CHAPTER 3

PROPOSED SYSTEM

The proposed method aims to develop an efficient machine learning-based system for heart disease prediction by leveraging advanced feature selection and optimization techniques. The process begins with data preprocessing, where missing values are handled, and the dataset is standardized. Fast Conditional Mutual Information Maximization (FCMIM) is used to select the most significant features, reducing dimensionality while preserving crucial information. The selected features are then scaled using StandardScaler to ensure consistency across different variables. A Support Vector Machine (SVM) classifier is trained using these refined features, and its performance is optimized through Grid Search, Bayesian Optimization, and Randomized Search. Each optimization technique fine-tunes hyperparameters to enhance classification accuracy and model robustness. The final optimized model is evaluated using accuracy scores, confusion matrices, and ROC curves, ensuring its reliability in predicting heart disease risk. Additionally, the trained model is saved and deployed for real-time predictions, allowing users to input patient details and receive an immediate risk assessment, thereby facilitating early diagnosis and proactive healthcare interventions.

3.1 ARCHITECTURE OF THE PROPOSED SYSTEM

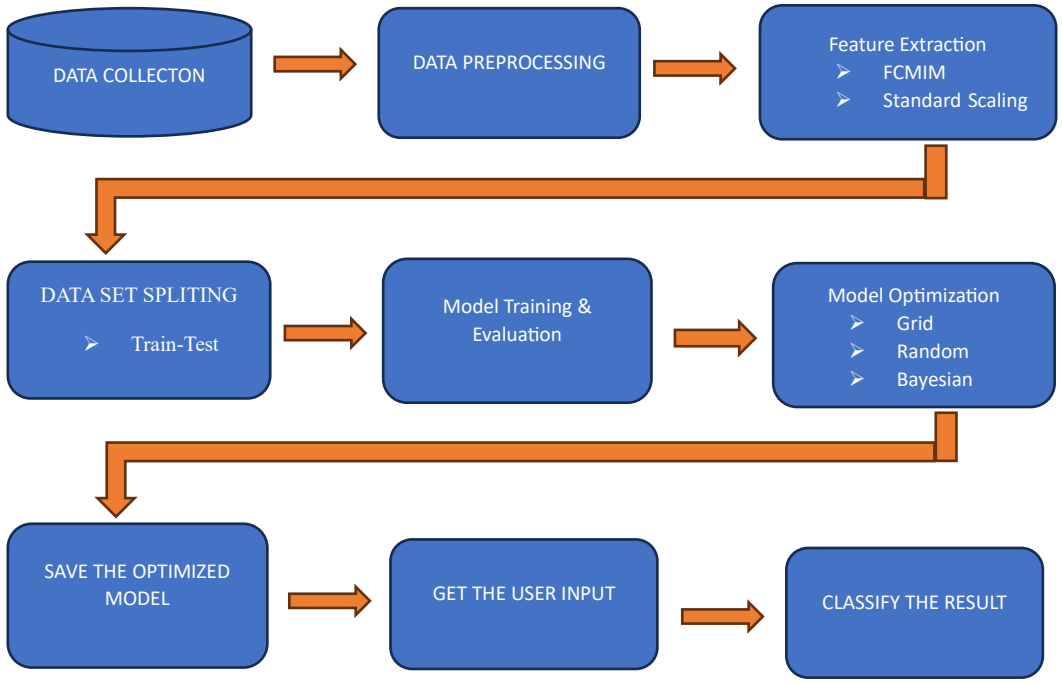

Fig. 3.1 Architecture of Proposed System.

The architecture of the Heart Disease Prediction System is designed to classify patients' heart disease risk efficiently using Support Vector Machine (SVM) with various optimization techniques. The system begins with data collection, where patient medical records are loaded from a dataset. In the preprocessing stage, missing values are handled, and feature selection is performed using Fast Conditional Mutual Information Maximization (FCMIM) to select the most relevant attributes. The selected features are then standardized using StandardScaler to normalize the data distribution. The dataset is split into training and testing sets using Stratified Sampling to maintain class balance.

For classification, an SVM model with a linear kernel is initially trained on the dataset. To improve performance, hyperparameter optimization techniques such as Grid Search, Bayesian Optimization (BayesSearchCV), and Randomized Search are applied. These methods fine-tune the C (Regularization Parameter), Kernel Type, and Gamma values, ensuring the best model configuration. Bayesian Optimization, inspired by probabilistic models, efficiently searches for the best hyperparameters by minimizing the validation loss. Once the optimal parameters are identified, the final SVM model is trained using the best configuration.

