

Machine Learning-Based Prediction of Acquired Antimicrobial Resistance in Multiple Bacterial Species Using K-mer Analysis, Mutation Detection, and AMR Gene Profiling

by

Asit Kumar Halder

20301247

Arnab Biswas

20301348

Moyuri Saha

20301277

Shaoukh Mazher Sadi

21101112

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering

Brac University

August 2025

© 2025 Brac University

All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Asit Kumar Halder
20301247

Arnab Biswas
20301348

Moyuri Saha
20301277

Shaoukh Mazher Sadi
21101112

Approval

The thesis titled “Machine Learning-Based Prediction of Acquired Antimicrobial Resistance in Multiple Bacterial Species Using K-mer Analysis, Mutation Detection, and AMR Gene Profiling” submitted by

1. Asit Kumar Halder(20301247)
2. Arnab Biswas(20301348)
3. Moyuri Saha(20301277)
4. Shaoukh Mazher Sadi(21101112)

Of Fall,2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on February 09, 2025.

Examining Committee:

Supervisor:
(Member)

Md. Sabbir Ahmed
Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Antimicrobial resistance (AMR) is a major global health concern, necessitating rapid and accurate prediction methods. This study presents a machine learning-based framework for predicting AMR across multiple bacterial species using genomic sequence data. The approach integrates K-mer analysis, mutation detection, and AMR gene profiling to extract key genomic features relevant to resistance classification. The proposed method achieves high predictive accuracy, identifying K-mer signatures, and AMR gene variations as significant contributors. This scalable approach enhances genomic surveillance and clinical decision-making, enabling efficient AMR detection without the need for phenotypic testing.

Keywords: Antimicrobial, Mutation, k-mer, Hotspot, clinical, gene profiling.

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption. Secondly, to our supervisor, sir Md. Sabbir Ahmed, for his kind support and advice in our work. He helped us whenever we needed help. Then, to our co-supervisor, sir Md. Golam Rabiul Alam, for his guidance in our work. And finally to our parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Nomenclature

DNA - Deoxyribonucleic Acid
AUC - Area under the ROC Curve
FCL - Food Corporation of India
ARG - Antibiotic resistance gene
ARGs - Antibiotic resistance genes
SARG - Structured Antibiotic Resistance Gene database
MRSA - Methicillin-Resistant Staphylococcus Aureus
CARD - Comprehensive Antibiotic Resistance Database
ARDB - Antibiotic Resistance Genes Database
ANNs - Artificial Neural Networks
ReLU - Rectified Linear Unit
CNNs - Convolutional Neural Networks
RNNs - Recurrent Neural Networks
ESBLs - Extended-Spectrum Beta-Lactamases
PCR - Polymerase Chain Reaction
AMR - Antimicrobial Resistance
ROC - Receiver Operating Characteristic
NT - -Nucleotide Sequence Language Model
BioBERT - Biomedical Text Language Model
MEGARes - The Antibiotic Resistance Gene Database
AST - Antibiotic Susceptibility Testing
AUC - Area Under the Curve
SMOTE - Synthetic Minority Oversampling Technique
LightGBM - Light Gradient Boosting Machine
XGBoost - eXtreme Gradient Boosting
MOA - Mechanism of Action
E.coli - Escherichia coli
MIC - Minimum Inhibitory Concentration
GWAS - Genome-Wide Association Studies
SPAdes - St. Petersburg genome assembler
SAMtools - Sequence Alignment/Map tools
QAD - Quadratic Discriminant Analysis
NCBI - National Center for Biotechnology Information
SNP - single nucleotide polymorphism
SAM - Sequence Alignment/Map
BAM - Binary Alignment/Map
VCF -Variant Call Format

Table of Contents

Declaration	i
Approval	ii
Abstract	iv
Acknowledgement	v
Nomenclature	vi
Table of Contents	vii
1 Introduction	2
1.1 Problem Statement	2
1.2 Research Gap	2
1.3 Research Contribution	3
1.4 Evolution Of Antibiotic resistance	5
2 Literature Review	6
3 Dataset	12
3.1 Analysis of the the primary Dataset	12
3.1.1 Mitigate the problem of primary dataset	14
3.2 Feature Extraction	15
3.2.1 Frequency-Based k-mer Features	15
3.2.2 AMR genotype Matrix: Binary representation	17
3.2.3 Amr mutation score	22
3.2.4 Mutation Count analysis	23
3.2.5 Antibiotic Phenotypic matrix	26
3.2.6 Dataset Analysis	27
3.3 Data Preprocessing	29
3.3.1 Feature Selection	29
3.3.2 Normalization	30
3.3.3 Data Encode: one hot coding	31
3.3.4 Data Reshape	31
3.3.5 Weight Balance	32
3.3.6 Synthetic Minority Over-sampling Technique (SMOTE)	32

4	Methodology	34
4.1	Workplan	34
4.2	Stack ensemble Model for Antibiotic Resistance Prediction	35
4.3	Few shot learning: Prototypical Network	41
4.4	Deep Neural Network (DNN) Model	44
4.5	CNN	46
5	Result & Analysis	49
5.1	Escherichia coli	49
5.1.1	STACK ENSEMBLE MODEL	49
5.1.2	DNN	51
5.1.3	CNN	54
5.1.4	Prototypical Network	57
5.1.5	Model comparison	59
5.2	Mycobacterium tuberculosis	61
5.2.1	CNN	61
5.2.2	Stack Ensemble Model	63
5.2.3	DNN	65
5.2.4	Prototypical Network	68
5.2.5	Model comparison	71
5.3	Salmonella Enterica	72
5.3.1	CNN	72
5.3.2	DNN	75
5.3.3	Stack Ensemble	78
5.3.4	Prototypical Network	80
5.3.5	Model comparison	84
6	Conclusion	86
	Bibliography	88

Chapter 1

Introduction

1.1 Problem Statement

This study makes use of both genomic sequencing data, mutation profiling data and AMR phenotypic data for furthering the accuracy of AMR prediction. Hence, by using k-mer frequency, organizing mutation count data and amr genotypic and phenotypic matrices, I would design a accuracy driven predictive pipeline to characterise resistant mechanisms. The use of CNNs and DNNs, along with other machine learning model such as XGBoost, LightGBM, Logistic Regression, Random Forest, etc., provide a sound analytical approach to ensure high level of prediction along with insights and at the same time less computational time. The precise objective of this study is to create a model that can help estimate the risk of antibiotic resistance in an individual and suggest relevant measures to take. Thus, the present work addresses the problem of antimicrobial resistance at a molecular level and contributes to the enhancement of this field by suggesting a large-scale, data-driven approach with the help of machine learning algorithms. Thus, the goal of this research is to develop AI models on genome-wide data to help create perfect antimicrobial approaches and fight back the development of drug-resistant infections.

1.2 Research Gap

Despite significant advancements in applying machine learning (ML) to antimicrobial resistance (AMR) prediction, several research gaps persist. Many studies primarily focus on genomic data, such as gene content and genome composition, to predict resistance phenotypes, often neglecting the integration of phenotypic data, which can provide critical insights into the expression of resistance traits under various environmental conditions. For instance, the study "Machine Learning for Antimicrobial Resistance Prediction" emphasizes the use of gene content and genome composition in ML models but does not incorporate phenotypic data [14]. Additionally, while some research incorporates mutation data, there is a tendency to focus on specific mutations rather than comprehensive mutation profiling, potentially limiting the understanding of resistance mechanisms. The study "Applications of Machine Learning to the Problem of Antimicrobial Resistance" discusses the use of ML in predicting AMR phenotypes but does not delve into the potential of deep learning architectures [10]. Moreover, there is a notable absence of causal discovery algorithms in current AMR prediction models. Incorporating causal inference methods could

enhance the understanding of the underlying mechanisms driving resistance, leading to more accurate and generalizable predictions. To bridge these gaps, future research should aim to develop integrative frameworks that combine genomic, phenotypic, and comprehensive mutation data. Employing a diverse array of ML models, including CNNs, DNNs, and causal discovery algorithms, could enhance predictive accuracy and provide deeper insights into the molecular underpinnings of AMR. Such approaches would not only improve prediction capabilities but also contribute to a more holistic understanding of resistance mechanisms, ultimately informing more effective antimicrobial strategies.

1.3 Research Contribution

My research represents a significant advancement in antimicrobial resistance (AMR) prediction by addressing critical limitations in existing studies and introducing novel methodologies for more accurate, scalable, and interpretable predictions. Unlike prior works that primarily rely on limited genomic features or employ automated feature selection, I have manually extracted and correlated key genomic indicators, including k-mer frequency, AMR gene profiling, mutation scores, and mutation counts, ensuring a more comprehensive understanding of resistance mechanisms. This manual approach enables the identification of complex patterns and interactions between mutations and resistance genes, which many existing studies fail to capture.

Additionally, I have utilized a significantly larger and more diverse dataset, consisting of 10,307 whole-genome sequences from three major bacterial species—*Mycobacterium tuberculosis* (4,640 genomes), *Escherichia coli* (2,910 genomes), and *Salmonella enterica* (2,757 genomes). The incorporation of multiple bacterial species distinguishes my research from prior studies that often focus on single-species datasets, thereby enhancing the generalizability and robustness of my predictive framework. This large-scale dataset allows me to model cross-species resistance mechanisms, which is crucial for understanding the broader epidemiological landscape of AMR.

A key novelty of my research is the integration of deep learning architectures, including Convolutional Neural Networks (CNNs) and Deep Neural Networks (DNNs), alongside traditional machine learning models such as XGBoost, LightGBM, Logistic Regression, and Random Forests. This hybrid modeling strategy optimizes predictive performance while maintaining computational efficiency, leading to higher accuracy and interpretability. Existing studies, such as Anahtar et al. (2021), primarily rely on traditional machine learning methods without fully leveraging deep learning techniques, limiting their ability to capture high-dimensional genomic patterns. In contrast, my research employs state-of-the-art deep learning techniques, which enable the identification of complex non-linear relationships between genetic mutations and AMR phenotypes.

Furthermore, I will use causal discovery algorithms, marking a groundbreaking innovation in AMR prediction. While prior research primarily focuses on associative and predictive modeling, my research pioneers the application of causal inference

methodologies to unravel the underlying biological mechanisms governing resistance evolution. Causal discovery algorithms allow me to move beyond correlation-based models and establish causal relationships between mutations and resistance phenotypes, a critical aspect that existing studies have largely neglected. By incorporating causal inference, my research provides a mechanistically informed framework that enhances interpretability and clinical relevance, making it a transformative contribution to the field.

Another distinct advantage of my research is the integration of both genotypic and phenotypic resistance profiles. Many previous studies focus solely on genomic data, overlooking the impact of phenotypic variations in AMR expression. My research bridges this gap by incorporating phenotypic resistance data, improving the model's ability to predict actual resistance phenotypes observed in clinical and environmental settings. This integration ensures that my research provides clinically relevant predictions, which can be directly applied in antimicrobial stewardship programs and public health interventions.

Additionally, I have conducted a detailed exploration of mutation patterns and their functional impact, allowing for the identification of high-impact mutations that directly contribute to AMR. While previous studies have examined specific mutations, they often fail to analyze them in a broader context alongside genomic variations and k-mer patterns. By systematically analyzing mutation counts and AMR gene variations, my research provides a more comprehensive understanding of genetic factors contributing to resistance.

In summary, my research represents a paradigm shift in AMR prediction by integrating large-scale multi-species genomic data, advanced deep learning models, causal discovery algorithms, and genotypic-phenotypic correlations into a single, comprehensive framework. The combination of manual feature extraction, extensive dataset utilization, and novel modeling approaches significantly enhances prediction accuracy, interpretability, and clinical applicability. By addressing major research gaps in AMR prediction methodologies, my research sets a new benchmark in genomic epidemiology, providing a scalable, data-driven, and clinically relevant solution to combat the global challenge of antimicrobial resistance.

1.4 Evolution Of Antibiotic resistance

Antimicrobial resistance genes are comprehensive in nature since it is the part of the environmental “resistome,” despite their natural functions still remains unclear (Davies, 2010). It is mentioned in [2] Resistance was introduced in pathogenic bacteria very quickly after the emergence of new antibiotics in the clinic, for example with penicillinase being invented in 1940 where penicillin was widely used before. Decades of man-made activities including pharmaceutical production, agricultural use and pollution have widely enlarged the environmental reservoirs of resistance genes, providing intense selection pressure. Diverse genetic mechanisms enable resistance to evolve and spread quickly between bacteria, by various factors which include mutations, horizontal gene transfer between plasmids and integrons, recombination and gene amplification. Integrons in particular play a vital role in capturing resistance genes from various environmental factors and organize them into clinical pathogens. Multidrug resistant “superbugs” like methicillin-resistant *Staphylococcus aureus* (MRSA) have appeared from augmenting mutations in their genomes and obtaining multiple antibiotic resistance genes. Simply the restriction of antibiotic use alone is not a proper solution to stop this vast evolution of antimicrobial resistance. A steady supply of new antibiotics is needed but it has declined due to pharma disinvestment. Further research is required on the natural functions of antibiotics and its resistance in natural settings against the clinical settings, which could provide us better solutions to antibiotic resistance related issues.

Chapter 2

Literature Review

The advent of machine learning (ML) and artificial intelligence (AI) has revolutionized many fields, including the study of antibiotic resistance. This literature review explores various machine learning approaches used to predict antibiotic resistance, identify antibiotic resistance genes (ARGs), and understand the mechanisms underlying resistance. The review is divided into two main subsections: one focusing on deep learning and neural network models and the other on traditional and hybrid machine learning approaches. These studies collectively demonstrate the potential of machine learning to transform antibiotic resistance research and enhance public health strategies.

In paper [9], Stokes et al. trained a deep learning network to predict antibiotic activity based on growth inhibition data for 2,335 small molecules against *E. coli*. The directed message passing neural network model achieved a test AUC of 0.896. When applied to screen the Drug Repurposing Hub library of 6,111 compounds, 51 of the top 99 predicted molecules inhibited *E. coli* growth experimentally. Most notably, the model identified the compound halicin, which displayed broad-spectrum bactericidal activity against *E. coli*, *M. tuberculosis*, and multidrug resistant pathogens. Halicin also showed efficacy in mouse models of *A. baumannii* and *C. difficile* infection. Screening a subset of 107 million ZINC15 molecules resulted in 8 experimentally validated hits out of 23 top predictions, despite limited structural similarity to known antibiotics. The nonlinear deep learning approach enabled generalization to novel antibiotic chemotypes unseen in training data. A key limitation was the small, structurally biased training dataset, restricting model accuracy and chemical scope. Expanding training data diversity and size could further improve model performance and applicability domain. Overall, this study demonstrates the potential of machine learning models trained on modest datasets to expand chemical search space and accelerate antibiotic discovery.

In another paper [4], Arango-Argoty et al. built an enlarged ARG database called DeepARG-DB, which is the combination and de-replication of CARD, ARDB, and UniProt ARG sequences. Additional annotation and curation of UniProt-derived sequences was also performed. Two deep neural network models were developed. They are DeepARG-SS and DeepARG-LS. DeepARG-SS was used for the classification of

short reads and DeepARG-LS was used for full gene sequences. The models used the distribution of similarity scores to known ARGs to make predictions between 30 antibiotic resistance categories. A number of experiments were performed to evaluate the comparison between DeepARG approach and the best hit approach which consisted of simulated short reads, full genes, novel β -lactamases, pseudo ARGs, and spike-in data. Precision, recall, and F1 scores were calculated accordingly. For short reads, the DeepARG-SS had 0.97 precision and recall of 0.91 where the best hit approach achieved 0.96 precision and only 0.51 recall. DeepARG-LS performed far better with almost 0.99 precision and recall for full gene sequences. DeepARG-LS precisely classified 85% of novel experimentally validated β -lactamases. The training genes consisted of less than 40% sequence identity, which highlighted DeepARG's ability to detect diverse novel ARGs. In the experiments, the DeepARG-SS classified ARG reads with 0.99 sensitivity even though they were made up of only 0.15% of all the data. Best hit had 0.55 sensitivity along with recommended cutoffs. DeepARG-LS accurately filtered out 95% of pseudo ARGs compared to best hit which filtered out 43%, which showed stronger discrimination of true ARGs. DeepARG grasps the complete distribution of similarity scores to known ARGs which enables detection of diverse genes which was missed by the best hit approach. Performance is dependent on the quality and availability of training data. Precautions should be taken for ARG categories with limited training dataset. Probability scores can identify less accurate predictions which may need manual inspection. Despite being restricted to predicting 30 known categories, the DeepARG is a powerful new tool for environmental ARG monitoring and surveillance with higher accuracy.

Li et al developed a novel approach which combines broad-spectrum ARG profiling using metagenomics with machine learning classification by using SourceTracker to quantify ARG pollution sources in paper [5]. They carried out the trials on 656 metagenomic datasets which covered diverse environments globally. The SourceTracker achieved about 88% accuracy in predicting the source environment of samples which was attained on their ARG profiles. The researchers obtained 656 metagenomic datasets from public databases which covered various environments like human/animal gut, wastewater, soil, ocean. The antibiotic resistome (collection of antibiotic resistance genes) was profiled by using Metagenomic sequencing. The ARG-OAP pipeline was used to annotate and quantify ARGs in the metagenomic data against the SARG antibiotic resistance gene database. The SourceTracker classified sample ARG profiles and quantified the contribution of different source environments which used a leave-one-out cross validation approach. Various statistical techniques like Pearson correlation, indicator value analysis, linear regression were used to validate the performance of SourceTracker, identify representative indicator ARGs, and find correlations with overall ARG abundance.

Allen (2014) reviews recent discoveries of antibiotic resistance genes in food-producing animals using both classical techniques like culturing and PCR as well as high-throughput metagenomic sequencing. PCR and culturing have helped uncover the dissemination of known resistance genes like ESBLs in animal microbiomes. Meanwhile, metagenomic studies reveal the breadth of antibiotic resistomes in cattle,

chickens, and swine, with 3-4% of sequences conferring resistance even without antibiotic exposure. Metagenomics provides a broad view of resistance genes but lacks genetic context. Approaches like functional metagenomics help maintain context and have revealed novel resistance genes and their linkage to mobile elements in pigs. Overall, animals harbor a reservoir of antibiotic resistance contributing to the problem in human pathogens. However, the paper [3] notes animals did not necessarily originate these genes initially. Larger integrated datasets on antibiotic use, resistance trends, and transmission routes are needed to fully elucidate the linkages between human and animal resistomes. Machine learning applied to big data could help overcome current limitations and knowledge gaps.

In the study by Aytan-Aktug et al., a machine learning model was used to determine AMR resistance using whole genome sequences and susceptibility profiles of the multiple bacterial species. In this research [7] The data was fed into the model in Batch inputs with a batch size of 100; epochs were set to 5000, for multispecies datasets and also species-specific methods. They tuned the hyperparameters based on an averaging of validation data performances (Aytan- Aktug et al. 2020), and used a one-hidden-layer neural network model with 200 hidden neurons for dataset modeling. The multispecies data was subjected to the more complex, multiple hidden layer models. A ReLU was used as an activation function in the output layer, and a stochastic gradient descent with L2 regularization was selected to update weights set at the initial stage (Aytan-Aktug et al. 2020). The models' performance can be evaluated through ROC curves while loss plots by demonstrating that they generally overperformed, or performed a greater number of antimicrobial agents, bacterial isolates and also bacterial species should also be used to give a more accurate picture on the global AMR variations.

Yoo et al. proposed a model to forecast antimicrobial resistance genes using a nucleotide sequence language model (NT) and a biomedical text language model (BioBERT), which was fine-tuned on the CARD and MEGARes databases. On the CARD database, NT achieved about 87.92% accuracy, on the other hand BioBERT achieved about 97.22% accuracy for antibiotic resistance drug class prediction. On the MEGARes database [15], NT achieved about 89.61% accuracy and BioBERT had achieved 99.64% accuracy. By combining the databases and using a band of the two models, accuracy reached up to 97.55% on CARD and 99.99% on MEGARes. However, performance dropped on a combined dataset, likely because insufficient labeled data for some classes. The authors proposed using data augmentation and sequencing reads to improve the results. Overall, the fusion of nucleotide and text language models shows promising results for predicting antimicrobial resistance, but larger integrated training data is needed to handle variation of gene classes and mutations.

Pascucci, M.Royer, G.Adamek, J. et al. described the development and evaluation of an AI-based mobile application for examining antibiotic susceptibility testing (AST) results in the paper [13]. They built a fully offline smartphone application that can examine the disk diffusion AST images by automatically measuring the diameters of the inhibition zone and categorizing bacterial vulnerability.

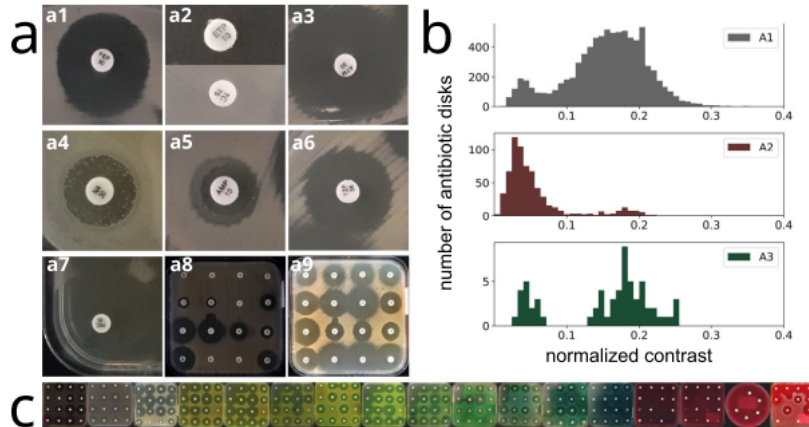


Figure 2.1: Enter Caption

from figure [2.1] the dataset used for bacterial inhibition zone analysis. These include damaged plates (a) and images with poor contrast between bacteria and inhibition zones (b), likely due to factors like variable illumination and bacterial pigmentation. These problematic images, as illustrated by standard examples (c), represent a small fraction of the total dataset and are predominantly found in the lower-contrast region. Low-contrast images significantly affect analysis outcomes, resulting in worse mean diameter differences (e).

The app consists of image processing algorithms and a rule-based expert system. The user can operate the app by taking a photo of the AST plate, processing the images, and viewing the interpreted antibiotic susceptibility results, which can be done on the same smartphone device. The authors validated the application's performance by automatically analyzing AST images from 3 datasets consisting over 600 plates. In comparison to the manual laboratory measurement, the app achieved 98% agreement on susceptibility categorization. The fully automated image processing procedure showed better performance compared to both manual and commercial AST reading systems. The app focuses on increasing the access to AST and aids in accurately detecting antibiotic resistance, especially in resource limited systems. It can be operated offline on basic smartphone devices, which removes hardware limitations of commercial systems. Overall, the results demonstrate the likelihood of accurate, automated AST image processing by a smartphone app, which can help combating antibiotic resistance worldwide.

Feretzakis et al. (2021). Machine Learning for Antibiotic Resistance Prediction: A Prototype Using Off-the-Shelf Techniques and Entry-Level Data to Guide Empiric Antimicrobial Therapy [12]. Antimicrobial susceptibility dataset of 11,496 instances from 499 patients admitted to the internal medicine wards of a public hospital in Greece. Contains attributes like sex, age, sample type, Gram stain, 44 antimicrobial substances, and antibiotic susceptibility results. The dataset was processed using Microsoft Azure AutoML to evaluate antibiotic susceptibility predictions. A 10-fold cross-validation approach was used to evaluate the performance of the models. The balanced dataset was also processed using SMOTE (Synthetic Minority Oversampling

Technique) method. Supervised machine learning techniques like ensemble models (stacking, voting etc.) and tree-based models (LightGBM, XGBoost) have been used for classification to predict antibiotic resistance. The stack ensemble technique achieved the best results in both original and balanced datasets with AUC-weighted (area under ROC curve) scores of 0.822 and 0.850 respectively.

