[Thesis Title Here]

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of
Master of Science (MSc)
in
[Department Name]

[Your Name]
[Student ID]
[University Name]

Month, Year

Declaration

I hereby declare that the work presented in this thesis is my own and has not been submitted elsewhere for the award of any degree.

[Your Name]
[Date]

Abstract

Acknowledgement

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

1.2 Problem Statement

1.3 Objectives

1.4 Significance of the Study

1.5 Thesis Organization

Chapter 2

THEORETICAL BACKGROUND

2.1 Overview

Antimicrobial Resistance (AMR) prediction from genomic data represents a critical intersection of machine learning and computational biology. This chapter provides the theoretical foundation for machine learning techniques, feature engineering, ensemble learning, and interpretability frameworks employed in this research for predicting AMR from resistance gene profiles.

2.2 Machine Learning Fundamentals

2.2.1 Supervised Learning

Supervised learning learns a mapping function $f : X \to Y$ from labeled data $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where x_i is the feature vector and $y_i \in \{0, 1\}$ is the class label (1 = resistant, 0 = susceptible). The dataset is partitioned into training and testing sets using stratified splitting to preserve class proportions. Stratified k-fold cross-validation assesses performance by training k times on different partitions, averaging results across iterations.

2.3 Feature Engineering and Selection

2.3.1 Feature Engineering

Feature engineering transforms raw data to improve model performance. Min-Max normalization scales features to $[0, 1]$: $x' = \frac{x - \min(x)}{\max(x) - \min(x)}$. Derived features capture

domain knowledge through aggregation (sum, mean, count), ratios, interactions, or binning.

2.3.2 Feature Selection

Feature selection addresses dimensionality, computational efficiency, interpretability, and noise reduction. This research employs a hybrid approach combining filter and embedded methods.

ANOVA F-test: Evaluates features independently using the F-statistic: $F = \frac{\text{Between-group variability}}{\text{Within-group variability}}$. Higher F-statistics indicate greater class separation with statistical significance.

XGBoost Gain-Based Importance: Measures average loss function improvement: Importance(feature) = $\sum$(Gain from splits). Features are ranked and selected until cumulative importance reaches a threshold (e.g., 85%).

Hybrid Strategy: Combining methods (union approach) leverages statistical univariate relationships and multivariate interactions for comprehensive coverage.

2.4 Class Imbalance Handling

2.4.1 The Imbalance Problem

Class imbalance occurs when the majority class significantly outnumbers the minority class. Imbalance ratio = $\frac{\text{Majority samples}}{\text{Minority samples}}$. Ratios $> 1.5 : 1$ are imbalanced, causing models to bias toward the majority class and poorly predict the critical minority class.

2.4.2 SMOTE and SMOTE-Tomek

SMOTE (Synthetic Minority Over-sampling Technique): Generates synthetic minority samples by interpolating between existing instances. For sample x with nearest neighbor x_{nn}: $x_{\text{synthetic}} = x + \lambda \times (x_{nn} - x)$, where $\lambda \in [0, 1]$. SMOTE creates diverse samples but may generate unrealistic instances in sparse spaces.

SMOTE-Tomek: Combines SMOTE with Tomek links removal. A Tomek link is a pair (x_i, x_j) from different classes that are mutual nearest neighbors. The process: (1) Apply SMOTE, (2) Identify Tomek links, (3) Remove linked samples. This balances classes while improving boundary clarity, removing noisy samples, and reducing overfitting—particularly effective for high-dimensional, sparse data.

2.5 Classification Algorithms

2.5.1 Logistic Regression

Models binary outcome probability using the sigmoid function: $P(y = 1 \mid x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\cdots+\beta_n x_n)}}$. Creates linear decision boundaries, is computationally efficient, produces calibrated probabilities, and offers interpretable coefficients but cannot capture complex non-linear patterns.

2.5.2 Support Vector Machine (SVM)

Finds the optimal separating hyperplane maximizing margin: $f(x) = \text{sign}(w \cdot x + b)$. The kernel trick maps inputs to higher dimensions for non-linear separation. The RBF kernel $K(x, x') = \exp(-\gamma\|x - x'\|^2)$ handles non-linearity with tunable smoothness. Effective in high-dimensional spaces but computationally expensive for large datasets.

