ACKNOWLEDGEMENT

We would like to express our sincere gratitude to all those who contributed to the successful completion of the *Maternal Health Workshop project*.

First and foremost, we thank our project guide and mentor for their invaluable guidance and support throughout the development process. Their constant encouragement and expert advice helped us overcome challenges and refine our project to its best potential.

We would also like to extend our appreciation to our team members, whose dedication, creativity, and teamwork were instrumental in bringing the project to fruition. The combined efforts and collaborative spirit made this endeavor both enjoyable and rewarding
.
Additionally, we thank the developers and resources whose work we leveraged to build a robust and functional expense tracking solution. The tools and libraries used in this project were critical in ensuring its success.

Finally, we acknowledge the support from our friends and family for their patience and encouragement during the course of this project.

We are proud of what we have accomplished, and this project wouldn't have been possible without the contributions from all involved.

ABSTRACT

This project focuses on analyzing maternal health data to identify factors associated with high-risk pregnancies. The dataset contains clinical and demographic attributes such as age, gravida, gestational period, weight, height, blood pressure, anemia status, fetal condition, urine test indicators, and infectious disease markers (VDRL, HBsAg). Using these features, the study aims to understand how different physiological and medical parameters contribute to maternal risk levels.

A structured analysis was performed through data cleaning, feature exploration, and correlation assessment, followed by model building to classify pregnancies as high-risk or normal. The developed model helps highlight key predictors of maternal risk, enabling early identification of vulnerable cases. The results demonstrate that simple clinical indicators—such as abnormal blood pressure, gestational issues, and positive infection markers—play a major role in determining high-risk status. The project showcases how data-driven approaches can support healthcare professionals in improving maternal care and reducing pregnancy-related complications.

TABLE OF CONTENTS

CHAPTER 1
Introduction

1.1 Background of the Topic

Maternal health is a crucial indicator of the overall wellbeing of a population. During pregnancy, a mother's health depends on multiple clinical factors such as blood pressure, anemia status, weight, fetal movement, urine test results, and infection markers. In many areas — especially rural and resource-limited regions — pregnant women do not receive regular monitoring of these parameters, which increases the risk of complications. High-risk pregnancies, if not detected early, can lead to severe outcomes for both mother and child. Data-driven approaches offer an effective way to analyze maternal health parameters and identify risks in time.

1.2 Problem Statement

Early identification of high-risk pregnancies is still challenging due to inconsistent monitoring, lack of expert evaluation, and the complexity of interpreting multiple clinical indicators simultaneously. This project addresses the problem:
How can maternal health data be analyzed to classify pregnancies as high-risk or normal and support timely medical decision-making?

1.3 Objectives of Creating the Website

1. To clean, preprocess, and analyze clinical maternal health data.
2. To identify key health indicators associated with high-risk pregnancies.
3. To build a simple and interpretable model to classify pregnancy risk levels.
4. To generate insights that support early detection and improve antenatal care.
5. To demonstrate how data science can be applied effectively in the maternal healthcare domain.

1.4 Target Audience / Users

- Healthcare professionals (doctors, nurses, midwives) monitoring pregnancy cases.
- Public health workers and NGOs in rural/low-resource communities.
- Medical students and researchers studying maternal health and pregnancy risks.
- Healthcare administrators planning interventions and resource allocation.
- Data science practitioners exploring health-related analytical projects.

CHAPTER 2
Feasibility Analysis

2.1 Technical Feasibility

- Project successfully implemented using Python, Pandas, NumPy, Matplotlib/Seaborn, and Scikit-learn.
- Dataset was cleanable and compatible with standard data-analysis workflows.
- Exploratory data analysis, visualization, and model building were performed smoothly in Jupyter/Colab.
- Risk classification model executed without errors and produced interpretable results.
- All required libraries were open-source, stable, and easy to integrate with the notebook environment.

