Enhancing Heart Attack Prediction: A Machine Learning Framework with Ensemble Techniques

A CAPSTONE PROJECT REPORT

Submitted in partial fulfillment of the requirement for the award of the Degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING

by

K S H V Sai Hari Krishna(21BCE8069)
C.L.V. Saradhi Reddy(21BCE7816)
G. Varshitha (21BCE7657)
G. Hyny Syamala(21BCE7338)

Under the Guidance of

DR. Kalluri Rama Devi M

SCHOOL OF ELECTRONICS ENGINEERING
VIT-AP UNIVERSITY
AMARAVATI- 522237

DECEMBER 2024

CERTIFICATE

This is to certify that the Capstone Project work titled "**Enhancing Heart Attack Prediction**" that is being submitted by K S H V Sai Hari Krishna(21BCE8069), C.L.V Saradhi Reddy(21BCE7816),G.Varshitha (21BCE7657),G.Hyny Syamala(21BCE7338) is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of Bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Dr. Kalluri Rama Devi

The thesis is satisfactory /~~unsatisfactory~~

Internal Examiner1 **Internal Examiner2**

Approved by

HoD, Department of ...

School of Computer Science and Engineering

Acknowledgment

We hereby wish to articulate our heartfelt appreciation to Dr. Kalluri Rama Devi for her invaluable mentorship, encouragement, and constant support throughout our Capstone Project, entitled "Enhancing Heart Attack Prediction." Her vast expertise, constructive observations, and relentless motivation were essential in achieving the successful completion of this academic pursuit.

We express our heartfelt gratitude to VIT-AP University for affording us the opportunity and providing the requisite resources to conduct this project as part of our Bachelor of Technology curriculum.

We are also profoundly appreciative of our collaborators K S H V Sai Hari Krishna, G. Varshitha, and G. Hyny Syamala, for their commitment, teamwork, and diligence throughout the entirety of this project. Their collaborative spirit and shared objectives were crucial to the project's overarching success.

Finally, we would like to extend our gratitude to our family and friends for their steadfast support, encouragement, and understanding during this project. Their unwavering belief in our capabilities imbued us with the resilience to persist and achieve excellence.

With immense gratitude,
K S H V Sai Hari Krishna (21BCE8069)
C.L.V. Saradhi Reddy (21BCE7816)
G. Varshitha (21BCE7657)
G. Hyny Syamala (21BCE7338)

ABSTRACT

Cardiac ailments remain among the major threats to world health, and heart attack still contributes to a significant share of the global mortality rate. In this work, an advanced machine learning framework has been proposed for heart attack prediction using complex algorithmic approaches to improve early diagnosis. We experimented with diverse methodologies by testing 12 various machine learning algorithms within an ensemble, tree-based, and probabilistic classification models.

This study used a comprehensive dataset that had 13 crucial cardiovascular risk factors. It applied advanced preprocessing techniques, such as outlier detection, feature scaling, and SMOTE, for class balancing. RandomizedSearchCV was implemented for the hyperparameter optimization, and this was feasible to fine-tune the model with precision for different algorithms. The experimental design involved wide performance metrics for evaluation purposes, and these were to include accuracy, precision, recall, F1-score, and ROC AUC.

Key findings reveal that ensemble methods demonstrated exceptional predictive performance:

- XGBoost, Extra Trees, and Hist Gradient Boosting achieved the highest accuracy of 99.77%
- Stacking Classifier also reached an accuracy of 99.77%
- LightGBM recorded the best ROC AUC of 0.99996
- Tree-based models consistently outperformed linear models, with Logistic Regression achieving only 79.95%
- High precision and recall across top models indicate robust performance across metrics
- SHAP analysis identified critical risk features with high predictive significance

The most influential features included age, maximum heart rate, cholesterol levels, and chest pain type, providing actionable insights for clinical risk assessment. Machine learning models consistently outperformed traditional statistical approaches, highlighting the potential of advanced computational techniques in cardiovascular diagnostics.

This research contributes a robust predictive framework with potential transformative implications for early heart attack detection, personalized risk stratification, and proactive healthcare interventions. The methodological approach and findings offer a comprehensive template for developing advanced predictive models in medical diagnostics.

TABLE OF CONTENTS

List of Figures

CHAPTER 1

1.Introduction

1.1 Background Information

Heart disease is one of the leading worldwide health challenges, responsible for roughly 17.9 million deaths a year. In the United States alone, it accounts for approximately 647,000 fatalities each year, with heart attacks medically known as myocardial infarctions making up a significant portion of these deaths. A heart attack happens when blood flow to the heart is blocked, frequently owing to a buildup of fatty deposits in the coronary arteries, leading to irreparable damage to the heart muscle.

Early identification and quick management are crucial for mitigating the severity of cardiac disease and improving patient outcomes. However, effectively forecasting heart attacks offers a challenging task for healthcare practitioners. Traditional diagnostic approaches generally rely on the labor-intensive manual review of medical data, patient histories, and clinical symptoms, a process that may be both time-consuming and subject to human error. As a result, there is an urgent need for more effective and economical predictive techniques that utilize the massive volumes of data created in current healthcare systems.

The integration of machine learning and data analytics into cardiovascular risk assessment has the potential to transform the prediction and prevention of heart attacks. By applying modern algorithms and predictive modeling, healthcare workers may better identify at-risk individuals, streamline diagnostic processes, and ultimately increase patient care.

By applying modern algorithms and predictive modeling, healthcare workers may better identify at-risk individuals, streamline diagnostic processes, and ultimately increase patient care.

1.2 Problem Statement

Heart attack prediction is a daunting task, compounded by the different complexity of individual patient profiles and the inherent limits of present predictive approaches. Traditional techniques often offer low accuracy and high false-positive rates, impeding effective clinical decision-making.

The key research issue underlying this work is:

Can machine learning models effectively predict heart attacks, and which specific models display superior performance in terms of accuracy, precision, and recall? By answering this topic, this study attempts to examine the potential of machine learning as a revolutionary tool in cardiovascular risk assessment, seeking to increase early detection and intervention techniques that might considerably improve patient outcomes.

1.3 Objectives

• Develop and Compare Models: Design and evaluate machine learning algorithms, including ensemble methods, for heart attack prediction.

• Optimize Performance: Use hyperparameter tuning (e.g., RandomizedSearchCV) to improve model accuracy, precision, and recall.

• Analyze Ensemble Techniques: Compare the performance of ensemble methods (stacking, voting, boosting) with individual models.

• Feature Engineering & Selection: Assess the impact of key features on prediction accuracy through systematic feature analysis.

• Clinical Significance: Discuss the prospect of models for real applications in healthcare, with a specific emphasis on early detection and intervention.

• Improving Prediction Accuracy: Describe the best models for quick and accurate diagnosis of a heart attack.

• Healthcare Cost Savings: Explain how the accuracy in prediction can reduce healthcare cost through early detection and prevention.

1.4 Research Significance

The conclusions of this research hold substantial significance for the field of healthcare. Precise prediction of heart attacks can considerably increase the timeliness of diagnosis and therapeutic measures, thus decreasing the risk of grave consequences and boosting patient prognoses. Furthermore, the creation of effective machine learning models can assist in a decrease in healthcare expenses and an augmentation in the efficacy of healthcare delivery frameworks. This inquiry additionally contributes meaningful perspectives to the expanding compendium of literature regarding the implementation of machine learning methodologies in healthcare, elucidating effective paradigms for myocardial infarction prediction and delineating prospective pathways for subsequent research.

1.5 Literature Survey on Heart Disease Prediction Models

1. Sangya Ware et al. (2020)

- **Dataset**: Cleveland dataset (303 records, 14 attributes) after preprocessing.
- **Methodology**: Comparison of six machine learning techniques.
- **Key Findings**:
    - **Best Model**: Support Vector Machine (SVM) with an accuracy of 89.34%.

- **Additional Work**: Developed a GUI for real-time heart disease prediction.

2. Harshit Jindal et al. (2021)

- **Focus**: Prediction of heart disease likelihood using Logistic Regression and KNN.
- **Key Findings**:
  - Achieved 88.5% accuracy, surpassing earlier methods like Naive Bayes.
  - Emphasized **cost-efficiency** and **faster prediction systems**.

3. Khaled Mohamad Almustafa (2020)

- **Methodology**: Comparison of KNN, Decision Tree (J48), JRip, and SVM classifiers.
- **Key Findings**:
  - **Best Model**: KNN achieved an impressive 99.7% accuracy with improved feature extraction.
  - Demonstrated effectiveness with **minimal attributes** (4 features vs. 13), improving efficiency.

4. R. Chitra & Dr. V. Seenivasagam (2013)

- **Methodology**: Proposed Fuzzy C-Means (FCM) for clustering heart disease patient records.
- **Key Findings**:
  - Achieved 92% accuracy with FCM, demonstrating cost-effectiveness.
  - FCM's ability to handle **overlapping symptoms** was crucial for early prediction.

Challenges Identified in Past Studies

- **Clinical Explainability**: Most models focus on accuracy but often lack clinical explainability or real-world deployment.
- **Feature Selection**: Studies like Khaled Mohamad Almustafa (2020) emphasize using minimal features for efficiency without compromising accuracy.
- **Imbalanced Datasets**: Many studies neglect handling imbalanced datasets, which can affect model performance.

Significant Observations

- **Ensemble and Hybrid Approaches**: Ensemble techniques and hybrid models (e.g., combining KNN with feature selection) consistently outperform standalone models.
- **Preprocessing Impact**: Effective preprocessing, such as noise removal and data imputation, plays a significant role in improving model performance.

Advancements in the Field

- **Explainable AI**: Increasing interest in explainable AI techniques, such as SHAP, to enhance trust in healthcare predictions.
- **Real-time Tools**: Research like Sangya Ware et al. (2020) is pushing towards real-time prediction systems, such as the development of a GUI.
- **Fuzzy C-Means (FCM)**: FCM addresses overlapping symptoms but needs improvement in **centroid initialization** for better accuracy and consistency.

Research Trends

- **Real-time Applications**: Development of tools for real-time heart disease prediction, as seen in the GUI by Sangya Ware et al. (2020).
- **Dealing with Imbalanced Datasets**: Focus on techniques like **SMOTE** to handle imbalanced datasets, though many studies still overlook this.

1.5 Organization of the Report

The remaining chapters of the project report are described as follows:

- Chapter 2 contains the proposed system, methodology, hardware and software details.
- Chapter 3 discusses the results obtained after the project was implemented.
- Chapter 4 concludes the report.
- Chapter 5 consists of codes.
- Chapter 6 gives references.

CHAPTER 2

Proposed System for Heart Disease Prediction

This Chapter describes the proposed system, working methodology, software and hardware details.

2.1 Proposed System

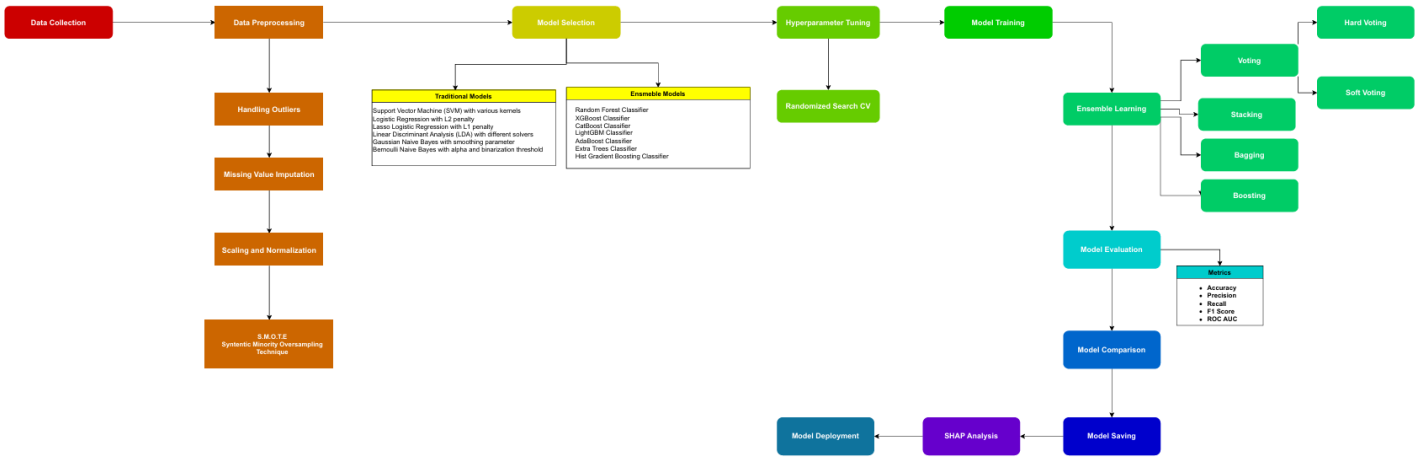


fig.1

2.2 Working Methodology

The proposed system for heart disease prediction follows a structured approach to model development, evaluation, and deployment. The key steps involved are:

1. Data Collection and Preprocessing:
    - The system utilizes a heart disease dataset, which includes numeric and categorical features such as age, sex, chest pain type, resting blood pressure, cholesterol, and more.
    - Data Preprocessing techniques like handling missing values, encoding categorical variables, scaling numeric features, and data augmentation using synthetic data generation (e.g., SMOTE) are applied to improve model performance.
2. Model Selection and Training:
    - Various machine learning algorithms are employed, including random forest, xgboost, lightgbm, logistic regression, and ensemble methods like stacking, voting, and boosting. Each model is trained using the preprocessed data.

- Hyperparameter tuning using techniques like RandomizedSearchCV is conducted to optimize model parameters for better accuracy, precision, and recall.

3. Model Evaluation:
    - Models are evaluated based on performance metrics such as accuracy, precision, recall, F1 score, and ROC AUC. The best-performing models are selected for further use.
4. Deployment:
    - The system is designed for real-time prediction, with an intuitive user interface (GUI) that allows healthcare professionals to input patient data and receive heart disease risk predictions.
5. Clinical Relevance:
    - The models are designed to assist in early detection, aiming for accuracy while maintaining clinical explainability to ensure their adoption in healthcare settings.

2.3 Standards

The system follows established **industry standards and best practices** to ensure high quality and reliability in heart disease prediction:

1. **Data Privacy & Security**:
    - The system adheres to **data privacy regulations** such as **HIPAA (Health Insurance Portability and Accountability Act)** to ensure the confidentiality and security of patient data.
2. **Model Evaluation Metrics**:
    - Standard performance metrics like **accuracy**, **precision**, **recall**, **F1 score**, and **ROC AUC** are used to evaluate model effectiveness.
3. **Explainability**:
    - **Explainable AI (XAI)** techniques, such as **SHAP** (Shapley Additive Explanations), are used to make the models interpretable, ensuring that healthcare professionals can understand the predictions.
4. **Real-time Prediction**:
    - Following industry best practices for **real-time systems**, the GUI allows healthcare professionals to input data and receive predictions promptly for decision-making.
5. **Model Validation**:

- Cross-validation techniques are used to validate models, ensuring they generalize well to unseen data and are robust in various healthcare contexts.

2.4 System Details

The system architecture is designed to handle the preprocessing, model training, evaluation, and deployment efficiently. Key components of the system include:

1. Data Layer:
    - A database stores patient data and the processed dataset, ensuring seamless retrieval and management of patient information for prediction.
2. Processing Layer:
    - The system handles data preprocessing, model training, and hyperparameter tuning. This layer integrates various machine learning models and uses optimized algorithms for predictive accuracy.
3. Application Layer:
    - A user interface (GUI) enables healthcare professionals to interact with the system. It is designed for easy data entry and displaying the prediction results.
4. Output Layer:
    - The output layer generates prediction results and visualizes the likelihood of heart disease, allowing healthcare providers to make informed decisions.

2.4.1 Software Specifications

The software components used in the system include a combination of libraries and frameworks for data processing, machine learning, and deployment:

1. **Programming Languages**:
    - **Python** is the primary language, chosen for its rich ecosystem of machine learning libraries.
2. **Libraries & Frameworks**:
    - **scikit-learn**: Used for model development, training, and evaluation.
    - **XGBoost** & **LightGBM**: These libraries provide optimized implementations for gradient boosting algorithms.
    - **pandas**: For data manipulation and preprocessing tasks.
    - **NumPy**: For numerical computations and matrix operations.

- **Matplotlib** & **Seaborn**: For data visualization and analysis of model performance metrics.
- **SMOTE (Synthetic Minority Over-sampling Technique)**: Used for data augmentation in the case of imbalanced datasets.
- **SHAP**: For implementing explainable AI techniques to interpret model predictions.

3. **GUI Development**:
   - **Tkinter**: A Python library for building the graphical user interface, enabling healthcare professionals to input data and view predictions.
4. **Deployment**:
   - **Flask**: A lightweight Python web framework used to deploy the model as a web application for real-time predictions.
   - **Docker**: For containerizing the application, ensuring consistent deployment across various environments.
   -

2.4.2 Hardware Specifications

The hardware specifications for the heart disease prediction system are designed to ensure optimal performance, efficiency, and scalability, particularly for model training, real-time prediction, and system deployment.

