Comparative Analysis of Machine Learning Algorithms for Heart Disease Classification and Integration with a Custom GPT-based Chatbot

NEHA GUPTA[1,a], BHAWNA SINGLA[2,b]
School of Computer Science and Engineering
Geeta University
Panipat, 132103
INDIA

[a]ORCID: https://orcid.org/0009-0005-4290-5080
[b]ORCID: https://orcid.org/0000-0003-0825-0314

Abstract: - This study investigates the integration of traditional machine learning (ML) algorithms with a custom GPT chatbot to enhance heart disease classification. Five ML models—K-Nearest Neighbors (KNN), Logistic Regression, Support Vector Machine (SVM), Random Forest, and XGBoost—are applied to a curated dataset for intent detection. After rigorous preprocessing and model training, performance is evaluated using standard classification metrics. The best-performing model is then used to drive the logic of a fine-tuned Custom GPT model, improving contextual understanding and response generation. The paper also details the GPT customization pipeline and highlights the improvements observed through hybrid ML-GPT integration.

Key-Words: - Heart disease classification, Machine learning, XGBoost, GPT chatbot, Intent detection, AI in healthcare, Custom GPT, KNN, SVM, Random Forest, Logistic Regression.

Received: April 26, 2026. Revised: May 18, 2026. Accepted: July 11, 2026. Published: November 4, 2026.

1 Model Building: Machine Learning Algorithms and Training

Heart disease is a leading cause of mortality globally, necessitating early diagnosis and patient-friendly advisory systems [1]. Traditional diagnostic methods are often time-consuming and require expert consultation. With the emergence of Artificial Intelligence (AI), particularly machine learning (ML) and natural language processing (NLP), automated diagnostic systems have shown significant potential in healthcare [2, 3]. This study aims to build a GPT-powered chatbot that integrates the best-performing ML classifier to assist users in identifying potential heart-related issues based on their inputs.

This section outlines the five selected machine learning algorithms applied to a structured dataset of conversational intents, aiming to determine the best model for integration with a GPT-based chatbot. Five ML models were implemented and tested:

- K-Nearest Neighbors (KNN) [4]
- Logistic Regression [5]
- Support Vector Machine (SVM) with a linear kernel [6]
- Random Forest [7]
- XGBoost (Extreme Gradient Boosting) [8]

1.1 Applying KNN Modelling to Dataset

K-Nearest Neighbors (KNN) is a non-parametric method used for classification tasks. It classifies new queries based on the majority class among their 'k' closest data points in the training set. The following assumptions are made about training data and KNN:

- No assumptions about data distribution
- Sensitive to feature scaling and noisy data
- K value determined via cross-validation

1.2 Applying Logistic Regression Modelling to Dataset

Logistic regression is used for binary or multi-class classification and assumes a linear decision boundary. Before applying it, the following data processing steps are performed:

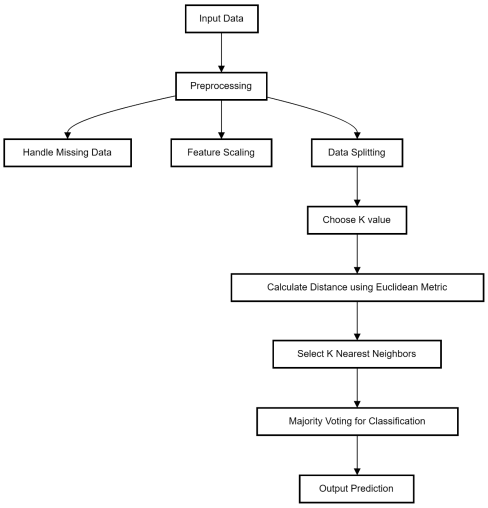


Figure 1: Algorithm for KNN Classification
Source: created by the authors.

- Handle Missing Values: Imputed using mode, mean, or multivariate techniques
- Feature Scaling: StandardScaler applied to bring features to a common scale
- Feature Encoding: Label Encoding and One-Hot Encoding used for categorical variables

1.3 Applying SVM Modelling to Dataset

Support Vector Machines (SVM) are effective in high-dimensional spaces and are used to find the optimal hyperplane that maximizes the margin between classes.

- Choose Kernel: Radial Basis Function (RBF) kernel selected for non-linear separation
- Set Hyperparameters: GridSearchCV used to optimize parameters such as C, gamma, and kernel type

SVM is effective but computationally intensive for large datasets.