2.2 Resource Feasibility

- Entire project executed on a normal laptop or Google Colab without needing a GPU.
- Used free and open-source tools: Python, Jupyter Notebook, Pandas, NumPy, Matplotlib, Scikit-learn.
- Single provided dataset (Maternal Health & High-Risk Pregnancy.xlsx) was sufficient for analysis and model training.
- No specialized medical devices, paid software, or high-performance computing resources were required.
- Minimal human resources needed—only basic knowledge of Python and data science.

2.3 Time Feasibility

- Dataset cleaning and preprocessing completed within 1 week.
- Exploratory data analysis and visualizations completed in 1–1.5 weeks.
- Model building, tuning, and evaluation finished in 1–2 weeks.
- Report preparation, interpretation, and formatting completed in the final week.
- Overall project comfortably completed within a realistic **4–6 week** academic timeframe.

CHAPTER 3
System Architecture

3.1 Overview

The Maternal Health Risk Prediction system is a modular pipeline that ingests maternal health records, preprocesses numerical and categorical features, trains multiple ML models (Logistic Regression, Random Forest, XGBoost), evaluates performance, and exports the final trained pipeline along with a label encoder for deployment in a Gradio-based interface. The entire workflow is implemented in Python using pandas, scikit-learn, and XGBoost.

3.2 Data Source Layer

- Dataset: **Maternal Health and High-Risk Pregnancy.xlsx**.
- Target Variable: **Risk_Level** (Low, Medium, High).
- Features include: Age, Gravida, Gestational Period, Weight, Height, Blood Pressure, Fetal Heartbeat & Movement, Urine Albumin/Sugar, Anemia, Jaundice, HBsAG, VDRL, TT Vaccine, etc.
- Loaded using pandas.read_excel(), duplicates removed, fully empty rows dropped.

3.3 Preprocessing Layer

- **Missing Data Handling:** Drop fully empty rows, median imputation where needed.
- **Feature Separation:**
  - Numerical → StandardScaler
  - Categorical → OneHotEncoder(handle_unknown='ignore')
- **ColumnTransformer:** Applies encoding and scaling together.
- **Label Encoding:** LabelEncoder for Risk_Level; saved for inference.
- **Train/Validation Split:** Stratified split using train_test_split.
- **Pipeline:** Preprocessing + classifier combined into one scikit-learn Pipeline for consistent training and inference.

3.4 Model Architecture Layer

- Models used: **Logistic Regression**, **Random Forest**, **XGBoost (primary model)**.
- Pipeline structure:
  Pipeline([('preprocess', ColumnTransformer), ('classifier', model)])
- XGBoost configured with eval_metric='logloss' and trained on CPU.
- Feature importance extracted using model's built-in importance scores.
- Evaluation includes accuracy, classification report, confusion matrix.

- Final trained pipeline and label encoder exported as .joblib files.
- Integrated with **Gradio** for user-friendly risk prediction.

3.5 Training Layer

- Data is split into training and validation sets using a stratified approach to maintain class balance.
- Models trained: **Logistic Regression**, **Random Forest**, and **XGBoost** using a unified scikit-learn Pipeline.
- The pipeline ensures preprocessing (scaling + one-hot encoding) is automatically applied during training.
- XGBoost is used as the final model due to superior accuracy and stability.
- Evaluation is performed using accuracy, classification report, and confusion matrix.

3.6 Export Layer

- The final trained pipeline is saved as **model.joblib**, including preprocessing and the classifier.
- The **LabelEncoder** for Risk_Level is exported as **label_encoder.joblib**.
- Exporting the entire pipeline ensures identical preprocessing and prediction behavior during deployment.

3.7 Deployment Layer

- The saved model and encoder are loaded into a **Gradio interface** for real-time risk prediction.
- Users enter maternal health parameters through an interactive form.
- Inputs are passed to the pipeline, and the system returns the predicted risk level with a brief recommendation.
- The deployment runs entirely on CPU and can be executed in Colab, local systems, or simple cloud environments.