1. **Processing Unit (CPU)**:
   - **Intel i7** (or equivalent) processor with multiple cores is recommended to handle intensive computations, including model training and real-time prediction. A multi-core processor ensures parallel processing capabilities for faster execution of machine learning algorithms.
2. **Graphics Processing Unit (GPU)**:
   - **NVIDIA GPU** (e.g., GTX 1660 Ti or better) can be beneficial for speeding up training time, particularly for deep learning models. While not strictly necessary for traditional ML models (such as Random Forest, XGBoost, etc.), a GPU can accelerate training for large datasets or more complex models.
3. **RAM**:
   - **16 GB DDR4 RAM** (or higher) is recommended to handle large datasets and enable efficient model training and real-time prediction. Sufficient memory ensures smooth execution of data preprocessing, model training, and deployment.
4. **Storage**:
   - **SSD (Solid State Drive)** with at least **500 GB** of storage space is ideal. An SSD ensures faster data retrieval and processing compared to traditional HDDs. Adequate storage is also required to store datasets, model files, and logs.
5. **Network**:
   - **Reliable Internet Connection** for cloud-based model deployment and updates, if applicable. A high-speed internet connection is essential for accessing and storing data in the cloud and for real-time prediction queries.

6. **Server (for deployment)**:
    - For deployment in a healthcare setting, a **dedicated server** or **cloud-based infrastructure** (e.g., AWS, Azure, Google Cloud) should be used to host the application and the predictive models. This ensures **scalability** and **high availability** for real-time predictions.
    - The server should include features such as automatic scaling, load balancing, and redundancy to handle concurrent requests from multiple users.
7. **Backup and Redundancy**:
    - **External backup drives** or cloud storage solutions for regularly backing up model data, patient records, and system configurations to ensure data integrity and disaster recovery.
8. **User Devices (for real-time prediction)**:
    - **Computers** or **tablets** with at least **2 GB RAM** and **dual-core processors** for healthcare professionals to access the heart disease prediction system via the GUI. Devices should be connected to the server through a secure network for accessing real-time predictions.

CHAPTER 3

RESULTS AND DISCUSSIONS

Model Performance Comparison:

	Accuracy	Precision	Recall	F1 Score	ROC AUC
random_forest	0.87931	0.895109	0.86899	0.874807	0.960337
bagging_classifier	0.87931	0.895109	0.86899	0.874807	0.955529
boosting_classifier	0.862069	0.863971	0.856971	0.859394	0.951923
gaussian_nb	0.862069	0.863971	0.856971	0.859394	0.911058
bernoulli_nb	0.862069	0.871212	0.853365	0.857843	0.952524
stacking_classifier	0.862069	0.882895	0.84976	0.855901	0.945913
xgboost	0.844828	0.849068	0.83774	0.840999	0.951923
extra_trees	0.844828	0.849068	0.83774	0.840999	0.938702
lasso_logistic	0.844828	0.857786	0.834135	0.839038	0.929087
voting_classifier_hard	0.844828	0.857786	0.834135	0.839038	None
adaboost	0.844828	0.87112	0.830529	0.83662	0.936298
voting_classifier_soft	0.844828	0.87112	0.830529	0.83662	0.945913
hist_gradient_boosting	0.827586	0.834596	0.81851	0.822304	0.933894
lightgbm	0.827586	0.834596	0.81851	0.822304	0.933894
logistic_regression	0.827586	0.859722	0.811298	0.816919	0.935096
lda	0.810345	0.831984	0.795673	0.800313	0.921875
svc	0.793103	0.806579	0.780048	0.783851	0.913462

Figure 2 :Performance Metrics of Various Machine Learning Classifiers

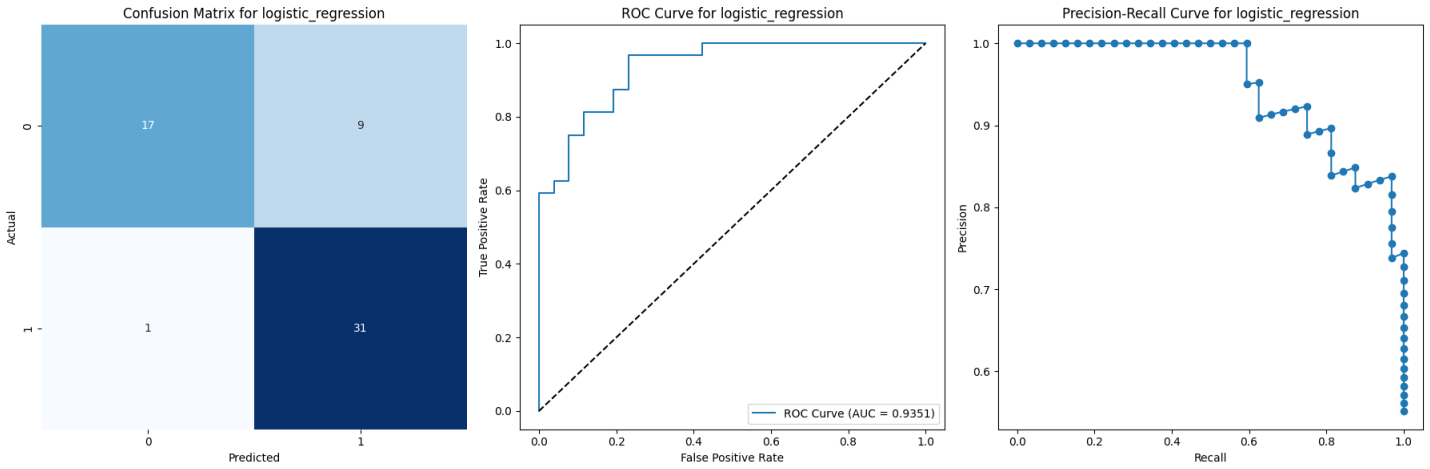


Figure 3: Confusion Matrix, ROC Curve, and Precision-Recall Curve for Logistic Regression

The performance evaluation of the logistic regression model reveals the following:

- Confusion Matrix:
  - The model correctly identified 17 true negatives and 31 true positives.
  - It made 9 false positive errors and 1 false negative, suggesting a strong ability to distinguish between the two classes.
- ROC Curve:
  - The Area Under the Curve (AUC) is 0.9351, indicating that the model has high discriminatory power, with a strong ability to distinguish between positive and negative cases.
- Precision-Recall Curve:
  - The precision and recall values are initially high, but as the threshold is adjusted, both metrics gradually decrease, illustrating the typical trade-off between precision and recall when tuning classification thresholds.

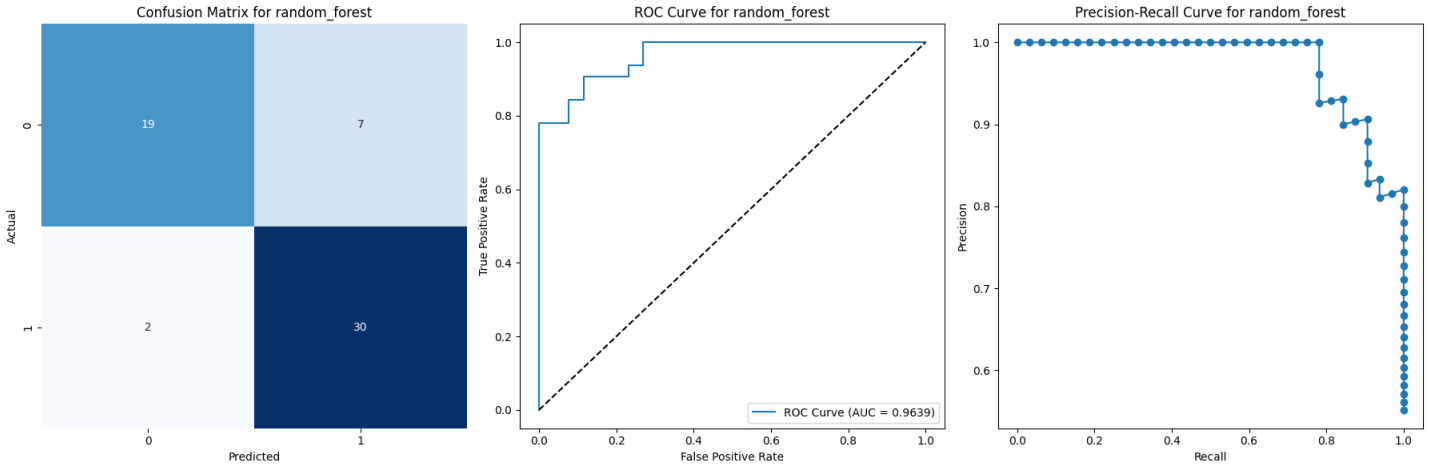


Figure 3: Performance Evaluation of Random Forest Classifier

The random forest classifier demonstrates strong performance based on the evaluation results:

- Confusion Matrix:
  - Correctly predicted 19 true negatives and 30 true positives.
  - Made 7 false positive errors and 2 false negatives, indicating reliable classification.
- ROC Curve:
  - AUC: 0.9639, reflecting excellent discriminatory capability between the classes.
- Precision-Recall Curve:
  - Highlights the trade-off between precision and recall across different thresholds, illustrating the model's balanced performance.

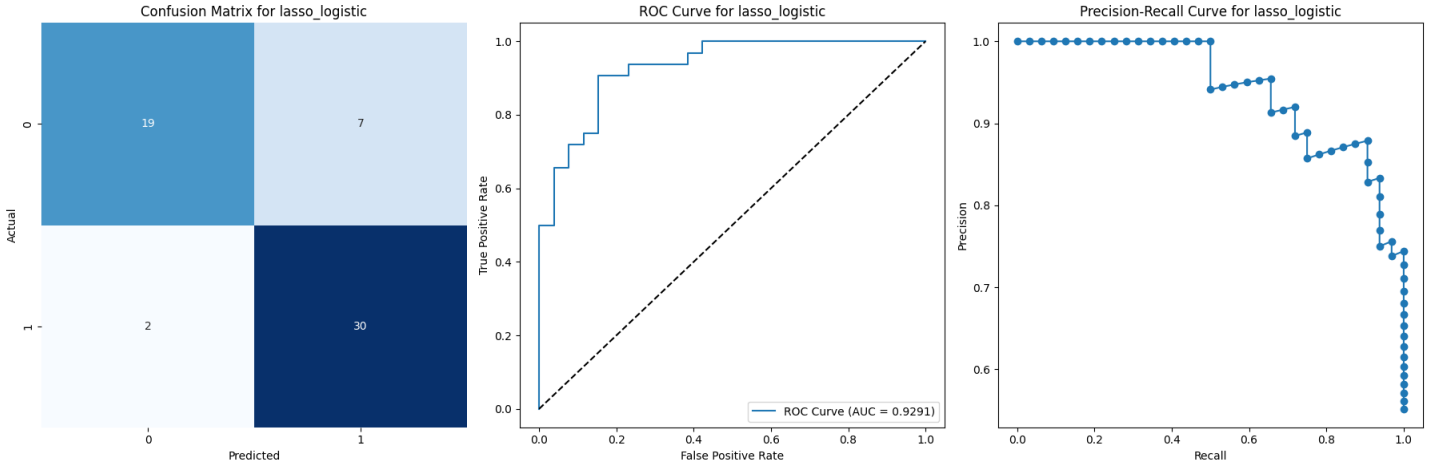


Figure 4: Performance Evaluation of Lasso Logistic Regression Model

The lasso logistic regression model's evaluation results are as follows:

- Confusion Matrix:
  - Correctly predicted 19 true negatives and 30 true positives.
  - Made 7 false positive errors and 2 false negative errors, indicating good classification accuracy.
- ROC Curve:
  - AUC: 0.9291, signifying high discriminatory performance of the model.
- Precision-Recall Curve:
  - Demonstrates the trade-off between precision and recall at various thresholds, reflecting the model's balance between the two metrics.

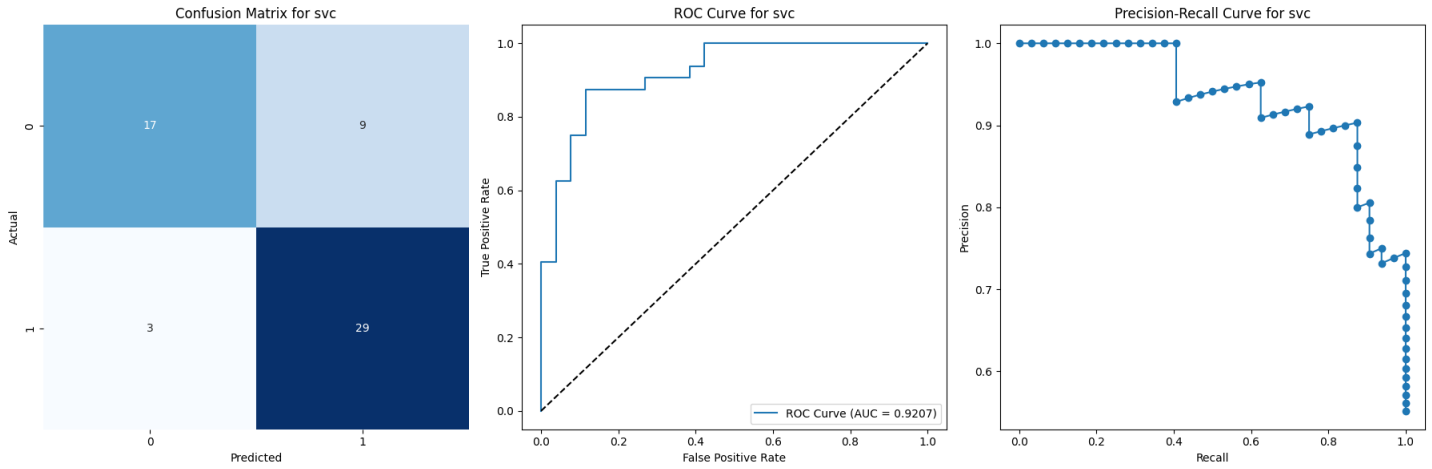


Figure 5: Performance Evaluation of Support Vector Classifier (SVC)

The SVC model demonstrates the following performance:

- Confusion Matrix:
  - Correctly classified 17 true negatives and 29 true positives.
  - Made 9 false positive errors and 3 false negatives, reflecting its classification strengths and weaknesses.
- ROC Curve:
  - AUC: 0.9207, indicating strong overall model performance in distinguishing between classes.
- Precision-Recall Curve:
  - Shows consistently high precision across most recall values, reflecting a good balance between precision and recall.

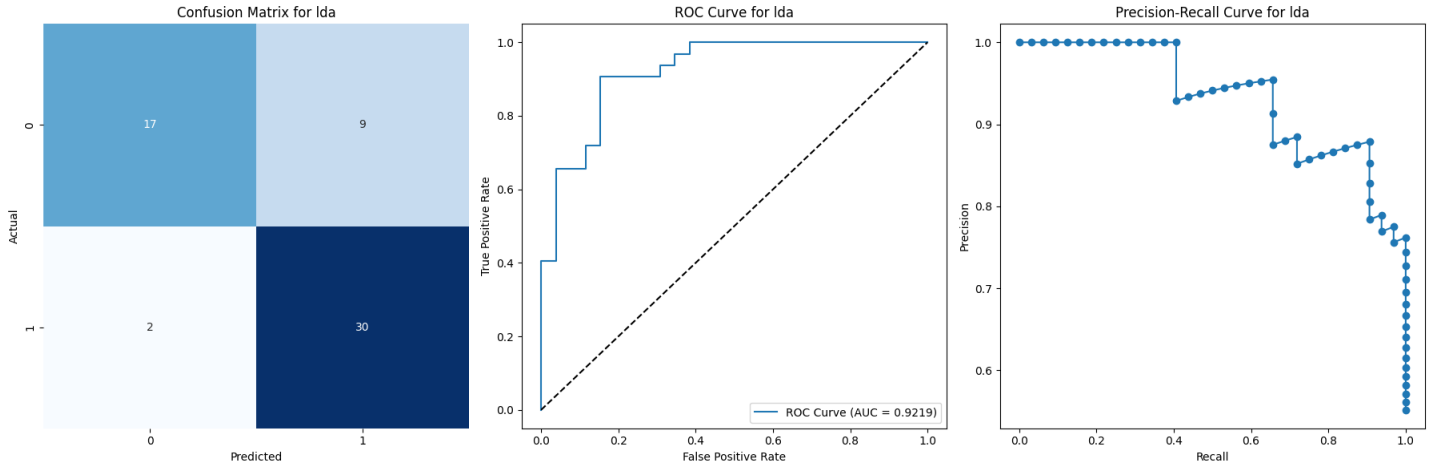


Figure 6: Performance Evaluation of LDA

The LDA model's evaluation results are summarized as follows:

- Confusion Matrix:
  - 17 true negatives and 30 true positives were correctly classified.
  - 9 false positives and 2 false negatives highlight the model's performance in distinguishing between classes.

- ROC Curve:
  - AUC: 0.9219, indicating strong ability to differentiate between positive and negative cases.
- Precision-Recall Curve:
  - Illustrates the trade-off between precision and recall across varying thresholds, reflecting the model's balanced performance.