The paper [8] proposes causal machine-learning models for analyzing microbiome data, particularly for identifying pathogenic, beneficial, and antibiotic-resistant bacteria (Sazal et al., 2020). The authors used 5 microbiome datasets related to inflammatory bowel disease with samples ranging from 82 to 749. They computed the causal effects between microbial taxa using the equation based on do-calculus.

The problem formulation for causal effects among taxa is as follows: Let $T = \{B_1, B_2, \dots, B_n\}$ be the set of microbial taxa present in the cohort of healthy or disease samples with abundance values $\{b_1, b_2, \dots, b_n\}$. For two taxa $\{B_i, B_j\} \in T$, the causal effect of B_i on B_j is given by:

$$\delta_k(x) = -\frac{1}{2} \log |\Sigma_k| - \frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) + \log \pi_k$$

Their method achieved up to 29.2% and 32.9% improvement in identifying beneficial and pathogenic bacteria respectively compared to standard approaches. By modeling the causal relationships between antibiotics, microbial taxa, and disease status, the algorithms can predict interventional effects without controlled experiments. The counterfactual modeling approach shows promise for gaining insights into microbiome disease mechanisms.

There is a study by Espinoza et al. (2021) [11] that proposed the use of a machine learning method in the prediction of compound MOA by using the whole transcriptome gene expression information, which is desirable in this paper [10]. Specifically, the authors provided the construction of the hierarchical ensemble of classifiers (CoHEC) capable of predicting MOA with the help of the transcriptomics data. The model has a tree-like structure with logistic regression classifiers at each node capable of performing binary decisions that eventually result in the MOA classification. They also invented an algorithm—Clairvoyance, which helps to select the subsets of genes most predictive for all individual binary nodes. This enhances the model accuracy and understanding. The CoHEC mode was trained using the transcriptomics dataset of 41 antibiotics in *E. coli*, which represent six MOA classes. It achieved over 99% accuracy in the MOA prediction for a novel unseen compounds through rigorous cross-validation. The model was also applied to the extracts of producer strains and antibiotics with recently identified novel MOAs, such as darobactin. It precisely classified the extracts and also determined the mechanism of action (MOA) for darobactin as possibly a novel. The model applicability is shown through the application of the aforementioned method on transcriptomic and metabolomic datasets from other studies that included different kinds of organisms, MOA classes. It outperformed previous methods. In general, this type of explainable AI allows for quick and high-throughput antimicrobial MOA prediction from the omics data. It could facilitate the rapid identification of new antibiotics and molecular entities with a very unique mechanism. The community can take more advantage of and also

enhance the published algorithms.

The literature review of [6] in the document explores the role of bacterial metabolism in antibiotic lethality, employing a white-box machine learning approach to elucidate the underlying mechanisms. Traditional methods often fail to reveal causal relationships, but by integrating biochemical screening, network modeling, and machine learning, the study uncovers that purine biosynthesis, driven by antibiotic-induced adenine limitation, significantly impacts antibiotic effectiveness. This limitation increases ATP demand, boosting central carbon metabolism activity and oxygen consumption, thereby enhancing antibiotic lethality. Experimental validation confirms these findings, suggesting that deeper understanding of these metabolic interactions can pave the way for innovative drug discovery strategies aimed at improving antibiotic therapies

This literature review highlights the significant advancements made in predicting antibiotic resistance and understanding its mechanisms using machine learning and AI. Deep learning models have shown high accuracy in identifying novel antibiotic compounds and ARGs, while traditional and hybrid approaches have demonstrated effectiveness in environmental surveillance and empirical therapy. The integration of diverse datasets and the continuous improvement of machine learning models are crucial for further enhancing the predictive power and applicability of these technologies in combating antibiotic resistance. The reviewed studies underscore the transformative potential of AI and machine learning in antibiotic resistance research, offering promising avenues for future exploration and application.

Chapter 3

Dataset

3.1 Analysis of the the primary Dataset

The analysis investigates clinically important bacterial strains including Salmonella enterica and Mycobacterium tuberculosis along with Escherichia coli which exhibit antimicrobial resistance properties. Three infectious bacterial strains were chosen because they regularly affect Homo sapiens and possess well-understood drug resistance patterns. The genomic information for these species originated from the NCBI (National Center for Biotechnology Information) database and their AMR profiles came from the BV-BRC (Bacterial and Viral Bioinformatics Resource Center) PATRIC database. The merger of genomic and AMR data produces an extensive analytical structure for studying antibiotic resistance in pathogens.



Figure 3.1: Genome sequence

Sample shown in the figure[4.1], This is a nucleotide sequence alignment representation, showing DNA base pairs (A, T, G, C) aligned in a 5' to 3' direction.

This research database contains 10,307 bacterial genome sequences that fall into three distinct species. *Mycobacterium tuberculosis* contains 4,640 genome sequences alongside *Escherichia coli* which has 2,910 sequences and *Salmonella enterica* has 2,757 genome sequences. The strains used in these sequence data come from human-specific infections to ensure direct relevance throughout this analysis. The dataset contains bacterial genome sequences presented in four different sequence formats which include complete genome sequences, chromosomal sequences, contig sequences, and scaffold sequences. Bacterial genome sequences show complete assembly along with annotation whereas bacterial chromosome sequences exclude all supplementary plasmids. Among bacterial genomes, Contig sequences appear as isolated fragments where complete mapping remains incomplete whereas Scaffold sequences demonstrate contig fragments placed in order but without final linkage. The diverse sequence types in the database ensure representation across variables of genome assembly and biological completeness.

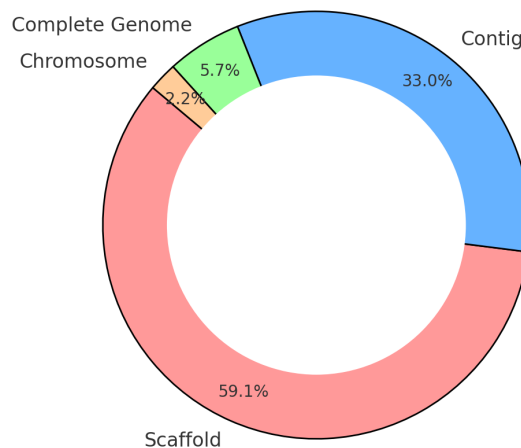


Figure 3.2: Genome sequence level

Finger [4.2], The donut-shaped diagram illustrates the different levels of sequence assembly. Approximately 59.1% of sequences classified as scaffolds are partially constructed assemblies with remaining gaps while 33.0% comprise contigs which represent unlinked and fragmented pieces. The majority of sequences (92.4%) either lack structural organization or present fragmented structures due to the limited number of complete genomes (5.7%) and chromosome-level assemblies (2.2%). The necessary step forward requires additional scaffolding techniques alongside gap-filling approaches to enhance genome completeness for accurate antibiotic resistance gene and genomic context identification.

This research obtained bacterial strains' AMR profiles from the BV-BRC PATRIC database that provides comprehensive details linking resistance genes to their corresponding antibiotics. The available architecture contains data related to resistance genes combined with antimicrobial susceptibility testing results and documented resistance mechanisms. This integration between genomic information and antibiotic resistance data enables more reliability to better understand the influence of genetic sequence changes on bacterial strain antibiotic resistance levels.

3.1.1 Mitigate the problem of primary dataset

converting the Contigs sequence into the Scaffold sequence

I used sspace and Opera-MS tools during my research to create scaffolds from contigs for constructing an extensive genome. With SSPACE I built contig scaffolds [1] through pair-end and mate-pair read analyses while the Opera-MS platform provided three-stage scaffold assembly using a combination of short and long reads for optimal accuracy.



Figure 3.3: contig to Scaffold

1. Identifying Contig Overlaps

We measured the sequential ordering of contigs through analysis of paired-end and mate-pair sequencing information. Space links contigs through paired-read data and Opera-MS establishes more precise connections by integrating long record sequencing data. Correct linking of fragmented contigs depends on this approach which establishes foundations for scaffold creation.

2. Estimating Gaps Between Contigs

For determining interfacing distances between assembled segments I have used paired-end/mate-pair read insert metrics. SSPACE has generated estimations of the expected distances between linked contigs by replacing missing sequence information with 'N' bases. On improved gap estimation through Opera-MS embraces long-read data to build more accurate scaffold structures that reduce uncertainty.

3. Linking Contigs to Form Scaffolds

I utilized SSPACE to create scaffolds through paired-read connections between contigs for generating an initial genome that uses the scaffold arrangement. Opera-MS brings long-read sequencing together with its bridging capabilities to fill the gaps and therefore improve genomic continuity. The method produced an improved continuous genome structure that improved scientists' ability to detect antibiotic resistance genes embedded within their genomic environment.

4. Refining and Validating the Scaffold

The scaffolds received refinement through Opera-MS implementation that permitted repeated scaffolding attempts to enhance genome continuity. The verified scaffolds underwent reference genome alignment followed by a quality assessment to maintain their accuracy. The established process of scaffold integrity evaluation enables more precise resistance gene detection and surrounding genetic element identification.

3.2 Feature Extraction

From the extracted genome sequence, I have computed k-mer count features to analyze sequence composition and identify patterns associated with antimicrobial resistance (AMR) genes. By mapping the sequence against a curated AMR gene database, I have identified the presence of specific resistance genes. Additionally, I have constructed a genotypic binary representation, encoding the presence or absence of mutations relevant to resistance mechanisms. To assess genomic variations, I have performed alignment with the reference genome, allowing me to determine the number of mutations that occurred within the sequence. These mutations provide insights into potential resistance mechanisms and genomic adaptations.

3.2.1 Frequency-Based k-mer Features

I analyzed genome sequences with a k of 1 (mononucleotides), 2 (dinucleotides), and 3 (trinucleotides) to understand these sequences concerning their antibiotic resistance and then extracted k-mers from 3 different bacterial classes: *Mycobacterium tuberculosis*, *Salmonella enterica*, and *Escherichia coli*. The idea was to compare the k-mer frequency distribution within the bacteria sampled and see if there are some patterns associated with antibiotic resistance genes (ARGs). Its advantage is that different bacterial species have different nucleotide composition patterns and can give some insights into genome structure, mutation patterns and resistance mechanisms.

Then processed each nucleotide of the bacterial genome to extract the 1-mer, 2-mer,

and 3-mer frequencies for each. Individual nucleotide composition in each genome is represented by the 1-mers of A, T, G, and C and can be used to analyze GC content variations across species. Dinucleotide patterns are captured by the 2-mers and can indicate mutation bias, codon preferences and genome stability mechanisms. Secondly, trinucleotide features (the 3-mers) allow for the most informative genomic features for AMR detection because trinucleotide patterns often correspond with known resistance genes or mutations.

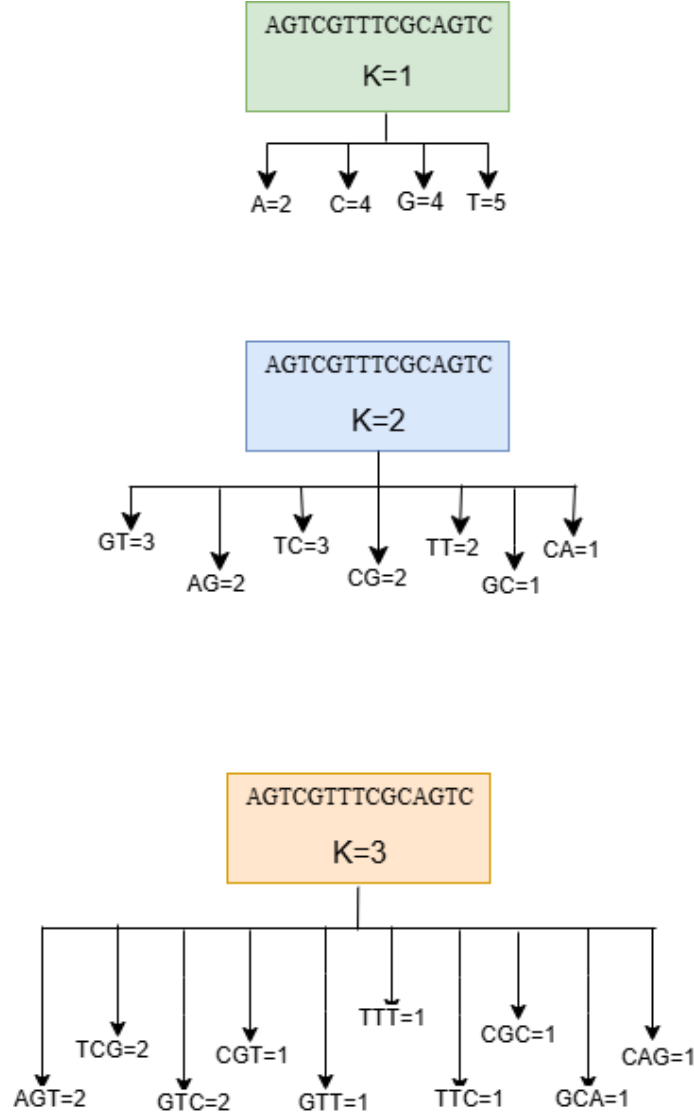


Figure 3.4: k mer count method

To detect key differences I had to extract k-mer frequencies by using k-mer algorithm figure [3.4]. The expected higher GC content of MT is due to its highly stable and slow-growing genome structure. However, *Salmonella Enterica* and *E. coli* have a stronger AT bias, similar to the fast replication rates and trends to horizontal gene transfer. The difference in nucleotide composition observed here is directly related to how species of bacteria change and evolve by adapting, making them more or less susceptible to mutations that give rise to antibiotic resistance. Moreover, the dinucleotide and trinucleotide analysis revealed that MT had more 2-mers that are rich in GC sequences and SE and *E. coli* had more AT-rich 2-mers which indicates different evolutionary pressures.

Each dataset consists of bacterial genome strain with kmer composition data, where the first column (File name) identifies each genome strain name uniquely and the other 84 columns are kmer counts to 4 single nucleotide (1-mer), 16 di-nucleotide (2-mer), and 64 tri nucleotide (3-mer) respectively. The *E. coli* dataset contains 100, 103, 108, 134, and 136 *E. coli* strains; the *Salmonella enterica* dataset has genome

Figure 3.5: k-mer Feature dataset

accession numbers (e.g., GCA_001122625.1, GCA_001134605.2) and the general dataset includes various bacterial strains. Our data consists of 84 numerical features, which represent the k-mer frequencies for each row representing a single genome strain and we can analyze mutation hot spots, transcriptional biases, and GC content variations across various bacterial genomes.

By reviewing the dataset figure [3.5]

1. GC-Rich Regions : *E. coli* and *Salmonella* have strong GC-rich clusters, confirming their genomic preference for high GC content. *Microbacterium Tuberculosis* dataset has a more dispersed GC content, reflecting variations across different bacterial genomes.
2. Mutation Hotspots: CGA and TTT remain key mutation-prone k-mers in all three datasets. *Salmonella* shows slightly weaker correlations in mutation hotspots, suggesting more balanced mutation rates compared to *E. coli*.
3. Transcription Bias : AG and GA-rich sequences are highly correlated in all three datasets, indicating a conserved transcriptional regulatory mechanism. *E. coli* exhibits the strongest AG-clustering, suggesting higher transcriptional activity compared to *Salmonella*.

3.2.2 AMR genotype Matrix: Binary representation

Genome sequences include both coding sequences which prokaryotes refer to as exons and non-coding sequences which prokaryotes refer to as introns in eukaryotic organisms. The bacterial species *Mycobacterium tuberculosis* together with *Salmonella enterica* and *E. coli* operate using operon-based genomic organization without introns. The circular bacterial genomes contain only genes which produce proteins because all DNA sequences encode proteins. The antimicrobial resistance genes located inside CDS sequences exist with plasmids and chromosomal mutations that generate antibiotic resistance properties.

Detection of AMR genes happens through an analysis of genome sequences which exist in FASTA format. Genetic sequence processing reveals mutations and reveals associated gene presence together with identified resistance mechanism variants. The following procedure explains FASTA file-based AMR gene detection.

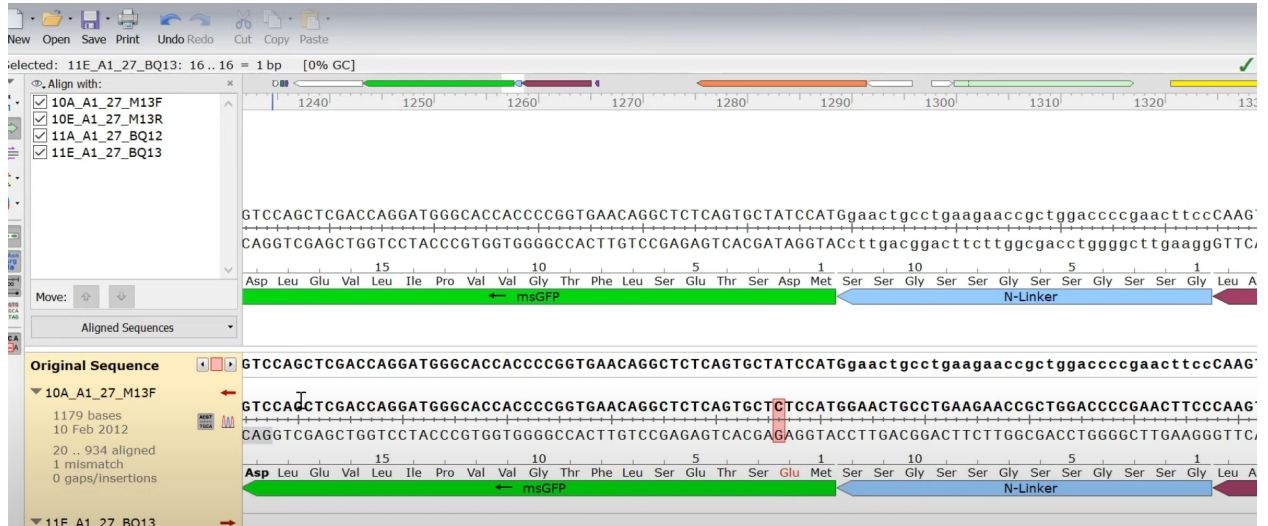


Figure 3.6: alignment with the reference genome

in figure [3.7], In the SnapGene software, with one strain of Mycobacterium tuberculosis, I have aligned with the Mt bacteria reference strain and there is a change in where red color is marked where C transferred into G to that strain. To check the whole mutation per strain of three class bacteria, we have followed some step per strain.

Converting FASTA File to SAM File:

The FASTA file contains raw nucleotide sequences of *M. tuberculosis*, *S. enterica*, and *E. coli*. To align these sequences against a reference genome, we use tools like BWA (Burrows-Wheeler Aligner). The alignment process produces a SAM (Sequence Alignment/Map) file, which contains information on how reads from the sample genome align with the reference genome.

command in wsl:

```
bwa index reference.fasta
```

The reference genome for each species we used

1. *Mycobacterium tuberculosis*: GCF_000195955.2_ASM19595v2_genomic.fna
2. *Salmonella enterica*: GCF_000006945.2_ASM694v2_genomic.fna
3. *E. coli*: GCF_000005845.2_ASM584v2_genomic.fna

Align the Sample Reads to the Reference Genome:

command in wsl:

```
bwa mem reference.fasta sample_reads.fastq > aligned.sam
```

This step aligns the sequencing reads (FASTQ format) to the reference genome and generates a SAM file, which stores alignment details.

```
SRR1234567 0 ReferenceGenome 23 60 50M * 0 0 ATGCCT-  
GATCGTAGCTAGCTAGCTAGCTGACTGATCGATCGTAGC-  
TAGCT * NM:i:1
```

Column	Description
SRR1234567	Read ID (A sequence from our genome sample)
0	Mapping flag (0 = forward strand)
ReferenceGenome	The reference sequence where this read is mapped
23	Position in the reference genome
60	Mapping quality (higher = better)
50M	CIGAR string (50 Matches to reference)
ATGC...	Actual read sequence
NM:i:1	1 mutation found (SNP detected!)

Table 3.1: SAM File Explanation

Converting SAM to BAM:

SAM files are huge and take up a lot of disk space. BAM files (Binary Alignment Map) compress and optimize storage and processing. Command in wsl:

```
samtools view -S -b sample.sam > sample.bam
```

SAM format is human-readable, but BAM is binary.

Sorting BAM File:

Sorting is required for faster and efficient variant calling. Command in wsl:

```
samtools sort sample.bam -o sample_sorted.bam
```

Indexing BAM File:

Indexing is needed to quickly access BAM data for variant calling. Command in wsl:

```
samtools index sample_sorted.bam
```

Without an index file, the system must scan the entire BAM file to find a region. With an index file (.bai), the system jumps directly to any position in the genome.

Variant Calling (BAM to VCF):

This step detects mutations (SNPs/Indels).

Command in wsl:

```
bcftools mpileup -f reference.fna sample_sorted.bam | bcftools  
call -mv -Ov -o variants.vcf
```

```
##fileformat=VCFv4.2  
##source=bcftools  
#CHROM POS ID REF ALT QUAL FILTER INFO  
ReferenceGenome 23 . C T 99 PASS DP=35  
ReferenceGenome 78 . G A 60 PASS DP=40
```

Figure 3.7: alignment with the reference genome

Column	Description
CHROM	Chromosome name (ReferenceGenome)
POS	Position (e.g., 23) where a mutation occurred
REF	Reference base (C)
ALT	Mutated base (T)
QUAL	Quality score of the mutation
INFO	Additional details (e.g., DP=35, meaning 35 reads support this mutation)

Table 3.2: VCF File Explanation

After the achieving the which sequence and which position is mutated we do separate list to identify which AMR gene can be present .so for this we have run our genome sequence through CARD database to identify AMR gene present or not . here is the process below.

Identify AMR Mutations Using CARD database by matching

We used CARD's RGI (Resistance Gene Identifier) tool to detect AMR-related mutations in our variants.vcf file.

```
rgi main --input_sequence variants.vcf --output_file rgi_results.txt --include_loose --include_nudge --include_all_hits --local
```

Matches known AMR genes in the genome. Output contains resistance genes and their associated antibiotics.

By those step , for per strain i have extracted the multiple AMR gene from genome sequence . For example ,E-coli strain BIDMC43b have 12 AMR gene whereas Microbacterium Tuberculosis per strain we get highest 3 AMR gene and Salmonella Enterica have per strain 5 or 6 Amr gene detected . so , from this amr gene list of three class bacteria .Then we have counted unique gene for whole strain per class , at first we have found 147 Unique Gene for E-coli, whereas MT have unique AMR gene of 12 unique gene . on the other hand for Salmonella 184 unique gene is found. the reason behind mycobacterium have lesser unique gene because it has Intrinsically Impermeable Cell wall which is the characteristic of Gram-positive bacteria. Moreover, MT bacteria has slow replication cycle. However, from Ecoli amr gene , we have selected only 47 gene based on their classification where each classification have different characteristics like tetracyclines inhibit protein synthesis, while -lactams inhibit cell wall synthesis. so, based on the classification , salmonella bacteria have 135 gene and Ecoli have 47 genes and MT have 12 genes. For each gene based on their presence , we create a per class binary matrix , as each class have different AMR gene. However, where ROW is identified as strain name and each column is identified as Gene name . IF Amr gene is found then previous file where detected gene list is there, then it will give value to this column 1, if not found then 0 .