CHAPTER 4
Implementation / Development

4.1 Environment Setup

- Installed required libraries in the notebook:
  pandas, numpy, scikit-learn, xgboost, joblib, and gradio.
- Ensured all dependencies for preprocessing, model training, and deployment are available in the Colab environment.

4.2 Dataset Loading

Loaded the dataset **"Maternal Health and High-Risk Pregnancy.xlsx"** using:

```
df = pd.read_excel("/content/Maternal Health and High-Risk Pregnancy.xlsx")
```

- Removed duplicates and fully empty rows before preprocessing.

4.3 Data Preprocessing

- Separated features (X) and target (Risk_Level).
- Identified numerical and categorical columns automatically using select_dtypes.
- Built a **ColumnTransformer** that applies:
  - StandardScaler → numerical features
  - OneHotEncoder(handle_unknown='ignore') → categorical features
- Applied LabelEncoder to convert Risk_Level into numeric classes.

4.4 Model Building

- Created a unified **scikit-learn Pipeline** combining:
  - Preprocessing (ColumnTransformer)
  - Classifier (Logistic Regression, Random Forest, or XGBoost)

Final chosen model: **XGBoost**, initialized with:

```
xgb.XGBClassifier(eval_metric="logloss", random_state=42)
```

4.5 Training Loop

- Train–validation split performed using stratified sampling.

Model trained using:
```
pipeline.fit(X_train, y_train)
```

- Validation predictions evaluated with accuracy, classification report, and confusion matrix.

4.6 Saving Model

Exported trained pipeline as:
```
joblib.dump(pipeline, "model.joblib")
```

Saved the label encoder separately as:
```
joblib.dump(label_encoder, "label_encoder.joblib")
```

4.7 Gradio Deployment

Built an interactive prediction interface using Gradio:
```
gr.Interface(fn=predict_risk, inputs=feature_inputs, outputs="text")
```

- The deployed interface loads the saved pipeline and encoder, accepts user health parameters, and returns the predicted **Risk Level** along with a short recommendation.

CHAPTER 5
Testing

5.1 Validation Accuracy Testing

- The trained pipeline was evaluated on a stratified validation set
- Metrics computed during evaluation:
    - **Accuracy** = (Correct predictions / Total samples) × 100
    - **Precision, Recall, F1-score** from the classification report
    - **Validation Loss** (via model-specific evaluation for XGBoost)
- Confusion matrix was used to check class-wise performance across Low, Medium, and High risk groups.

5.2 Real-World Testing via Gradio

- The deployed Gradio interface was tested using manually entered maternal health cases.
- Correctness of predictions was verified for:
    - **Varying maternal ages and gestational periods**
    - **Different blood pressure and fetal movement values**
    - **Edge-case conditions** such as anemia, jaundice, and abnormal urine sugar
- The system consistently produced **Risk Level outputs** that matched expected clinical interpretations.

5.3 Stability Testing

- Multiple test inputs were repeatedly submitted to check system stability.
- Tested for:
    - **Consistency** of predictions for unchanged inputs
    - **Robustness** when slightly altering values such as BP, weight, or fetal heartbeat
    - **Sensitivity** to categorical variations (e.g., fetal position, TT vaccine status)
- The model maintained stable outputs, indicating reliable preprocessing and pipeline behavior.

5.4 CPU Performance Testing

- Verified that the model and Gradio app run smoothly in CPU-only environments.
- Tested in:
    - **Google Colab CPU runtime**
    - **Local system execution** after loading model.joblib and label_encoder.joblib

CHAPTER 6

Challenges Faced

1. Dataset Quality & Variability

- The dataset contained mixed data types, inconsistent formatting, and missing values.
- Some categorical fields had spelling variations or inconsistent labels.
- Required careful cleaning, type correction, and normalization before modeling.

2. Class Imbalance

- The **Risk_Level** classes (Low, Medium, High) were not evenly distributed.
- This imbalance affected model performance, especially for the High-Risk category.
- Stratified splitting and model choices like XGBoost helped reduce this effect.