2.5.3 Decision Tree

Recursively partitions data using impurity measures. Gini impurity: $\text{Gini}(S) = 1 - \sum p_i^2$. Entropy: $\text{Entropy}(S) = -\sum p_i \times \log_2(p_i)$. Trees are interpretable, require minimal preprocessing, and capture non-linear relationships but are prone to overfitting and instability.

2.5.4 Random Forest

Ensemble of decision trees using bootstrap sampling and random feature selection. Algorithm: (1) Create B bootstrap samples, (2) For each sample, grow a tree selecting m random features at each split, (3) Aggregate predictions via majority voting. Feature importance: Importance $= \frac{1}{B}\sum$ Decrease in impurity. Reduces overfitting, provides robust importance estimates, handles high-dimensional data well but is less interpretable than single trees.

2.5.5 XGBoost (eXtreme Gradient Boosting)

Sequential tree ensemble where each tree corrects previous errors. Optimizes regularized objective: $\text{Obj} = \sum L(y_i, \hat{y}_i) + \sum \Omega(f_t)$, where $\Omega(f_t) = \gamma T + \frac{\lambda}{2}\|w\|^2$ penalizes complexity (T = leaves, w = weights). Key features: L1/L2 regularization, tree pruning, missing value handling, column/row subsampling. Important hyperparameters: n_estimators, max_depth, learning_rate, subsample, colsample_bytree, gamma,

lambda. Achieves state-of-the-art performance with built-in regularization but requires careful tuning.

2.6 Ensemble Learning

2.6.1 Ensemble Principles

Combines multiple models to reduce variance (averaging predictions), reduce bias (capturing diverse patterns), and improve robustness. Effectiveness requires diversity—models making different errors through different algorithms, training subsets, feature subsets, or hyperparameters.

2.6.2 Weighted Soft Voting

Predicts the class with highest average probability: $\hat{y} = \arg\max \sum w_i \times P_i(y = c \mid x)$, where w_i are importance weights ($\sum w_i = 1.0$) based on validation performance (e.g., F1-score). Leverages probability confidence, assigns greater influence to better models, and combines diverse algorithm strengths.

2.7 Model Evaluation Metrics

2.7.1 Confusion Matrix and Basic Metrics

The confusion matrix contains True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$ measures overall correctness but misleads with imbalanced data.

Precision = $\frac{TP}{TP+FP}$ measures positive prediction correctness (important when false positives are costly).

Recall = $\frac{TP}{TP+FN}$ measures actual positive identification (critical when missing positives is costly).

F1-Score = $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$ harmonically balances both, penalizing extreme values and proving robust to imbalance.

2.7.2 ROC and Precision-Recall Curves

AUROC (Area Under ROC Curve): Plots True Positive Rate vs. False Positive Rate across thresholds. Represents the probability that a randomly chosen positive

ranks higher than a negative. Provides threshold-independent evaluation but may be overly optimistic for imbalanced data.

AUPRC (Area Under Precision-Recall Curve): Plots Precision vs. Recall across thresholds. Focuses on positive class performance, making it more informative than AUROC for imbalanced datasets where the minority class is critical.

2.8 Model Interpretability

2.8.1 Importance of Interpretability

Interpretability enables trust and adoption, debugging and validation, regulatory compliance, and scientific insight. Models have intrinsic interpretability (linear models, trees) or require post-hoc methods (SHAP, permutation importance).

2.8.2 Permutation Feature Importance

Measures performance degradation when a feature is randomly shuffled, breaking its relationship with the target. Algorithm: (1) Compute baseline performance, (2) For each feature, permute values and compute performance, (3) Importance = Baseline - Permuted performance. Model-agnostic, considers features contextually, but can be unreliable with correlated features.