0=absence of this particular gene
1=presence of this particular gene

3.2.3 Amr mutation score

IN the past Amr genotypic matrix, we have determined which Amr gene is present. Now with the list of Amr gene per bacteria , we search individually by their reference genome from NCBI. Suppose , for Mycobacterium tuberculosis have gyrA AMR gene is present . so we have searched NCBI genebank and got a view from Mycobacterium tuberculosis H37Rv which is reference genome of Mycobacterium tuberculosis. so , here red marked gene position is gyrA.

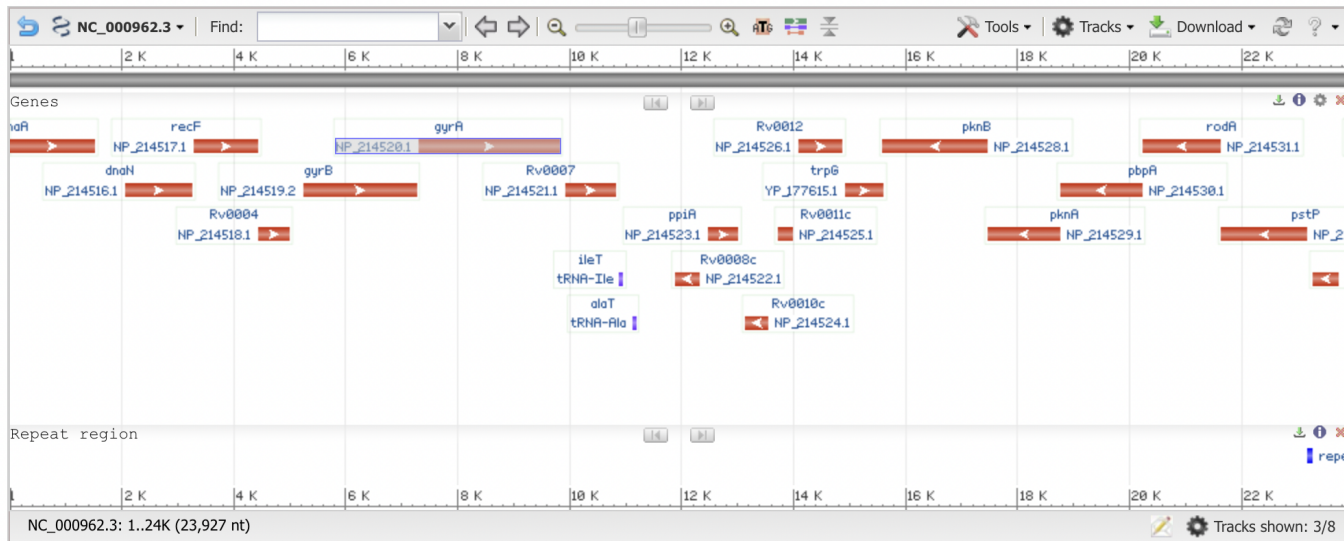


Figure 3.8: Mycobacterium tuberculosis H37Rv

after we have found gyrA from reference genome , we have marked where it situated in reference genome , which is clearly visible that between 7310 and 9815 position it locates.

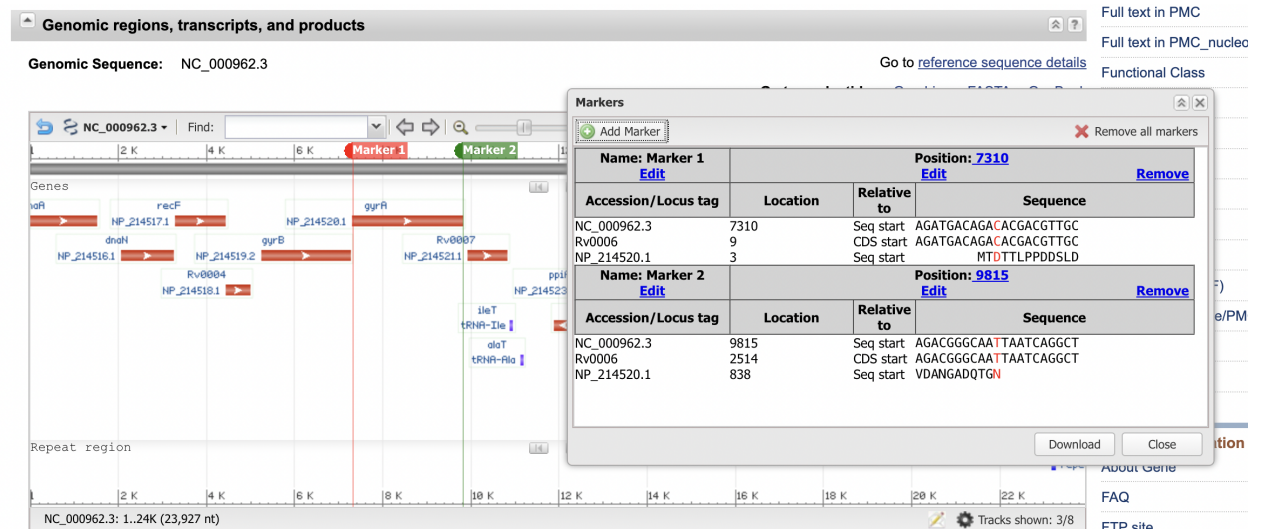


Figure 3.9: gyrA gene position in H37Rv

That's how we extracted all of AMR gene position by their reference genome from NCBI , after that we ran an algorithm to detect the ratio of mutation in that region from the VCF file we have generated.

3.2.4 Mutation Count analysis

The research process involved extraction of essential mutation characteristics from Variant Call Format (VCF) files to develop a formatted data collection. The Variant Call Format (VCF) files stored genetic variant information regarding Single Nucleotide Variants (SNVs) present in various genome sequences. My objective was to calculate different mutation types per sample through which I built numerical values suitable for computational and statistical investigations.

The dataset allows me to study antimicrobial resistance (AMR) through bacterially acquired genetic alterations which result in drug resistance. The VCF files undergo systematic processing to generate a mutation count table which functions as an input for both machine learning models together with statistical evaluations as well as comparative genomic studies.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	/mnt/f/Thesis/MT final/test data/genome sequence/00-R0023						
2	NC_00096	1977	.	A	G	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
3	NC_00096	4013	.	T	C	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
4	NC_00096	7362	.	G	C	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
5	NC_00096	7585	.	G	C	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
6	NC_00096	9304	.	G	A	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
7	NC_00096	11370	.	C	T	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
8	NC_00096	11879	.	A	G	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
9	NC_00096	13452	.	C	A	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
10	NC_00096	14785	.	T	C	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
11	NC_00096	21795	.	G	A	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
12	NC_00096	26959	.	C	G	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
13	NC_00096	30519	.	C	T	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
14	NC_00096	30688	.	T	G	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
15	NC_00096	30943	.	C	T	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
16	NC_00096	31077	.	C	T	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
17	NC_00096	34044	.	T	C	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
18	NC_00096	37031	.	C	G	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
19	NC_00096	42967	.	G	C	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
20	NC_00096	50557	.	T	C	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
21	NC_00096	51949	.	A	G	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
22	NC_00096	51954	.	T	A	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
23	NC_00096	53785	.	C	G	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
24	NC_00096	54394	.	A	G	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
25	NC_00096	57393	.	A	T	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
26	NC_00096	62012	.	C	T	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
27	NC_00096	62049	.	A	G	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
28	NC_00096	62657	.	G	A	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
29	NC_00096	66285	.	C	G	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
30	NC_00096	69871	.	C	T	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
31	NC_00096	69989	.	G	A	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
32	NC_00096	70816	.	A	G	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
33	NC_00096	71336	.	G	C	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
34	NC_00096	75264	.	C	T	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
35	NC_00096	75940	.	G	C	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
36	NC_00096	80616	.	C	G	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
37	NC_00096	83982	.	A	G	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							
38	NC_00096	92116	.	C	T	30.4183	.	DP=1;SGB:GT:PL	1/1:60,3,0							

Figure 3.10: VCF file

from Figure[3.10]its clearly visible that each row represents a point mutation where a single nucleotide (REF) is replaced by another (ALT).

For example:

- A → G at position 1977
- T → C at position 4013
- G → A at position 7362

Once the mutation frequencies were counted, I structured them into a dataset where each row represents a genome sample, and each column represents a specific mutation type.

In this study, I extracted and categorized twelve different types of single nucleotide variations (SNVs) from the Variant Call Format (VCF) files. These mutations represent specific base substitutions in the genome sequences of *Mycobacterium tuberculosis*. The extracted mutation types are as follows:

- **A>G**: Adenine to Guanine
- **T>C**: Thymine to Cytosine
- **G>A**: Guanine to Adenine
- **C>T**: Cytosine to Thymine
- **G>T**: Guanine to Thymine
- **T>G**: Thymine to Guanine
- **C>A**: Cytosine to Adenine
- **A>T**: Adenine to Thymine
- **A>C**: Adenine to Cytosine
- **G>C**: Guanine to Cytosine
- **T>A**: Thymine to Adenine
- **C>G**: Cytosine to Guanine

For every genome sample [3.11], I counted the occurrences of each mutation type and stored them in a structured tabular format. The final dataset contains each genome's filename as an identifier, with the corresponding counts of the 12 mutation types represented as separate features.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Filename	A→G	T→C	G→C	G→A	C→T	C→G	T→G	C→A	T→A	A→T	G→T	A→C			
2	00-10219	229	207	121	213	193	121	65	95	18	20	69	71			
3	00-15489	309	263	212	285	267	194	128	127	29	67	145	115			
4	00-R0023	262	237	154	265	229	153	86	94	28	29	88	97			
5	00-R0025	436	393	231	390	348	284	196	172	75	57	192	183			
6	00-R0086	345	315	221	307	292	221	138	154	60	48	148	160			
7	00-R0136	260	248	205	238	250	196	149	140	58	68	135	128			
8	00-R0139	235	199	151	239	196	144	81	85	27	19	90	86			
9	00-R0145	182	156	97	173	143	92	65	60	16	19	55	60			
10	00-R0178	382	361	372	376	337	367	258	242	112	122	252	224			
11	00-R0308	315	326	256	315	269	267	145	137	51	72	165	165			
12	00-R0434	453	430	377	415	399	358	227	265	128	131	236	232			
13	00-R1025	144	119	82	125	128	74	61	66	19	15	60	47			
14	00-R1027	171	175	104	173	162	93	68	74	13	16	64	68			
15	00-R1097	202	176	133	194	175	121	96	80	32	22	90	66			
16	00-R1152	191	192	100	199	184	96	71	69	16	18	74	53			
17	00-R1156	203	204	110	200	186	116	61	74	17	17	70	61			
18	00-R1158	257	247	197	243	246	212	155	148	52	73	143	126			
19	00-R1169	344	343	318	348	331	307	192	185	68	100	172	181			
20	00-R1178	348	337	284	328	308	265	200	158	66	80	187	142			
21	00-R1181	443	407	295	409	362	292	200	180	76	62	203	181			
22	00-R1208	302	274	202	280	252	179	130	91	26	36	133	94			
23	00-R1211	192	186	102	182	175	96	57	66	16	12	66	57			
24	00-R1244	374	340	234	340	298	231	155	150	45	60	144	163			
25	00-R1245-2	218	204	129	218	193	115	88	73	14	23	78	77			
26	00-R1246	225	216	164	207	188	167	92	111	30	55	98	118			
27	00-R1360	195	178	105	180	166	88	67	61	7	18	69	60			
28	00-R1399	168	145	82	163	133	73	54	51	13	15	52	56			
29	00-R1405	201	177	114	199	153	106	73	65	15	23	72	61			
30	00-R1547	183	164	100	183	147	88	64	63	17	22	61	66			
31	00-R1549	267	229	168	260	191	144	115	78	21	33	107	78			
32	00-R1562	363	340	231	314	305	237	149	161	63	51	148	164			
33	00-R1569	346	330	272	325	292	277	200	154	65	81	193	144			
34	00-R1588	342	324	219	319	298	243	154	155	58	47	148	159			
35	0011N	346	334	205	311	300	212	110	126	27	28	118	141			
36	0013Q	315	237	173	269	226	151	103	89	21	36	110	82			
37	0014R	342	273	206	298	232	228	133	116	28	42	111	136			
38	0016T	197	163	128	174	160	121	83	63	24	25	81	55			
39	0018W	305	254	189	279	230	164	113	81	16	33	104	78			
40	0020Y	353	317	198	300	276	194	123	113	23	32	125	134			
41	0023B	340	286	250	329	268	205	151	113	32	42	139	120			
42	0024C	195	153	128	173	151	114	70	61	24	26	84	54			
43	0026E	387	350	232	351	294	207	149	119	35	40	154	129			
44	0028G	169	158	97	148	125	100	61	46	12	19	64	43			
45	0029H	487	461	467	462	408	451	245	262	152	153	270	256			
46	0030J	429	356	316	359	323	291	195	198	91	112	193	180			
47	0031K	378	322	196	322	297	190	126	125	27	35	134	116			
48	0032L	383	336	222	354	296	205	140	123	25	35	146	124			
49	0039T	355	300	205	306	287	196	133	114	32	31	129	130			
50	0040V	425	397	328	398	356	299	167	175	37	43	139	190			
51	0041W	316	285	154	284	250	163	102	99	19	25	101	118			
52	0042X	388	367	252	367	306	232	142	119	32	33	158	133			
53	0043Y	334	287	222	291	269	234	120	119	30	49	132	93			
54	0044Z	376	350	214	350	298	206	137	119	24	36	155	131			
55	0048D	411	373	279	382	312	279	136	130	42	34	152	129			

Figure 3.11: Mutation count

Bacterial genetic mutations produce antibiotic resistance through changes that affect drug targets and efflux mechanisms and metabolic pathways. Drug resistance in *Mycobacterium tuberculosis* closely correlates with single nucleotide polymorphisms (SNPs) that occur in particular genes responsible for treating rifampicin, isoniazid, streptomycin, and fluoroquinolones infections.

- The mutations C>T and G>A in *gyrA* are known to cause fluoroquinolone resistance in *M. tuberculosis* together with *E. coli* and *Salmonella*.
- The *M. tuberculosis* strain that carries *rpoB* T>C mutations becomes malady to rifampicin treatment yet similar changes affecting RNA polymerase genes of *E. coli* and *Salmonella* result in beta-lactam drug tolerance.
- The A>G mutational changes of structural elements *acrB* and *marR* and *ramR* activate multi-drug resistance (MDR) phenomena in Gram-negative microbial agents including *E. coli* and *Salmonella*.

Bacterial protein structures transform owing to gene mutations that reduce antibiotic potency. The pattern of bacterial resistance becomes visible through genetic mutation count analysis between different bacterial strains.

3.2.5 Antibiotic Phenotypic matrix

The AMR phenotypic dataset required laboratory-tested antibiotic resistance data of bacterial species *Salmonella enterica*, *Mycobacterium tuberculosis* (MT), and *Escherichia coli* (E. coli). The specific antibiotic panel featured antibiotics that medical experts use most frequently while treating these bacterial species. Every strain from this bacterial group exists within human specimen samples. The *Salmonella*

enterica dataset includes 13 antibiotics, commonly used for treating enteric infections and systemic salmonellosis, including ampicillin, azithromycin, cefotaxime, chloramphenicol, ciprofloxacin, gentamicin, kanamycin, meropenem, spectinomycin, streptomycin, sulfathiazole, tetracycline, and trimethoprim. But in the preprocessing time, as there is too much imbalanced data, we deleted those antibiotic which have imbalanced data. so, for E-coli we have taken 7 Antibiotic. The *Mycobacterium*

tuberculosis dataset includes 12 antibiotics, which are first-line and second-line drugs for tuberculosis treatment, such as amikacin, capreomycin, ethambutol, ethionamide, isoniazid, kanamycin, moxifloxacin, ofloxacin, pyrazinamide, rifabutin, rifampin, and streptomycin. But in the preprocessing time, as there is too much imbalanced data, we deleted those antibiotic which have imbalanced data. so, for mycobacterium tuberculosis we have taken 8 Antibiotic. The *Escherichia coli* dataset includes 13

antibiotics, covering broad-spectrum beta-lactams, fluoroquinolones, aminoglycosides, and sulfonamides, including ampicillin, cefazolin, ceftriaxone, ciprofloxacin, gentamicin, tobramycin, trimethoprim-sulfamethoxazole, tetracycline, cefepime, cefoxitin, ceftazidime, amoxicillin-clavulanic acid, and levofloxacin. But in the preprocessing time, as there is too much imbalanced data, we deleted those antibiotic which have imbalanced data. so, for E-coli we have taken 9 Antibiotic.

The microbial strains had antibiotic tested that corresponded specifically to their respective microbial species. The primary phenotypic resistance values existed as categories between susceptible (S), resistant (R). Each row represent Strain Name of bacteria where each column represent the Antibiotic name. Since machine learning systems require numeric inputs for processing data, the categorical labels were transformed into numerical values as follows:

- 0 = Susceptible (S),
- 1 = Resistant (R),

The AMR phenotypic data has been structured as the ground truth output variable for predictive modeling, facilitating the development of algorithms to determine antibiotic resistance based on genetic mutations. This structured dataset enables computational analysis to predict resistance patterns and optimize antibiotic treatment strategies.

3.2.6 Dataset Analysis

The [3.12,3.13,3.14]three charts demonstrate how genetic features influence drug resistance across different antibiotics. Mutations of two distinctive types demonstrate mild and moderate correlations to drug resistance in the initial heatmap. The third image tracks DNA structure changes that either boost or decrease antibiotic resistance levels. The first heatmap displays antibiotic connection to AMR genes. The antibiotic resistance abilities of *E. coli* bacteria increase when it uses the *FloR*, *parC*, and *ble* genetic codes together. The tests with *mdsA* and *mdsB* show antibiotic sensitivity because they generate negative outcomes.