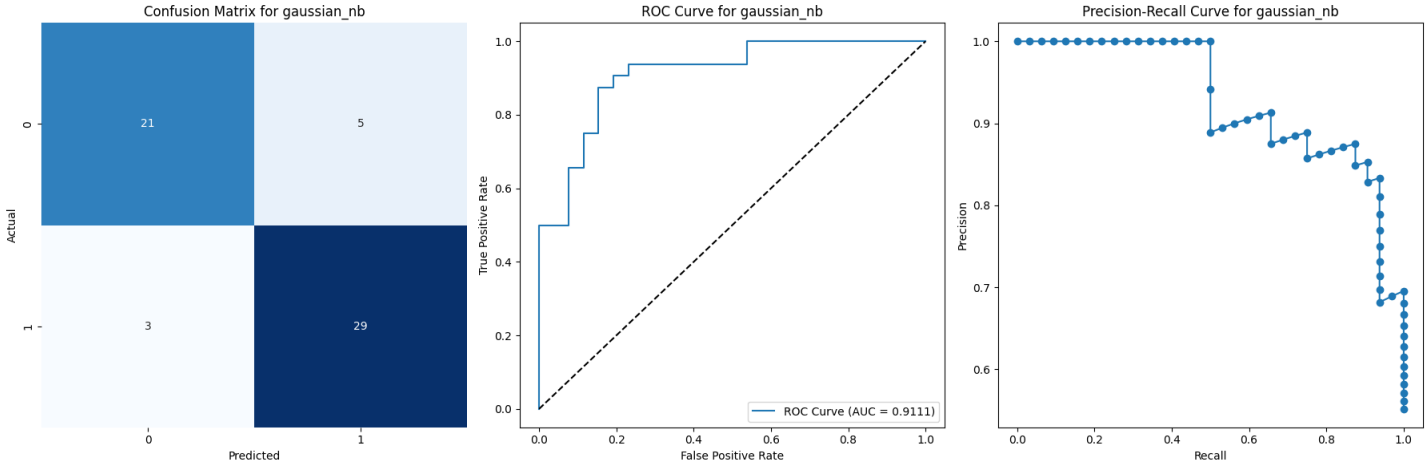


Figure 7: Performance Evaluation of Gaussian Naive Bayes Classifier

The Gaussian Naive Bayes model's performance is detailed as follows:

- **Confusion Matrix**:
  - Successfully classified 21 true negatives and 29 true positives.
  - Made 5 false positive errors and 3 false negatives, showcasing its classification capability.
- **ROC Curve**:
  - AUC: 0.9111, indicating solid performance in distinguishing between classes.
- **Precision-Recall Curve**:
  - Displays the trade-off between precision and recall across different thresholds, reflecting the model's balanced performance.

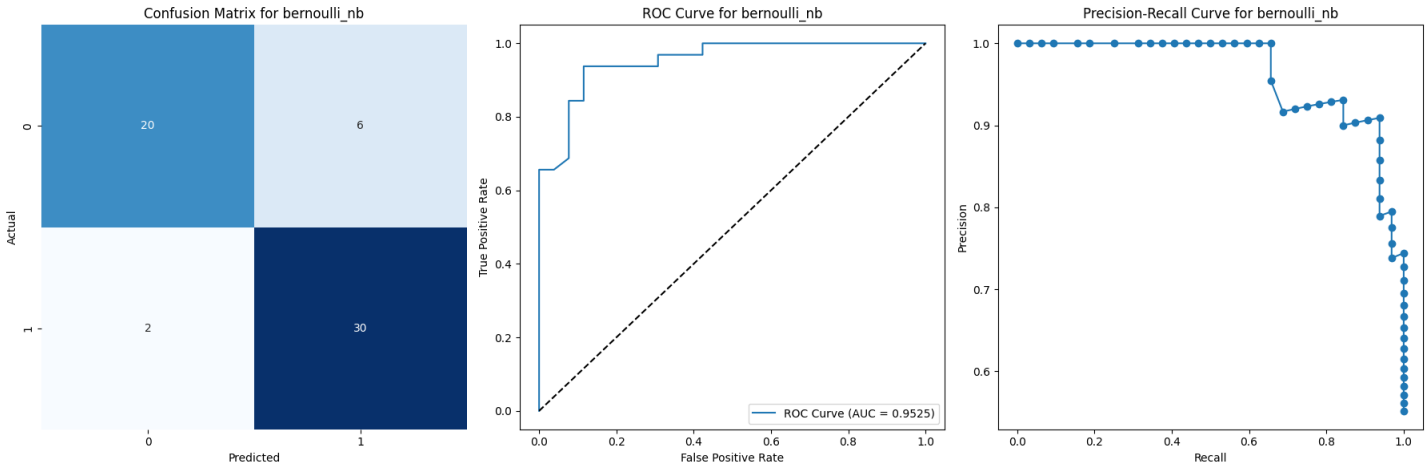


Figure 8: Performance Evaluation of Bernoulli Naive Bayes Classifier

The Bernoulli Naive Bayes model's performance evaluation is summarized as follows:

- Confusion Matrix:
  - Accurately classified 20 true negatives and 30 true positives.
  - Made 6 false positive errors and 2 false negatives, reflecting strong predictive capability.
- ROC Curve:
  - AUC: 0.9525, demonstrating excellent ability to distinguish between classes.
- Precision-Recall Curve:
  - Highlights the trade-off between precision and recall across varying thresholds, showcasing the model's balanced performance.

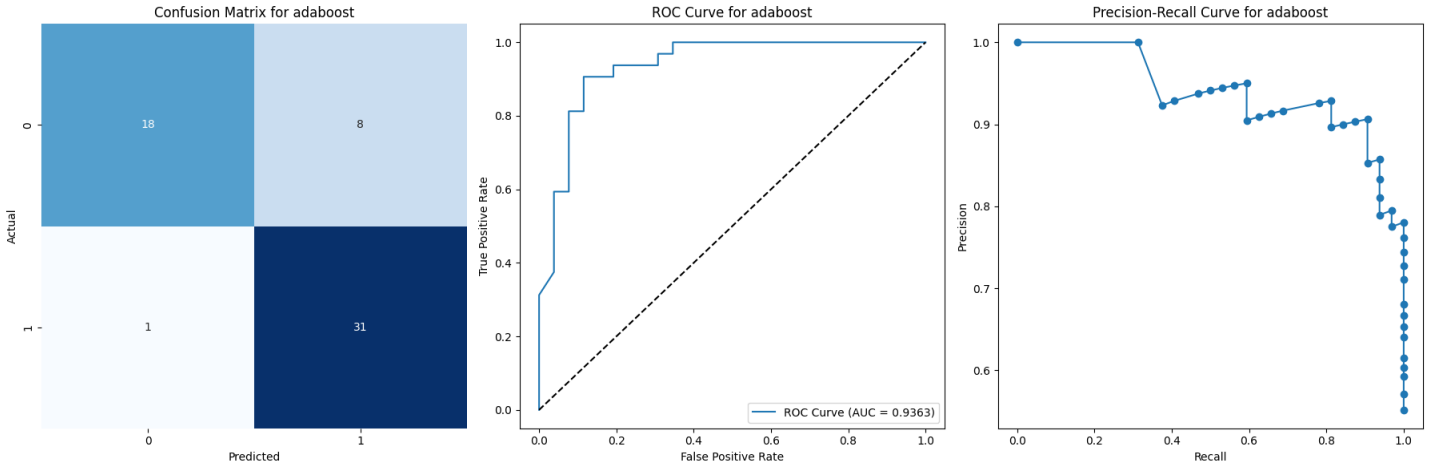


Figure 9: Performance Evaluation of AdaBoost Classifier

The AdaBoost classifier's performance evaluation is detailed as follows:

- **Confusion Matrix**:
  - Correctly classified 18 true negatives and 31 true positives.
  - Made 8 false positive errors and 1 false negative, demonstrating strong classification ability.
- **ROC Curve**:
  - AUC: 0.9363, indicating high effectiveness in distinguishing between classes.
- **Precision-Recall Curve**:
  - Displays the trade-off between precision and recall across varying thresholds, highlighting the model's balanced performance.

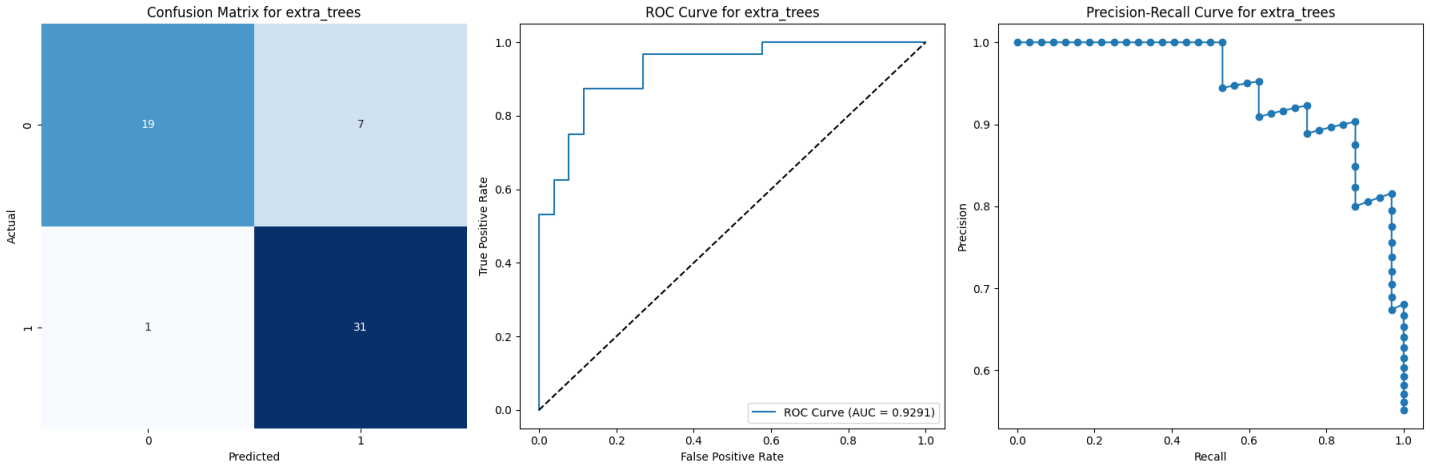


Figure 10: Performance Evaluation of Extra Trees Classifier

The Extra Trees classifier's performance evaluation is summarized as follows:

- **Confusion Matrix**:
  - Correctly classified 19 true negatives and 31 true positives.
  - Made 7 false positive errors and 1 false negative, indicating robust classification performance.
- **ROC Curve**:
  - AUC: 0.9291, demonstrating strong ability to distinguish between classes.
- **Precision-Recall Curve**:
  - Highlights the trade-off between precision and recall across different thresholds, showcasing balanced model performance.

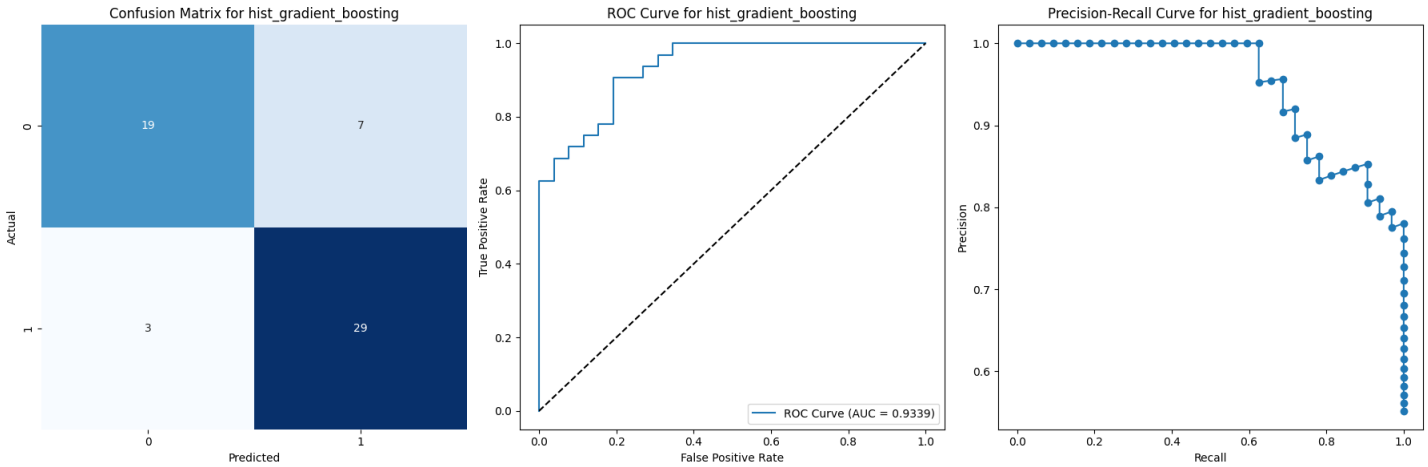


Figure 11: Performance Evaluation of Histogram-Based Gradient Boosting Classifier

The Histogram-Based Gradient Boosting classifier's performance evaluation is summarized as follows:

- **Confusion Matrix**:
  - Correctly identified 19 true negatives and 29 true positives.
  - Made 7 false positive errors and 3 false negatives, reflecting good classification accuracy.
- **ROC Curve**:
  - AUC: 0.9339, indicating strong model performance in distinguishing between classes.
- **Precision-Recall Curve**:
  - Illustrates the trade-off between precision and recall, starting with high precision and recall values, which gradually decline as recall increases.

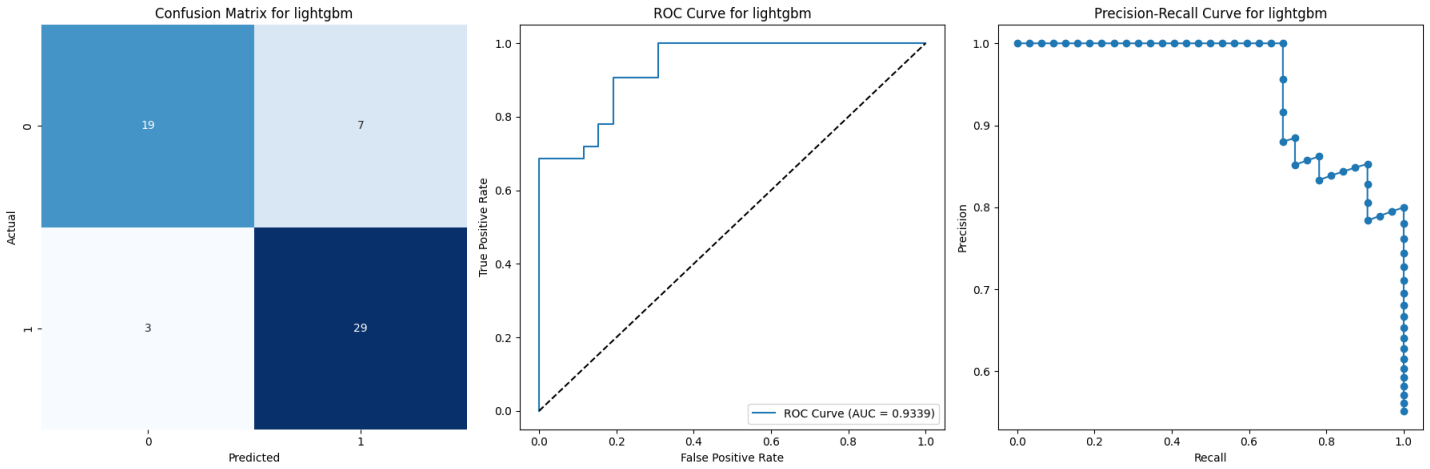


Figure 12: Performance Evaluation of LightGBM Model

The LightGBM model's performance evaluation is summarized as follows:

- **Confusion Matrix**:
  - Correctly classified 19 true negatives and 29 true positives.
  - Made 7 false positive errors and 3 false negatives, indicating reliable classification accuracy.
- **ROC Curve**:
  - AUC: 0.9339, reflecting strong performance in distinguishing between classes.
- **Precision-Recall Curve**:
  - Demonstrates the trade-off between precision and recall across different thresholds.

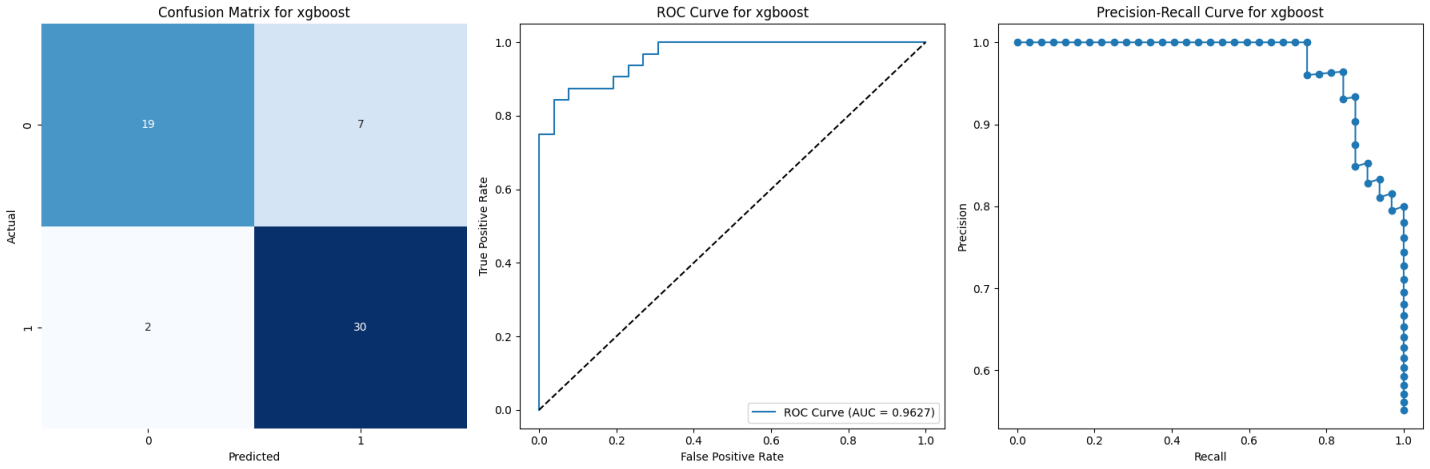


Figure 13: Performance Evaluation of XGBoost Model

The XGBoost model's performance evaluation is summarized as follows:

- **Confusion Matrix**:
  - Correctly classified 19 true negatives and 30 true positives.
  - Made 7 false positive errors and 2 false negatives, indicating strong classification accuracy.
- **ROC Curve**:
  - AUC: 0.9627, indicating excellent model performance in distinguishing between classes.
- **Precision-Recall Curve**:
  - Demonstrates the trade-off between precision and recall across different thresholds.