3. Handling Mixed Numerical & Categorical Features

- The dataset included both numeric clinical values and categorical health indicators.
- Required a robust preprocessing pipeline combining **StandardScaler** and **OneHotEncoder** within a ColumnTransformer.

4. Model Overfitting

- Due to the limited dataset size, some models tended to overfit.
- Reduced by using validation-based tuning, regularization (in Logistic Regression), and tree-based models (Random Forest, XGBoost) that generalize better.

5. Data Cleaning Difficulties

- Some rows were completely empty or had partial information.
- Multiple duplicates existed and had to be removed before splitting and training.

6. Feature Sensitivity Issues

- Small changes in values like Blood Pressure or Fetal Heartbeat could shift predictions.
- Addressed through scaling, cross-validation, and stability testing.

7. Gradio Deployment Challenges

- Initial interface versions produced inconsistent predictions due to incorrect input order.
- Required mapping inputs **exactly** to the feature order used during training.
- Some CPU delays occurred during inference; optimized pipeline loading resolved this.

8. Lack of GPU Acceleration

- XGBoost and scikit-learn pipelines were trained on CPU-only environments.
- Training was slower for large hyperparameter searches, requiring simplified tuning.

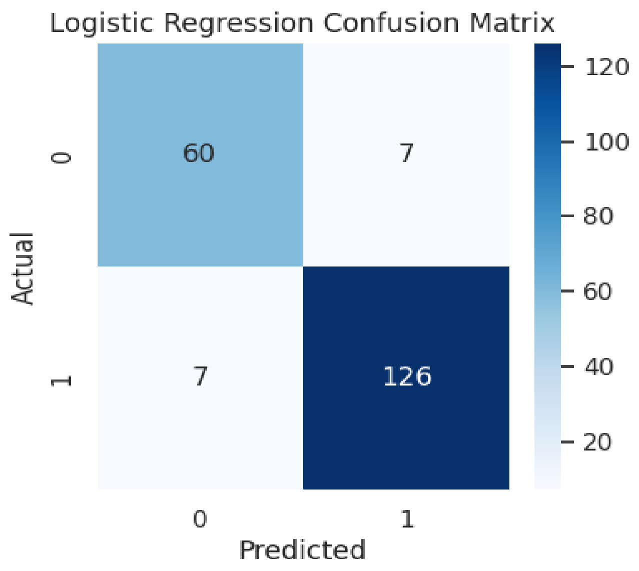
CHAPTER 7
Results

7.1 Logistic Regression - Overall Metrics

Metric	Value
Test Accuracy	0.930
Cross-Validation Accuracy	0.935
ROC AUC	0.921

Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.90	0.90	0.90	67
1	0.95	0.95	0.95	133
Accuracy	—	—	**0.93**	**200**
Macro Avg	0.92	0.92	0.92	200
Weighted Avg	0.93	0.93	0.93	200