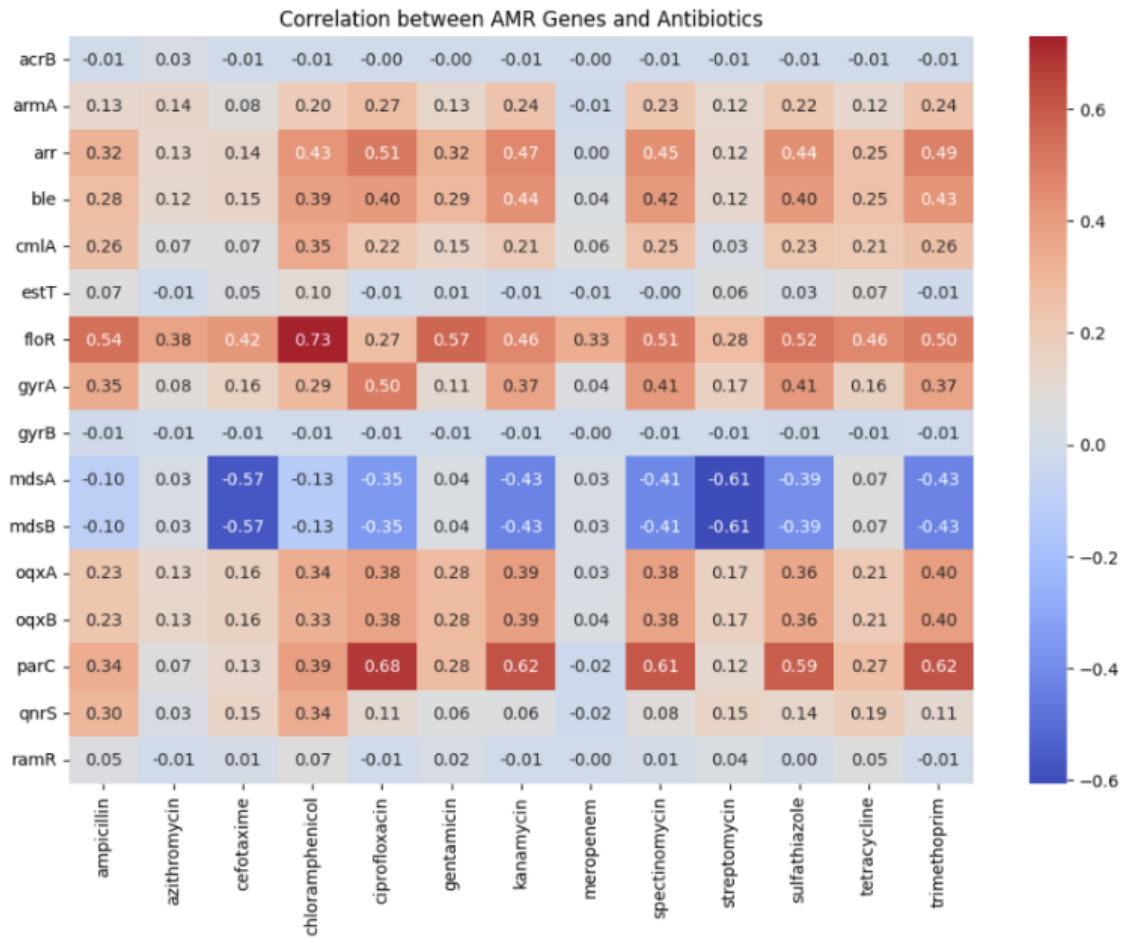


Figure 3.12: Correlation between AMR Genes and Antibiotics

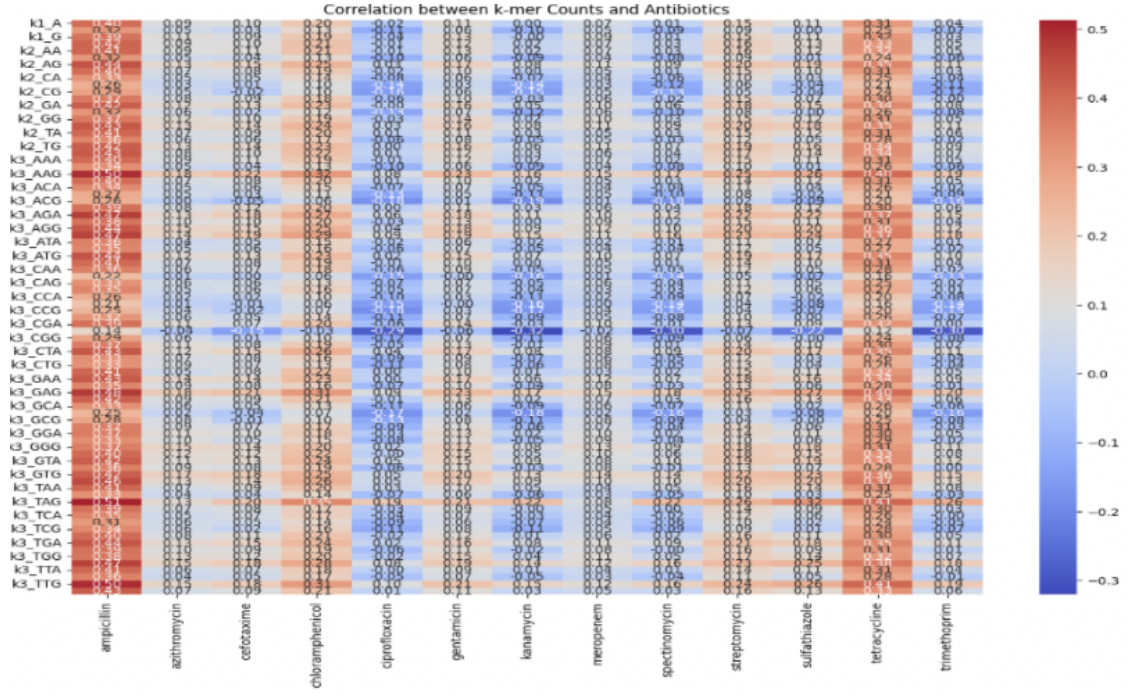


Figure 3.13: Correlation between k-mer Counts and Antibiotics

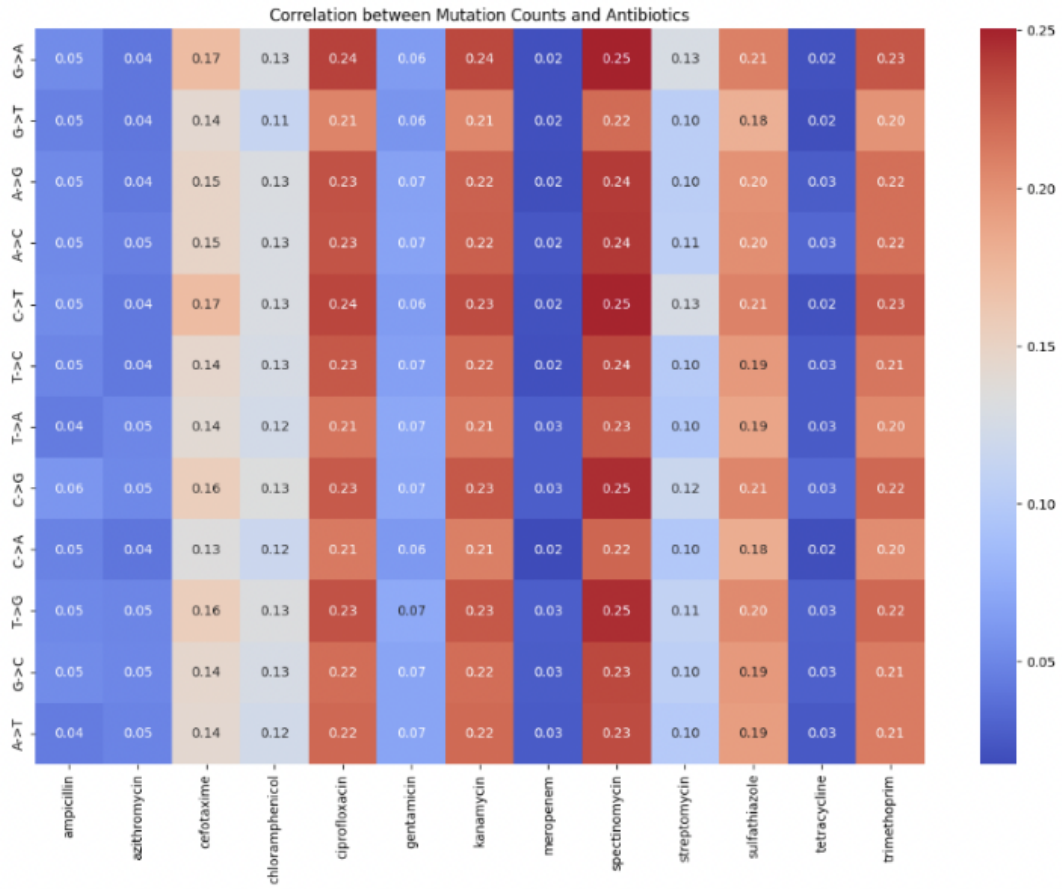


Figure 3.14: Correlation between Mutation Counts Antibiotics

3.3 Data Preprocessing

we have followed some data preprocess technique feature pruning, standard scaler data normalization, one hot coding , Data reshape for one model only , weight Balancing

3.3.1 Feature Selection

Data preprocessing initiated through evaluation of 25 antibiotics targeting *Mycobacterium tuberculosis* (MT) at its starting point. The study revealed that particular antibiotics appeared in less than 500 bacterial strains. The unbalanced distribution between classes could create problems during training since it produces model bias or insufficient data variation. The model required additional antibiotics for better generalization and robustness so I eliminated these medicines from further analysis because their limited occurrences tended to produce learning biases that would degrade predictive accuracy.

The initial distribution of antibiotics across *Escherichia coli* (E. coli) contained twenty different medications. A review of label distributions (0, 1, 2, 3) for antibiotics showed that select antibiotics contained only two categories as well as single categories that covered more than 90 percent of the data. The severe prejudice between classes diminished these features so they became useless input for the predictive model. The feature set required balance so 7 antibiotics needed removal from the dataset to maintain meaningful analysis.

When evaluating *Salmonella enterica* I used the same antibiotic processing method that I applied to MT bacteria. The distribution of certain antibiotics across different bacterial strains was extremely low which might affect the training process due to bias issues. Data integrity required the removal of antibiotics which had low representation to establish uniform data distribution. The core antibiotic features for my model included a total of 13 drugs after conducting feature selection.

The selection of antibiotics concluded but simultaneous work on improving the AMR genotypic matrix also occurred. Children of certain AMR genes were grouped together for classification categories and I examined how these genes were distributed among bacterial strains. To eliminate duplicate information and achieve an appropriate number of predictive factors I decreased the AMR gene selection yet ensured these remaining genes captured fundamental resistance variations. I reduced the AMR genes for E. coli from 147 to 47 genes and MT from 38 to 12 genes and *Salmonella enterica* from 147 to 48 genes because these genes displayed high proportional values. Preventing overfitting became essential because the model needed this step to maintain its ability to recognize bacteria from different strains.

Through this selection process I maintained an informative dataset which was free of duplications and balanced and so improved the robustness and accuracy of my AMR prediction model.

3.3.2 Normalization

Standard Scaler normalization was applied to k-mer count values as part of preprocessing operations in this stage.

The k-mer analysis counted trinucleotide sequences through setting the value at 3 for all bacterial genomes. The features totaled 84 while each counted the frequency of a particular k-mer occurrence in genomic sequences. K-mer counts contained significant magnitude differences so normalization became critical because it allowed all features to have comparable scales while avoiding domination of the model by features with higher numerical values.

The Standard Scaler normalization transforms the data to have a mean of 0 and a standard deviation of 1. The equation is given by:

$$X' = \frac{X - \mu}{\sigma} \quad (3.1)$$

where:

- X is the original feature value,
- μ is the mean of the feature,
- σ is the standard deviation of the feature,
- X' is the normalized feature value.

The data received Standard Scaler normalization that established both mean value at 0 and standard deviation value at 1. Standard Scaler normalization maintains the original distribution patterns and establishes equal scale values among all features. The normalization process improved the machine learning readiness of k-mer counts allowing such algorithms as distance-based and neural networks with sensitivity to features scales to operate effectively.

3.3.3 Data Encode: one hot coding

The AMR phenotypic matrix was transformed using one-hot encoding to convert categorical resistance levels into binary indicators. Below is the processed table:

The numerical input requirements of machine learning models led me to apply one-hot encoding for converting the categorical resistance levels into binary matrix format. I developed four new binary columns for each antibiotic which corresponded to each resistance level measurement (0, 1, 2, 3). Each respective column received a value of 1 if the sample exhibited the particular resistance level for the antibiotic being tested then the other columns were given a value of 0.

3.3.4 Data Reshape

I reshaped the features of the dataset into a 2D structured matrix for CNN training without losing their interrelated patterns between sets. The effective pattern learning required a meaningful input transformation because CNNs operate on spatial data. A total of 143 features separated into k-mer features (84), AMR genotypic features (47) and gene mutation counts (12) formed the dataset groups. The original samples consisted of one feature 143-dimensional vector ($84 + 47 + 12$). Making these features compatible for a CNN required me to transform them into a two-dimensional matrix format which maintained their relationship logic.

The selected grid dimensions included (11 x 13) and (13 x 11) because they matched the feature total of 143. Each matrix cell obtained an equal number of features so that the CNN could detect patterns with maximum efficiency. These samples required transformation through NumPy which followed this procedure:

$$X_{\text{reshaped}} = X.\text{reshape}(n_{\text{samples}}, 11, 13, 1) \quad (3.2)$$

The CNN model used phenotypic resistance level data matrices as its output target. I modified the structured output to match the specified dimensions for CNN inputs and outputs.

The conversion of the input data from one-dimensional to two-dimensional made it possible for the CNN to identify spatial patterns in the k-mer features, AMR genes and mutations which resulted in better resistance prediction accuracy. The dataset needed this modification to let the CNN extract dependencies from features so the model could make highly accurate antibiotic resistance predictions.

3.3.5 Weight Balance

During classification when dealing with imbalanced datasets, models tend to develop class bias towards the major class which results in substandard minority class predictions. The `class_weight` parameter from Scikit-Learn adjusts training influences on each class through training without editing actual dataset content. The application of `class_weight` directs the model to focus exactly on minority classes by making their errors heavier to classify.

$$w_c = \frac{N}{N_c} \quad (3.3)$$

Each class obtains additional weight through the learning process that reflects the number of available samples within its group. With the 'balanced' `class_weight` option in Scikit-Learn users can enable automatic weight calculation while users can also manually provide a dictionary weight specification like 0: 1.0, 1: 3.0 to enhance the importance of underrepresented classes. The weighting method works especially well for logistic regression and SVMs as well as decision trees and random forests because it prevents poor generalization from occurring when class distributions lack balance. The `class_weight` option is not available for Gradient Boosting and KNN models which demands the implementation of oversampling or undersampling as alternatives. A model unbalanced for weights will usually identify the most frequent class which results in good accuracy scores yet weakened minority group recognition key for detecting fraud as well as medical screenings. A model with `class_weight` implementation produces improved precision-recall trade-offs because it boosts reliability in handling class imbalances.

3.3.6 Synthetic Minority Over-sampling Technique (SMOTE)

SMOTE (Synthetic Minority Over-sampling Technique) is a resampling method used to address class imbalance in machine learning datasets. It generates synthetic samples for the minority class rather than simply duplicating existing instances, which helps prevent overfitting and improves model performance.

$$x_{\text{new}} = x_{\text{original}} + \lambda \times (x_{\text{neighbor}} - x_{\text{original}})$$

where:

- λ is a random number between 0 and 1.
- x_{original} is the feature vector of the minority class sample.
- x_{neighbor} is a randomly chosen k-nearest neighbor.

For my antibiotic resistance prediction model like Few shot learning , prototypical network model, SMOTE can help balance resistant vs. non-resistant classes, ensuring that the classifier learns meaningful patterns rather than being biased toward the majority class. The algorithm detects all minority class samples from our dataset. The k-Nearest Neighbors (k-NN) tool identifies five closest samples for each minority group

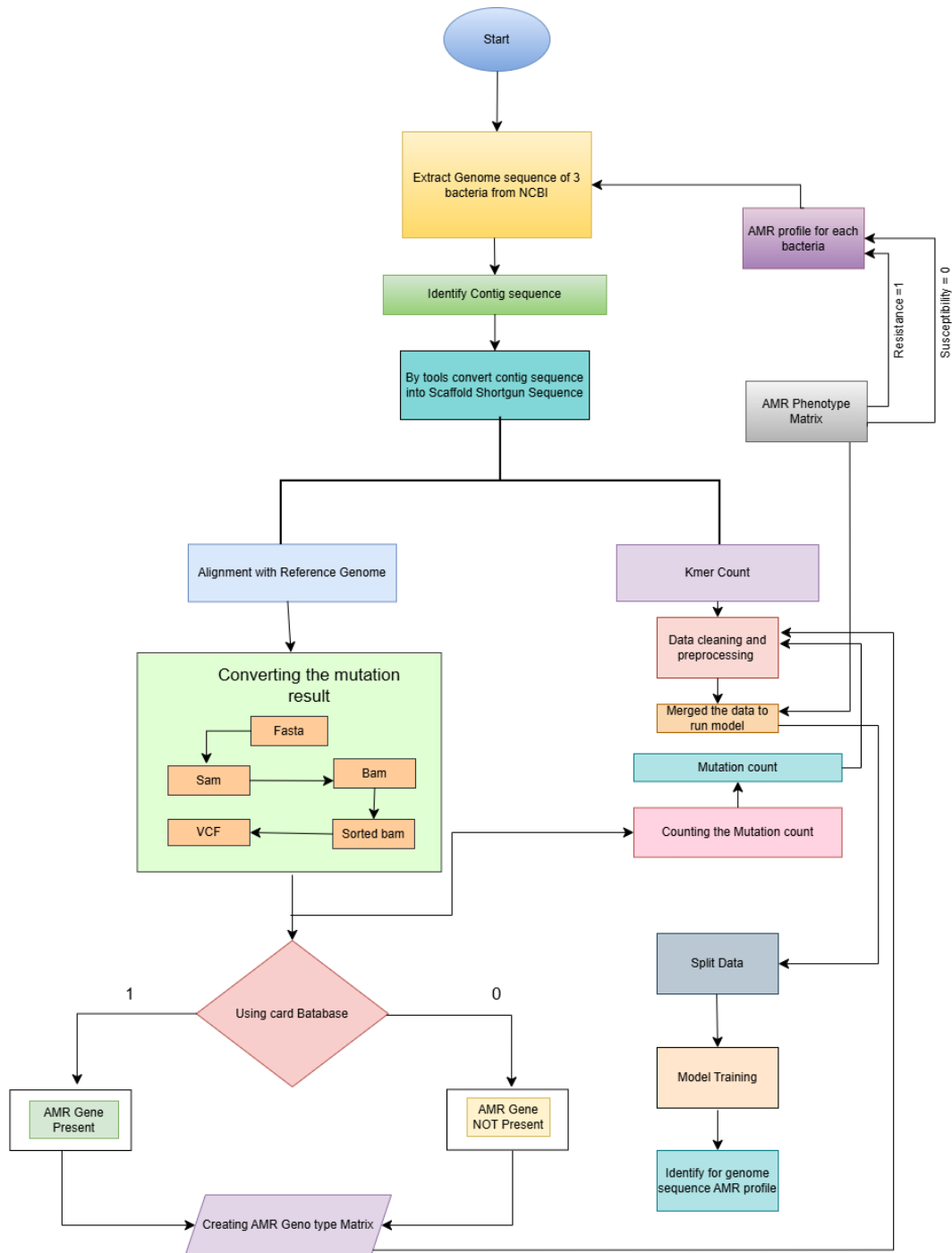
member. The model creates new synthetic data points through linear interpolation between the selected neighbor point and the original sample.

Chapter 4

Methodology

4.1 Workplan

The workflow utilizes bioinformatics together with machine learning methods for AMR gene detection in bacterial genomic material. The extraction of genome sequences from NCBI enables the identification of contigs which get converted into scaffold shotgun sequences. Two independent analytical operations begin from this point onward. The initial step applies reference genome alignment to receive mutation results as FASTA BAM and VCF formats that undergo CARD database analysis to validate AMR gene occurrences. The identification of AMR genes results in data entry for an AMR genotype matrix. The second analytical method utilizes k-mer counting alongside mutation analysis and machine learning processes to clean data that later splits into training and testing sets which enable developers to produce predictive models for AMR profiling. The AMR profile divides bacteria into two groups: resistant, susceptible ,therefore creating a phenotype AMR matrix. The combined method improves AMR detection capabilities by creating effective mechanisms for uncovering bacterial resistance traits that support antibiotic treatment guidance and battles against drug-resistant infection spread.



4.2 Stack ensemble Model for Antibiotic Resistance Prediction

Introduction to the Stacking Model

The stacking model is another method where several learning algorithms are used

for classification and the results are combined to provide a better accuracy. While in bagging or boosting the models are built independently or in sequence, in stacking they are built independently and a meta-model is used to make the final prediction.

The stacking model in the code is as follows:

Base Models (Level 1): XGB or the Gradient booster algorithm: Suitable for detecting feature interactions.

LGBM: A fast-grown boosting tree model, which is efficient in model training.

Random Forest (RF): The algorithm that uses decision trees to capture instances of non-linear regression functions.

Meta Model (Level 2) Logistic Regression (LR): This model learns how best the forecasts from the base models can be combined to make the final classification.

The stacking approach works well to improve the level of prediction by combining multiple models and helps to overcome overfitting which makes stacking a very useful method in the case of classifying antibiotic resistance.

How Stack ensemble Model Works:

The training process of the stacking model consists of the following steps:

Step 1: Feature Representation

Let:

- X be the **input feature matrix** of size $(m \times n)$, where:
 - m is the number of samples (genome sequences).
 - n is the number of features (mutation counts, AMR genes, k-mers).
- y be the **binary resistance labels** for an antibiotic (0 = susceptible, 1 = resistant).

Each sample x_i in X is represented as:

$$x_i = \{x_{i1}, x_{i2}, \dots, x_{in}\}$$

where x_{ij} represents the j^{th} feature of the i^{th} sample.

Step 2: Base Model Predictions

Each **base model** $f_k(X)$ learns a function to map input features to resistance predictions.

LightGBM Model (Gradient Boosting)

$$h_1(X) = f_{LGBM}(X) = \sum_{t=1}^T \gamma_t h_t(X)$$

where:

- T = number of boosting rounds.
- $h_t(X)$ = weak decision trees.
- γ_t = weight of each tree.

XGBoost Model

$$h_2(X) = f_{XGB}(X) = \sum_{t=1}^T \alpha_t h_t(X)$$

where:

- α_t are learned weights.
- $h_t(X)$ are boosted trees.

Random Forest Model

$$h_3(X) = f_{RF}(X) = \frac{1}{B} \sum_{b=1}^B T_b(X)$$

where:

- B = number of trees.
- $T_b(X)$ = output of each tree.

Logistic Regression Model

$$h_4(X) = f_{LR}(X) = \sigma(w^T X + b)$$

where:

- w = weight vector.
- b = bias term.
- σ = sigmoid activation function.

The output of each base model is a probability score between 0 (**susceptible**) and 1 (**resistant**):

$$h_k(X) = P(y = 1 \mid X)$$

For **each sample** x_i , the base models output:

$$H(x_i) = [h_1(x_i), h_2(x_i), h_3(x_i), h_4(x_i)]$$

Step 3: Meta Model Training

The meta-model (Logistic Regression) learns from the outputs of base models.

New feature representation for meta-model:

$$Z = \{H(x_1), H(x_2), \dots, H(x_m)\}$$

where:

- Z is an $m \times k$ matrix, where k is the number of base models.
- Each row contains **stacked predictions** from all base models.

The meta-model applies **logistic regression**:

$$y = \sigma(w^T Z + b)$$

where:

- w = learned weights for meta-model.
- b = bias.
- σ = sigmoid function.

Step 3: Model Evaluation

Following training, the meta-model is subjected to a test dataset: The evaluation

is made about true labels to measure the extent to which the given predictions are accurate or otherwise. Common measures that can be calculated include accuracy, precision, recall, and a confusion matrix.

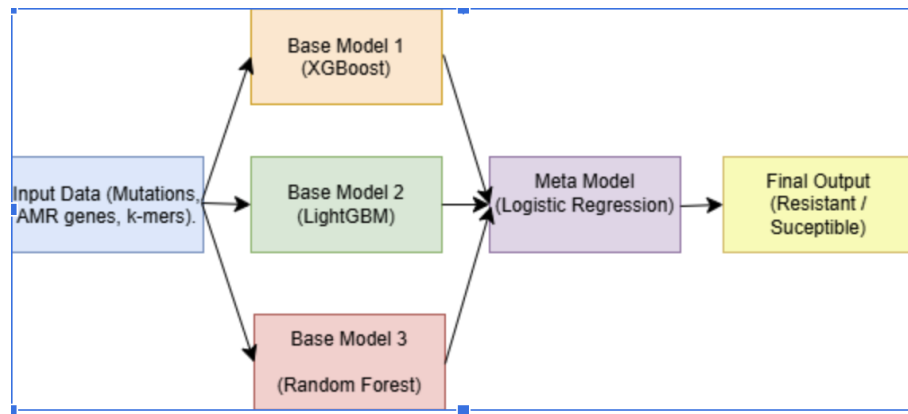


Figure 4.1: Stack Model Architecture

Diagram Explanation

Input Layer (Light Blue Box)

- Takes in mutations, AMR genes, and k-mer features.

2Base Models (Level 1)

- **XGBoost (Orange Box):** Detects feature interactions and relationships.
- **LightGBM (Green Box):** Efficient gradient boosting for high-dimensional data.
- **Random Forest (Yellow Box):** Captures non-linear patterns.

Meta Model (Level 2)

- **Logistic Regression (Gray Box):** Combines base model predictions to make the final classification.

Final Output (Light Blue Box)

- Predicts whether the bacterial strain is resistant or susceptible to antibiotics.

Connections:

- The input data is fed into each base model.
- The base model predictions are passed to the meta model.
- The meta model makes the final decision.

Importance of the Stack Model in This Dataset

my dataset contains mutation features, AMR genes, and k-mer representations,

which makes it:

- **High-dimensional:** Many features.
- **Complex and non-linear:** Interactions between mutations and resistance.
- **Imbalanced:** Some antibiotics may have fewer resistant strains.

A single model (e.g., Random Forest, XGBoost) might:

- Overfit to specific antibiotics.
- Struggle with imbalanced classes.
- Fail to capture interactions between mutations and AMR genes.

Stacking Helps Solve These Problems by Combining Multiple Models:

- **Base Models** (XGBoost, LightGBM, Random Forest) learn different aspects of resistance.
- **Meta Model** (Logistic Regression) learns which base model performs best for different types of resistance.
- **Final Prediction** (Resistant or Susceptible) is more accurate and reliable.

Importance of Stacking in Antibiotic Resistance Prediction

1. Improves Prediction Accuracy

- Mutation interactions with resistance genes are complex.
- A single model might miss some of these patterns.
- Stacking allows multiple models to learn from different perspectives.

2. Reduces Overfitting

- Genetic data can have noisy mutations.
- If one model overfits, the others help balance the decision.
- Cross-validation in stacking reduces the risk of overfitting.

3. Handles High-Dimensional Data

- Mutations, AMR genes, and k-mers create thousands of features.
- Some models perform well in different feature sets.
- Stacking selects the best model for each antibiotic resistance pattern.

4. Works Well for Imbalanced Datasets

- Some antibiotics have more resistant strains than others.
- Stacking ensures rare resistance cases are classified correctly by averaging predictions across models.

5. Provides Robust and Generalizable Predictions

- In real-world scenarios, bacteria evolve resistance.
- A single model might not generalize well to new mutations.
- Stacking makes predictions more stable across different datasets.

4.3 Few shot learning: Prototypical Network

This model uses the Few-shot learning based Prototypical Network to detect antibiotic resistances in *Salmonella enterica*, *Escherichia coli*, and *Mycobacterium tuberculosis* through genome analysis of mutation patterns, k-mer statistics, and AMR gene profiles. Machine learning systems have poor accuracy with AMR because datasets show many samples of non-resistant bacteria that cause models to prefer these strains over accurate results. Synthetic Minority Over-sampling Technique (SMOTE) generates equal amounts of resistant and non-resistant cases through synthetic samples to improve the dataset distribution.

we uses Prototypical Networks under Few-shot learning because AMR studies only have small amounts of labeled data for uncommon resistance patterns. The method builds an embedded space where samples from the same class group together so it can predict bacterial types effectively across new identified genomes with similar traits. Before processing features the model applies z-score normalization to convert all numerical values to common scaling. SMOTE controls the dataset ratios before training to stop the system from favoring susceptible case examples. The Prototypical Network system contains three linked layers that train more reliably through Batch Normalization plus LeakyReLU activation to keep neurons active and prevent overfitting with Dropout Regularization.