The system is evaluated using cross-validation accuracy and assessed on the test data through accuracy scores, confusion matrices, and classification reports to analyze precision, recall, and F1-score. Additionally, ROC curves and AUC scores are computed to compare the performance of different SVM models. The optimized model is then saved and deployed, allowing real-time heart disease risk prediction based on patient input. This architecture ensures an efficient and accurate decision-making system for predicting heart disease, aiding in early diagnosis and medical intervention.

3.2. Support Vector Machine (SVM) Algorithm

The Support Vector Machine (SVM) is a powerful supervised machine learning algorithm used for both classification and regression tasks. It is widely applied in areas such as medical diagnosis, image recognition, text classification, and financial forecasting due to its ability to handle high-dimensional data efficiently [66]. SVM works by finding the optimal hyperplane that best separates different classes in a dataset, maximizing the margin between data points of different categories [67].

SVM operates based on the structural risk minimization principle, making it robust against overfitting. Given a set of training data, the algorithm finds a decision boundary (hyperplane) that maximizes the separation between different class labels.

Data points that lie closest to the hyperplane are called support vectors, and they play a crucial role in defining the classification boundary [66].

3.2.1 Working of SVM

The key steps involved in the SVM algorithm are:

Selecting the Kernel Function: SVM can use different kernel functions to transform data into a higher-dimensional space where it becomes linearly separable.

Finding the Optimal Hyperplane: The algorithm determines the hyperplane that maximizes the margin between the nearest data points of different classes.

Identifying Support Vectors: Data points closest to the hyperplane are identified as support vectors, which define the decision boundary.

Classifying New Data Points: Based on the trained model, SVM predicts the class of new data points by evaluating on which side of the hyperplane they lie.

3.2.2 Kernel Functions in SVM

SVM utilizes different kernel functions to map non-linearly separable data into higher-dimensional spaces:

Linear Kernel: Used when data is linearly separable.

Polynomial Kernel: Extends SVM to non-linear problems.

Radial Basis Function (RBF) Kernel: Handles complex relationships effectively.

Sigmoid Kernel: Similar to neural networks.

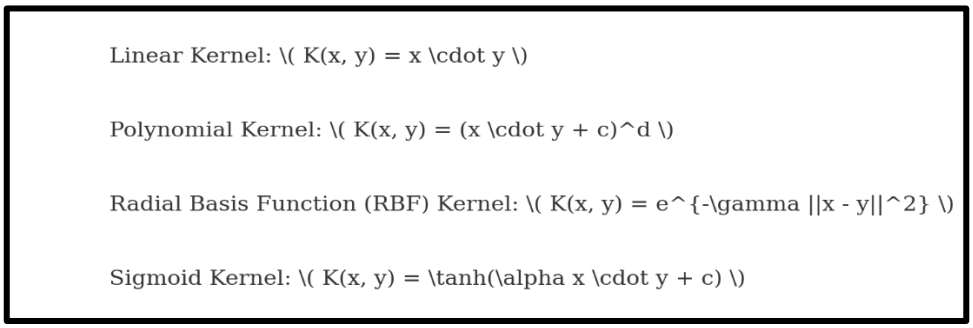

Fig. 3.2 Kernel formulas

3.2.3 Choosing the Hyperparameters

To optimize SVM performance, selecting the right hyperparameters is crucial:

C (Regularization Parameter): Controls the trade-off between achieving a low error and maximizing the margin. A smaller C leads to a larger margin but allows some misclassification, while a larger **C** tries to classify all points correctly.

Gamma (γ) Parameter (for RBF Kernel): Determines how far the influence of a single data point reaches. A higher γ captures more details but risks overfitting.

Kernel Type: The choice of kernel depends on the data distribution. Linear kernels work well for linearly separable data, while RBF kernels are effective for complex patterns.