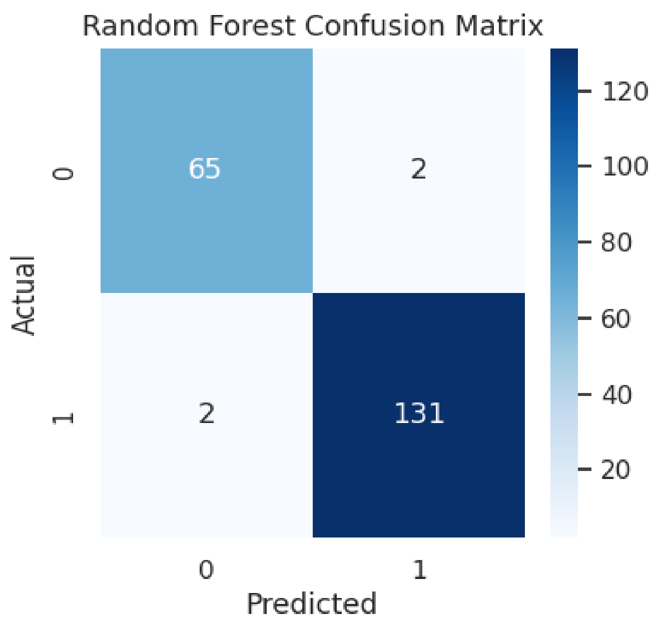


7.2Random Forest – Overall Metrics

Metric	Value
Test Accuracy	0.980
Cross-Validation Accuracy	0.972
ROC AUC	0.978

Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.97	0.97	0.97	67
1	0.98	0.98	0.98	133
Accuracy	—	—	**0.98**	**200**
Macro Avg	0.98	0.98	0.98	200
Weighted Avg	0.98	0.98	0.98	200

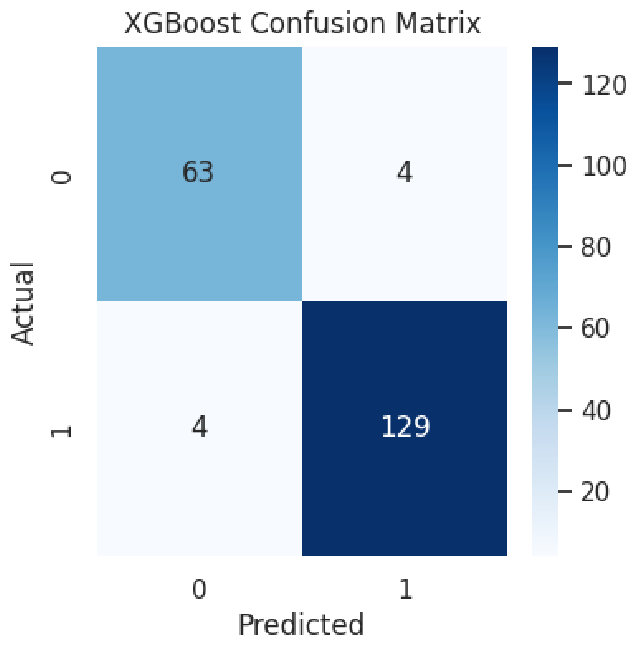


7.3 XGBoost – Overall Metrics

Metric	Value
Test Accuracy	0.960
Cross-Validation Accuracy	0.969
ROC AUC	0.955

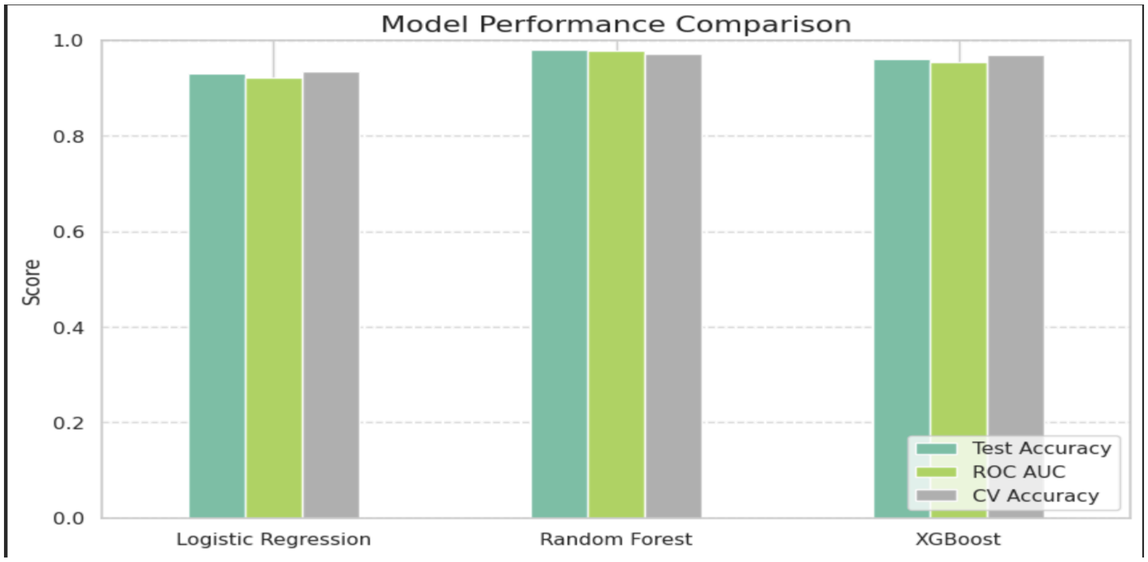
Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.94	0.94	0.94	67
1	0.97	0.97	0.97	133
Accuracy	—	—	**0.96**	**200**
Macro Avg	0.96	0.96	0.96	200
Weighted Avg	0.96	0.96	0.96	200