2.8.3 SHAP (SHapley Additive exPlanations)

Assigns each feature a contribution value based on Shapley values from game theory. The SHAP value ϕ_j for feature j:

$$\phi_j = \sum_{S \subseteq F \setminus \{j\}} \frac{|S|!(M - |S| - 1)!}{M!} \times [f(S \cup \{j\}) - f(S)] \tag{2.1}$$

SHAP satisfies local accuracy ($f(x) = \phi_0 + \sum \phi_j$), consistency, and additivity. TreeExplainer efficiently computes exact values for tree models.

Global Explanations: Mean absolute SHAP values indicate overall feature importance.

Local Explanations: Instance-level SHAP values show which features contributed to specific predictions.

Ensemble SHAP: For weighted voting with weights w_i: $\phi_{j,\text{ensemble}} = \frac{\sum w_i \times \phi_{j,i}}{\sum w_i}$.

2.9 Hyperparameter Optimization

2.9.1 Grid Search with Cross-Validation

Hyperparameters control the learning process (learning rate, tree depth, regularization). Grid search exhaustively evaluates predefined combinations using cross-validation: (1) Train with cross-validation, (2) Compute average performance, (3) Select best combination. Guaranteed to find optimal grid combination and parallelizable but computationally expensive.

2.9.2 Cross-Validation Protocol

Ensures unbiased selection: (1) Split data into train and test sets, (2) Use cross-validation on training set for hyperparameter selection, (3) Train final model with selected hyperparameters, (4) Evaluate on held-out test set. Prevents overfitting to test set performance.

2.10 Data Leakage Prevention

2.10.1 Definition and Sources

Data leakage occurs when test set information influences training, causing overly optimistic estimates. Sources include: (1) Preprocessing applied before splitting (training "sees" test data), (2) Resampling applied to both sets (synthetic test samples use training information).

2.10.2 Correct Protocol

(1) Split data (stratified), (2) Fit preprocessing on training data only, (3) Apply preprocessing to both sets, (4) Apply resampling only to training set, (5) Train on preprocessed, resampled training set, (6) Evaluate on preprocessed (not resampled) test set. Ensures test set remains unseen and represents real-world conditions.

2.11 Binary Classification Formulation

2.11.1 Problem Definition

Binary classification assigns instances to one of two classes. Input: $x \in \mathbb{R}^n$, Output: $y \in \{0, 1\}$, Objective: Learn $f : \mathbb{R}^n \to \{0, 1\}$ minimizing prediction error.

2.11.2 Decision Threshold and Calibration

Classifiers output probability $P(y = 1 \mid x)$. Predicted class determined by threshold τ: $\hat{y} = 1$ if $P(y = 1 \mid x) \geq \tau$, else 0. Default $\tau = 0.5$, adjustable based on class imbalance and cost trade-offs.

Well-calibrated classifiers produce probabilities reflecting true likelihood. Calibration methods: Platt Scaling (logistic regression on outputs), Isotonic Regression (non-parametric monotonic function). Logistic Regression naturally produces calibrated probabilities; tree models may require calibration.

2.12 Chapter Summary

This chapter established the theoretical foundation for AMR prediction, covering supervised learning, feature engineering and selection (hybrid ANOVA-XGBoost), class imbalance handling (SMOTE-Tomek), classification algorithms (Logistic Regression, SVM, Decision Tree, Random Forest, XGBoost), weighted soft voting ensembles, evaluation metrics (accuracy, precision, recall, F1-score, AUROC, AUPRC), model interpretability (permutation importance, SHAP), hyperparameter optimization, and data leakage prevention. These concepts form the methodological foundation for the subsequent research.

2.13 Chapter Summary

This chapter established the theoretical foundation for AMR prediction, covering supervised learning, feature engineering and selection (hybrid ANOVA-XGBoost), class imbalance handling (SMOTE-Tomek), classification algorithms (Logistic Regression, SVM, Decision Tree, Random Forest, XGBoost), weighted soft voting ensembles, evaluation metrics (accuracy, precision, recall, F1-score, AUROC, AUPRC), model interpretability (permutation importance, SHAP), hyperparameter optimization, and

data leakage prevention. These concepts form the methodological foundation for the subsequent research.