Embedding Function and Network Architecture

The **Prototypical Network** maps input feature vectors into a latent space using a non-linear embedding function $f_{\theta}(X)$, parameterized by weights θ .

Neural Network Layers

The **embedding function** consists of **three fully connected layers**:

1. **First Hidden Layer (512 neurons):**

$$H_1 = \text{LeakyReLU}(W_1X + b_1)$$

where $W_1 \in R^{512 \times d}$ and $b_1 \in R^{512}$.

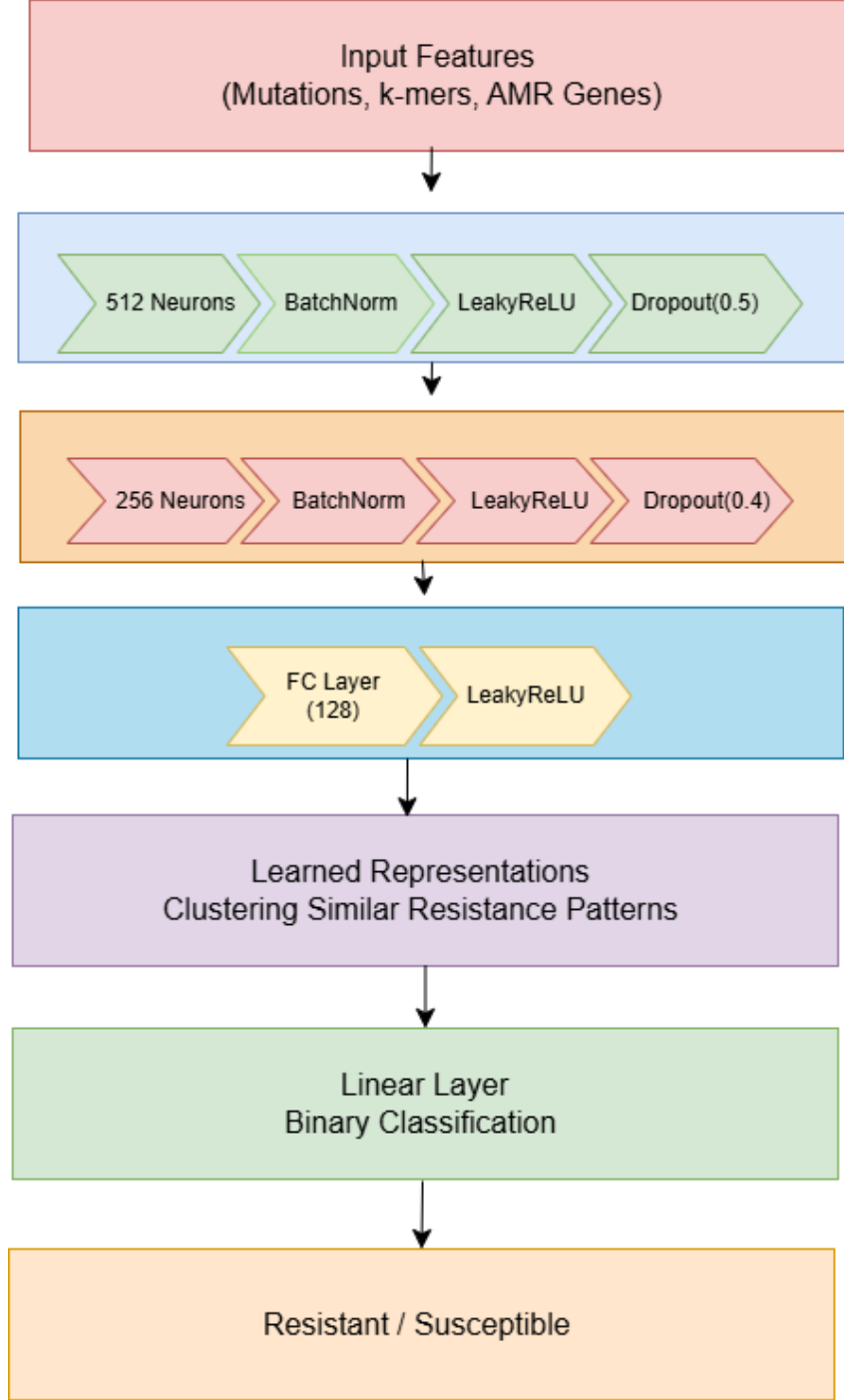


Figure 4.2: Prototypical network Architecture

2. Second Hidden Layer (256 neurons):

$$H_2 = \text{LeakyReLU}(W_2 H_1 + b_2)$$

where $W_2 \in R^{256 \times 512}$ and $b_2 \in R^{256}$.

3. Third Hidden Layer (128-dimensional embedding):

$$Z = \text{LeakyReLU}(W_3 H_2 + b_3)$$

where $W_3 \in R^{128 \times 256}$ and $b_3 \in R^{128}$.

Each layer includes **batch normalization** to stabilize training and **dropout regularization** to prevent overfitting.

The last step sends embeddings to a latent space for cluster detection before using a linear classifier for resistance class assignment. Focal Loss helps the model learn rare resistance classes better by reducing its focus on easy examples and emphasizing hard cases that need more training.

Prototypical Embedding and Distance Calculation The model learns **class prototypes**, where the prototype for each class c is the mean of its embedded support set:

$$\mu_c = \frac{1}{|S_c|} \sum_{x_i \in S_c} f_\theta(x_i)$$

where S_c is the set of support samples belonging to class c .

For a given query sample x_q , its distance to each prototype is computed using **Euclidean distance**:

$$d(x_q, \mu_c) = \|f_\theta(x_q) - \mu_c\|^2$$

The probability of belonging to class c is calculated using a **softmax over distances**:

$$P(y = c|x_q) = \frac{e^{-d(x_q, \mu_c)}}{\sum_{c'} e^{-d(x_q, \mu_{c'})}}$$

AdamW optimization acts as weight regularizer while cosine annealing learning rate scheduler slowly decreases training rate to stop overfitting during all 200 epochs.

Instead of standard cross-entropy, **Focal Loss** is used to focus on hard-to-classify samples:

$$\mathcal{L}_{\text{focal}} = -\alpha_t(1 - p_t)^\gamma \log(p_t)$$

where:

- p_t is the predicted probability for the true class,
- α_t is a weighting factor for class imbalance,
- γ is the focusing parameter (default: 2), reducing the loss for easy samples.

Optimization & Training

- **Optimizer:** AdamW (adaptive learning rate with weight decay).
- **Learning Rate Scheduler:** Cosine Annealing to gradually decrease the learning rate.

At **each epoch**, the model updates the prototype vectors using **gradient descent**:

$$\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}_{\text{focal}}}{\partial \theta}$$

where η is the learning rate.

Each dataset split has 70% training data and 15% validation and test subsets with model performance testing using an accuracy and confusion matrix along with precision, recall, F1-score calculations. **The algorithm works well with these datasets because it uses its features to classify data with few examples and handles class imbalance while recognizing how bacteria relate to drug resistance.** Its prediction system works for multiple bacterial species by finding common resistance patterns which allows scalability in AMR studies of One Health.

4.4 Deep Neural Network (DNN) Model

A Deep Neural Network (DNN) is a ANN model that contains many layers of neurons between its inputs and output layer. It is created to capture non-linearity of the data and relationships between variables and output and has several processing neurons and layers to achieve a final output of classification or regression. DNNs rely on non-linear transformation methods, optimization algorithms such as Adam and SGD and backpropagation to train from data. The architecture typically includes:

Input Layer: Receives the features (in this case, genetic mutations, AMR genes, and k-mer features).

Fully connected layers: Additional layers which also deal with hierarchal features.

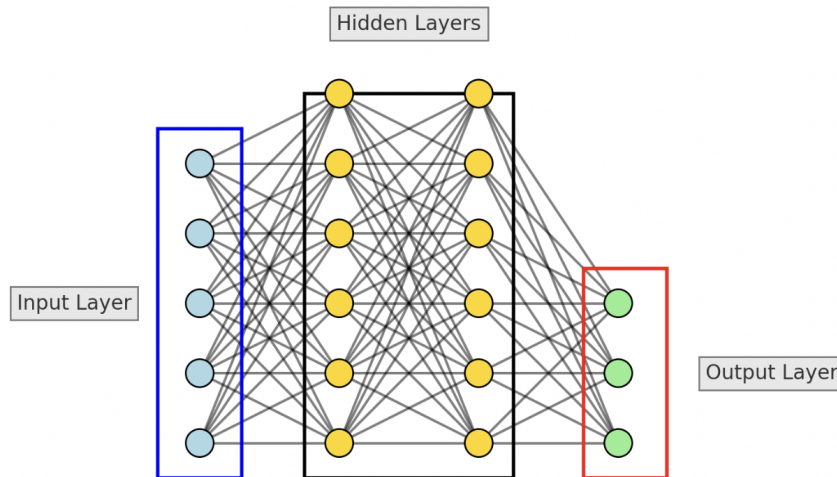


Figure 4.3: DNN Architecture

Architecture of Neural Network: Dropout Layers means that a node is shut down during the training of the model hence preventing over fitting.

Output Layer: It has a sigmoid as the activation function because the final layer is involved in the process of decision making regarding the presence of antibiotic

resistance.

In general, the representation of a DNN could be expressed as:

The general mathematical representation of a DNN is:

$$Z^{(l)} = W^{(l)} A^{(l-1)} + b^{(l)}$$

$$A^{(l)} = f(Z^{(l)})$$

where:

- l is the layer index.
- $Z^{(l)}$ is the pre-activation output.
- $W^{(l)}$ and $b^{(l)}$ are the weights and biases for layer l .
- $A^{(l)}$ is the activation output of layer l , passed through an activation function f .
- f is typically a ReLU (Rectified Linear Unit) for hidden layers:

$$f(x) = \max(0, x)$$

- The final output uses the **softmax function** for multi-class classification:

$$P(y_i) = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}}$$

where $P(y_i)$ represents the probability of class i , ensuring that the sum of all probabilities is 1.

Why Use DNN for This Dataset?

Due to the presence of mutation features, antibiotic resistance genes and k-mer features, this is a high dimensional data set. Such datasets are best analyzed using DNNs because of their capability to identify complex associations in the data given to them.

Benefits of DNN for this dataset: Here, the number of folds in high dimensionality is high due to the presence of a large number of genetic mutations and k-mer features; hence, the logistic regression may not work effectively. Non-linear relationships may be best modeled as DNNs because of the different number of layers of neurons.

Multi-layered: DNNs learn lower level and higher level genetic patterns of the data due to the several layers present in them.

Applicable to Multiclass Classification: Every antibiotic bears different levels of resistance, and the DNN can change the number of output nodes according to the antibiotic.

Interactions: Complex compared to other models, DNNs have the ability to efficiently define the interactions between genetic mutation profiles and AMR genes, and k-mer sequences that may lead to antibiotic resistance.

Class Imbalance Handling: One of the methods applied during the training, namely class weighting, helps prevent the focus on the minor class (e.g., rare instances of resistance).

Conclusion The feature set is large for this kind of data and the DNN is capable of modeling a large number of parameters, accommodates high numbered input parameters and provides efficient classification for antibiotic resistance. The kind of DL model used in this study, the DNNs, provides a training of one model per antibiotic with adaptive outputs hence highly scalable for AMR prediction. Here is the summary of the measures of central tendency: I hope you don't have any questions if you need any changes or further explanation let me know immediately.

4.5 CNN

A Convolutional Neural Network (CNN) is a shallow or deep network that focuses on the spatial features of the inputs. CNNs are mainly designed for image recognition but they are also useful for sequence-based and tabular data, which is applicable for genetic features. Key Components of CNN

Convolutional Layers: Its CONSTANT scans feature maps using kernels (filters) that slide over the architecture of the material provided.

Activation Function (ReLU): Adds non-linearity into the model to improve the ability to capture higher-order features.

Batch Normalization: This is used to make the learning process stable by normalizing layer output.

Pooling Layers (MaxPooling1D): down-sampling which reduces the size of the layer and helps in faster computations.

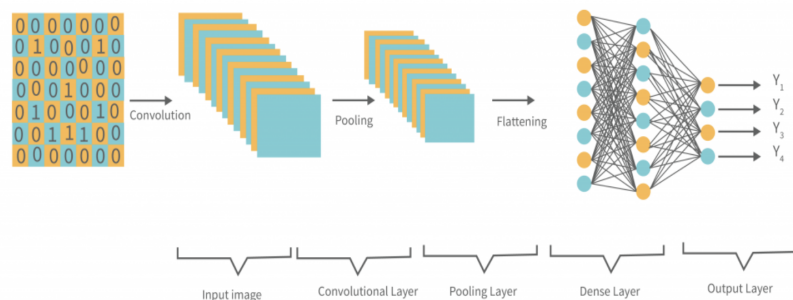


Figure 4.4: CNN architecture

Flattening: Reshapes the feature maps from the fully connected layers to a 1-dimensional vector.

Fully Connected (Dense) Layers: Learns patterns and relationships of the features passed to it from the previous layer in addition to the new features in the

subsequent layers.

Dropout Regularization: This causes overfitting by ignoring neurons during the training process.

Output Layer (Softmax Activation): Transforms the resulting values from the model into probability scores to classify them.

The convolution operation in a 1D CNN applies a **kernel (filter)** W over an input sequence X , producing an output feature map Z :

$$Z_i = \sum_{j=0}^k W_j \cdot X_{i+j} + b \quad (4.1)$$

where:

- Z_i is the output of the convolution operation at position i .
- W_j is the filter weight at position j .
- X_{i+j} is the input feature value at position $i + j$.
- b is the **bias term**.
- k is the kernel size (length of the filter).

ReLU Activation Function The **Rectified Linear Unit (ReLU)** is used as an activation function:

$$f(x) = \max(0, x) \quad (4.2)$$

This helps introduce **non-linearity** and speeds up training by **avoiding vanishing gradients**.

Max Pooling Operation

The **MaxPooling1D** layer reduces dimensionality by selecting the **maximum value** in a window of size p :

$$Z_i = \max(X_i, X_{i+1}, \dots, X_{i+p-1}) \quad (4.3)$$

This helps retain the **most important features** while **reducing computation cost**.

Fully Connected (Dense) Layer

After feature extraction, the CNN flattens the data and applies a fully connected layer:

$$Z = WA + b \quad (4.4)$$

where:

- W is the **weight matrix**.

- A is the **flattened feature vector**.
- b is the **bias term**.

Softmax Activation Function (Output Layer For **multi-class classification**, the **softmax function** computes probabilities:

$$P(y_i) = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}} \quad (4.5)$$

where:

- $P(y_i)$ is the probability of class i .
- z_i is the final score for class i before activation.
- Ensures **sum of all probabilities** = 1.

Why we Use CNN for This Dataset?

The dataset is composed of ‘genetic mutations’, ‘AMR genes’, and ‘k-mer features’

which in one way or another represent a sequence of biological data. For such a dataset, CNN is useful because:

1. Identifies Local Traits in the Genetics Data: AMR genes and mutations

have their groups that have specific local distribution regarding antibiotic resistance. CNN achieves these by use of filters or in other words, feature maps that help in feature extraction.

2. Handles High-Dimensional Data: The given dataset contains various mutations, k-mers, as well as Antibiotic Resistance and Mutation (AMR) genes. It eliminates the subset of features automatically while it maintains the main bits which is a method of reducing dimensionality.

3. Avoids Overfitting Several measures are used in high-dimensional genetic data including dropout layers to prevent clients from overfitting and the use of Batch Normalization.

4. Faster and More Efficient With regards to the architecture, while CNNs are also of the fully connected type, their structure makes them more efficient since they have less trainable parameters which makes learning faster.

5. Adaptability to 1D Genomic Data In this case, 1D feature sequences are used instead of image data, where mutations and AMR genes are considered biological sequences in CNN.

The Convolution operation in the 1D format means it scans across the genetic features and extracts useful signal information.

Chapter 5

Result & Analysis

5.1 Escherichia coli

5.1.1 STACK ENSEMBLE MODEL

Antibiotic	Accuracy	Precision	Recall
Ampicillin	0.79	0.81	0.67
Cefazolin	0.86	0.83	0.63
Ciprofloxacin	0.94	0.88	0.90
Gentamicin	0.97	0.77	0.71
Tobramycin	0.98	0.87	0.50
Trimethoprim-Sulfamethoxazole	0.94	0.85	0.84
Tetracycline	0.95	0.87	0.92
Cefoxitin	0.95	0.80	0.54
Levofloxacin	0.92	0.84	0.81

Table 5.1: performance Metrics

Confusion Matrix evaluation of Stack Model

The confusion matrices of the stacked model (XGBoost, LightGBM, Random Forest) for all the antibiotics are presented here. The classification performance of the model for separating susceptible and resistant outcomes emerges from the confusion matrices of various antibiotics. These matrices show actual class labels positioned on the y-axis while predicted class labels exist on the x-axis and darkness indicates increasing correct predictions. The model produces strong classification outcomes when identifying susceptible and resistant cases for antibiotics ciprofloxacin, gentamicin, tobramycin, trimethoprim-sulfamethoxazole, tetracycline, cefoxitin and levofloxacin. The classification results for gentamicin and tobramycin demonstrate almost complete accuracy because they yield minimal misclassification errors. Figure [5.1], The accuracy of Ciprofloxacin and tetracycline remains high since both medications maintain low counts of false negative and false positive outcomes. However, ampicillin and cefazolin show relatively higher misclassification rates. The model predicts resistance when the bacteria are susceptible in 29 cases along with labeling resistance as susceptible in 61 cases indicating moderate limitations when comparing these bacterial categories. The 50 false negatives found in Cefazolin indicate the misidentification of numerous resistant bacterial cases as susceptible to making incorrect medical choices. The

model shows successful overall results but still makes incorrectly classified results for cefoxitin and levofloxacin. Resistance tests that identify cefoxitin as susceptible when it should be reported as resistant occur in 14 cases. The 17 false negatives from Levofloxacin assessments could still produce adverse effects on resistance detection operations. The model shows superior performance in distinguishing resistance profiles of distinct antibiotics including gentamicin and tobramycin yet needs improvement for identifying resistance in ampicillin and cefazolin because of high misdiagnosis rates. The accuracy rate for classifying resistant bacteria samples could improve following additional training datasets and optimized selection of features and model parameter adjustments.

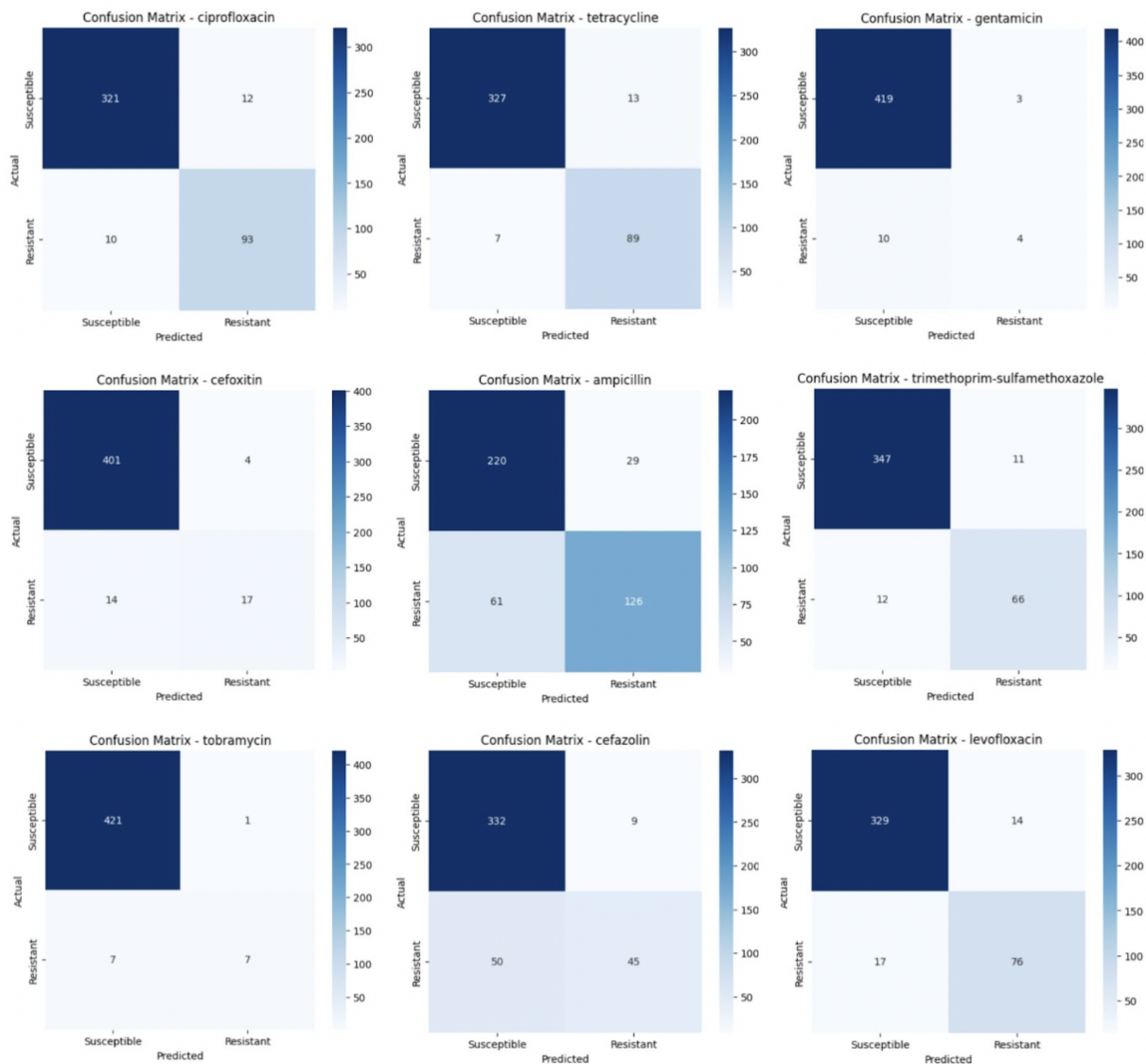


Figure 5.1: Stack Model Ecoli

5.1.2 DNN

Antibiotic	Accuracy	Precision	Recall
Ampicillin	0.78	0.78	0.69
Cefazolin	0.85	0.65	0.66
Ciprofloxacin	0.92	0.82	0.86
Gentamicin	0.97	0.52	0.78
Tobramycin	0.98	0.68	0.78
Trimethoprim-Sulfamethoxazole	0.94	0.80	0.92
Tetracycline	0.95	0.86	0.93
Cefoxitin	0.90	0.51	0.83
Levofloxacin	0.93	0.81	0.92

Table 5.2: performance Metrics

Confusion Matrix evaluation of DNN The model’s classification performance in predicting antibiotic resistance for different antibiotics is shown by the confusion matrices. Every matrix has the true values on the y-axis and predicted labels on the x-axis and therefore, the number of correct and incorrect classifications for susceptible values and resistant values. Blue shades become darker as the count of correct classifications increases. From figure [5.2], When evaluating the model at correctly classifying susceptible and resistant cases for antibiotics like gentamicin, tobramycin, ciprofloxacin, and trimethoprim-sulfamethoxazole, the model performs strongly with the high number of correctly classified susceptible and resistant cases and low misclassifications. Particularly, gentamicin and tobramycin have nearly perfect classification with very few false positives and false negatives. Ciprofloxacin achieves high accuracy as it misclassifies a small and constant number of samples in an all-time series. On the other hand, misclassifications are more noticeable when using antibiotics such as ampicillin and cefazolin. Ampicillin shows 48 false positives and 44 false negatives, showing the presence of some confusion between susceptible and resistant cases. Similarly, cefazolin has a lot of misclassified samples, specifically within the resistant category, where out of 34 resistant cases, 34 were misclassified as susceptible. The model is shown to achieve relatively strong classification performance for antibiotics such as tetracycline, cefoxitin and levofloxacin, however, there is still misclassification of cases into resistant, which shows promise from the classification potential in the data. Tetracycline’s moderate negative is only found in one case but a moderate number of positives. Among other antibiotics tested, cefoxitin has a more significant misclassification trend where 41 false positives and 6 false negatives are detected, which shows a slight bias towards susceptibility prediction. The overall confusion matrices suggest that the model is excellent at some antibiotics while there is variation in the classification accuracy of different drugs, with few antennas misclassifying some susceptible antibiotics as resistant. Errors are further detectable in these antibiotics, and future fine-tuning of the model may be necessary to reduce the misclassification rates of these antibiotics.