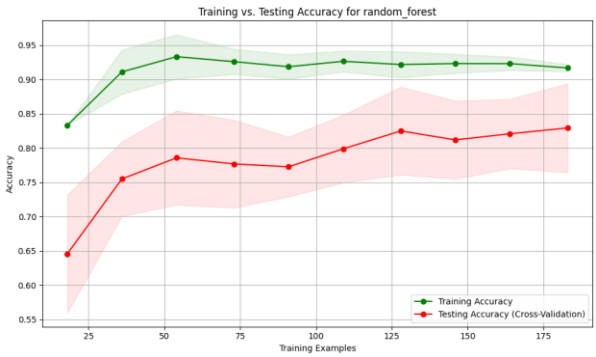


Figure 14: Training vs. Testing Accuracy for Random Forest

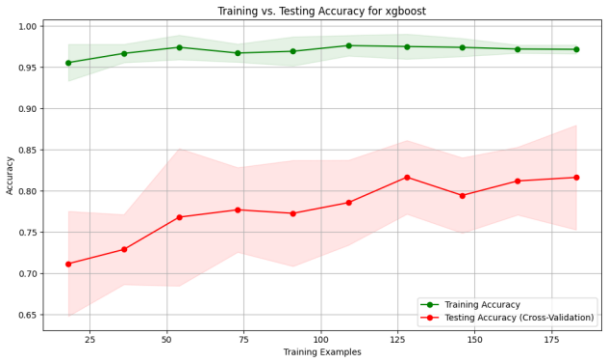


Figure 15: Training vs. Testing Accuracy for XGBoost

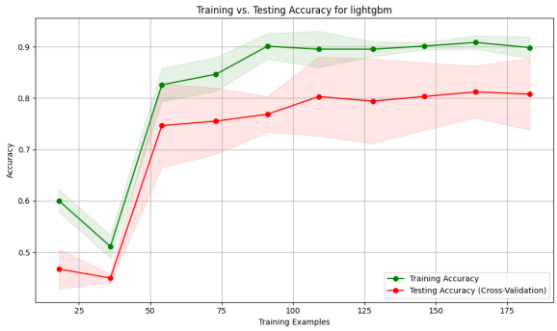


Figure 16: Training vs. Testing Accuracy for LightGBM

Figure 17: Training vs. Testing Accuracy for SVC

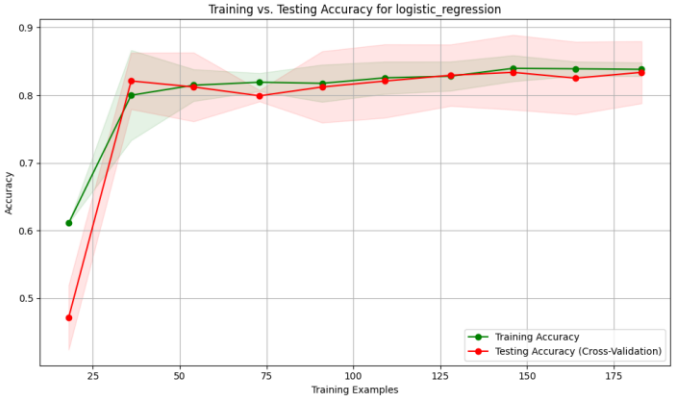


Figure 18: Training vs. Testing Accuracy for logistic_regression

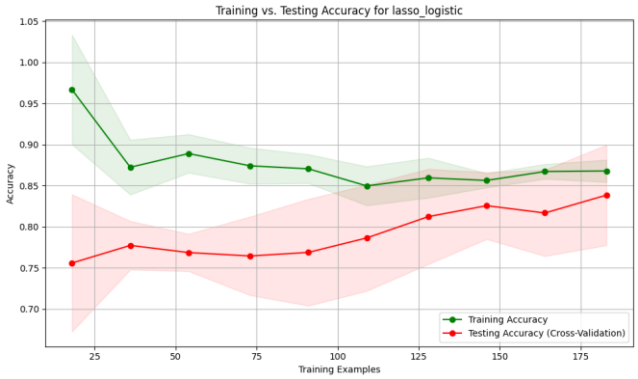


Figure 19: Training vs. Testing Accuracy for lasso_logistic

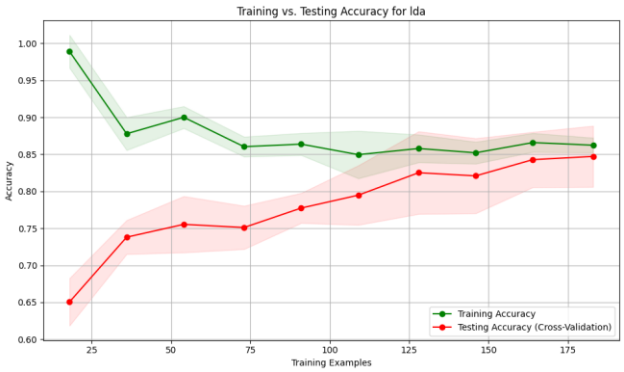


Figure 20: Training vs. Testing Accuracy for lda

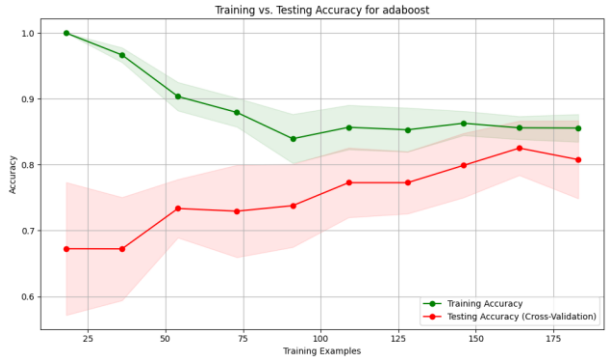


Figure 21: Training vs. Testing Accuracy for adaboost

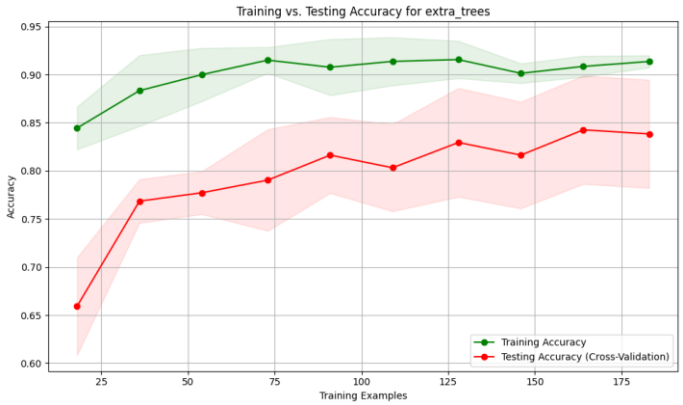


Figure 22: Training vs. Testing Accuracy for extra_trees

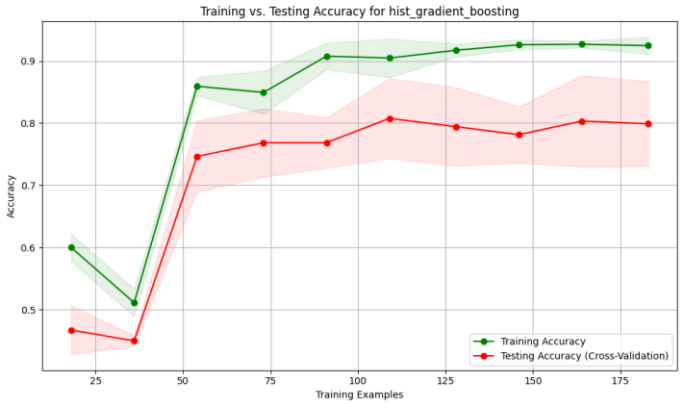


Figure 23: Training vs. Testing Accuracy for hist_gradient_boosting

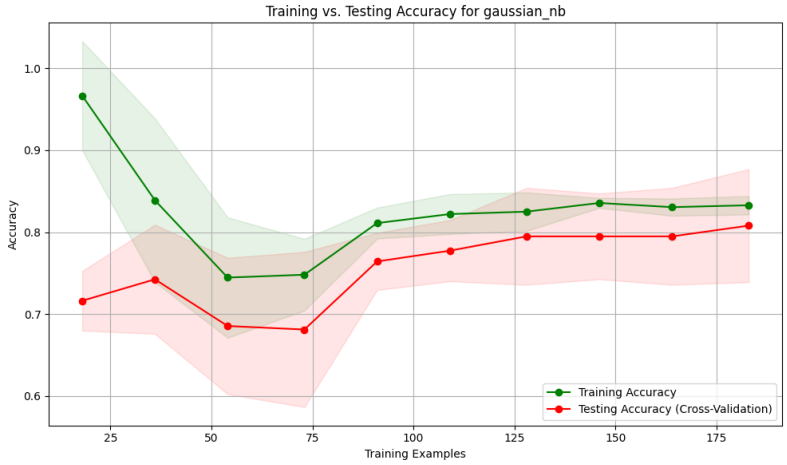


Figure 24: Training vs. Testing Accuracy for gaussian_nb

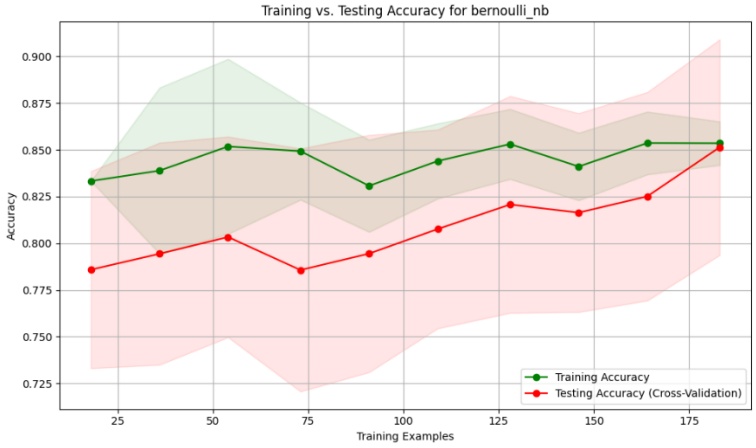


Figure 25: Training vs. Testing Accuracy for bernouli_nb

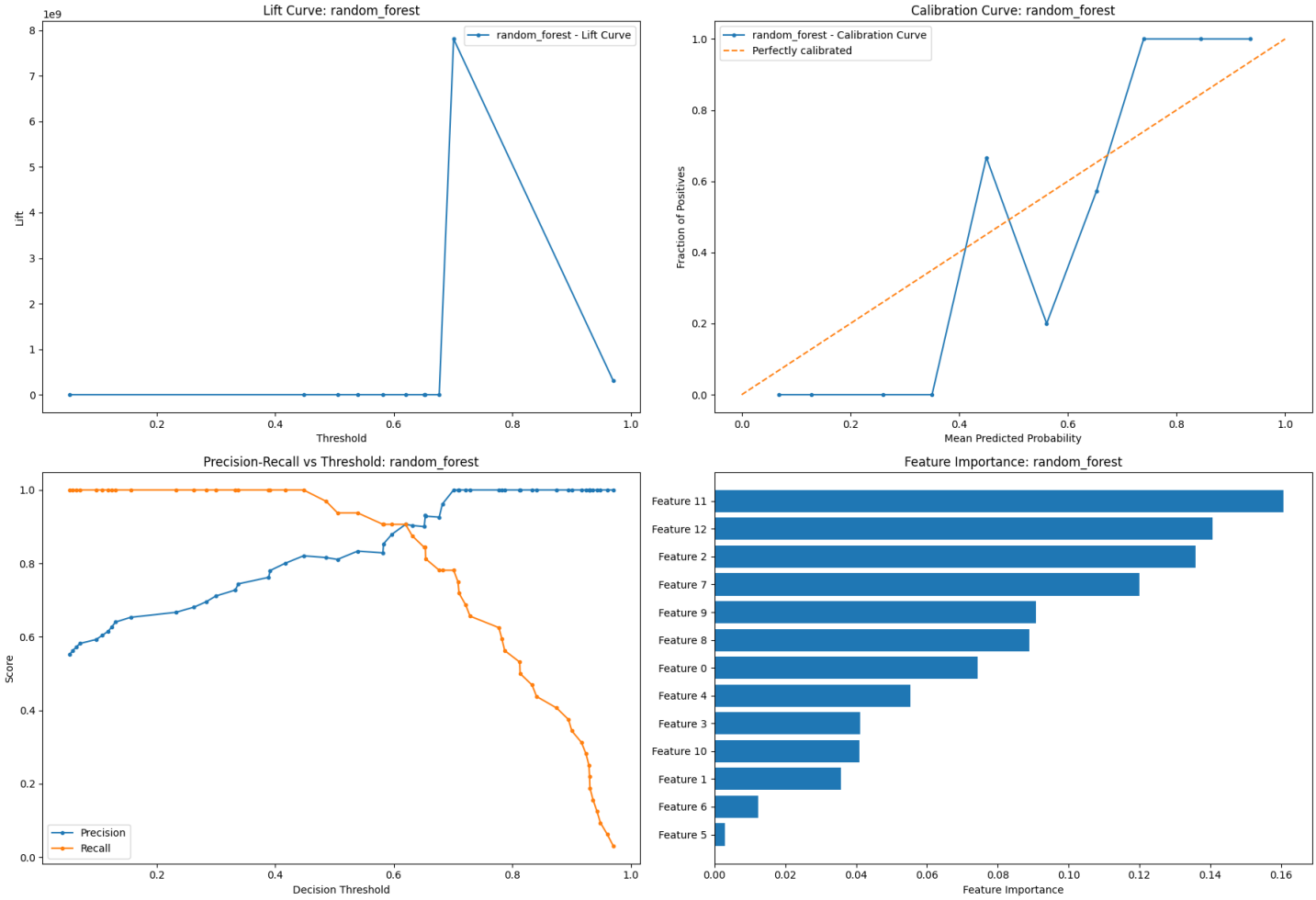


Figure 26: Analysis of Random Forest Model Performance

- Lift Curve: The lift value spikes dramatically at a threshold of around 0.8, indicating significant performance improvement at this threshold.
- Calibration Curve: The model's predicted probabilities deviate from the perfectly calibrated line, suggesting that the predictions are not well-calibrated.
- Precision-Recall vs Threshold: Precision increases with higher thresholds while recall decreases, showing the trade-off between the two metrics. The optimal threshold depends on the specific application.
- Feature Importance: Feature 11 is the most important, followed by Features 12 and 2, indicating their significant contribution to the model's prediction

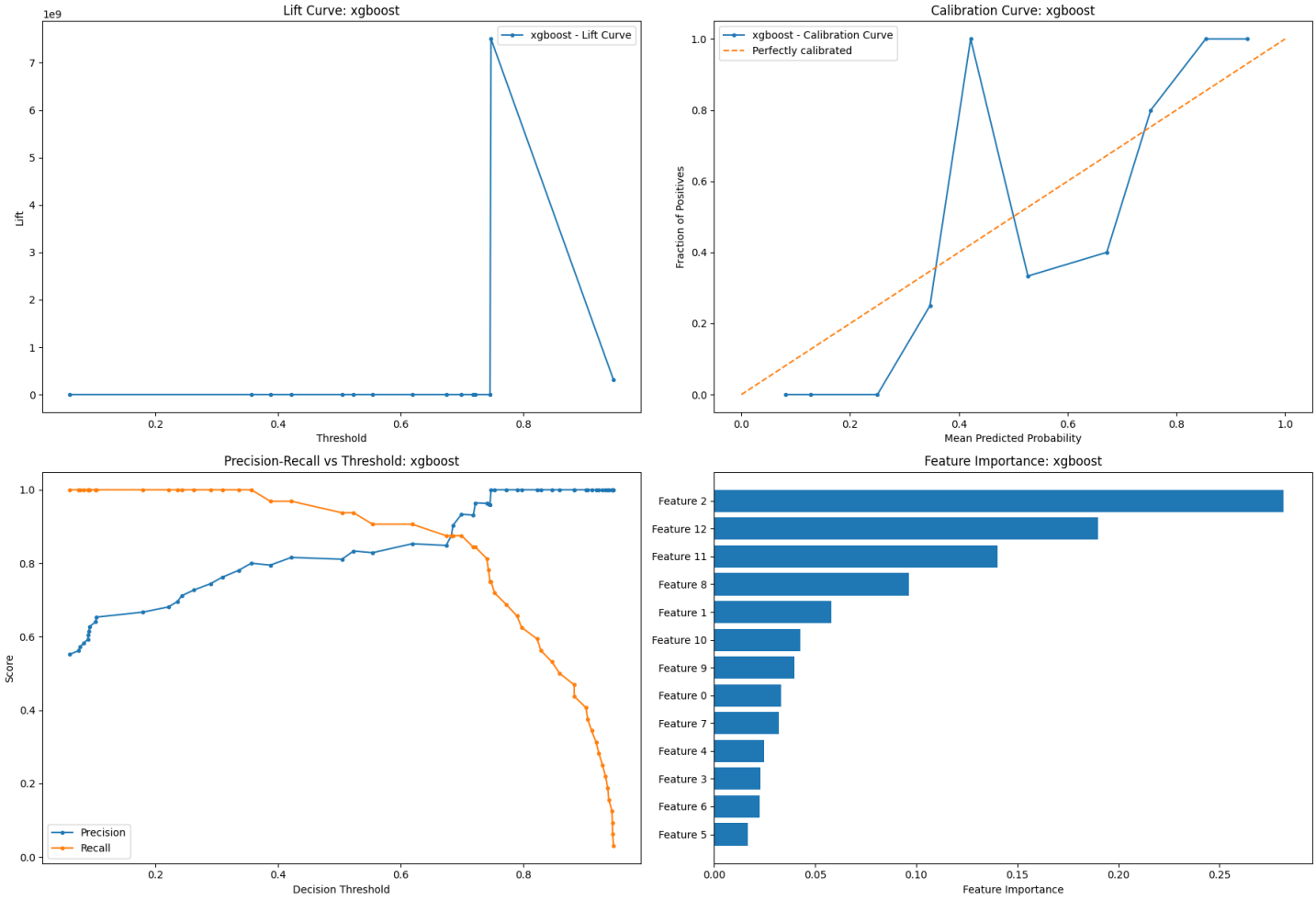


Figure 27: Analysis of XGBoost Model Performance

- Lift Curve: Significant improvement is observed at a threshold around 0.8.
- Calibration Curve: The model's predicted probabilities deviate from the perfectly calibrated line, indicating poor calibration.
- Precision-Recall vs Threshold: Precision increases with higher thresholds while recall decreases, showing their trade-off.
- Feature Importance: Feature 2 holds the highest importance, followed by Features 12 and 11.