Chapter 3

Literature Review

3.1 Antimicrobial Resistance Overview

3.2 Machine Learning in Genomics

3.3 Related Works

Table 3.1: **Summary of the main characteristics of the study.**

Authors and Country	Number of Microbial Strains	Antibiotics Studied	Critical and High-Priority Pathogens Studied	Machine Learning Model	Assessr Perforr
This Study (2025), Bangladesh	6,100	Meropenem, Doripenem, Imipenem, Ertapenem	*Escherichia coli*, *Klebsiella pneumoniae*, *Pseudomonas aeruginosa*	Logistic Regression, Decision Tree, Random Forest, XGBoost, SVM, R-Blend Ensemble	Accurac cision, F1-score Confusic trix
Sunuwar & Azad (2021), USA	724	Doripenem, Ertapenem, Imipenem, Meropenem	*Escherichia coli*, *Klebsiella pneumoniae*, *Pseudomonas aeruginosa*	LR, gNB, SVM, DT, RF, KNN, LDA, mNB, AdaBoost, GBDT, ETC, BG	Recall, cision, k-fold validatic

Chapter 4

Methodology

4.1 Overview

This chapter presents a comprehensive methodology for predicting Antimicrobial Resistance (AMR) using machine learning techniques applied to resistance gene profiles. The proposed approach demonstrates that resistance gene-based prediction can achieve performance comparable to, or exceeding, whole-genome sequence (WGS)-based methods while maintaining computational efficiency and biological interpretability.

The methodology encompasses six key stages:

1. Dataset Collection and Preprocessing
2. Feature Engineering
3. Feature Selection
4. Class Imbalance Handling
5. Model Training and Ensemble Construction
6. Model Interpretability and Explainability

The pipeline addresses specific challenges of AMR genomic data, such as high dimensionality, sparsity of binary gene matrices, and class imbalance. Figure **??** illustrates the complete workflow of the proposed methodology.

4.2 Dataset Collection and Preprocessing

4.2.1 Data Source

The study utilized antimicrobial susceptibility testing (AST) data and genomic profiles from the NCBI Pathogen Detection Database. Resistance gene annotations were obtained via the AMRFinderPlus pipeline. Twelve datasets were compiled:

- *Klebsiella pneumoniae*: Doripenem, Ertapenem, Imipenem, Meropenem
- *Escherichia coli* and *Shigella*: Doripenem, Ertapenem, Imipenem, Meropenem
- *Pseudomonas aeruginosa*: Doripenem
- *Salmonella enterica*: Streptomycin, Kanamycin
- *Campylobacter jejuni*: Clindamycin

4.2.2 Binary Gene Matrix Construction

Each isolate was represented as a binary gene presence vector $\mathbf{x}_i \in \{0,1\}^n$. Detection completeness metadata (e.g., COMPLETE, PARTIAL) was encoded as additional binary features. The label $y_i \in \{0,1\}$ denotes resistance status.

4.3 Feature Engineering

4.3.1 Resistance Gene Load Score (R-Score)

The R-Score is computed as:

$$\text{R-Score}_i = \sum_{j=1}^{n} x_{ij}$$

It is then normalized using min-max scaling:

$$\text{R-Score}'_i = \frac{\text{R-Score}_i - \min(\text{R-Score})}{\max(\text{R-Score}) - \min(\text{R-Score})}$$

4.3.2 Justification and Impact Analysis

Inclusion of R-Score improved model performance and provided biologically interpretable separation between resistant and susceptible classes.

4.4 Feature Selection

4.4.1 Stage 1: ANOVA F-test

Univariate statistical filtering was performed using ANOVA F-statistics. Features with $p \leq 0.30$ were retained.

4.4.2 Stage 2: XGBoost Importance

An XGBoost model (400 trees, max depth 6) was used to compute feature importances. Features accounting for 85% cumulative importance were selected.

4.4.3 Final Feature Set

Selected features were combined as:

$$S_{\text{final}} = S_{\text{ANOVA}} \cup S_{\text{XGB}}$$

4.5 Class Imbalance Handling

4.5.1 Resampling Strategy

The study compared several strategies (SMOTE, ADASYN, etc.) and selected SMOTE-Tomek based on classification performance and biological plausibility.