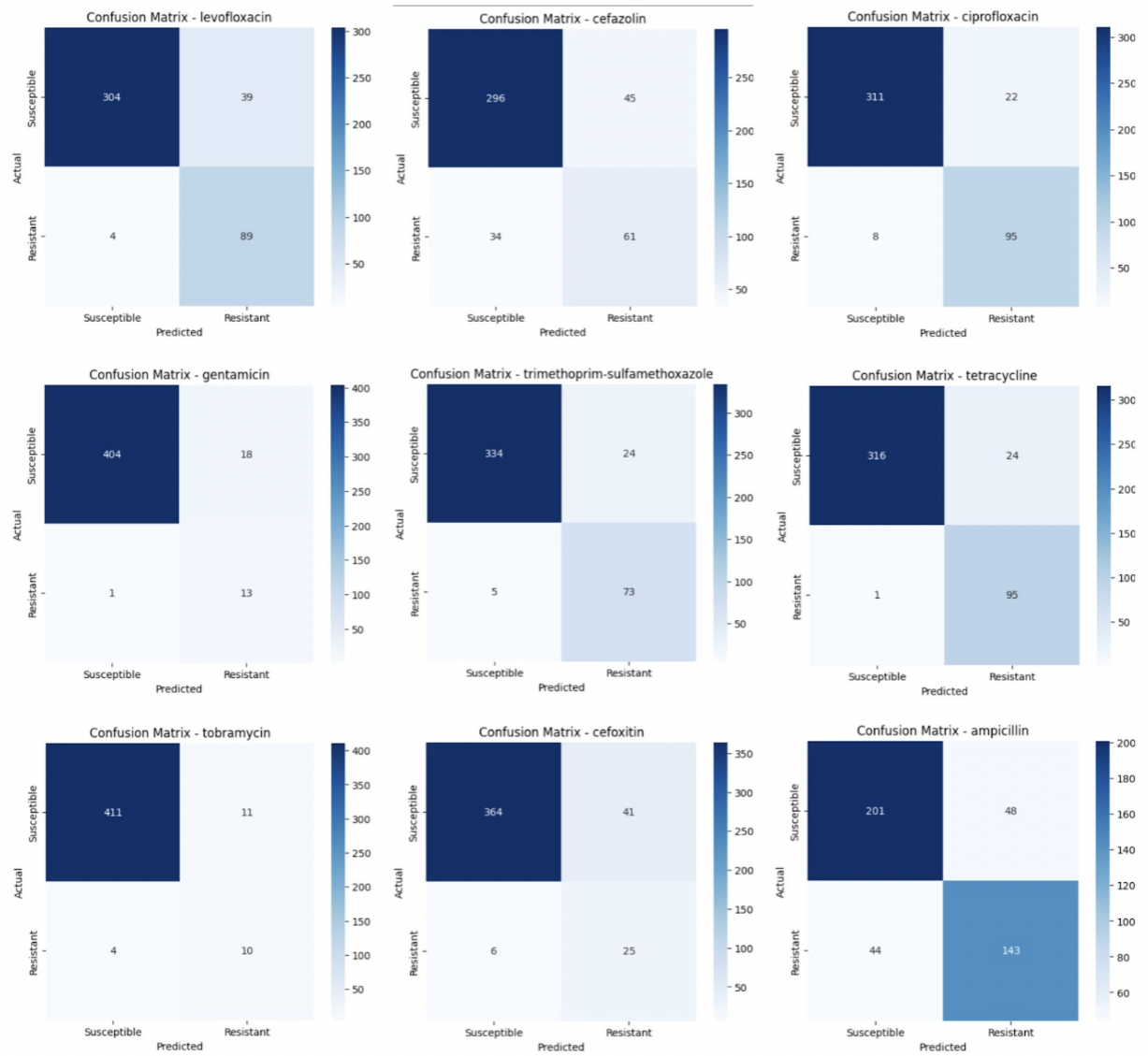


Figure 5.2: confusion matrix for DNN model Ecoli

Training and test data evaluation of DNN

Accuracies of training and validation of the given model for predicting antibiotic resistance across multiple antibiotics on inclusion of different features are shown in the graphs, as the model learns with different epochs. From figure [5.3] For each graph, one antibiotic was used and the x-axis is the number of epochs and the y-axis is accuracy. Training accuracy with the solid blue line and validation accuracy marked with the dashed red line can be compared and it can be easily visualized how the model trains on these data. The model can achieve consistently high accuracy for antibiotics like ciprofloxacin, gentamicin, tobramycin, and trimethoprim-sulfamethoxazole, with training and validation accuracy both over 95%, and very close to each other implying that the model generalizes well and overfit negligibly. On the other hand, while ampicillin and cefazolin show more fluctuation in the validation accuracy, there are also gaps between training and validation accuracies which might suggest variability and sensitivity in how well the model would predict. Training accuracy slowly rises to a plateau and validation accuracy follows a similar trajectory with a few oscillations i.e. curve starts increasing with some oscillations (but not much) before levelling off

over time. This leads me to believe that there is little amount of overfitting present in these antibiotics such as tetracycline and cefoxitin or some kind of generalization variability. Even with this, most models still tend to go up over time in terms of epochs to mean accuracy, indicating successful learning over time. All in all, the overall graphs make it clear that the model works well across distinct antibiotics, although some more work is needed to stabilize it and decrease variance.

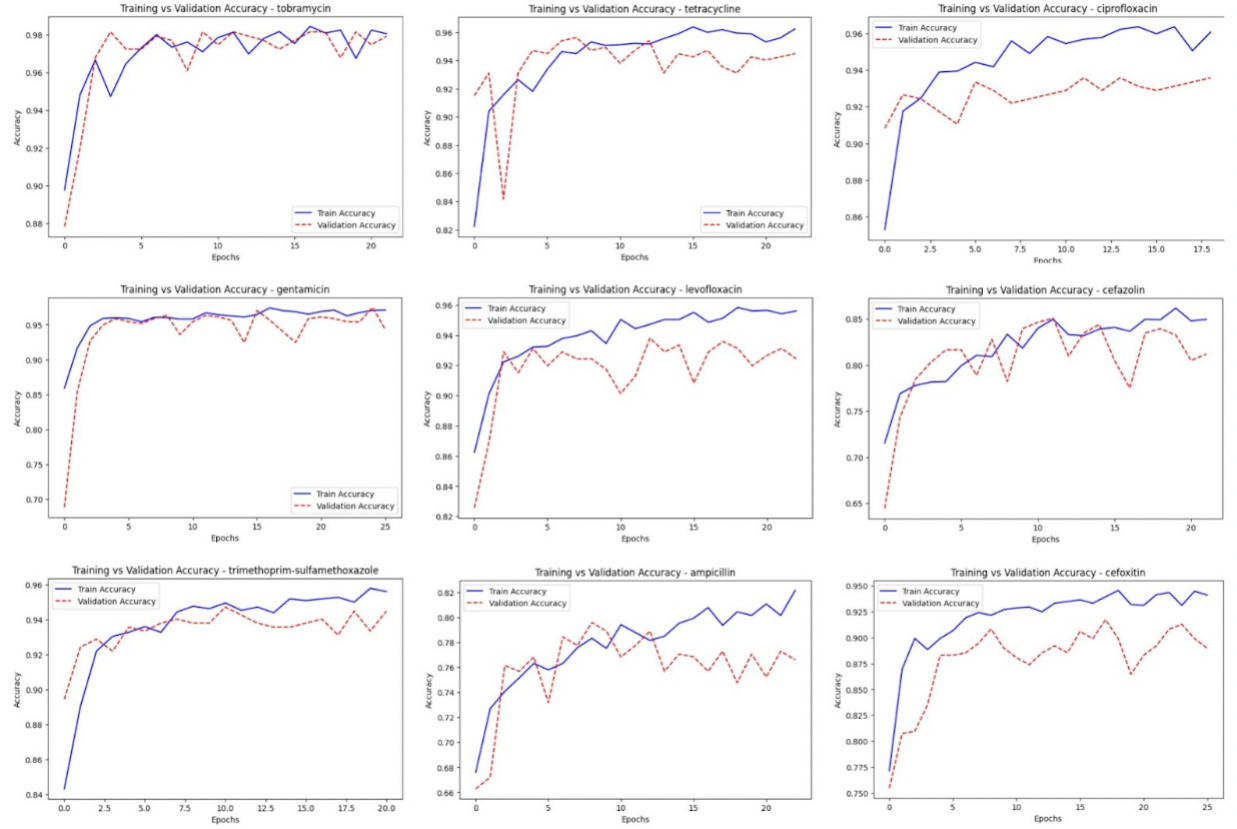


Figure 5.3: Train and Validation graph for DNN model Ecoli

5.1.3 CNN

Antibiotic	Accuracy	Precision	Recall
Ampicillin	0.79	0.75	0.76
Cefazolin	0.82	0.58	0.64
Ciprofloxacin	0.93	0.81	0.92
Gentamicin	0.96	0.42	0.93
Tobramycin	0.97	0.48	0.71
Trimethoprim-Sulfamethoxazole	0.93	0.75	0.94
Tetracycline	0.94	0.80	0.99
Cefoxitin	0.89	0.38	0.81
Levofloxacin	0.90	0.70	0.96

Table 5.3: Performance metrics

Confusion Matrix Evaluation of CNN

Different antibiotic confusion matrices show how the prediction model performs in antibiotic resistance classifications by depicting actual results on the y-axis and predicted results on the x-axis. The visualization shows dark blue sections to represent correct predictions while light blue sections signify incorrect predictions between actual and predicted values.

The evaluation model delivers robust performance when measuring ciprofloxacin, gentamicin, tobramycin, trimethoprim-sulfamethoxazole, tetracycline and levofloxacin by correctly identifying both resistant and susceptible bacterial strains from Table [5.3] ,. The model demonstrates exceptional reliability for antibiotic classification through its nearly perfect performance with Gentamicin and tobramycin because the numbers of false positives and false negatives remain very minimal. The classification results indicate that ampicillin and cefazolin lead to significant misclassification errors because the model identifies both false positive and false negative cases from figure [5.4]. The measurement of susceptible versus resistant shows that Ampicillin produces 35 false positive results and 57 incorrect negative outcomes thus indicating a possible imbalance in classification. The classification model shows a preference towards susceptible predictions for antibiotic cefazolin through its identification of 32 resistant cases that were wrongly marked as susceptible. The data suggests the model correctly identifies bacterial responses but produces a combination of 36 false positives along with 5 false negatives when testing cefoxitin. False classifications appear less frequently in tetracycline and levofloxacin antibiotic testing although some detection errors remain evident. Additional improvements to the model are necessary to minimize error results for ampicillin and cefazolin antibiotic classifications. The analytical indications show that the model delivers outstanding performance on antibiotics with discernible resistance patterns yet requires additional training examples or adjustments for those showing wide classification variations.

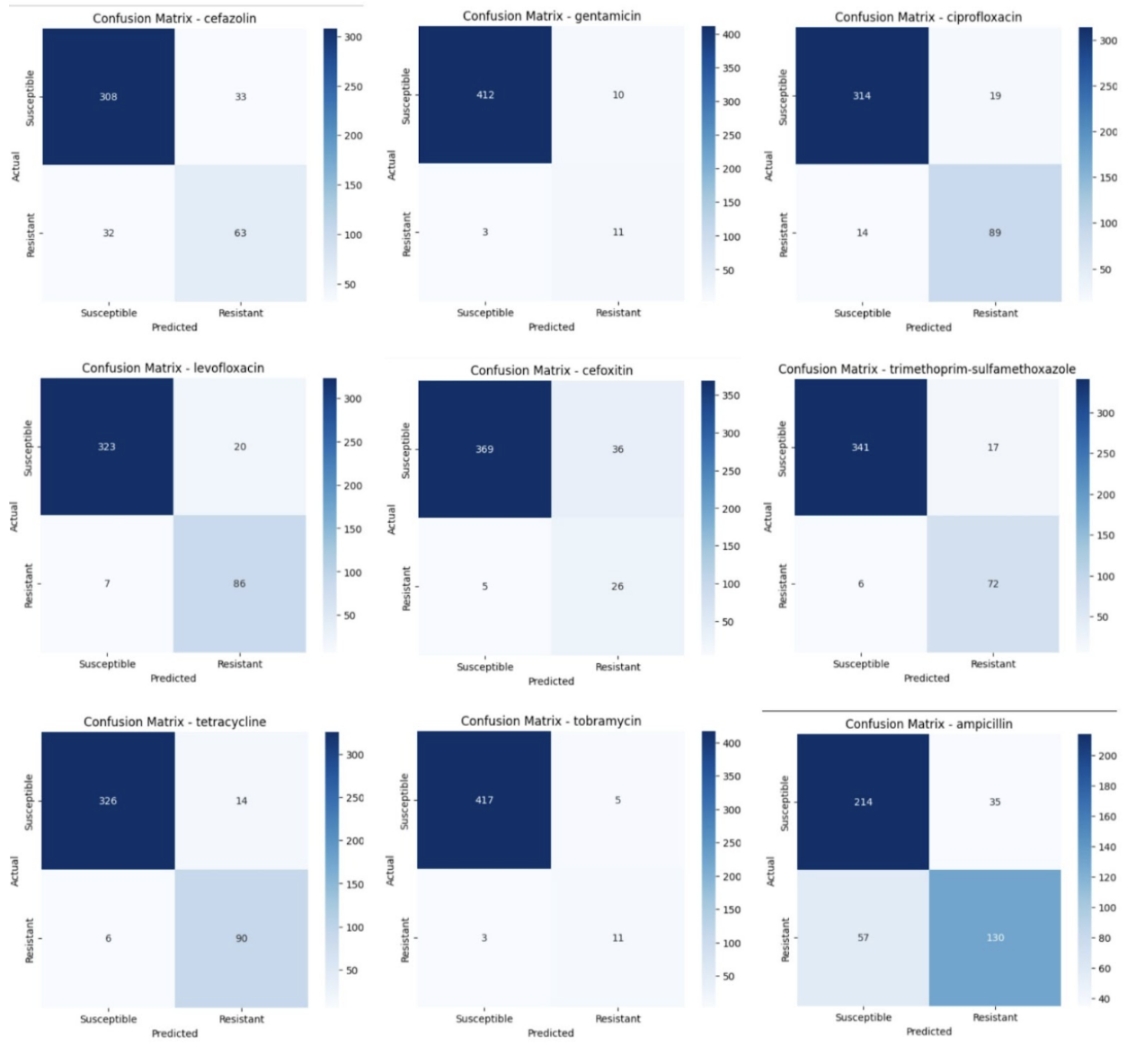


Figure 5.4: Confusion matrix for CNN model Ecoli

Training and test data evaluation

The graphs for learning performance of the model in the training vs. the validation accuracy of the model for different antibiotics having been trained for multiple epochs are shown. In this case, on the x-axis, we get a number of epochs, while on the y-axis, we get accuracy. Training accuracy is shown by the solid blue line, while the dashed red line is the validation accuracy. The model reaches high accuracy levels of over 95%, with the shortest divergence between training and validation accuracy, which implies generalisability for antibiotics such as ciprofloxacin, gentamicin, tobramycin and trimethoprim-sulfamethoxazole. For these antibiotics, the curves stabilize quickly to show good learning. On the contrary, the validation accuracy of ampicillin and cefazolin shows large fluctuations, signalling some instability of generalization. In particular, ampicillin shows an increase in the gap between training and validation accuracy in later epochs which might be due to overfitting. Furthermore, the accuracy of cefoxitin and levofloxacin in validation also has a degree of variation from training

accuracy, though, in the case of the former, it follows a slightly increasing trend in the gap. In the case of tetracycline, high accuracy is achieved, but the variance in validation accuracy is important, especially in the early epochs. But further training makes the trend stabilize. There is a sharp improvement and then some fluctuations in validation accuracy for Cefazolin that could be improved with more robust regularization. The model exhibits good overall performance for almost all antibiotics and possesses increasing accuracy trends. Although the fluctuation of validation accuracy for certain antibiotics may be caused by overfitting or differences in resistance classification, results indicate that further optimization or more training data may be necessary for stability.

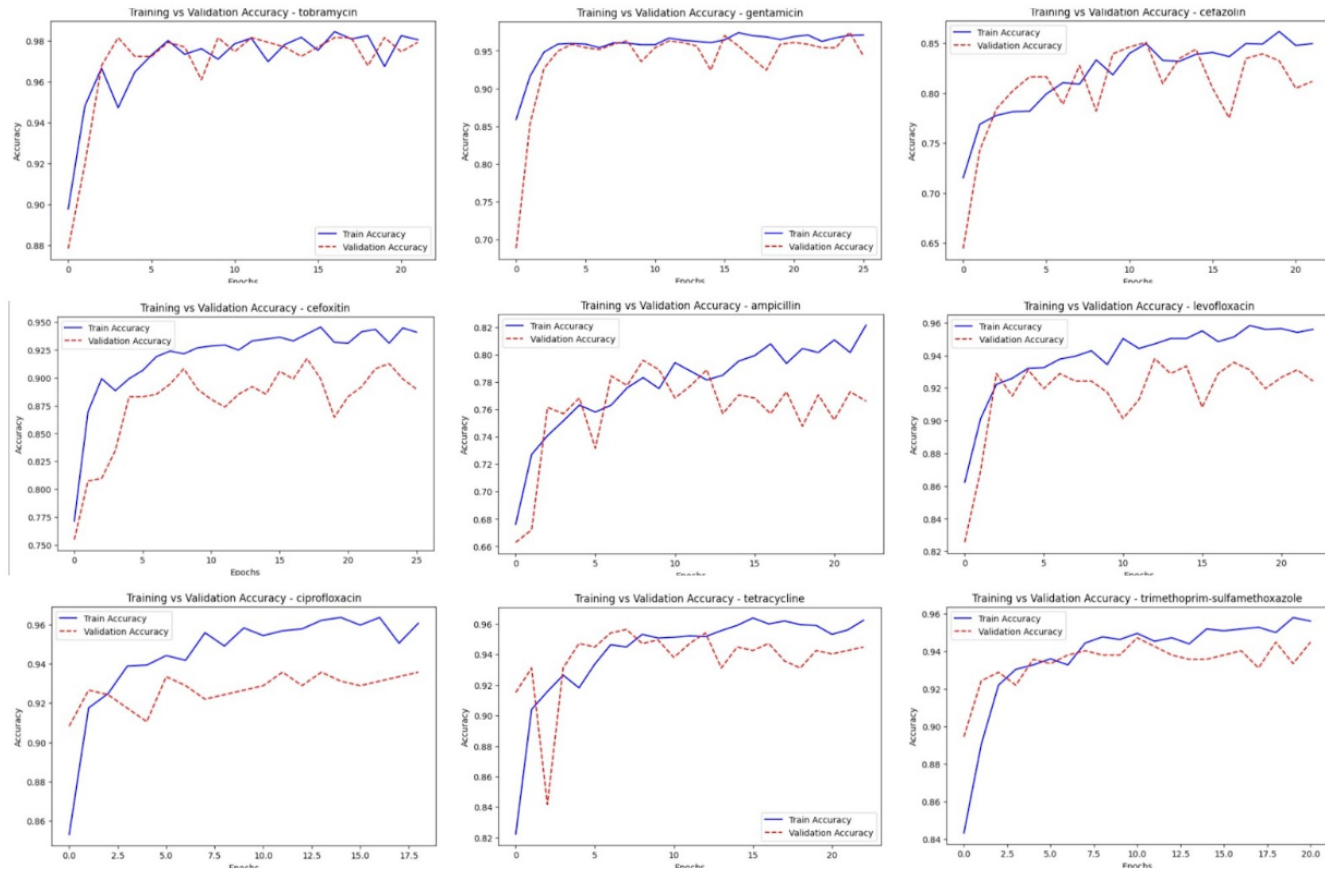


Figure 5.5: train and validation accuracy graph for CNN model Ecoli

5.1.4 Prototypical Network

Antibiotic	Accuracy	Precision	Recall	F1-Score
Ampicillin	0.77	0.77	0.76	0.76
Cefazolin	0.84	0.80	0.89	0.84
Ciprofloxacin	0.91	0.87	0.97	0.92
Gentamicin	0.97	0.95	1.00	0.97
Tobramycin	0.95	0.90	1.00	0.95
Trimethoprim-Sulfamethoxazole	0.86	0.81	0.94	0.87
Tetracycline	0.83	0.77	0.95	0.85
Cefoxitin	0.93	0.92	0.95	0.94
Levofloxacin	0.91	0.88	0.95	0.92

Table 5.4: Model Performance Metrics for Antibiotic Resistance

Cnnfusion Matrix Evaluation

These classification performances of the prototypical network model in predicting antibiotic resistance for *Escherichia coli* (*E. coli*) are represented by the confusion matrices for each case of predicting antibiotic resistance across tobramycin, ampicillin, cefazolin, ciprofloxacin, ceftriaxone, trimethoprim – sulfamethoxazole, tetracycline, cefoxitin, and levofloxacin. For each matrix, a breakdown is given of the true positives, true negatives, false positives and false negatives and they indicate the how accurate the model is classifying between resistant and susceptible strains. For the most matrices, strong diagonal dominance implies high classification accuracies with misclassification mostly limited to antibiotics cefoxitin and ciprofloxacin. Nevertheless, some antibiotics (e.g., ampicillin and tetracycline) show more false positives and false negatives, which indicates that the model has more difficulty with those classes. The performance variations of the different antibiotics depict the inherent complexity of resistance prediction and serve as an indication for further model optimization that could include feature selection or more training data.