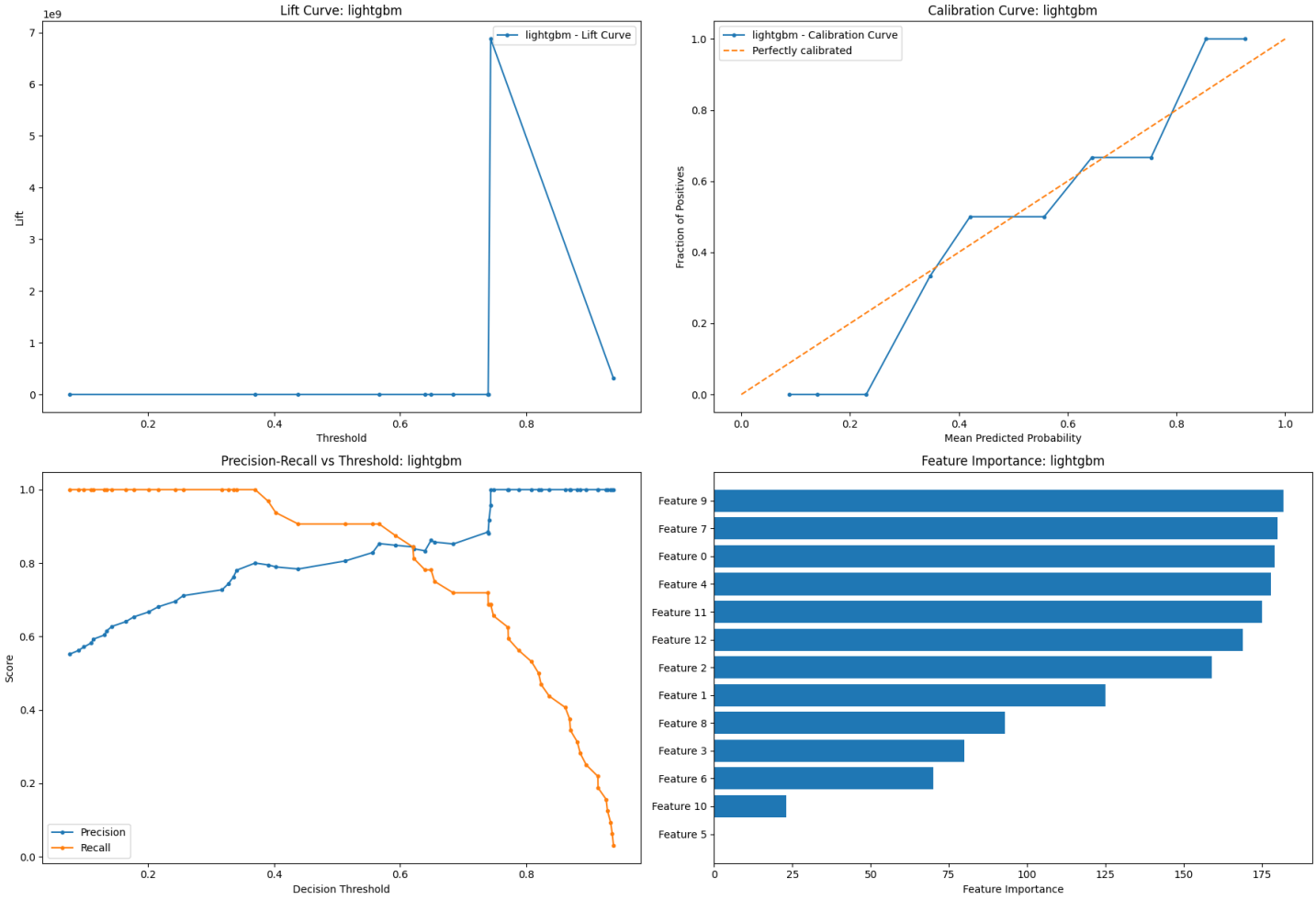


Figure 28: Analysis of LightGBM Model Performance

- Lift Curve: The lift curve indicates significant model improvement at a threshold near 0.8.
- Calibration Curve: The predicted probabilities deviate from the perfectly calibrated line, suggesting the model's predictions aren't well-calibrated.
- Precision-Recall vs Threshold: The precision increases as the threshold increases, while recall decreases, highlighting the trade-off.
- Feature Importance: Feature 9 is the most important, followed by Features 7 and 0.

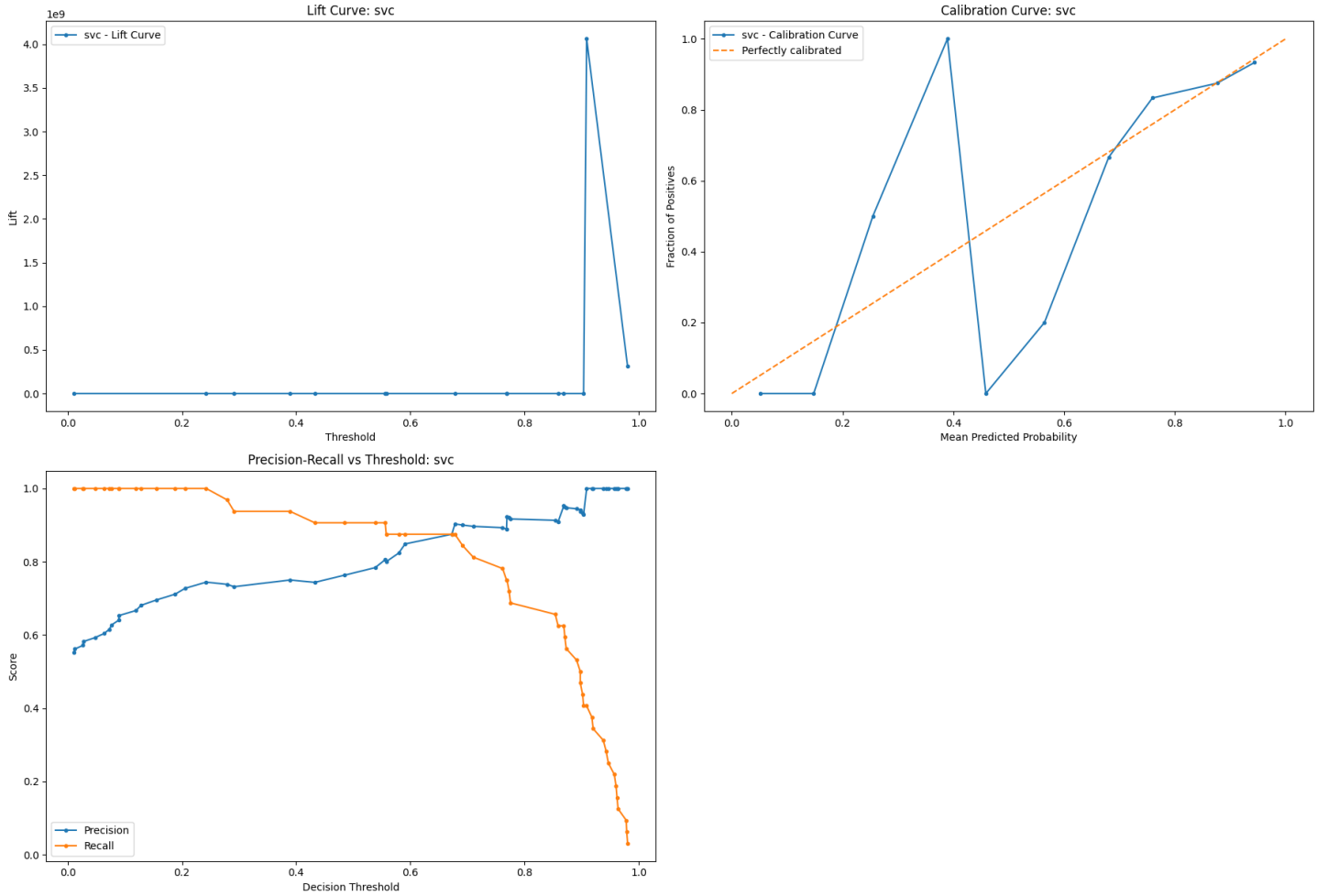


Figure 29: Analysis of Support Vector Classifier (SVC) Performance

- Lift Curve: The lift value shows a sharp increase at a threshold of around 1.0, indicating significant performance improvement at this threshold.
- Calibration Curve: The predicted probabilities deviate significantly from the perfectly calibrated line, suggesting poor calibration of the model.
- Precision-Recall vs Threshold: Precision increases with higher thresholds while recall decreases, indicating the trade-off between the two metrics.

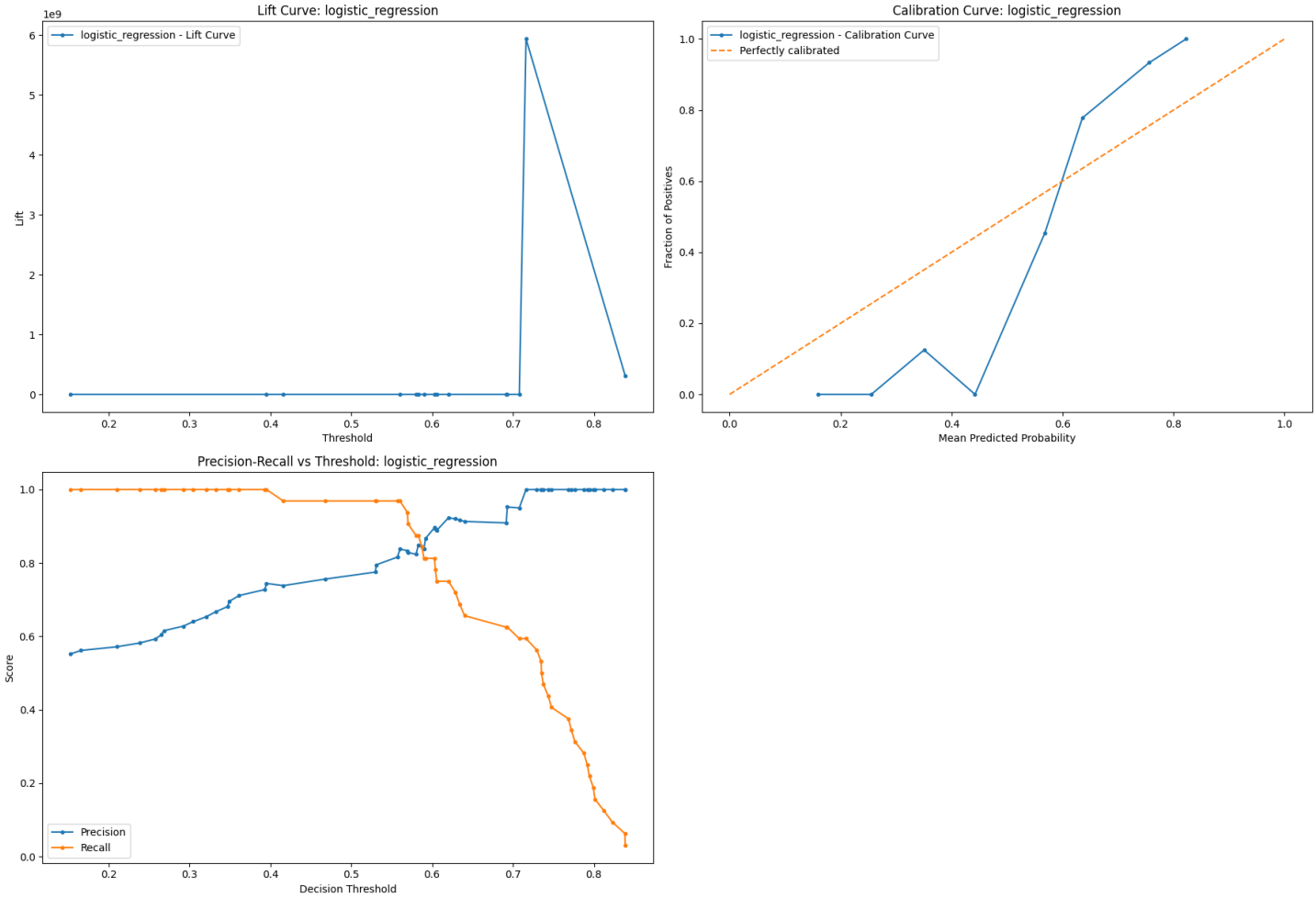


Figure 30: Performance Evaluation of Logistic Regression Model

- **Lift Curve:** The lift curve shows a significant spike at a threshold around 0.8, indicating enhanced model performance at this threshold.
- **Calibration Curve:** The predicted probabilities deviate from the perfect calibration line, suggesting the model's predictions aren't perfectly calibrated.
- **Precision-Recall vs Threshold:** Precision increases with higher thresholds while recall decreases, demonstrating the trade-off between precision and recall.