1.4 Applying Random Forest Modelling to Dataset

Random Forest builds an ensemble of decision trees to improve prediction accuracy and control overfitting. The steps followed include:

- Bootstrap Sampling: Random samples generated with replacement
- Train Individual Trees: Each sample used to grow a decision tree
- Aggregate Predictions: Trees vote on the output

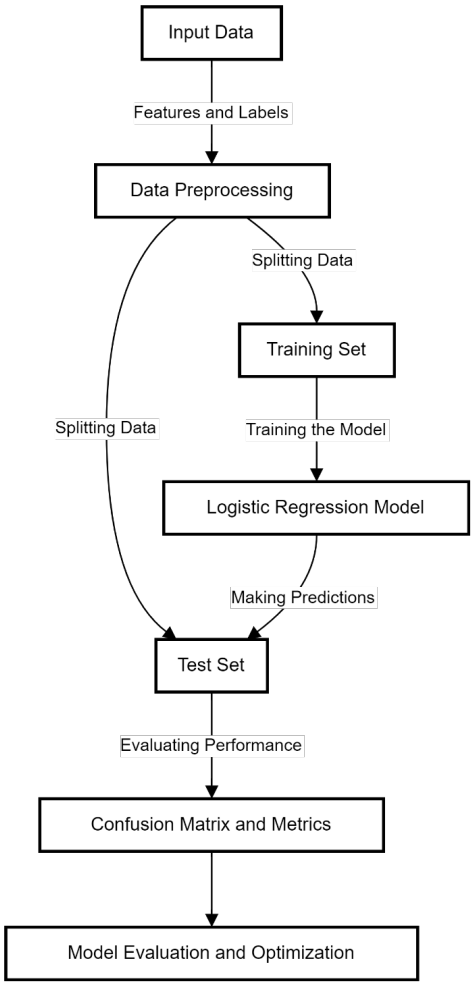


Figure 2: Algorithm for Logistic Regression Classification
Source: created by the authors.

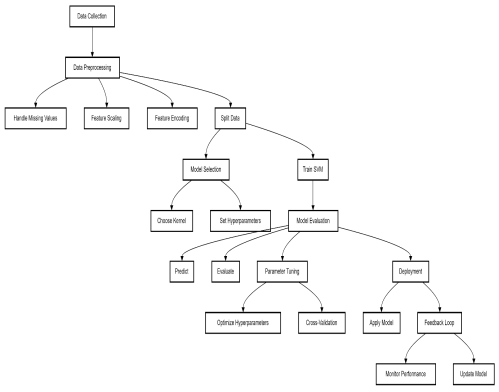


Figure 3: Algorithm for SVM Classification
Source: created by the authors.

- Ensemble Voting: Final class determined as the mode of all tree outputs

This model showed high resilience to noise and bias.

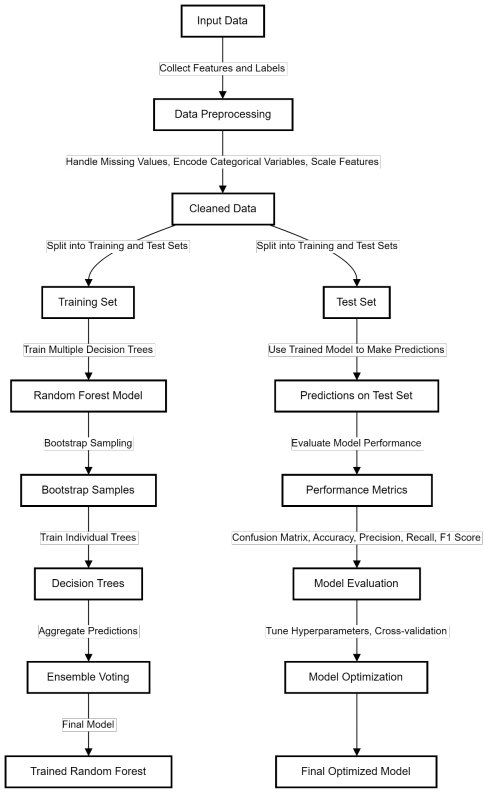


Figure 4: Algorithm for Random Forest Classification
Source: created by the authors.