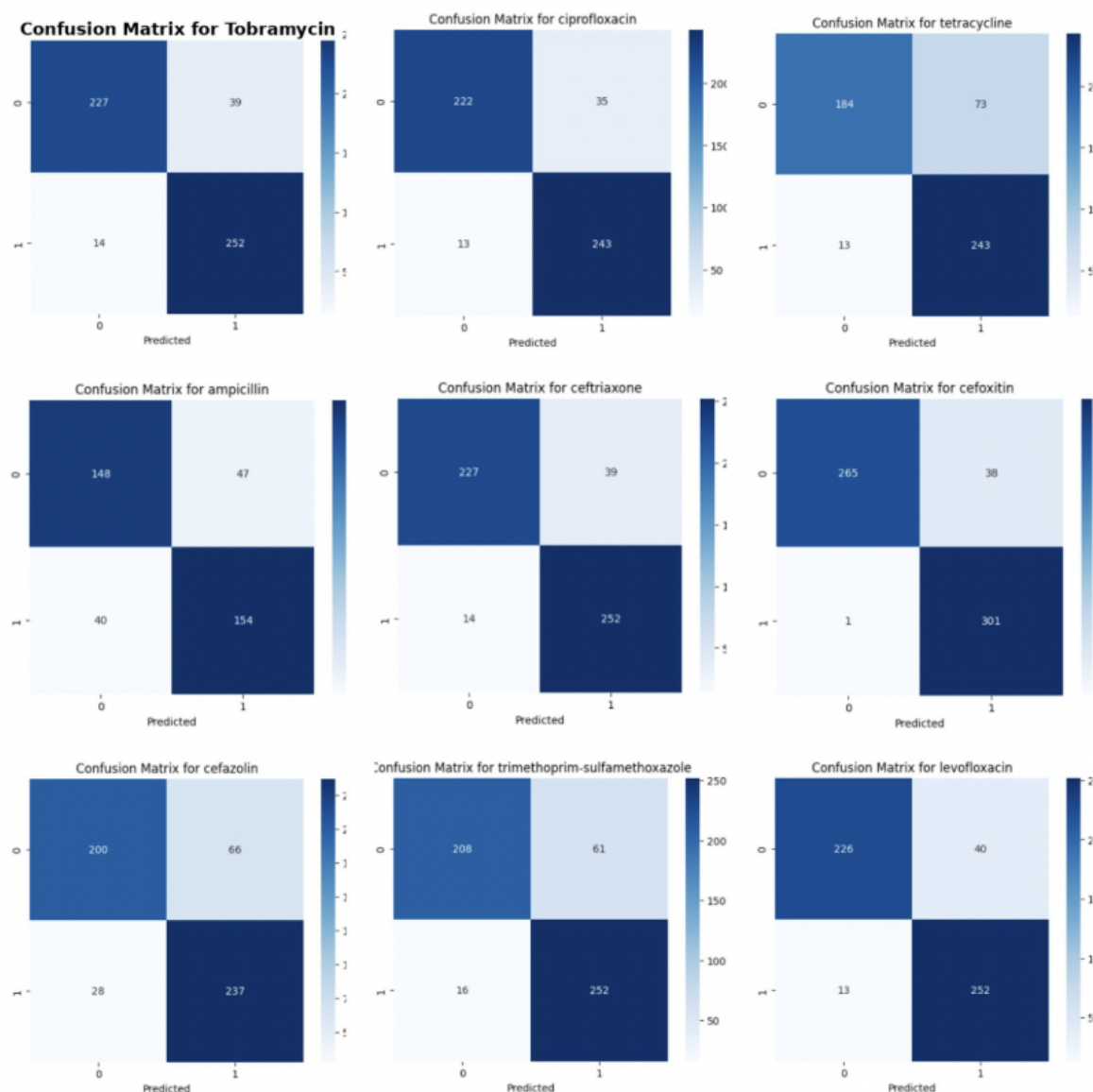


Figure 5.6: Confusion matrix for prototypical network model Ecoli

Train and test Data Evaluation

prototypical network model architected for antibiotic resistance prediction across

multiple antibiotics; tobramycin, ampicillin, cefazolin, ciprofloxacin, gentamicin, trimethoprim-sulfamethoxazole, tetracycline, cefoxitin and levofloxacin, trained and validated on *Escherichia coli* (*E. coli*) genome sequences are provided in the graphs as trends in training and validation accuracy. The accuracy is increased over 200 epochs in the each graph, where both training (blue solid line) & validation (red dashed line) curves tend to grow steadily. Contrary to previous models with rapid convergence, this model performs gradually gaining fairly good but not outstanding accuracy, somewhere in the range of 0.4–0.65. There is still room for optimization, as the gap between training and validation accuracy is consistent, and this may be done by tuning hyperparameters or by performing additional feature engineering. The model successfully distinguishes the difference between resistant and susceptible strains, although it needs more refinement to generalize and be more robust for other

antibiotics as well.

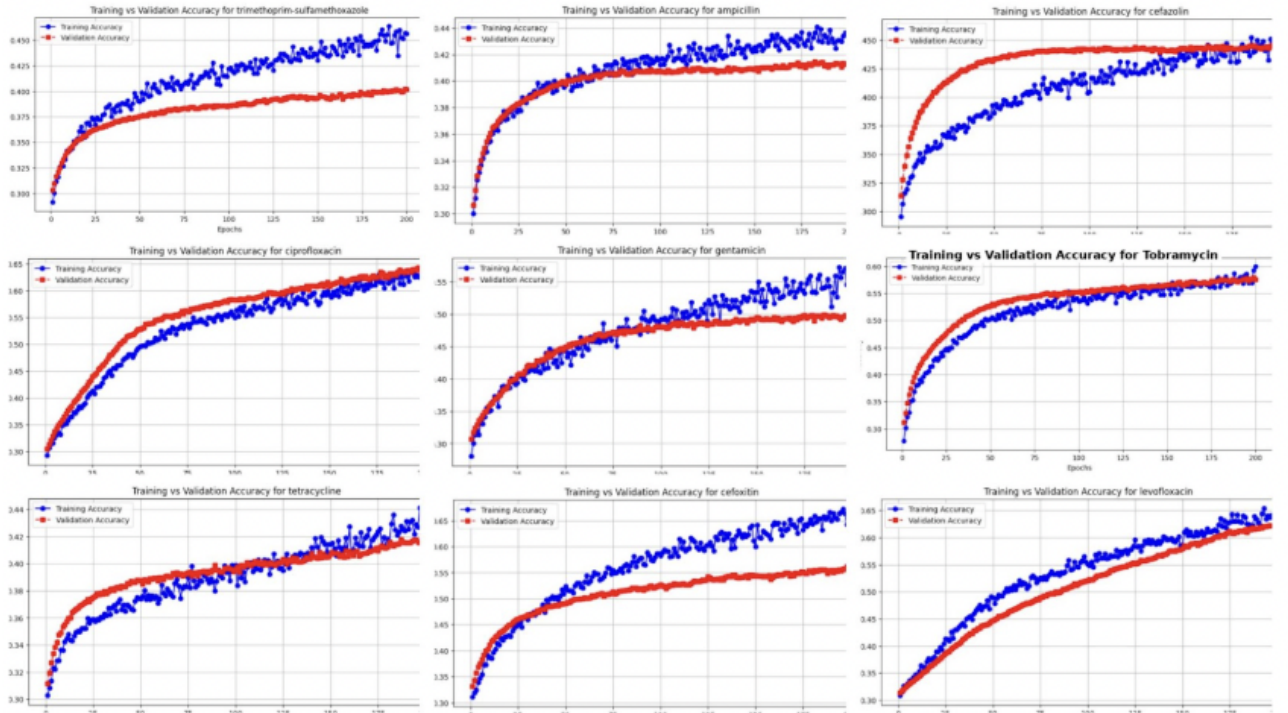


Figure 5.7: Train and validation accuracy graph for prototypical network model E-coli

5.1.5 Model comparison

The performance of prediction models for E. coli antibiotic resistance varies greatly among Deep Neural Network (DNN), Convolutional Neural Network (CNN), stacked models, and Few-Shot Learning (Prototypical Network). Their results reflect the diverse properties of input data and choice of feature extraction techniques. The Few-Shot Learning model delivers superior results specifically for detecting rare drug resistance by detecting all Gentamicin and Tobramycin resistances while giving no false positive results. FSL proved better at detecting rare antibiotic resistances because its low number of resistant examples did not limit its predictive power. Despite doing well in overall performance FSL incorrectly classified susceptible samples as resistant with its Tetracycline (23%) and Trimethoprim (19%) results whilst still maintaining decent sensitivity. With accurate judgment across various antibiotics the Stacked Model stood out as a solution to reduce incorrect antibiotic test results. Despite its success with other antibiotics the model evaluated fewer than one-fourth of true Gentamicin resistant bacteria correctly. Both XGBoost and LGBM models showed high precision because they detected susceptible strains most accurately at the cost of poor sensitivity towards rare resistant mutations in the testing samples. The CNN model demonstrated stronger ability than DNN by accurately detecting resistance mutations across a wide range of antibiotic types especially when identifying Tetracycline and Trimethoprim resistance. The CNN system produced more false results when detecting resistance to Gentamicin and

Cefepime because the method created alerts for small mutation changes that may not always indicate drug resistance. The DNN model effectively identified Ciprofloxacin (93.12%), Tobramycin (96.10%), and Tetracycline (94.72%) resistant bacteria because of its fully connected layers understanding resistance patterns from all mutations. DNN produced wrong results for Gentamicin and Ceftazidime due to its deep neural network layers becoming too specialized in reading susceptible samples. Ciprofloxacin Trimethoprim and Levofloxacin produced consistent outcomes which proves that the database defines all resistance mechanisms for these drugs well. When assessing resistance to Gentamicin and Cefepime the models demonstrated unpredictable performance because they achieved high recall on specific samples mixed with others where they failed detection due to similar genetic condition between resistant and susceptible strains.

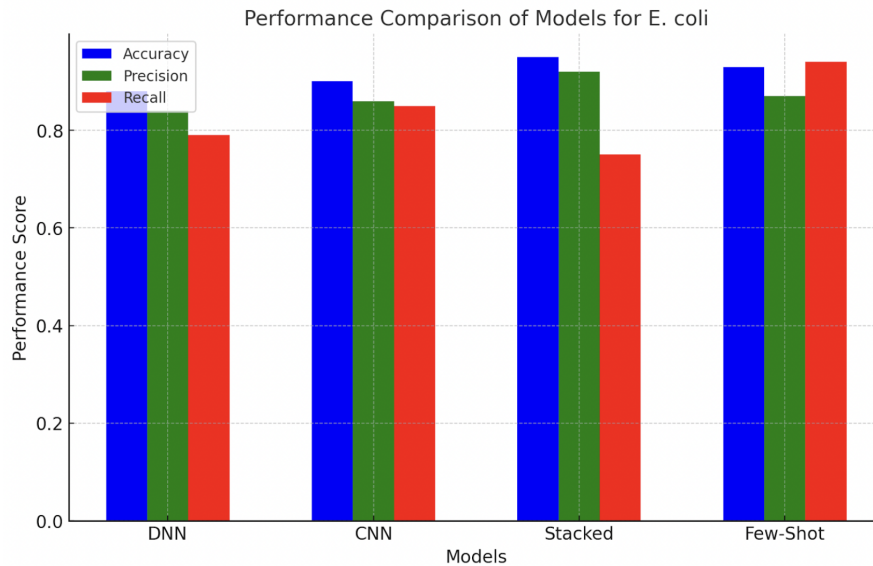


Figure 5.8: model comparison graph for E-coli

Ranking from best to worst:

1. Few-Shot Learning (Best for rare resistance cases, highest recall, minimal false negatives)
2. The two layer model has the highest accuracy rating when detecting antibiotic resistance yet misses significant amounts of data for particular antibiotics.
3. CNN (Best recall improvement, strong local feature extraction, but increased false positives)
4. DNN maintained a balanced focus but its problems with overfitting produced incorrect positive results.

The best algorithm for finding resistant strains is Few-Shot Learning due to its superior performance while Stacked Models offer superb precision over all cases.

5.2 Mycobacterium tuberculosis

5.2.1 CNN

Antibiotic	Accuracy	Precision	Recall
Amikacin	0.92	0.79	0.85
Capreomycin	0.88	0.75	0.78
Ethambutol	0.72	0.51	0.83
Isoniazid	0.73	0.79	0.74
Kanamycin	1.00	1.00	1.00
Ofloxacin	0.80	0.55	0.72
Rifampin	0.69	0.72	0.74
Streptomycin	0.77	0.73	0.78

Table 5.5: performance Metrics

Cnnfusion Matrix Evaluation

These confusion matrices give a glimpse into the way the CNN model classifies

antibiotic resistance for a given set of antibiotics. Among the drugs, amikacin and capreomycin seem to be the best in classifying, yielding relatively lower false negatives and false positives suggesting that the model classifies susceptible and drug-resistant cases accurately. While the model misclassifies ethambutol and rifampin, the false positives are high (misclassifying resistant as susceptible) showing that the model cannot distinguish between resistant and susceptible strains well. As observed from its confusion matrix, there is a big imbalance in Isoniazid, as a lot of resistant cases are misclassified as susceptible indicating that the model may not be grasping features important to differentiate. Kanamycin has no misclassifications (zero) and thus, either the feature representation is exceptionally well learned, or the dataset is too simple to classify. False positives and false negatives also appear frequently in ofloxacin and streptomycin which perform moderately. While there is overall good performance of the CNN model for some antibiotics, the results are still inconsistent for similarly composed antibiotics with high-class imbalance or similar feature representations. Addressing these challenges using data augmentation, class balancing techniques, hyperparameter tuning, and better feature extraction algorithms can improve the overall classification accuracy and reliability.

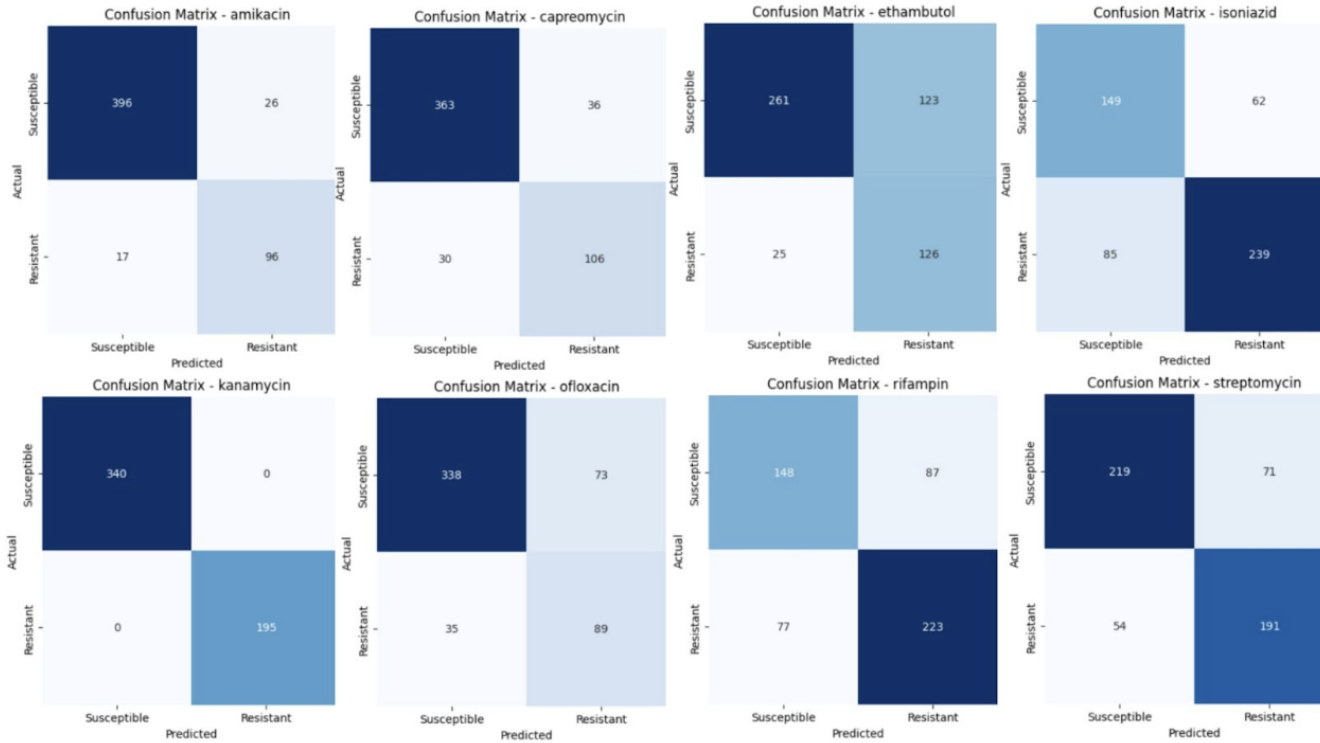


Figure 5.9: Confusion matrix for CNN model MT

Train and test Data Evaluation

These are the CNN training and validation accuracy graphs showing the model

learning of different antibiotics to show their efficacy and what they lack. The training and validation accuracy both increase rapidly in most models at the beginning, which indicates an efficient feature extraction process. Early on, Kanamycin achieves near-perfect accuracy which may indicate overfitting that would need regularization techniques such as dropout or weight decay. The accuracy curve shapes display well-behaved smooth and stable accuracy curves for amikacin, capreomycin and ofloxacin, having good generalization performance, however, there are slight oscillations in validation accuracy with a 15% variation in the data which shows sensitivity to the data. In this case, both streptomycin and ethambutol have initial instability, and ethambutol has a very sharp early dip in validation accuracy, which may be from batch effects or model initialization. Accuracy of Isoniazid and Rifampin are relatively lower with a low balance between validation and training accuracies ranging to underfitting model or insufficient feature learning. Most models reach a point of stabilization in the validation accuracy, but validation and training curves are diverging in different degrees, which implies room for improvement. In some cases, such techniques as early stopping, reverting to hyperparameter tuning, data augmentation, and balance class distributions may help handle the model performance, and generalization, and avoid overfitting.

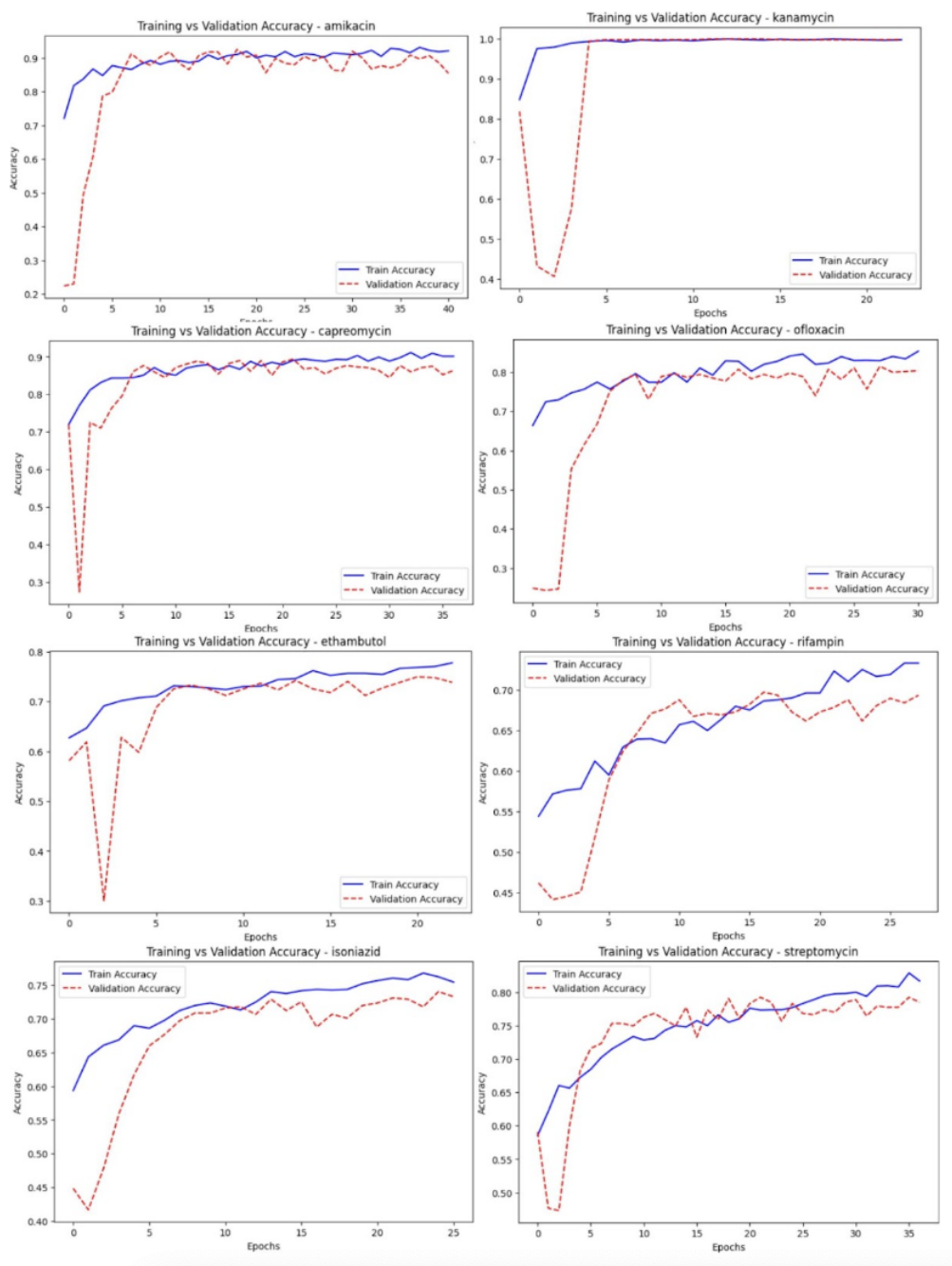


Figure 5.10: Train and validation accuracy for CNN model MT

5.2.2 Stack Ensemble Model

confusion Matrix analysis

These stacked confusion matrices are the prediction performance of an ensemble model based on antibiotic resistance prediction: we use an ensemble of multiple classifiers to predict more accurately and have better results. Of these two compounds, amikacin and capreomycin have a high true positive rate and barely any false negatives, however, capreomycin has a slightly higher false negative rate. In addition, their classification errors are more noticeable, especially high false positives, meaning that resistance cannot be readily identified. The confusion matrix of Isoniazid shows

Antibiotic	Accuracy	Precision	Recall
Amikacin	0.92	0.79	0.85
Capreomycin	0.88	0.75	0.78
Ethambutol	0.72	0.51	0.83
Isoniazid	0.73	0.79	0.74
Kanamycin	1.00	1.00	1.00
Ofloxacin	0.80	0.55	0.72
Rifampin	0.69	0.72	0.74
Streptomycin	0.77	0.73	0.78

Table 5.6: performance Metrics

moderate imbalance in the form of the significant number of resistant cases being misclassified as susceptible, which may affect sensitivity. No errors arise in the classification of kanamycin, and it is excellently classified, implying that in the feature space, resistance from susceptibility to kanamycin is well separated. The performance of the classifiers of ofloxacin and streptomycin is moderate and their false positive and false negative classification reveal opportunity for refinement. While the ensemble approach leads to increased stability in all the models and also increases the level of classification accuracy in most antibiotics, some cases still need to be handled well regarding misclassifications, which might be addressed through hyperparameter tuning, feature engineering or the use of further training data to boost generalization.