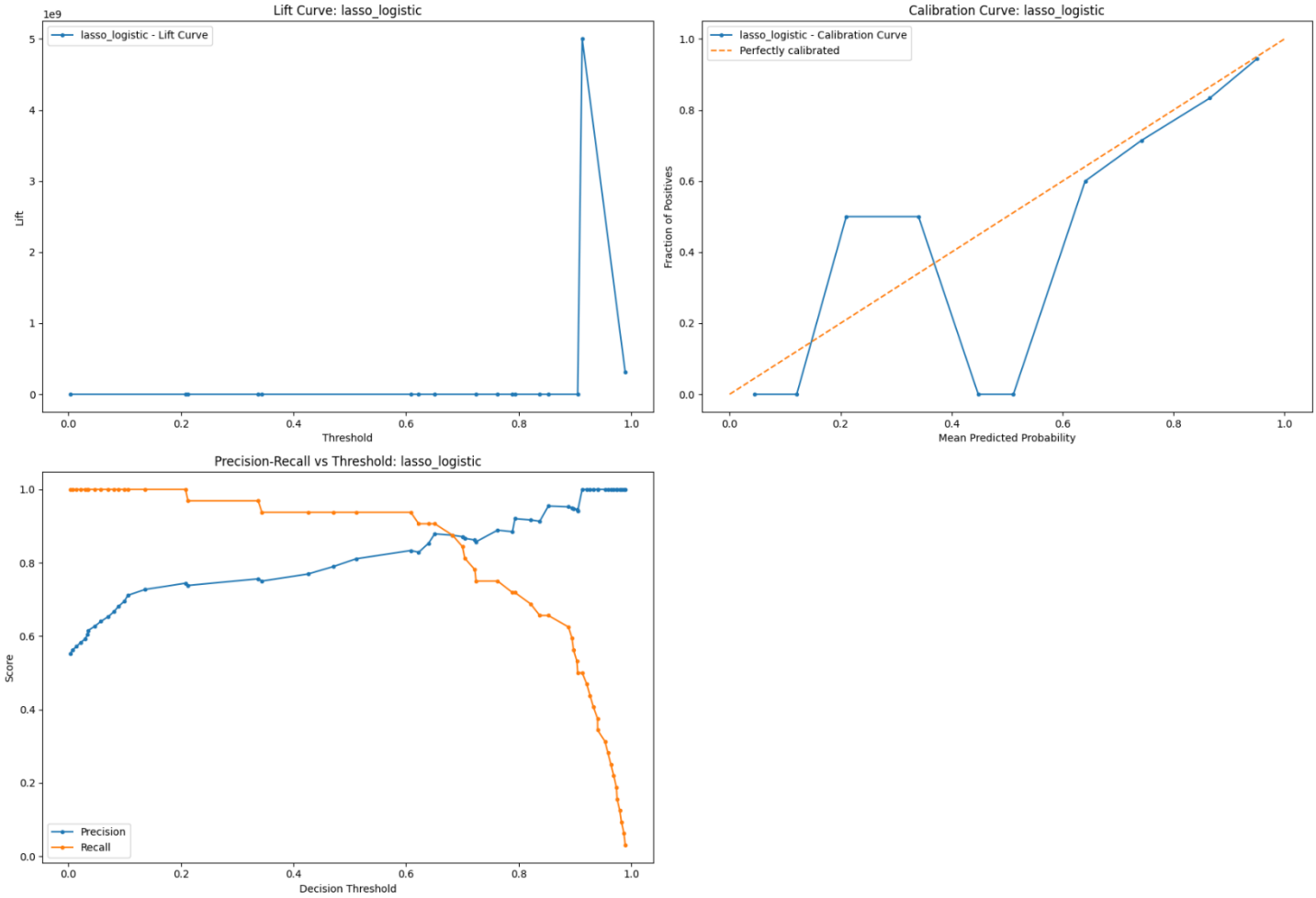


Figure 31: Performance Evaluation of Lasso Logistic Regression Model

- **Lift Curve:** Shows a significant spike at a threshold near 1.0, indicating improved model performance at this threshold.
- **Calibration Curve:** The model's predicted probabilities deviate from the perfect calibration line, suggesting the model's predictions aren't perfectly calibrated.
- **Precision-Recall vs. Threshold:** Precision increases as the threshold increases, while recall decreases, indicating the trade-off between precision and recall.

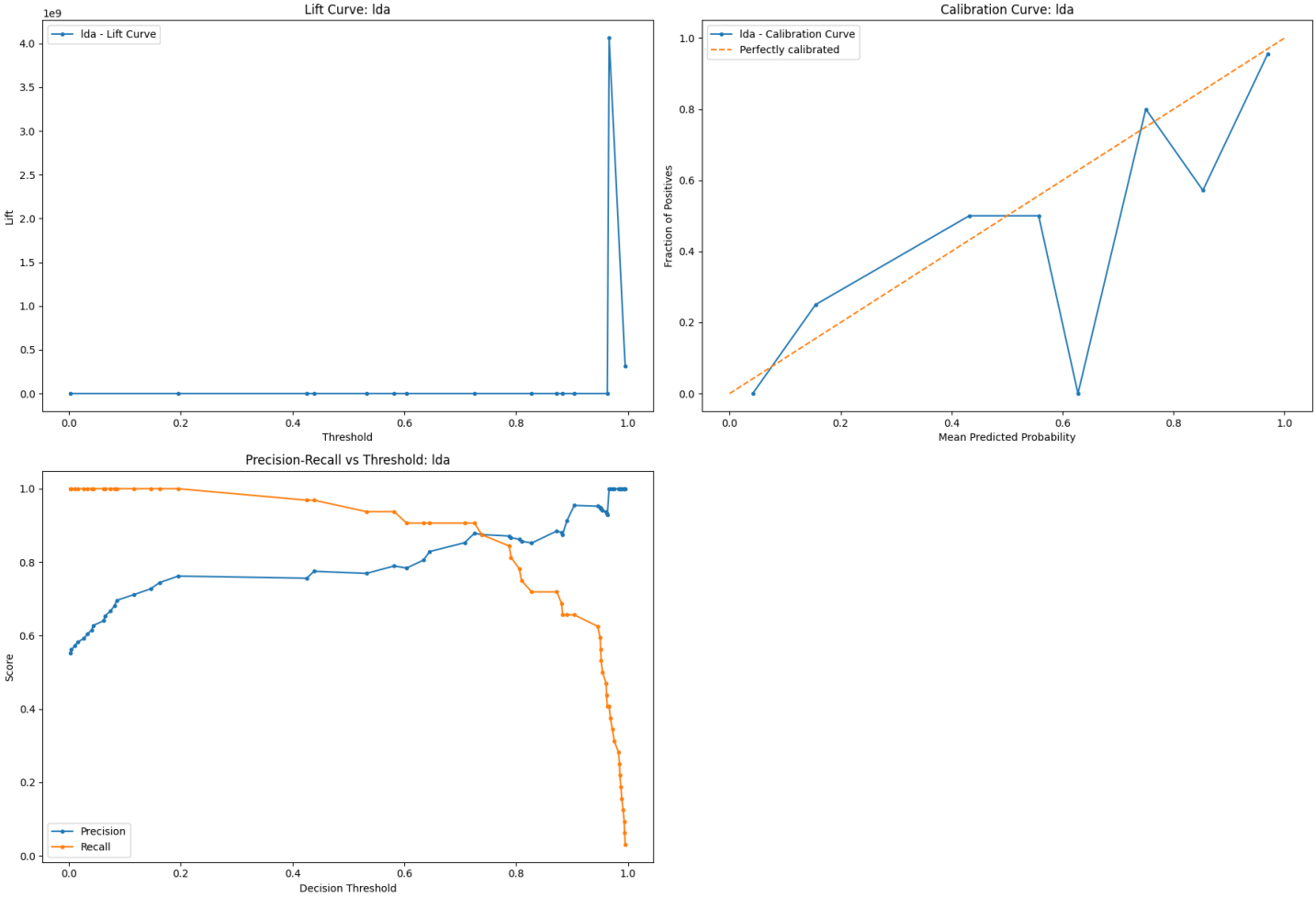


Figure 32: Performance Evaluation of LDA Model

- **Lift Curve:** The lift curve shows a significant spike at a threshold around 1.0, indicating improved model performance at this threshold.
- **Calibration Curve:** The model's predicted probabilities deviate from the perfect calibration line, suggesting the model's predictions aren't perfectly calibrated.
- **Precision-Recall vs. Threshold:** Precision decreases as the threshold increases, while recall initially increases and then sharply decreases after a certain threshold.

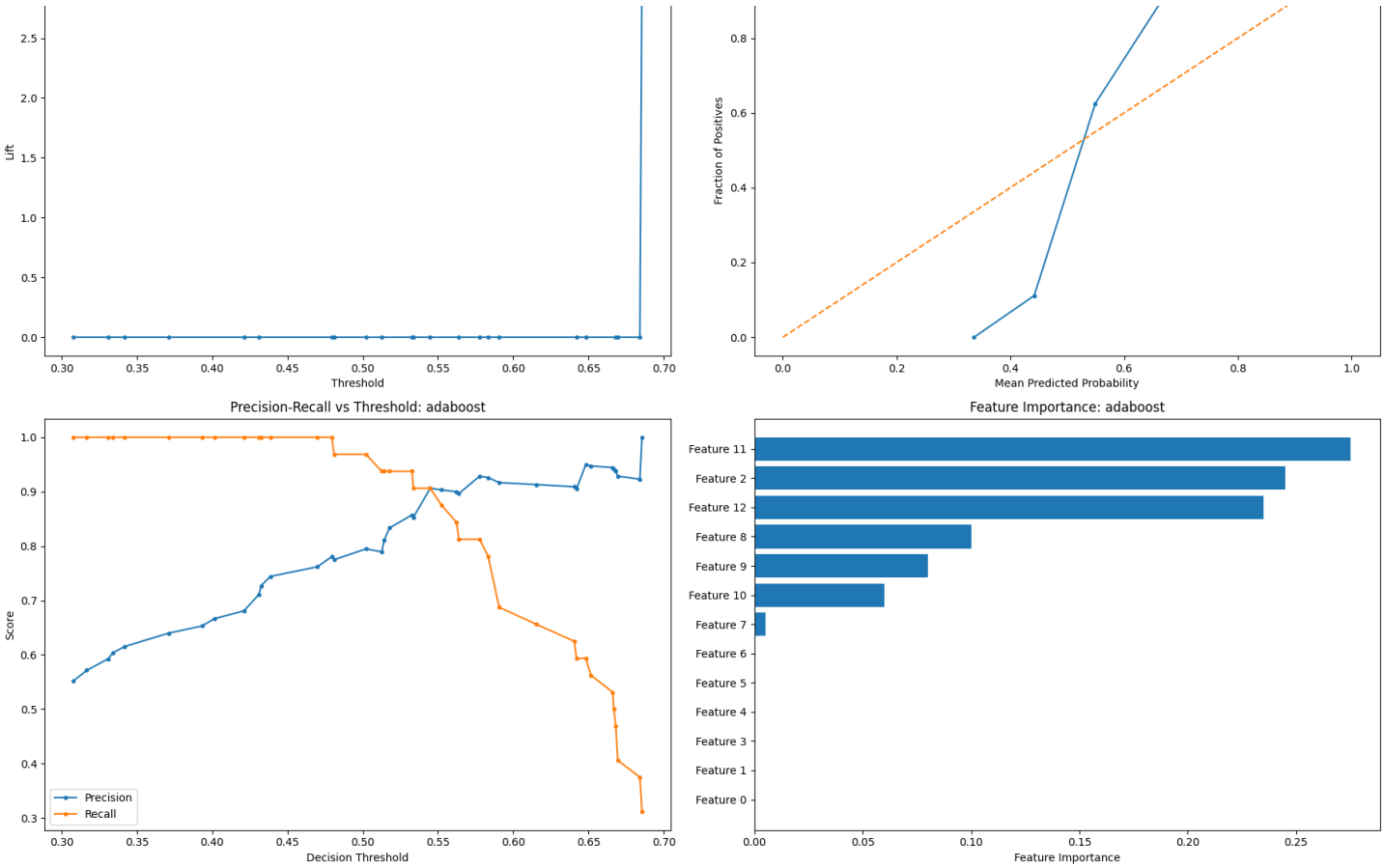


Figure 33: Performance Evaluation of AdaBoost Model

- **Lift Curve:** Indicates a significant performance improvement at higher threshold values.
- **Calibration Curve:** The model's predicted probabilities deviate from perfect calibration, suggesting poor calibration.
- **Precision-Recall vs. Threshold:** Precision increases with higher thresholds, while recall decreases, demonstrating the trade-off.
- **Feature Importance:** Feature 11 is the most important, followed by Features 12 and 2.

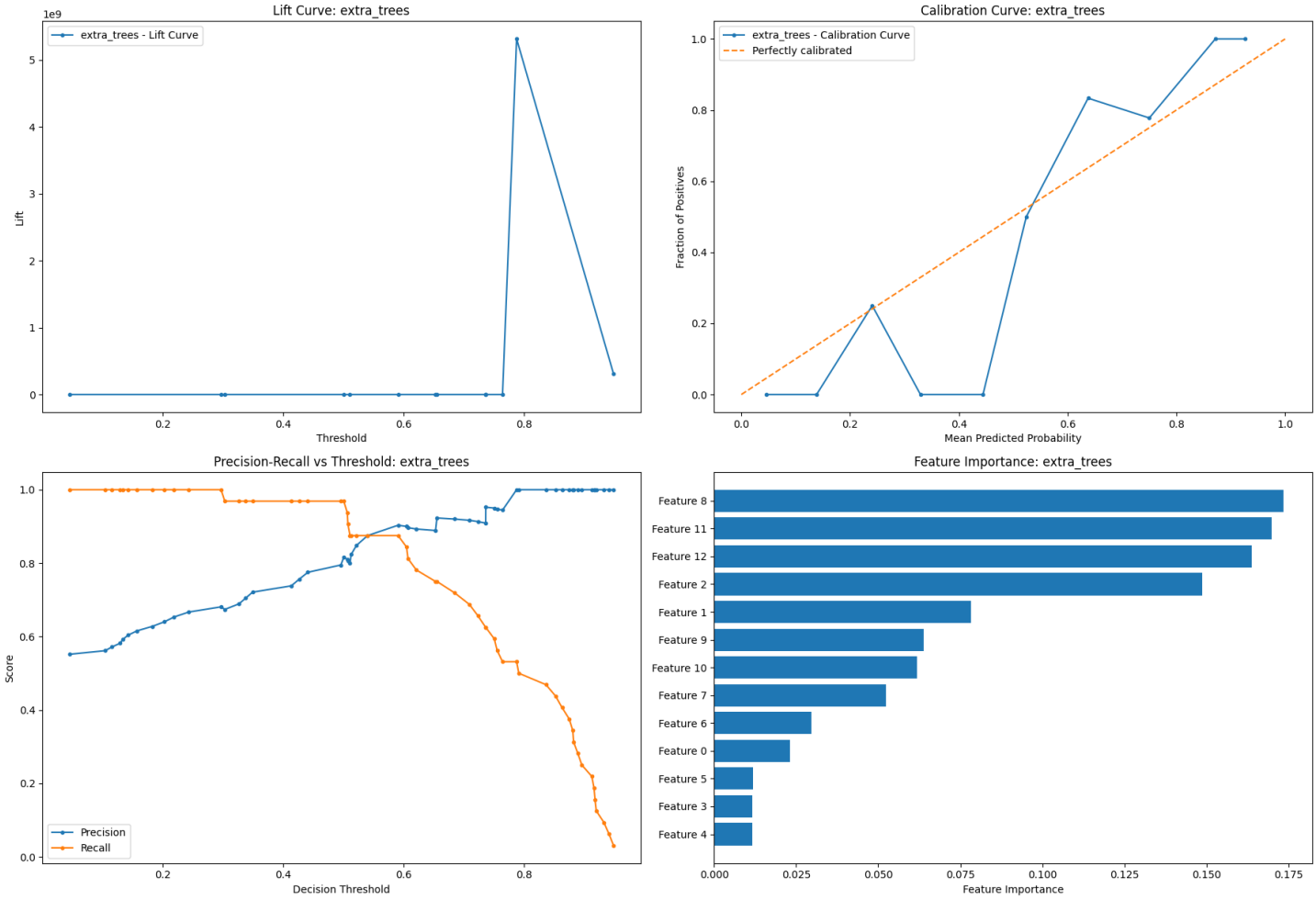


Figure 34: Performance and Analysis of Extra Trees Classifier

- **Lift Curve:** Indicates a significant performance improvement at higher threshold values, notably around 0.8.
- **Calibration Curve:** The predicted probabilities deviate from perfect calibration, suggesting the model's predictions aren't perfectly calibrated.
- **Precision-Recall vs. Threshold:** Precision increases with higher thresholds while recall decreases, demonstrating the trade-off.
- **Feature Importance:** Feature 8 is the most important, followed by Features 11 and 12.

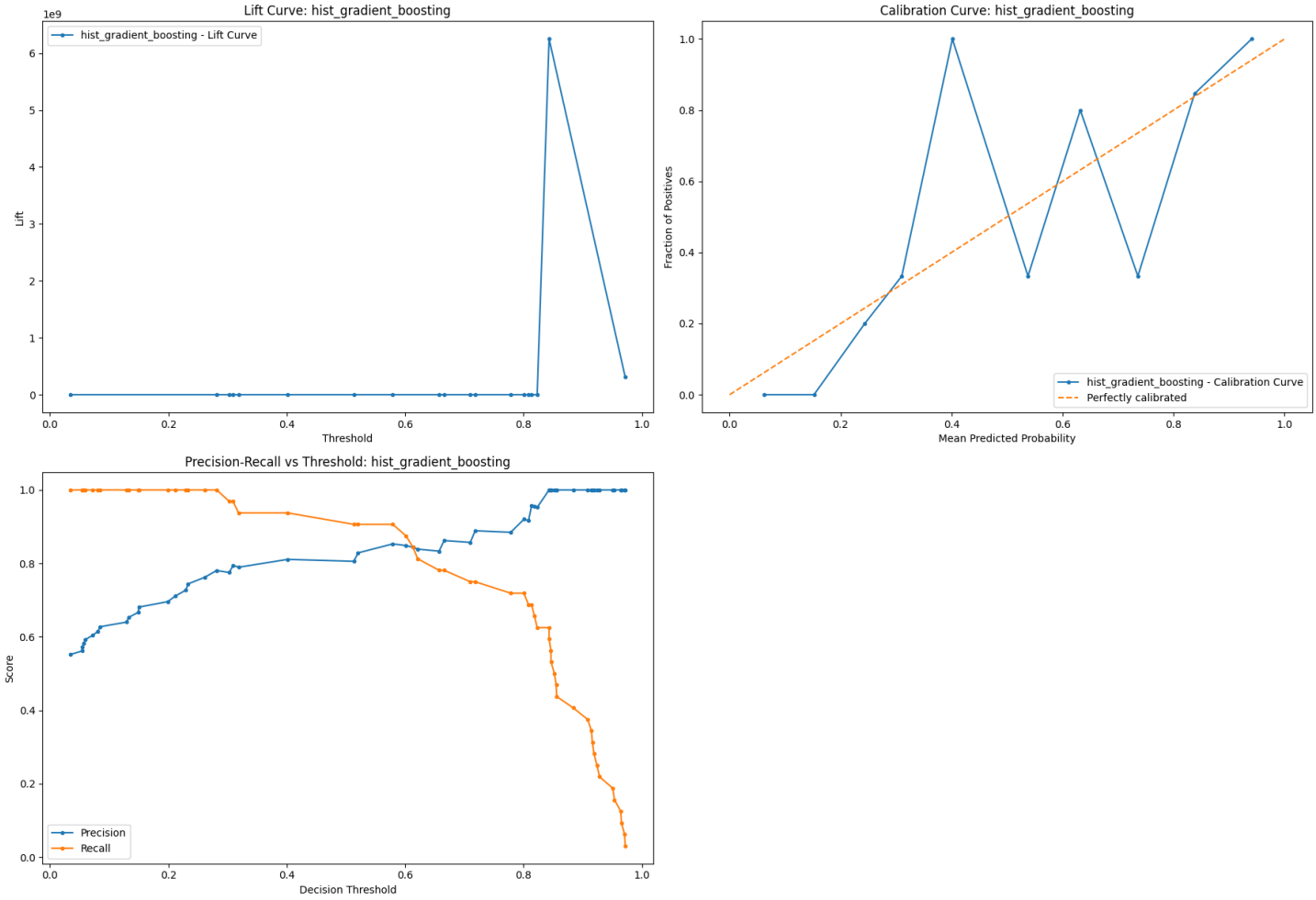


Figure 35: Performance Evaluation of HistGradientBoosting Model

- Lift Curve: Displays a significant spike around a threshold of 0.9, suggesting a performance boost at this threshold.
- Calibration Curve: The predicted probabilities deviate from the perfectly calibrated line, indicating room for improvement in calibration.
- Precision-Recall vs Threshold: Precision increases while recall decreases as the threshold increases, indicating the trade-off between these metrics. The curves intersect around a threshold of 0.6.