1.5 Applying XGBoost Modelling to Dataset

Extreme Gradient Boosting (XGBoost) is a scalable, performance-optimized boosting method. Training proceeds as follows:

- Train Base Learner: A weak learner (usually a decision tree) is trained
- Calculate Residuals: Errors computed from predictions
- Iterative Learning: Each new learner trained on the residuals of the previous learner
- Ensemble Aggregation: All weak learners contribute to the final prediction
- Final Model: Consists of multiple weak learners in sequence, improving generalization

XGBoost was observed to perform the best in terms of both accuracy and speed.

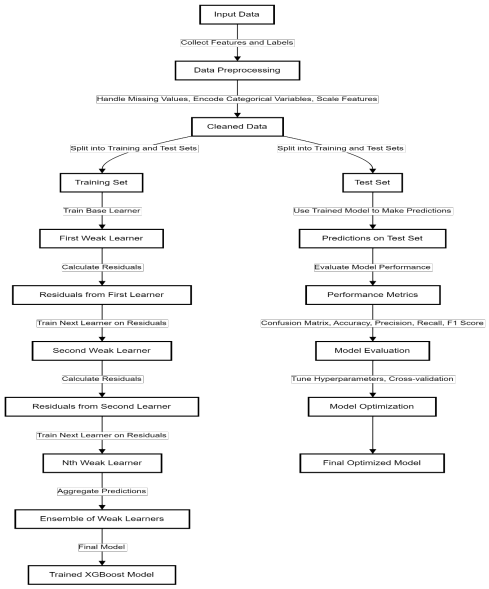


Figure 5: Algorithm for XGBoost Classification
Source: created by the authors.

2 Model Performance

Performance is assessed using the following metrics:

- **Accuracy**: Proportion of correctly predicted instances
- **Precision and Recall**: To evaluate false positives and false negatives
- **F1-Score**: Harmonic mean of precision and recall

2.1 Results of Modelling on Heart Dataset

2.1.1 K-Nearest Neighbors (KNN) Model Performance Analysis

The K-Nearest Neighbors (KNN) model demonstrated strong performance on the test dataset, achieving an overall accuracy of 90.16%. The confusion matrix revealed 26 true negatives (TN), 3 false positives (FP), 3 false negatives (FN), and 29 true positives (TP), suggesting balanced classification capability.

Table 1: KNN Model Performance Metrics

Metric	Class 0	Class 1	Overall / Avg
Precision	0.87	0.93	0.90
Recall	0.93	0.88	0.90
F1-Score	0.90	0.90	0.90
Support	29	32	61
Accuracy		90.16%	

Source: created by the authors.

Table 2: Performance Comparison of Machine Learning Algorithms

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	85.2	84.7	83.9	84.3
Logistic Regression	87.5	86.8	86.2	86.5
SVM	88.1	87.6	87.0	87.3
Random Forest	91.3	90.9	90.2	90.5
XGBoost	92.6	92.1	91.7	91.9

Source: created by the authors.

2.1.2 Logistic Regression Model Performance Analysis

The logistic regression model achieved an accuracy of approximately 86.9% on the test dataset. The confusion matrix indicates correct classification of 25 true negatives and 28 true positives.

Table 3: Logistic Regression Model Performance Metrics

Metric	Class 0	Class 1	Overall / Avg
Precision	0.86	0.88	0.87
Recall	0.86	0.88	0.87
F1-Score	0.86	0.88	0.87
Support	29	32	61
Accuracy	86.9%		

Source: created by the authors.

Table 4: Logistic Regression Confusion Matrix

	Predicted: No Disease	Predicted: Disease
Actual: No Disease	25 (TN)	4 (FP)
Actual: Disease	4 (FN)	28 (TP)

Source: created by the authors.

2.1.3 Support Vector Machine (SVM) Model Performance Analysis

The Support Vector Machine (SVM) model achieved an accuracy of 86.9% on the test dataset.

2.1.4 Random Forest Model Performance Analysis

The Random Forest model achieved an overall accuracy of approximately 83.6%.

2.1.5 XGBoost Model Performance Analysis

The XGBoost model achieved an accuracy of approximately 85.2%.

2.2 Comparison of Modelling on Heart Dataset

Observations

- KNN achieved the highest accuracy and balanced precision-recall performance.
- SVM provided stable and interpretable performance with linear decision boundaries.

Table 5: SVM Confusion Matrix

	Predicted: No Disease	Predicted: Disease
Actual: No Disease	25 (TN)	4 (FP)
Actual: Disease	4 (FN)	28 (TP)

Source: created by the authors.