4.6 Model Training and Ensemble Construction

4.6.1 Base Models

Evaluated models included:

- Logistic Regression (LogR)
- Support Vector Machine (SVM)
- Decision Tree (DT)
- Random Forest (RF)
- XGBoost (XGB)

4.6.2 R-Blend Ensemble

R-Blend is a soft-voting ensemble combining DT, LogR, and XGB with weights [1.0, 1.5, 1.0]:

$$P(y = c|\mathbf{x}) = \frac{\sum_i w_i \cdot P_i(y = c|\mathbf{x})}{\sum_i w_i}$$

4.7 Evaluation Metrics

The following metrics were used:

- Accuracy
- Precision
- Recall
- F1-Score
- AUROC
- AUPRC

4.8 Model Interpretability and Explainability

4.8.1 Permutation Importance

Assesses drop in performance when shuffling feature values.

4.8.2 SHAP Values

Computed using TreeExplainer (for DT/XGB) and LinearExplainer (for LogR):

$$f(\mathbf{x}) = \phi_0 + \sum_j \phi_j(x_j)$$

4.8.3 Ensemble SHAP Aggregation

$$\phi_j^{\text{ensemble}}(\mathbf{x}) = \frac{\sum_i w_i \cdot \phi_j^{(i)}(\mathbf{x})}{\sum_i w_i}$$

4.9 Experimental Design and Validation

- Ablation studies on R-Score, resampling, and ensemble impact
- Validation across 12 datasets with mean & std metrics
- Comparison with published methods (Her & Wu, Yasir et al.)

4.10 Implementation

Code was implemented in Python 3 using scikit-learn, XGBoost, imbalanced-learn, SHAP, pandas, and matplotlib. Experiments were run in Google Colab with GPU acceleration.

Chapter 5

Results

5.1 Descriptive Statistics

5.2 Model Performance

5.3 Feature Importance

Chapter 6

Discussion

6.1 Interpretation of Findings

6.2 Comparison with Previous Studies

6.3 Limitations

Chapter 7

Conclusion and Future Work

7.1 Conclusion

7.2 Future Research Directions

Bibliography

[1] Y. Chang, D. Lee, and H. Kim. Alignment-free identification of clones in b-cell receptor repertoires using tf-idf weighted k-mers. *Frontiers in Immunology*, 12: 791377, 2021. doi: 10.3389/fimmu.2021.791377.

[2] A. Kumar, S. Banerjee, and P. Roy. Bioset2vec: Extraction of k-mer dictionaries from biological sequences via tf-idf. *BMC Bioinformatics*, 24:62, 2025. doi: 10.1186/s12859-025-06261-7.

[3] Pavel Májek, Fajfr Oldřich, Náplavová Jana, Milan Kolář, and Jansa Petr. Genome-wide mutation scoring for machine-learning-based antimicrobial resistance prediction. *International Journal of Molecular Sciences*, 22(23):13049, 2021. doi: 10.3390/ijms222313049.

[4] NCBI Pathogen Detection. Pathogen detection isolate browser. URL `https://www.ncbi.nlm.nih.gov/pathogens`. Accessed: 2025-12-01.

[5] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] E. Rannon, T. Smith, and J. Lee. Dramma: A multifaceted machine-learning approach for novel antimicrobial resistance gene detection. *Microbiome*, 13(1):67, 2025. doi: 10.1186/s40168-025-1567-9.

[7] J. Sunuwar and R. Azad. Identification of novel antimicrobial resistance genes using machine learning, homology modeling, and molecular docking. *Microorganisms*, 10 (11):2102, 2022. doi: 10.3390/microorganisms10112102.

[8] Jitendra Sunuwar and Rajeev Azad. A machine learning framework to predict antibiotic resistance traits and yet unknown genes. *Briefings in Bioinformatics*, 22 (6):bbab179, 2021. doi: 10.1093/bib/bbab179.

[9] R. Zhang and L. Wang. Tf-idf based alignment-free methods for dna sequence comparison. *Computational Biology and Chemistry*, 94:107583, 2021. doi: 10.1016/j.compbiolchem.2021.107583.

Appendix A

Appendix A

Appendix B

Appendix B