3.2.4. Architecture of SVM Algorithm:

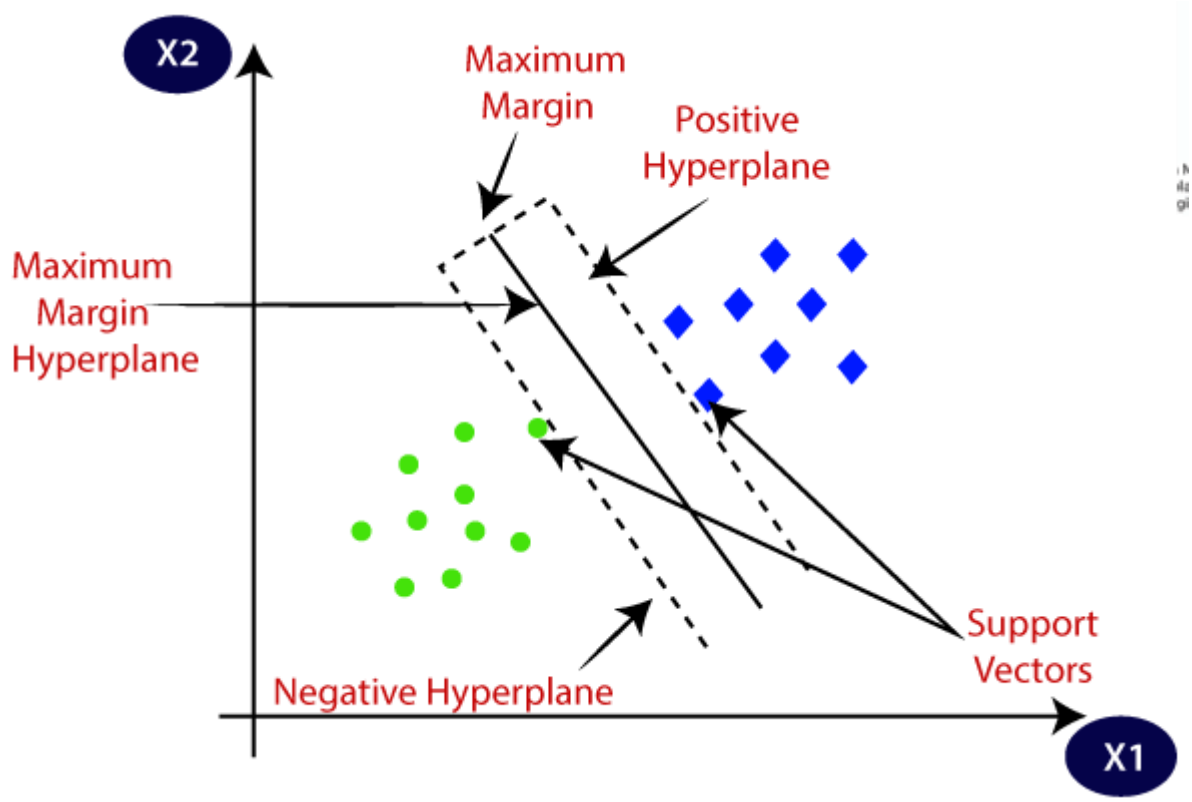

Fig.3.3 Architecture for SVM

3.2.5. FCMIM-SVM (Fast Conditional Mutual Information Maximization with SVM)

FCMIM-SVM is a feature selection-based approach that enhances the performance of an SVM classifier. FCMIM ranks features based on their relevance by maximizing conditional mutual information while reducing redundancy. This leads to improved classification accuracy, reduced computational cost, and better generalization.

Process Flow

1. Feature Relevance Evaluation: Compute conditional mutual information for each feature.

2. Selection Mechanism: Rank and choose the most informative features.

3. Model Construction: Train an SVM classifier with the refined feature set.

4.Performance Assessment: Evaluate classification accuracy and generalization ability.

3.2.6. Grid Search SVM

Grid Search is a hyperparameter optimization technique that systematically explores different parameter combinations to find the best configuration for an SVM model. It focuses on tuning parameters such as the kernel type, regularization parameter CCC, and kernel-specific values like gamma.

Process Flow

1. Parameter Space Definition: Establish a grid of potential values for hyperparameters.

2. Systematic Exploration: Test each parameter combination through cross-validation.

3. Optimal Selection: Identify the best hyperparameters based on performance metrics.

4.Final Model Training: Train the optimized SVM model with selected hyperparameters.

Table 3.1 Computational steps for Grid Search - SVM

SVM – GRID SEARCH
Input:heart_statlog_cleveland_hungary_final('/content/heart_statlog_cleveland_hungary_final.csv') **Output:** Classification (Have a heart disease or not have a heart disease)
Step 1: Load Dataset → Import heart disease dataset using pandas.read_csv(). **Step 2: Data Preprocessing** → Handle missing values, encode categorical features, and normalize numerical values. **Step 3: Feature Selection** → Apply **FCMIM** (Fast Conditional Mutual Information Maximization) to select relevant features. **Step 4: Class Balancing** → Implement **SMOTE (Synthetic Minority Over-sampling Technique)** to handle class imbalance. **Step 5: Standardization** → Scale numerical features using **StandardScaler()** to ensure all features have zero mean and unit variance. **Step 6: Split Data** → Divide dataset into training (80%) and testing (20%) sets. **Step 7: Define SVM Model** → Initialize Support Vector Machine (SVM) classifier with sklearn.svm.SVC(). **Step 8: Set Hyperparameter Grid** → Define a grid of hyperparameters (C, gamma, kernel) for tuning. **Step 9: Perform Grid Search** → Use GridSearchCV() to evaluate all hyperparameter combinations and find the best model.