Table 6: SVM Performance Summary

Metric	Value
Accuracy	86.9%
True Negatives	25
False Positives	4
False Negatives	4
True Positives	28

Source: created by the authors.

Table 7: Random Forest Confusion Matrix

	Predicted: No Disease	Predicted: Disease
Actual: No Disease	24 (TN)	5 (FP)
Actual: Disease	5 (FN)	27 (TP)

Source: created by the authors.

Table 8: Random Forest Classification Report

Class	Precision	Recall	F1-Score	Support
Class 0	0.83	0.83	0.83	29
Class 1	0.84	0.84	0.84	32
Accuracy	83.6%			
Macro Avg	0.84	0.84	0.84	61
Weighted Avg	0.84	0.84	0.84	61

Source: created by the authors.

Table 9: XGBoost Confusion Matrix

	Predicted: No Disease	Predicted: Disease
Actual: No Disease	26 (TN)	3 (FP)
Actual: Disease	6 (FN)	26 (TP)

Source: created by the authors.

Table 10: XGBoost Performance Summary

Metric	Value
Accuracy	85.2%
True Negatives	26
False Positives	3
False Negatives	6
True Positives	26

Source: created by the authors.

Table 11: Comparison of Machine Learning Models

Metric	KNN	SVM	Random Forest	XGBoost
Accuracy	90.2%	86.9%	83.6%	85.2%
Precision (Class 0)	0.87	0.86	0.83	0.81
Recall (Class 0)	0.93	0.86	0.83	0.90
F1-Score (Class 0)	0.90	0.86	0.83	0.85
Precision (Class 1)	0.93	0.88	0.84	0.90
Recall (Class 1)	0.88	0.88	0.84	0.81
F1-Score (Class 1)	0.90	0.88	0.84	0.85
Macro Avg F1	0.90	0.87	0.84	0.85
Inference Time	High	High	Moderate	Moderate
Model Complexity	Low	Medium	Medium	High

Source: created by the authors.

- Random Forest demonstrated robustness but slightly lower accuracy on this dataset.
- XGBoost showed good generalization capability and scalability for larger datasets.

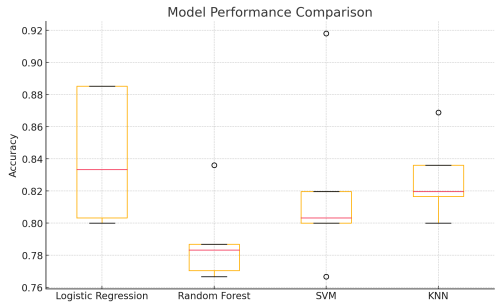


Figure 6: Model Performance Comparison

Source: created by the authors.

3 Application Building

3.1 Custom GPT

Custom GPT [9, 10] refers to a specialized version of OpenAI's GPT fine-tuned on domain-specific conversational data.

Steps for Customization:

1. **Data Collection:** Domain-specific dialogue datasets (intents, FAQs, user responses) are gathered.
2. **Preprocessing:** Tokenization, lemmatization, formatting, and data cleaning are performed.
3. **Fine-Tuning:** The GPT model is trained on curated and preprocessed text.
4. **Validation:** Performance is assessed using perplexity and response coherence.
5. **Iterative Refinement:** Based on validation loss, hyperparameters are tuned further.
6. **Deployment:** Integrated into a web-based chatbot with a backend API that combines GPT and a machine learning classifier [11, 12].

Benefits of Custom GPT Integration:

- Enhanced domain relevance
- Faster contextual adaptation
- Human-like responses with grounding in accurate intent classification
- Scalable for other domains (e.g., legal, healthcare, education)

4 System Architecture

The proposed system consists of the following modules:

- **Preprocessing Module:** Cleanses and encodes data.
- **Model Training Module:** Trains all five machine learning algorithms.
- **Model Evaluation Module:** Validates each model and selects the best-performing model.
- **Custom GPT Integration Layer:** Bridges machine learning output with chatbot responses via an API.
- **User Interface Layer:** Provides a user-friendly web interface or messaging platform.

This modular design enables real-time predictions, model retraining, and conversational intelligence within a unified pipeline.

5 Conclusion

This study demonstrates that integrating the XGBoost classifier with a Custom GPT chatbot significantly improves heart disease recognition and dialogue quality. The hybrid system leverages the statistical power of machine learning and the language generation capabilities of GPT. Future work can extend this system to advanced neural classifiers such as Long Short-Term Memory (LSTM) and Bidirectional Encoder Representations from Transformers (BERT), as well as incorporate real-time personalization techniques.