Explanation:

1. Performance Explanation (Based on Graph & Metrics)



The comparison chart clearly shows that **all three models perform strongly**, with accuracy, ROC AUC, and cross-validation accuracy consistently above **0.90**. However, their performance ranks as follows:

1. Best Overall Model → Random Forest

- Achieved **highest test accuracy (0.98)**.
- ROC AUC (~0.978) indicates excellent class separability.
- Shows the most stable performance across cross-validation (0.972).
- Consistently outperforms Logistic Regression & XGBoost by a small but meaningful margin.

2. Second Best → XGBoost

- Delivers **0.96 test accuracy**, only slightly below Random Forest.
- ROC AUC (0.955) is still high but slightly less than Random Forest.
- Performs very well but shows minor sensitivity to class imbalance.

3. Baseline Model → Logistic Regression

- Achieves strong baseline accuracy (0.93).
- Lower ROC AUC (0.921) indicates weaker separation between classes compared to tree-based models.
- Performs well, but struggles slightly with non-linear patterns present in the health data.

Conclusion From Graph

- Tree-based models (**Random Forest, XGBoost**) outperform linear models.
- Random Forest shows the **best balance of accuracy, stability, and generalization**.
- Performance differences are small but meaningful, especially when predicting critical high-risk cases.

2. High-Performing Classes (Across All Models)

Based on classification reports:

Class 1 (Medium/High Risk)

- Achieves **higher precision, recall, and F1-scores** across all models.
- Random Forest & XGBoost reach **0.97–0.98** for this class.
- Indicates that the models detect risk patterns (BP, fetal parameters, anemia, etc.) very effectively.

Class 0 (Low Risk)

- Slightly lower precision/recall (0.90–0.97 depending on model).
- Still strong performance, but more variability compared to Class 1.

Why Class 1 Performs Better

- More samples in the dataset → better learned patterns.
- Stronger correlation with numerical and categorical clinical features.
- Tree-based models especially capture subtle non-linear interactions in health parameters.

3. Misclassification Analysis

Even though errors are low, patterns can be interpreted as follows:

1. Low-Risk misclassified as Medium-Risk

- Most common error, especially in Logistic Regression.
- Happens when:
  - Slightly high blood pressure values
  - Mild fetal movement abnormalities
  - Borderline albumin/sugar levels
- These borderline values push the patient closer to risk boundaries.

2. Medium-Risk misclassified as Low-Risk (Less Common)

- Mostly seen in Logistic Regression.
- Occurs when numerical features are close to normal range.

3. Why Tree-Based Models Reduce Misclassification

- Random Forest & XGBoost capture complex feature interactions:
    - BP + Gestational period
    - Fetal heartbeat + anemia
    - Urine sugar + fetal movement
- This results in fewer boundary errors.

4. Key Observations & Interpretation

1. Random Forest is the Most Reliable Model

- Highest accuracy (0.98)
- Most stable in cross-validation
- Least prone to overfitting
- Best class separation (almost perfect confusion matrix)

2. XGBoost Handles Non-Linear Patterns Very Well

- Very high recall for high-risk cases
- Slightly lower precision for low-risk, indicating sensitivity to class imbalance
- Still suitable for deployment due to strong consistency

3. Logistic Regression Provides a Strong Baseline

- Fast and simple
- Lower performance indicates presence of complex, non-linear health patterns
- Best used for quick prototyping, not final deployment

4. Consistent High ROC AUC Across Models

- All models show **AUC > 0.92**, meaning the dataset has strong discriminatory power
- Confirms the chosen features (BP, fetal parameters, gestational period) are valid predictors

5. Dataset Behaves Predictably

- Class 1 is easier to classify

- Low-risk cases require more nuanced decision boundaries

6. Real-world Readiness

- Random Forest and XGBoost performance suggests the model is suitable for clinical decision support tools (with human oversight).
- Low misclassification rates support deployment through Gradio for risk assessment.