Step 10: Train & Evaluate Model → Train optimized SVM using the best hyperparameters and compute accuracy, precision, recall, and F1-score. **Step 11: Save Optimized Model** → Save the trained SVM model using joblib.dump() for future predictions

3.2.7. Bayesian Optimization SVM

Bayesian Optimization employs a probabilistic model to intelligently navigate the hyperparameter space instead of performing an exhaustive search. It utilizes an acquisition function to balance exploration and exploitation, leading to efficient hyperparameter tuning.

Process Flow

1. Prior Distribution Setup: Define an initial probability distribution for hyperparameters.

2. Adaptive Search Strategy: Use an acquisition function to decide the next set of hyperparameters to evaluate.

3. Iterative Refinement: Update the surrogate model with new evaluations for better predictions.

4. Optimized Model Training: Train SVM using the best-found hyperparameter combination.

Table:3.2 Computational steps for Bayes SVM

SVM – BAYES
Input:heart_statlog_cleveland_hungary_final('/content/heart_statlog_cleveland_hungary_final.csv') **Output:** Classification (Have a heart disease or not have a heart disease)
Step 1: Load Dataset → Import the heart disease dataset using pandas.read_csv(). **Step 2:** Data Preprocessing → Handle missing values, encode categorical features, and normalize numerical values. **Step 3:** Feature Selection → Apply FCMIM (Fast Conditional Mutual Information Maximization) to select relevant features. **Step 4:** Class Balancing → Implement SMOTE (Synthetic Minority Over-sampling Technique) to handle class imbalance. **Step 5:** Standardization → Scale numerical features using StandardScaler() to ensure all features have zero mean and unit variance.

Step 6: Split Data → Divide dataset into training (80%) and testing (20%) sets using train_test_split(). **Step 7:** Define SVM Model → Initialize Support Vector Machine (SVM) classifier with sklearn.svm.SVC(). **Step 8:** Define Search Space → Set a probabilistic distribution for hyperparameters (C, gamma, kernel) using Bayesian optimization. **Step 9:** Perform Bayesian Optimization → Use BayesSearchCV() from skopt to iteratively refine hyperparameter tuning based on past evaluations. **Step 10:** Train & Evaluate Model → Train the optimized SVM using the best hyperparameters and compute accuracy, precision, recall, and F1-score. **Step 11:** Save Optimized Model → Save the trained SVM model using joblib.dump() for future predictions.

3.2.8. Random Search SVM

Random Search is a hyperparameter tuning method that selects hyperparameter combinations randomly instead of evaluating all possibilities. It offers a more efficient alternative to Grid Search by covering a broader range of values in fewer trials.

Process Flow

1. Search Space Definition: Specify a range of possible values for each hyperparameter.

2. Randomized Sampling: Randomly select parameter configurations for testing.

3. Evaluation Phase: Train and assess model performance for each sampled combination.

4. Best Configuration Selection: Identify and finalize the best-performing parameter set.

Table 3.3 Computational steps for Random Search - SVM

SVM – RANDOM SEARCH
Input:heart_statlog_cleveland_hungary_final('/content/heart_statlog_cleveland_hungary_final.csv') **Output:** Classification (Have a heart disease or not have a heart disease)
Step1: Load Dataset → Import heart disease dataset using pandas.read_csv(). **Step2: Data Preprocessing** → Handle missing values, encode categorical features, and normalize numerical values.

Step3: Feature Selection → Apply **FCMIM** (Fast Conditional Mutual Information Maximization) to select relevant features.

Step4: Class Balancing → Implement **SMOTE** (Synthetic Minority Over-sampling Technique) to handle class imbalance.

Step5: Standardization → Scale numerical features using StandardScaler() to ensure all features have zero mean and unit variance.

Step6: Split Data → Divide dataset into **training (80%)** and **testing (20%)** sets.

Step7: Define SVM Model → Initialize Support Vector Machine (SVM) classifier using sklearn.svm.SVC().

Step8: Set Hyperparameter Distribution → Define a random distribution for hyperparameters (C, gamma, kernel) instead of a fixed grid.

Step9: Perform Random Search → Use RandomizedSearchCV() to randomly sample and evaluate hyperparameter combinations for optimization.

Step10: Train & Evaluate Model → Train optimized SVM with the best hyperparameters and compute **accuracy, precision, recall, and F1-score**.

Step11: Save Optimized Model → Save the trained SVM model using joblib.dump() for future predictions.