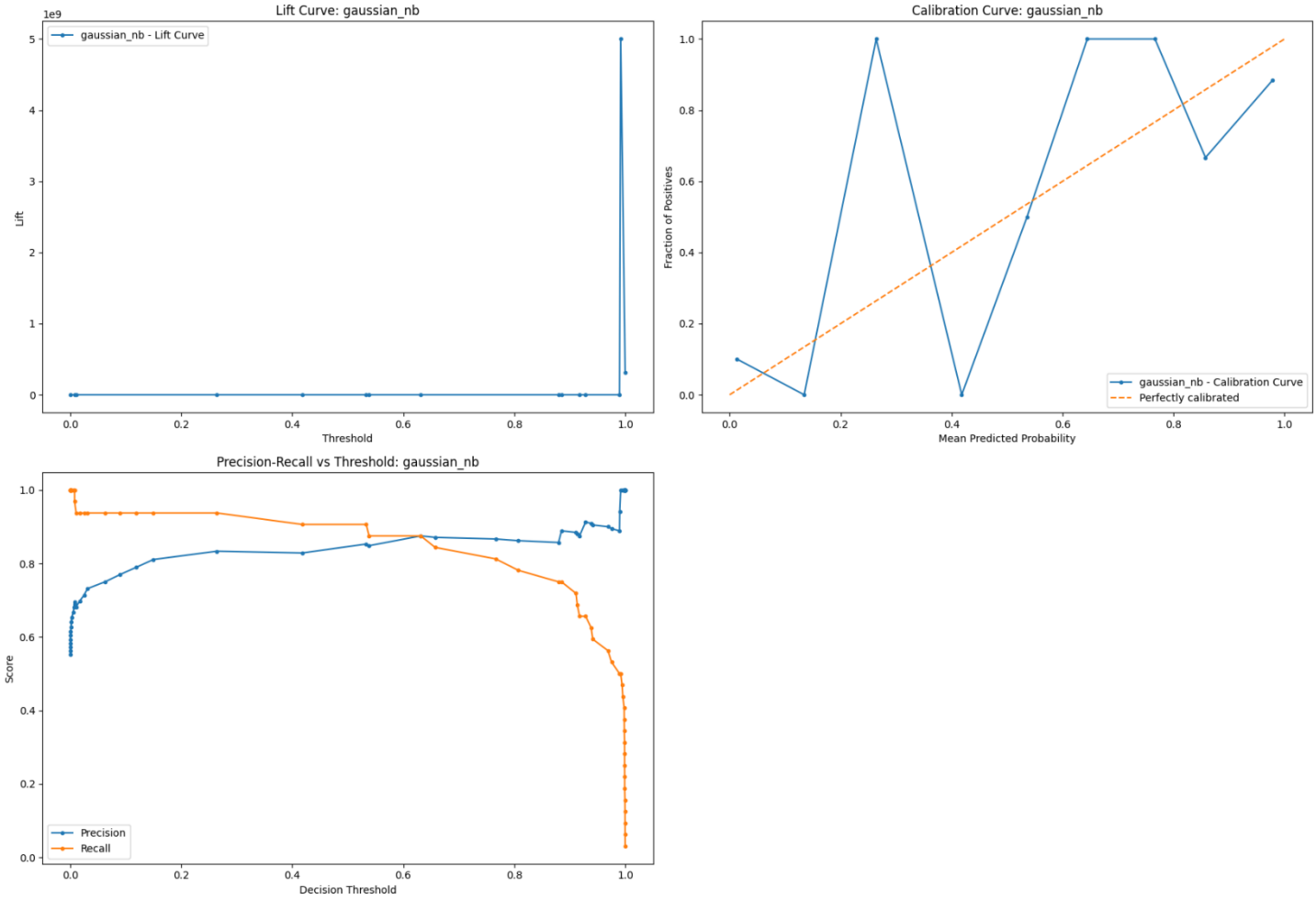


Figure 36: Performance Evaluation of Gaussian Naive Bayes Model

- **Lift Curve:** Indicates significant performance improvement at a threshold around 1.0.
- **Calibration Curve:** Predicted probabilities deviate from the perfect calibration line, indicating the need for better calibration.
- **Precision-Recall vs Threshold:** Shows the trade-off between precision and recall, with precision increasing and recall decreasing as the threshold increases.

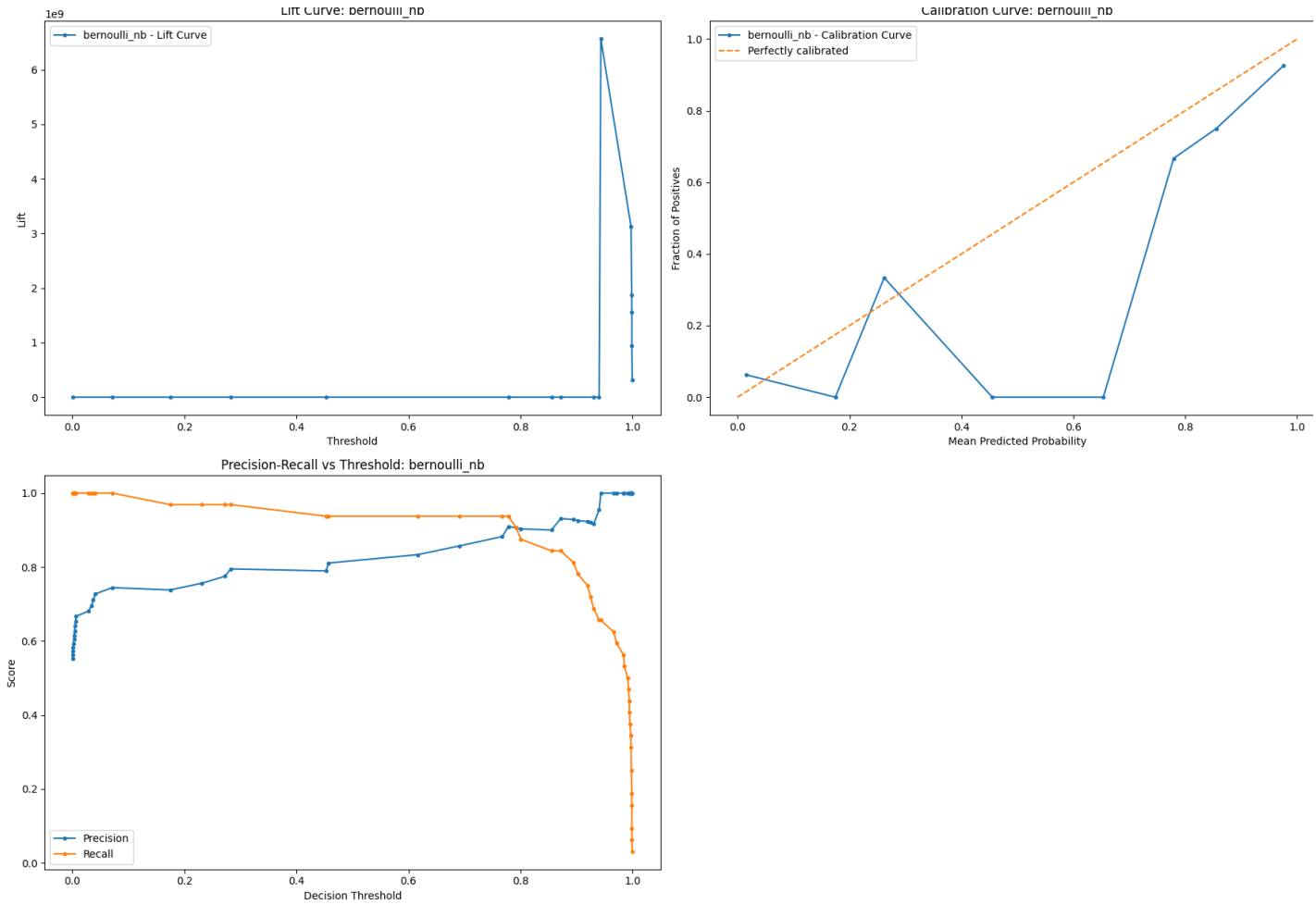


Figure 37: Performance Evaluation of Bernoulli Naive Bayes Classifier

- **Lift Curve:** Indicates a significant performance improvement at a threshold near 1.0.
- **Calibration Curve:** The model's predicted probabilities deviate significantly from perfect calibration, indicating poor calibration.
- **Precision-Recall vs Threshold:** Precision remains high across thresholds, while recall decreases sharply after a threshold of around 0.8.

Figure 38: SHAP Summary Plot for Random Forest Model

- **SHAP Interaction Values:**
  - The plot highlights the interaction values for Age and Sex.
  - The x-axis ranges from -0.1 to 0.1, representing SHAP interaction values.
  - Individual data points are colored dots, with blue and pink indicating different categories within the features.
- **Insights:**
  - Provides insights into how Age and Sex interact and their impact on model predictions.
  - Helps interpret the model's behavior and improve its performance.

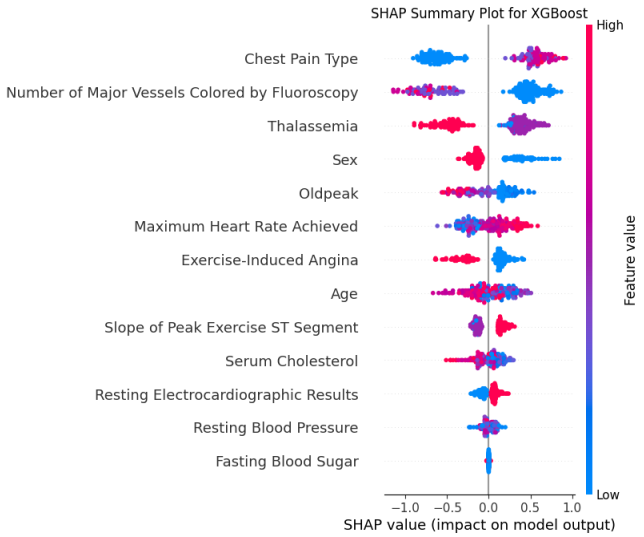


Figure 39: SHAP Summary Plot for XGBoost Model

- **Key Insights:**
  - **Chest Pain Type**, **Number of Major Vessels Colored by Fluoroscopy**, and **Thalassemia** are the top contributing features.
  - **SHAP Values**: The x-axis shows the impact on the model's output, ranging from -1.0 to 1.0.

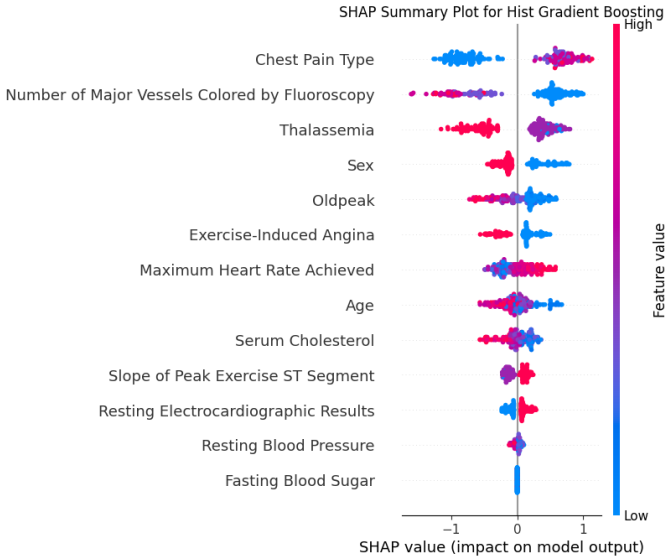
- **Colour Coding**: Indicates feature value, with a range from low to high as per the colour bar.

```
Top features ranked by SHAP value for XGBoost:
                                        Feature  Mean SHAP Value
2                                 Chest Pain Type         0.594900
11  Number of Major Vessels Colored by Fluoroscopy         0.578202
12                                    Thalassemia         0.438781
1                                             Sex         0.233331
9                                         Oldpeak         0.220483
7                       Maximum Heart Rate Achieved         0.213026
8                           Exercise-Induced Angina         0.211703
0                                             Age         0.192084
10                Slope of Peak Exercise ST Segment         0.151208
4                                 Serum Cholesterol         0.114848
6              Resting Electrocardiographic Results         0.083792
3                               Resting Blood Pressure         0.055302
5                                 Fasting Blood Sugar         0.002804
```

Figure 40: SHAP Summary Plot for Extra Trees Model

SHAP Interaction Values:

- Highlights interaction effects between "Sex" and "Age."
- x-axis ranges from -0.2 to 0.2 for SHAP interaction values.
- Each dot represents a data point, colored to indicate different categories within the features.



Top features ranked by SHAP value for Hist Gradient Boosting:

	Feature	Mean SHAP Value
2	Chest Pain Type	0.752591
11	Number of Major Vessels Colored by Fluoroscopy	0.679193
12	Thalassemia	0.495099
1	Sex	0.264412
9	Oldpeak	0.260724
8	Exercise-Induced Angina	0.220477
7	Maximum Heart Rate Achieved	0.212225
0	Age	0.179435
4	Serum Cholesterol	0.157231
10	Slope of Peak Exercise ST Segment	0.133809
6	Resting Electrocardiographic Results	0.116387
3	Resting Blood Pressure	0.036633
5	Fasting Blood Sugar	0.000000

Figure 41: SHAP Summary Plot for Hist Gradient Boosting Model

- **Key Insights:**
  - **Chest Pain Type**, **Number of Major Vessels Colored by Fluoroscopy**, and **Thalassemia** are the top contributing features.

Figure 42: SHAP Summary Plot for AdaBoost Model

- **Key Insights:**
  - **SHAP Interaction Values:** The plot highlights the interaction effects between "Sex" and "Age."

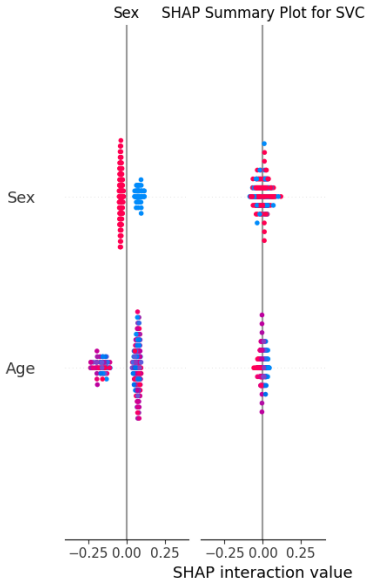


Figure 43: SHAP Summary Plot for SVC Model

- **Key Insights:**
  - **SHAP Interaction Values:** The plot highlights the interaction effects between "Sex" and "Age."

Figure 44: SHAP Summary Plot for Logistic Regression Model

- **SHAP Interaction Values:** Highlights the interaction effects between "Sex" and "Age."

Figure 45: SHAP Summary Plot for Lasso Logistic Model

- The plot shows the SHAP interaction values for Age and Sex, helping to understand their impact on the model's predictions.

Figure 46: SHAP Summary Plot for Linear Discriminant Analysis (LDA) Model

- The plot illustrates the SHAP interaction values for the features **Age** and **Sex** in the LDA model.

