Softmax activation function to classify students into performance categories (e.g., excellent, good, average, poor).
5. Optimize the model using Adam optimizer and evaluate using cross-entropy loss function.

**3.7 Development Tools**

* **Python**: Programming language for model implementation.
* **TensorFlow/Keras**: Framework for building deep learning models.
* **Google Colab**: Cloud-based platform for model training.
* **Android Studio**: Development environment for mobile application deployment.
* **SQLite**: Embedded database for storing student performance data in the mobile app.

**3.8 Performance Evaluation**

The model’s effectiveness is assessed using the following metrics:

* **Accuracy:** Measures the overall correctness of the model.
* **Precision:** Evaluates the proportion of correctly predicted positive cases.
* **Recall:** Assesses the ability to detect actual positive cases.
* **F1-score:** Harmonic mean of precision and recall.
* **Confusion Matrix:** Provides insights into classification performance by displaying True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN).

**3.9 System Deployment**

The trained model is integrated into a mobile application for real-time student performance prediction. The mobile app allows students, teachers, and parents to input relevant data and receive predictions regarding academic performance. The optimized MobileBERT ensures efficient performance on mobile devices with limited computational power.

**3.10 Summary**

This chapter detailed the methodology used in developing the mobile-based student performance prediction system. It outlined the system design, data collection and preprocessing, feature selection with the Genetic Algorithm, model development using LSTM and MobileBERT, performance evaluation, and deployment strategies. The next chapter will present the results and discuss the system’s performance based on the implemented methodology.