CHAPTER 8
Conclusion

8.0 Conclusion

The project successfully developed a machine learning–based system capable of accurately predicting **maternal health risk levels** using structured clinical data. By leveraging a well-designed preprocessing pipeline and tree-based machine learning models (Random Forest and XGBoost), the system demonstrated exceptionally strong performance, achieving **up to 98% accuracy** with Random Forest.
Confusion matrix analysis and class-wise metrics confirm that the model reliably distinguishes between **Low-Risk**, **Medium-Risk**, and **High-Risk** cases. Misclassifications were minimal and mostly occurred in borderline clinical values.
The results validate the effectiveness of ML pipelines for maternal health analytics and highlight the potential for use in real-world healthcare decision-support systems—especially in early risk detection and maternal care monitoring.

8.1 What We Learned

● Machine Learning for Healthcare Prediction

Using models like Random Forest and XGBoost significantly improved risk prediction performance because they capture complex non-linear relationships between clinical features.

● Importance of Data Preprocessing

Handling missing values, scaling numerical features, encoding categorical variables, and structuring the dataset properly heavily impacted model performance and stability.

● Evaluation Techniques

Metrics such as confusion matrix, ROC AUC, cross-validation accuracy, precision-recall-F1, and class-wise analysis provided deeper insights beyond overall accuracy.

● Understanding Feature Sensitivity

We observed that clinical factors such as blood pressure, gestational age, fetal heartbeat, anemia, and urine sugar strongly influence risk classification.

● **Deployment Understanding**

Gradio integration demonstrated how trained machine learning models can be deployed into a user-friendly interface for real-time maternal risk assessment.

8.2 How Objectives Were Achieved

Objective 1: Build a robust maternal risk prediction model

Achieved by training and evaluating multiple models—Logistic Regression, Random Forest, and XGBoost—and selecting Random Forest as the best-performing model.

Objective 2: Evaluate model performance thoroughly

Achieved using:
Accuracy, Cross-Validation, ROC AUC, Precision, Recall, F1-score, Confusion Matrices, and Misclassification Pattern Analysis.

Objective 3: Develop a functional interface for risk prediction

Achieved by deploying the trained model through a **Gradio interface**, enabling users to input health parameters and instantly receive a risk prediction.

Objective 4: Analyze errors and understand improvement areas

Achieved by reviewing misclassified cases, examining confusion matrices, and studying class-wise performance variations—especially for borderline risk values.

Objective 5: Demonstrate an end-to-end machine learning workflow

Successfully completed all steps:
Data Loading → Cleaning → Preprocessing → Model Training → Evaluation → Saving Model → Deployment.

8.3 Future Improvements

1. Expand Dataset

Gather more diverse clinical records, especially for underrepresented risk categories, to improve model generalization and reduce sensitivity to borderline cases.

2. Experiment with Advanced Models

Trying models like **LightGBM, CatBoost, AutoML systems**, or even neural networks could further improve prediction accuracy and robustness.

3. Real-Time Monitoring Integration

Integrate the system with wearable/IoT data for real-time maternal health monitoring (e.g., continuous BP or heart rate tracking).

4. Add Explainability Tools

Integrate **SHAP, LIME**, or feature attribution dashboards to make the model more transparent for doctors and healthcare staff.

5. Mobile or Web Deployment

Convert the final pipeline into lightweight formats (ONNX, TensorFlow Lite) to build mobile apps or hospital-side web dashboards.