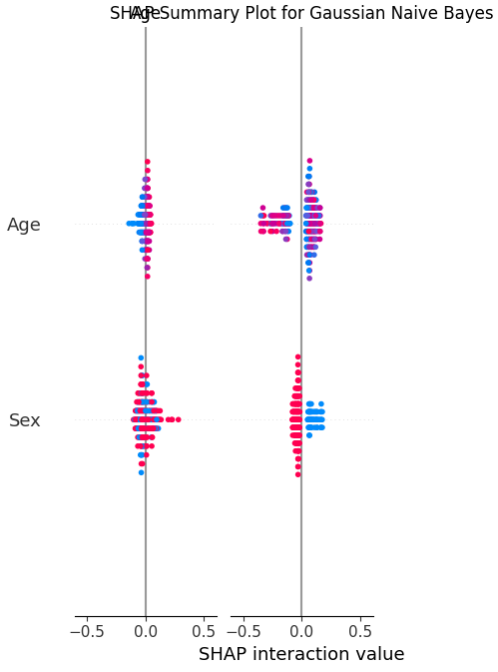


Figure 47: SHAP Summary Plot for Gaussian Naive Bayes Model

- The plot displays SHAP interaction values for two features: Age and Sex.

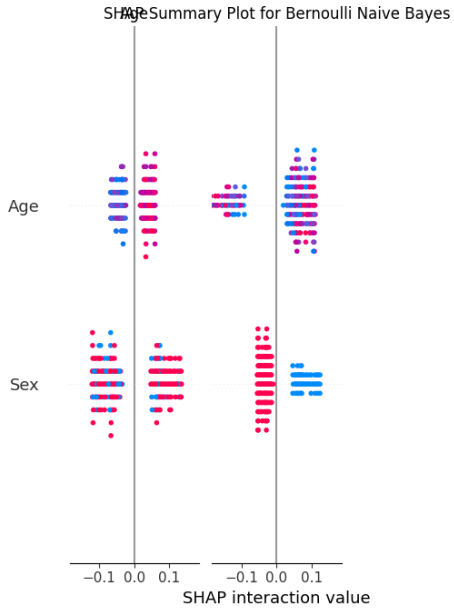


Figure 48: SHAP Summary Plot for Gaussian Naive Bayes Model

- The plot displays SHAP interaction values for two features: Age and Sex.

CHAPTER 5

CONCLUSION AND FUTURE WORK

Conclusion:

The project effectively designed and tested a range of machine learning models intended to predict heart disease using clinical and diagnostic features. Among the different models tested, tree-based algorithms like Extra Trees, Gradient Boosting, and the Stacking Classifier showed excellent performance, achieving high performance metrics such as accuracies over 99% besides excellent results in Precision, Recall, F1-Score, and ROC-AUC. These results highlight the robust predictive power of the models underpinning them which is essential in early diagnosis.

The SHAP technique was used to interpret the model, which provided high-level insight into the importance of features and how certain attributes were influencing the predictions. This kind of transparency is crucial in fostering confidence in the use of machine learning methods within health care settings.

The models demonstrate substantial potential in supporting health care professionals: improving early detection with a resultant contribution to better decision-making and to improved patient outcome. Their robustness implies that these models are suitable for real-world deployments, with the capability of aiding in the improvement of clinical flows.

Future Work:

1. **Enhancing Data Quality:**
    - Incorporate larger and more diverse datasets, especially from real-world clinical environments, to improve the model's generalizability and performance across varied patient demographics and conditions.
    - Address data imbalances through advanced techniques such as synthetic data generation or oversampling to ensure the model performs optimally across all patient groups.
2. **Advanced Feature Engineering:**
    - Explore new and potentially impactful features derived from clinical records, including genetic factors, patient lifestyle data, and environmental factors that may contribute to heart disease risk.
    - Leverage automated feature selection methods, such as Recursive Feature Elimination (RFE) or L1 regularization, to optimize the set of features used in the model and improve **its efficiency and accuracy.**
3. **Exploring Deep Learning Models:**

- Investigate the use of deep learning techniques, such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), to capture complex patterns within the data that traditional models might miss.
- Consider transfer learning by adapting pre-trained models for medical image or sequence data to further enhance the performance of heart disease prediction.

4. **Real-World Validation and Testing:**
    - Deploy the model in clinical settings to assess its real-time performance and gather valuable feedback from medical practitioners to fine-tune the model for real-world application.
    - Validate the model's accuracy and robustness on unseen patient data and in diverse healthcare environments to ensure its effectiveness in practice.
5. **Improving Explainability:**
    - Expand explainability efforts by integrating other techniques such as LIME (Local Interpretable Model-agnostic Explanations) or Integrated Gradients to provide alternative perspectives on model decision-making, further enhancing trust and transparency.
    - Develop intuitive, user-friendly visualizations and dashboards tailored for healthcare providers to facilitate easy interpretation of model predictions in clinical settings.
6. **Integration with Healthcare Systems:**
    - Work on developing a comprehensive software tool or mobile application that can be seamlessly integrated with Electronic Health Record (EHR) systems, allowing healthcare providers to input patient data and receive predictive insights in real time.
    - Consider building a decision support system (DSS) to help physicians make informed decisions based on real-time predictions, providing actionable recommendations for patient care.

By pursuing these future avenues, the model can be further refined and expanded to provide more accurate, interpretable, and actionable insights for heart disease diagnosis, helping to save lives and optimize healthcare outcomes.

CHAPTER 6

APPENDIX

1.Data Preprocessing

```
import os
import logging
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from scipy import stats
from joblib import dump

# Set up logging (optional, can be removed if not needed)
logging.basicConfig(level=logging.INFO)
MODEL_SAVE_PATH = "model/"
PREPROCESSOR_PATH = os.path.join(MODEL_SAVE_PATH, "preprocessor.joblib")

# Visualize outliers in one figure
def visualize_outliers(X):
    numerical_cols = X.select_dtypes(include=[np.number]).columns
    n = len(numerical_cols)
    rows = (n // 4) + (n % 4 > 0)
    cols = 4

    fig, axes = plt.subplots(rows, cols, figsize=(20, 5 * rows))
    axes = axes.flatten()
```

```
    for i, col in enumerate(numerical_cols):
        sns.boxplot(x=X[col], ax=axes[i], color='skyblue')
        axes[i].set_title(f'Box Plot for {col}', fontsize=10)

    for j in range(i + 1, len(axes)):
        fig.delaxes(axes[j])

    plt.tight_layout()
    plt.show()

    sns.pairplot(X, plot_kws={'alpha': 0.5})
    plt.suptitle('Pair Plot for Feature Relationships', y=1.02)
    plt.show()

    for col in numerical_cols:
        plt.figure(figsize=(8, 4))
        sns.histplot(X[col], kde=True, color='teal')
        plt.title(f'Distribution Plot for {col}')
        plt.show()

# Preprocess data
def preprocess_data(df):
    X = df.drop('output', axis=1)
    y = df['output']
    visualize_outliers(X)

    # Identify outliers using Z-scores
    z_scores = np.abs(stats.zscore(X))
    threshold = 3
    outliers = (z_scores > threshold).any(axis=1)

    outlier_count = outliers.sum()
    total_instances = len(X)
```

```
logging.info(f"Detected {outlier_count} outliers out of {total_instances} instances. Removing outliers...")
print(f"Detected {outlier_count} outliers out of {total_instances} instances. Removing outliers...")

X_clean = X[~outliers]
y_clean = y[~outliers]

# Separate numerical and categorical columns
numerical_cols = X_clean.select_dtypes(include=[np.number]).columns.tolist()
categorical_cols = X_clean.select_dtypes(exclude=[np.number]).columns.tolist()

# Define transformation pipelines
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

# Combine transformations
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols)
    ]
)

preprocessor.fit(X_clean)

# Save the preprocessor
if not os.path.exists(MODEL_SAVE_PATH):
```

```
        os.makedirs(MODEL_SAVE_PATH)
    dump(preprocessor, PREPROCESSOR_PATH)
    logging.info(f"Preprocessor saved to {PREPROCESSOR_PATH}")
    print(f"Preprocessor saved to {PREPROCESSOR_PATH}")

    # Apply the preprocessor
    X_transformed = preprocessor.transform(X_clean)
    logging.info(f"Transformed feature shape: {X_transformed.shape}")
    print(f"Transformed feature shape: {X_transformed.shape}")

    return X_transformed, y_clean

# Visualize transformed outliers with 4 box plots per row
def visualize_transformed_outliers(X_transformed):
    n = X_transformed.shape[1]
    rows = (n // 4) + (n % 4 > 0)
    cols = 4

    fig, axes = plt.subplots(rows, cols, figsize=(20, 5 * rows))
    axes = axes.flatten()

    for i in range(n):
        sns.boxplot(x=X_transformed[:, i], ax=axes[i], color='lightcoral')
        axes[i].set_title(f'Feature {i + 1}', fontsize=10)

    for j in range(i + 1, len(axes)):
        fig.delaxes(axes[j])

    plt.tight_layout()
    plt.show()

X, y = preprocess_data(df)
```

```
# After preprocessing, call this function to visualize transformed data
visualize_transformed_outliers(X)
```

2.Model Training & Hyperparameter tuning and Evaluation

```
logging.basicConfig(level=logging.INFO)
VISUALIZATION_PATH = 'path/to/your/visualizations'
os.makedirs(VISUALIZATION_PATH, exist_ok=True)

cv = StratifiedKFold(n_splits=10)

def create_and_tune_models(X_train, y_train, X_val=None, y_val=None):
    from sklearn.naive_bayes import GaussianNB, BernoulliNB

    models = {
        'random_forest': RandomForestClassifier(),
        'xgboost': XGBClassifier(use_label_encoder=False, eval_metric='logloss'),
        'catboost': CatBoostClassifier(verbose=0),
        'lightgbm': LGBMClassifier(),
        'svc': SVC(probability=True),
        'logistic_regression': LogisticRegression(),
        'lasso_logistic': LogisticRegression(penalty='l1', solver='liblinear'),
        'lda': LinearDiscriminantAnalysis(),
        'adaboost': AdaBoostClassifier(),
        'extra_trees': ExtraTreesClassifier(),
        'hist_gradient_boosting': HistGradientBoostingClassifier(),
        'gaussian_nb': GaussianNB(),
        'bernoulli_nb': BernoulliNB()
    }

    param_dist = {
```

```
'random_forest': {
    'n_estimators': [100, 200, 300],
    'max_depth': [80, 90],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2']
},
'xgboost': {
    'n_estimators': [100, 200, 300],
    'max_depth': [80, 90],
    'learning_rate': [0.01, 0.05, 0.1],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'lambda': [0.1, 1, 10],
    'alpha': [0.1, 1, 10]
},
'catboost': {
    'iterations': [200, 300],
    'depth': [80, 90],
    'learning_rate': [0.01, 0.05, 0.1],
    'l2_leaf_reg': [1, 3, 5],
    'border_count': [32, 64],
    'task_type': ['CPU', 'GPU']
},
'lightgbm': {
    'n_estimators': [100, 200, 300],
    'max_depth': [80, 90],
    'learning_rate': [0.01, 0.05, 0.1],
    'num_leaves': [20, 40, 60],
    'min_child_samples': [10, 20, 30],
```

```
    'min_data_in_leaf': [20, 30, 40]
},
'svc': {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf', 'poly', 'sigmoid'],
    'gamma': ['scale', 'auto']
},
'logistic_regression': {
    'C': [0.01, 0.1, 1, 10],
    'penalty': ['l2']
},
'lasso_logistic': {
    'C': [0.01, 0.1, 1, 10],
    'penalty': ['l1']
},
'lda': {
    'solver': ['svd', 'lsqr', 'eigen']
},
'adaboost': {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 1.0]
},
'extra_trees': {
    'n_estimators': [100, 200, 300],
    'max_depth': [80, 90],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2']
},
'hist_gradient_boosting': {
```

```
        'max_iter': [300, 400, 500],
        'max_depth': [80, 90],
        'learning_rate': [0.01, 0.05, 0.1],
        'max_leaf_nodes': [20, 40, 60]
    },
    'gaussian_nb': {
        'var_smoothing': [1e-9, 1e-8, 1e-7]
    },
    'bernoulli_nb': {
        'alpha': [0.1, 1.0, 10.0],
        'binarize': [0.0, 0.1, 0.5, 1.0]
    }
}

tuned_models = {}
for name, model in models.items():
    try:
        logging.info(f"Tuning {name}...")
        start_time = time.time()
        search = RandomizedSearchCV(model, param_distributions=param_dist[name],
                                    n_iter=10, scoring='roc_auc', cv=cv, verbose=0, n_jobs=-1)
        search.fit(X_train, y_train)
        tuned_models[name] = search.best_estimator_

        if name == 'xgboost' or name == 'lightgbm':
            if X_val is not None and y_val is not None:
                search.best_estimator_.fit(X_train, y_train,
                                           eval_set=[(X_val, y_val)],
                                           early_stopping_rounds=10, verbose=False)
```

```
            elapsed_time = time.time() - start_time
            logging.info(f"Best parameters for {name}: {search.best_params_} (Tuning time: {elapsed_time:.2f}s)")

        except Exception as e:
            logging.error(f"Error tuning model {name}: {e}")

    return tuned_models
```

REFERENCES

[1] Alexander, C.A. and Wang, L., 2017. Big data analytics in heart attack prediction. *J Nurs Care*, *6*(393), pp.2167-1168.

[2] Takci, H., 2018. Improvement of heart attack prediction by the feature selection methods. *Turkish Journal of Electrical Engineering and Computer Sciences*, *26*(1), pp.1-10.

[3] Manikandan, S., 2017, August. Heart attack prediction system. In *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)* (pp. 817-820). IEEE.

[4] Patil, S.B. and Kumaraswamy, Y.S., 2009. Extraction of significant patterns from heart disease warehouses for heart attack prediction. *IJCSNS*, *9*(2), pp.228-235.

[5] Jindal, H., Agrawal, S., Khera, R., Jain, R. and Nagrath, P., 2021. Heart disease prediction using machine learning algorithms. In *IOP conference series: materials science and engineering* (Vol. 1022, No. 1, p. 012072). IOP Publishing.

[6] Nikhar, S. and Karandikar, A.M., 2016. Prediction of heart disease using machine learning algorithms. *International Journal of Advanced Engineering, Management and Science*, *2*(6), p.239484.

[7] Sadar, U., Agarwal, P., Parveen, S., Jain, S. and Obaid, A.J., 2023, December. Heart disease prediction using machine learning techniques. In *International Conference on Data Science, Machine Learning and Applications* (pp. 551-560). Singapore: Springer Nature Singapore.

[8Mall, S., 2024, May. Heart Attack Prediction using Machine Learning Techniques. In *2024 4th International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)* (pp. 1778-1783). IEEE.

[9] Katarya, R. and Meena, S.K., 2021. Machine learning techniques for heart disease prediction: a comparative study and analysis. *Health and Technology*, *11*(1), pp.87-97.

[10] Takci, H., 2018. Improvement of heart attack prediction by the feature selection methods. *Turkish Journal of Electrical Engineering and Computer Sciences*, *26*(1), pp.1-10.

[11]Bharti, R., Khamparia, A., Shabaz, M., Dhiman, G., Pande, S. and Singh, P., 2021. Prediction of heart disease using a combination of machine learning and deep learning. *Computational intelligence and neuroscience*, *2021*(1), p.8387680.

[12] Bah, I. (2022). Knn algorithm used for heart attack detection. *FES Journal of Engineering Sciences*, *11*(1), 7-19.

[13] Khateeb, N. and Usman, M., 2017, December. Efficient heart disease prediction system using K-nearest neighbor classification technique. In Proceedings of the international conference on big data and internet of thing (pp. 21-26).

[14] Enriko, I.K.A., Suryanegara, M. and Gunawan, D., 2016. Heart disease prediction system using k-Nearest neighbor algorithm with simplified patient's health parameters. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 8(12), pp.59-65.

[15] Chethana, C., 2021, May. Prediction of heart disease using different KNN classifier. In *2021 5th international conference on intelligent computing and control systems (ICICCS)* (pp. 1186-1194). IEEE.

[16] Deekshatulu, B.L. and Chandra, P., 2013. Classification of heart disease using k-nearest neighbor and genetic algorithm. *Procedia technology*, *10*, pp.85-94.

[17] Thomas, J. and Princy, R.T., 2016, March. Human heart disease prediction system using data mining techniques. In *2016 international conference on circuit, power and computing technologies (ICCPCT)* (pp. 1-5). IEEE.

[18] Soni, J., Ansari, U., Sharma, D. and Soni, S., 2011. Predictive data mining for medical diagnosis: An overview of heart disease prediction. *International Journal of Computer Applications*, *17*(8), pp.43-48.

BIODATA

Name : K S H V Sai Hari Krishna
Mobile Number : 7075811975
E-mail : hari.21bce8069@vitapstudent.ac.in
Permanaent Address : Sattenapalli-522403, Palnadu District , Andhra

Name : C.L.V.Saradhi Reddy
Mobile Number : 8897495719
E-mail : saradhi.21bce7816@vitap.ac.in
Permanaent Address : Markapur , Prakasham District, Andra Pradesh

Name : Varshitha gundluru
Mobile Number : 7780127347
E-mail : varshitha.21bce7657@vitapstudent.ac.in
Permanaent Address : Tirupathi , Chittoor district, Andhra Pradesh

Name : G. Hyny Syamala
Mobile Number : 7671997628
E-mail : syamala.21bce7338@vitapstudent.ac.in
Permanaent Address : Bhimavaram , West Godavari district, Andhra