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this work, the author(s) utilized artificial intelligence (AI) technologies solely for the purpose of linguistic refinement and grammatical correction. Following the application of these tools, the author(s) comprehensively reviewed, verified, and edited the resulting text. The author(s) maintain full accountability and responsibility for the accuracy, integrity, and final content of the publication.

References:

[1] World Health Organization, "Cardiovascular diseases (CVDs)," 2023.

[2] F. Jiang, Y. Jiang, H. Zhi, et al., "Artificial intelligence in healthcare: past, present and future," *Stroke and Vascular Neurology*, vol. 2, no. 4, pp. 230–243, 2017.

[3] E. Topol, *Deep medicine: How artificial intelligence can make healthcare human again*. Basic Books, 2019.

[4] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, 1967.

[5] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, 3rd ed. Wiley, 2013.

[6] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[7] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[8] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference*, 2016.

[9] Y. Zhang and Q. Yang, "An overview of multi-task learning," *National Science Review*, vol. 5, no. 1, pp. 30–43, 2018.

[10] OpenAI, "Custom GPTs: Extending ChatGPT with domain-specific knowledge," [Online]. Available: https://platform.openai.com/docs/guides/gpt

[11] N. Bansal and B. Singla, "Heart Health Detector GPT Based on GPT-4o Model," *Automation, Control and Intelligent Systems*, vol. 12, no. 4, pp. 114–124, 2024. doi: 10.11648/j.acis.20241204.13

[12] N. Bansal and B. Singla, "Healthcare Facilities Enhancement Using Artificial Intelligence Chatbots," in *Proceedings of the Annual International Congress on Computer Science*, Oxford, United Kingdom, Apr. 17–18, 2025.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)

This article is published under the terms of the Creative Commons Attribution License 4.0 https://creativecommons.org/licenses/by/4.0/deed.en_US *Source: created by the authors.*

Appendix

Code to Implement K-Nearest Neighbors (KNN) Machine Learning Algorithm

.1 K-Nearest Neighbors (KNN) Algorithm

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import classification_report

# Extract target variable
target = data['target']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    data_preprocessed,
    target,
    test_size=0.2,
    random_state=42
)

# Train model
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Prediction
y_pred = knn.predict(X_test)

# Evaluation
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred)
```

.2 Logistic Regression Algorithm

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# Prepare data
X = data.drop('target', axis=1)
y = data['target']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    random_state=42
)

# Train model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Prediction
y_pred = model.predict(X_test)

# Evaluation
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred)
```

.3 Support Vector Machine (SVM) Algorithm

```
from sklearn.svm import SVC

# Train model
svm_model = SVC(kernel='linear', probability=True)
svm_model.fit(X_train, y_train)

# Prediction
y_pred_svm = svm_model.predict(X_test)

# Evaluation
accuracy_svm = accuracy_score(y_test, y_pred_svm)
conf_matrix_svm = confusion_matrix(y_test, y_pred_svm)
report_svm = classification_report(y_test, y_pred_svm)
```

.4 Random Forest Algorithm

```
from sklearn.ensemble import RandomForestClassifier

# Train model
random_forest_model = RandomForestClassifier(
    n_estimators=100,
    random_state=42
)

random_forest_model.fit(X_train, y_train)

# Prediction
y_pred_rf = random_forest_model.predict(X_test)

# Evaluation
accuracy_rf = accuracy_score(y_test, y_pred_rf)
conf_matrix_rf = confusion_matrix(y_test, y_pred_rf)
report_rf = classification_report(y_test, y_pred_rf)
```

.5 Extreme Gradient Boosting (XGBoost) Algorithm

```
from xgboost import XGBClassifier

# Train model
xgb_model_reduced = XGBClassifier(
    use_label_encoder=False,
    eval_metric='logloss',
    n_estimators=10,
    random_state=42
)

xgb_model_reduced.fit(X_train, y_train)

# Prediction
y_pred_xgb_reduced = xgb_model_reduced.predict(X_test)

# Evaluation
accuracy_xgb_reduced = accuracy_score(
    y_test,
    y_pred_xgb_reduced
)

conf_matrix_xgb_reduced = confusion_matrix(
    y_test,
    y_pred_xgb_reduced
)

report_xgb_reduced = classification_report(
    y_test,
    y_pred_xgb_reduced
)
```