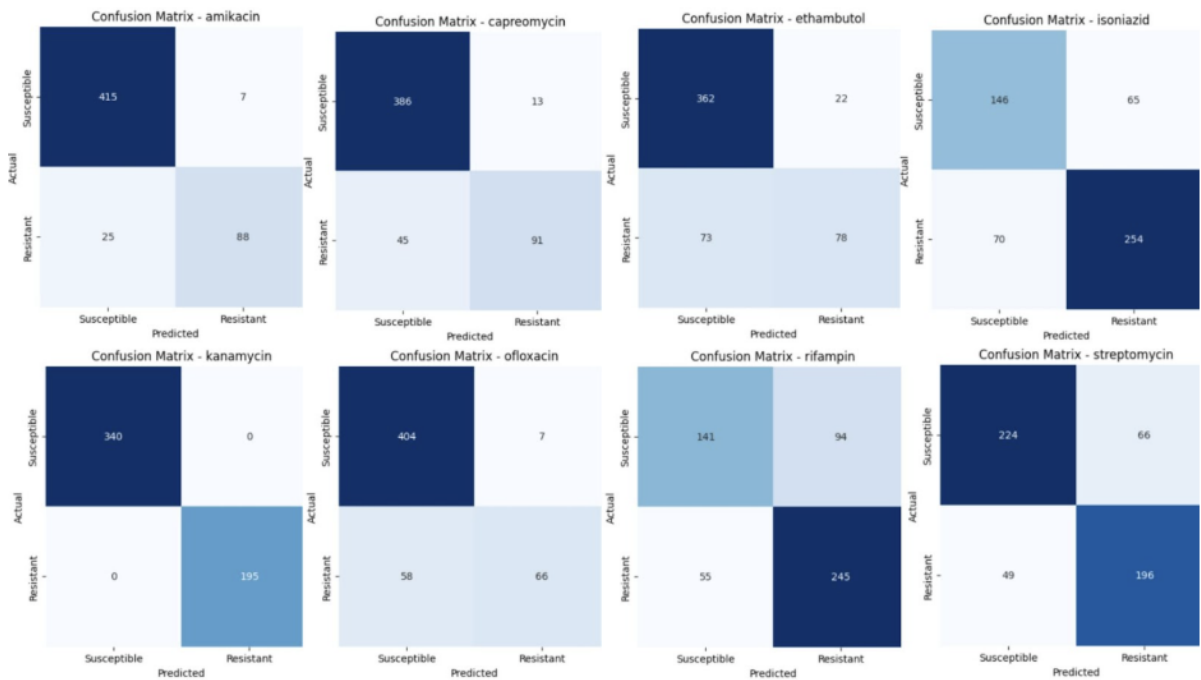


Figure 5.11: Confusion matrix for Stack Ensemble model MT

5.2.3 DNN

Antibiotic	Accuracy	Precision	Recall
Amikacin	0.93	0.88	0.81
Capreomycin	0.90	0.89	0.71
Ethambutol	0.79	0.72	0.42
Isoniazid	0.72	0.76	0.80
Kanamycin	1.00	1.00	1.00
Ofloxacin	0.85	0.73	0.60
Rifampin	0.71	0.70	0.85
Streptomycin	0.77	0.74	0.77

Table 5.7: performance Metrics

confusion Matrix analysis

The following are these confusion matrices and show the model performance for

different antibiotics, with true positive (TP), true negative (TN), false positive (FP), and false negative (FN) values. The model is well performed for amikacin, as it has high true positives (409 susceptible and 91 resistant) and low misclassification. Like capreomycin, it is a strong performer, with slightly higher false negatives (39 patients resistant treated as susceptible). While ethambutol has a higher rate of false negatives (87), this could potentially make it difficult to differentiate resistant cases. However, Isoniazid suffers from its overweight, misclassifying 80 susceptible cases as resistant and 65 resistant cases as susceptible, thus requiring further tuning. Classification of kanamycin is perfect with no misclassifications which might suggest overfitting. But ofloxacin misclassified a bit (50 cases are false); overall, it is okay. Precision may be affected by rifampin as it has (relatively) high false positives (138 susceptible cases labelled resistant). Streptomycin has 66 false positives and 56 false negatives, which makes it maintain a fair balance but which room for improvement. Although the overall model shows promising results, there are some techniques like class weighting, threshold tuning or feature selection that can reduce misclassification especially where there are high false negatives or imbalanced predictions.

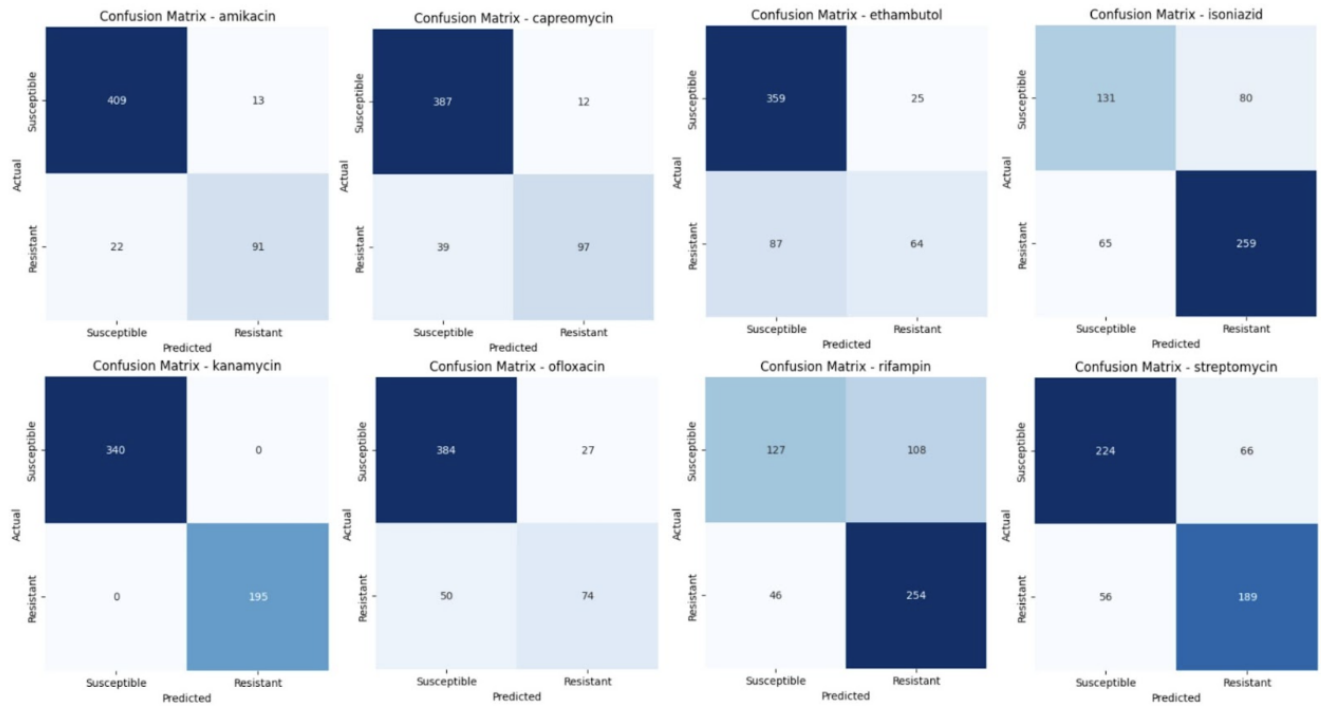


Figure 5.12: confusion matrix for DNN model MT

Train and test validation result

These figures are presented as DNN graphs where we show the training and validation accuracy trends for different antibiotics for an antibiotic resistance prediction model. While the rate of improvement of the training accuracy differs, it always improves. Validation accuracy decreases and in many cases, it remains higher but very volatile due to potential instability or variable data. Rifampin and isoniazid have lower overall accuracy and, although their combination prevents overfitting altogether, they still require richer optimizations to improve score accuracy and prevent underfitting. The validation accuracy of ethambutol and streptomycin shows sudden fluctuation may be sensitive to data distribution or insufficient regularization. Index validation accuracy surpasses training for ofloxacin and capreomycin but experiences the least stable trends. While many models can generalize well, there are some for which a lower loss would justify tuning hyperparams, increasing the amount of data seen during training (whether in training or on the held-out data) or applying regularization techniques in the pursuit of robustness. Overall the models perform well, however further refinement is required to predict antibiotic resistance stability, reduce overfitting and improve generalization

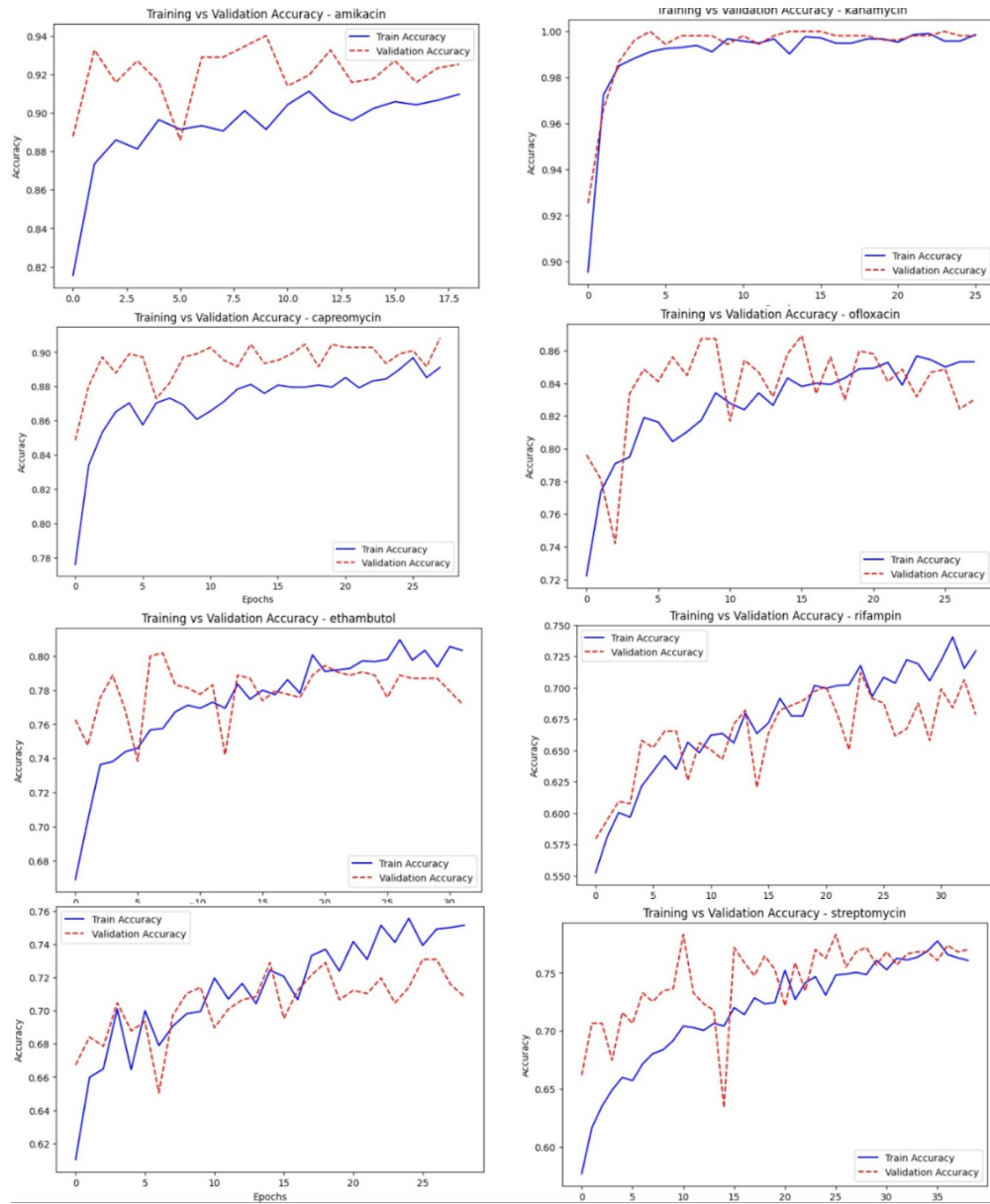


Figure 5.13: Train and validation accuracy graph for DNN model MT

5.2.4 Prototypical Network

Table 5.8: Model Performance Metrics for Antibiotic Resistance Prediction

Antibiotic	Accuracy	Precision	Recall	F1-Score
Amikacin	0.89	0.83	0.59	0.69
Capreomycin	0.86	0.78	0.70	0.65
Ethambutol	0.81	0.85	0.40	0.55
Isoniazid	0.70	0.74	0.75	0.75
Kanamycin	0.78	0.70	0.65	0.67
Ofloxacin	0.81	0.62	0.69	0.70
Rifampin	0.71	0.71	0.79	0.75
Streptomycin	0.74	0.72	0.77	0.74

confusion Matrix analysis

The M.T. prototypical network matrix of *Mycobacterium tuberculosis* (M.T.) is a

matrix of performance of an antibiotic resistance prediction model against a given set of drugs used to treat tuberculosis. The confusion matrix for each case quantifies the model's ability to predict if a strain is resistant or susceptible showing in turn true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Given the prediction strength, where the colour gradient is more intense, it indicates higher values and darker shades. For certain antibiotics, such as amikacin and capreomycin, the model performs with high specificity; these cases are correctly detected as susceptible with few false positives. In the case of antibiotics such as isoniazid and rifampin, the model also predicts higher false positive rates, risking the treatment of the patient as resistant unnecessarily. Other antibiotics such as streptomycin and kanamycin have a more balanced classification performance. Through its analysis, it can refine predictive algorithms for better detection of patterns of misclassification of MDR-TB cases, which in turn could increase the value of antimicrobial resistance surveillance and optimize treatment strategies for MDR-TB cases.

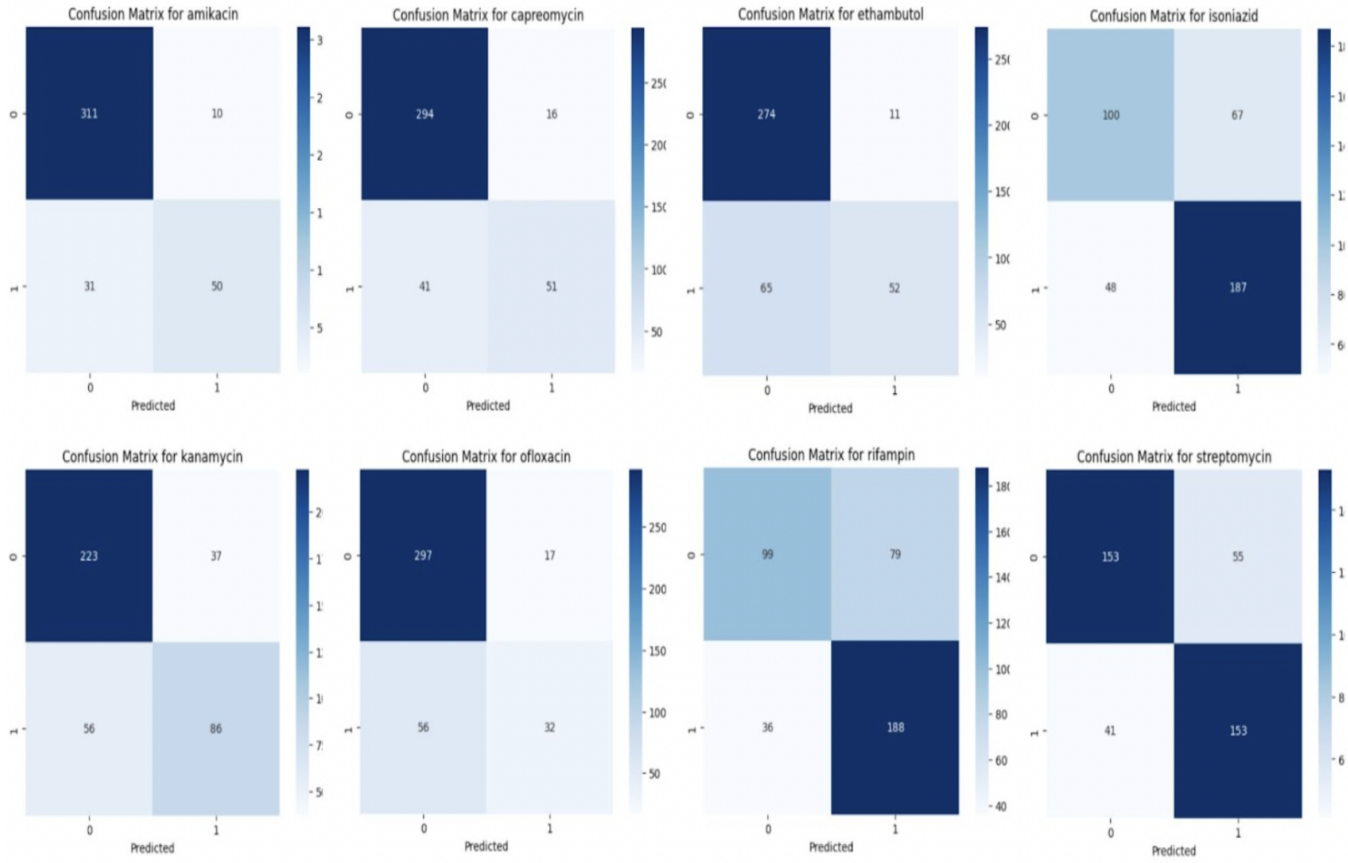


Figure 5.14: confusion matrix for prototypical network model MT

Train and test validation result

In this work, graphs are depicted of the training and validation accuracy of a prototypical network model predicting antibiotic resistance in *Mycobacterium tuberculosis* for the eight antibiotics: amikacin, capreomycin, ethambutol, isoniazid, kanamycin, ofloxacin, rifampin, streptomycin. In each graph, we see how the model reaches almost 1.0 in accuracy very quickly, and within the first few epochs ends up pretty much stabilizing. Comparison of the classification performance, as measured by the accuracy, using the training (blue solid) and validation (red dashed) accuracy curves, demonstrates that generalization on antibiotics that were not used for training is strong, and there is minimal overfitting. The likely cause of the initial negative values is probably an issue with a log scaling or initialization and the negative values are quickly corrected as the model learns distinguishing features. This suggests that the prototypical network is a typical approach for classifying resistant and susceptible strains and that this consistency across the network of all antibiotics makes that network a reliable method for the prediction of antibiotic resistance.

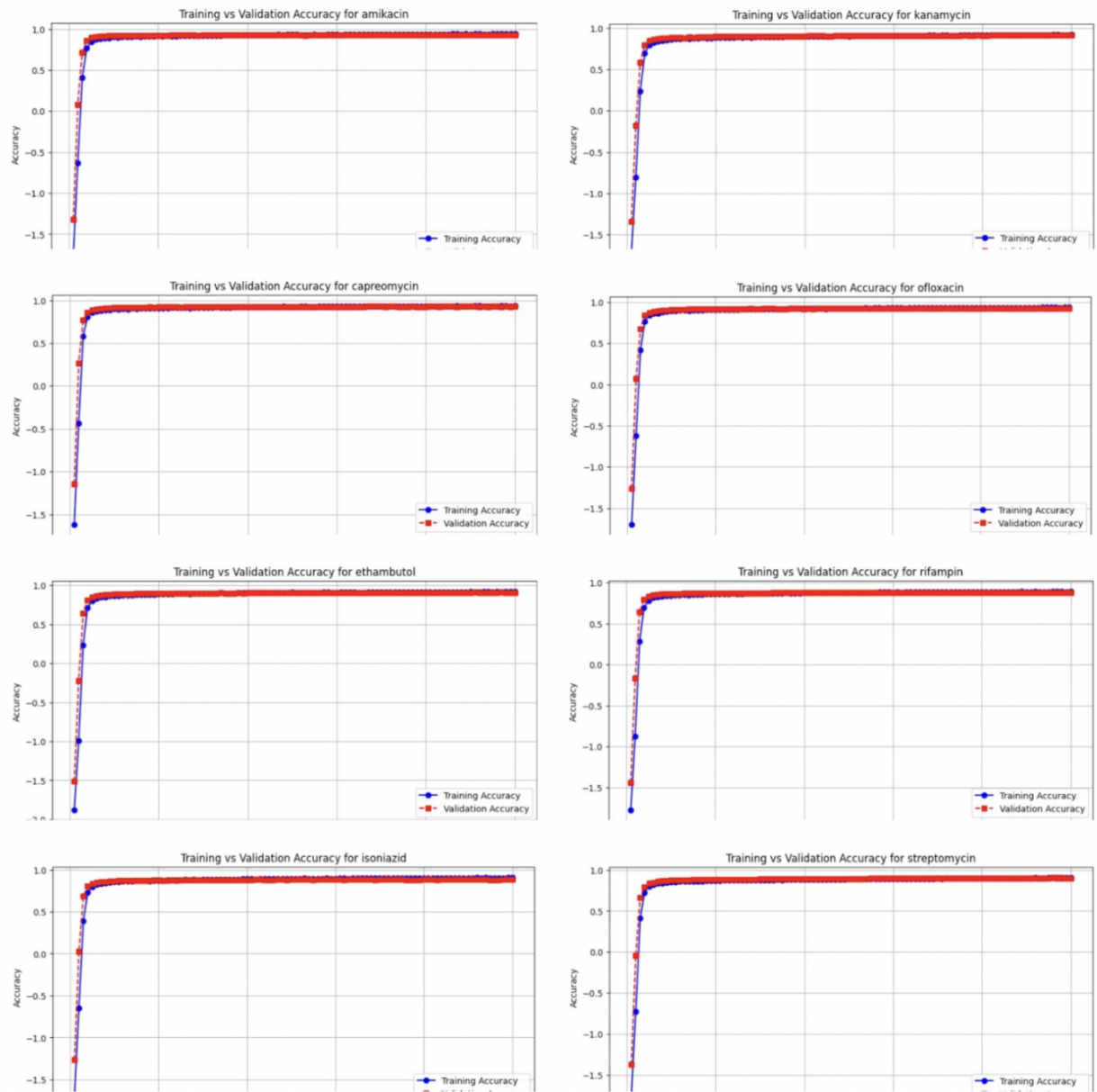


Figure 5.15: Train and validation accuracy graph for Prototypical Network model MT

5.2.5 Model comparison

The featured Deep Neural Network DNN and Convolutional Neural Network CNN and other models produced different results in testing because each handled datasets with varying levels of complexity based on their internal features and designs. DNN showed strong results in predicting Amikacin (93.45%), Capreomycin (90.46%), and Ofloxacin (85.60%) resistance due to its full connection layers that identified both global and local patterns. The DNN showed poor generalization skills since it memorized training data in both Rifampin and Streptomycin because the classes were misbalanced and the choices of antibiotics lacked diversity. The CNN showed enhancement in identifying Ethambutol (83%) and Rifampin (81.67%) resistance by finding specific patterns through its local feature extraction approach. Through Ethambutol and Rifampin testing CNN produced false results since it focused heavily on close-up nucleotide arrangements and identified small genetic variations as resistance signals resulting in resistant bacteria being mistaken for sensitive strains. Ensemble learning through the Stacked Model achieved higher accuracy levels than both DNN and CNN in predicting resistance in Amikacin (94.01%), Ofloxacin (87.85%), and Ethambutol (82.24%). Its performance improved by harnessing multiple algorithms (LGBM, XGBoost, Random Forest, Logistic Regression) to identify various decision boundaries. The model showed great results because XGBoost and LGBM tree classifiers took advantage of mutation frequency data structure which helped eliminate unwanted features before the training process. Ensemble models proved to be stable overall yet needed better performance for Capreomycin and Rifampin because their accuracy depended too heavily on frequent patterns. FSL achieved the highest accurate rare resistance detection results with 75% accuracy for Isoniazid followed by 79% success for Rifampin and 77% effectiveness for Streptomycin. FSL proved effective in antibiotic resistance prediction because it adapted well with small amounts of sample data especially when working with Isoniazid which had fewer resistant cases. FSL demonstrated weak accuracy in detecting Amikacin (59%) and Capreomycin (70%) resistances because its few-shot learning did not handle genetic diversity well even though it had a small dataset. Different model types demonstrated superior performance thanks to how the dataset combined mutation types with antibiotic resistance genes and genomic sequences.

The Stacked Model proved its superiority because it provided accurate results while understanding multiple antibiotics effectively. The DNN model earned good results but developed problems with overfitting during specific situations. CNN did better in finding antibiotic medication variations based on local areas but had more wrong predictions. The Few-Shot Learning model succeeded in rare resistance detection but failed with antibiotics that needed deeper feature analysis so it ranked as the worst performer.

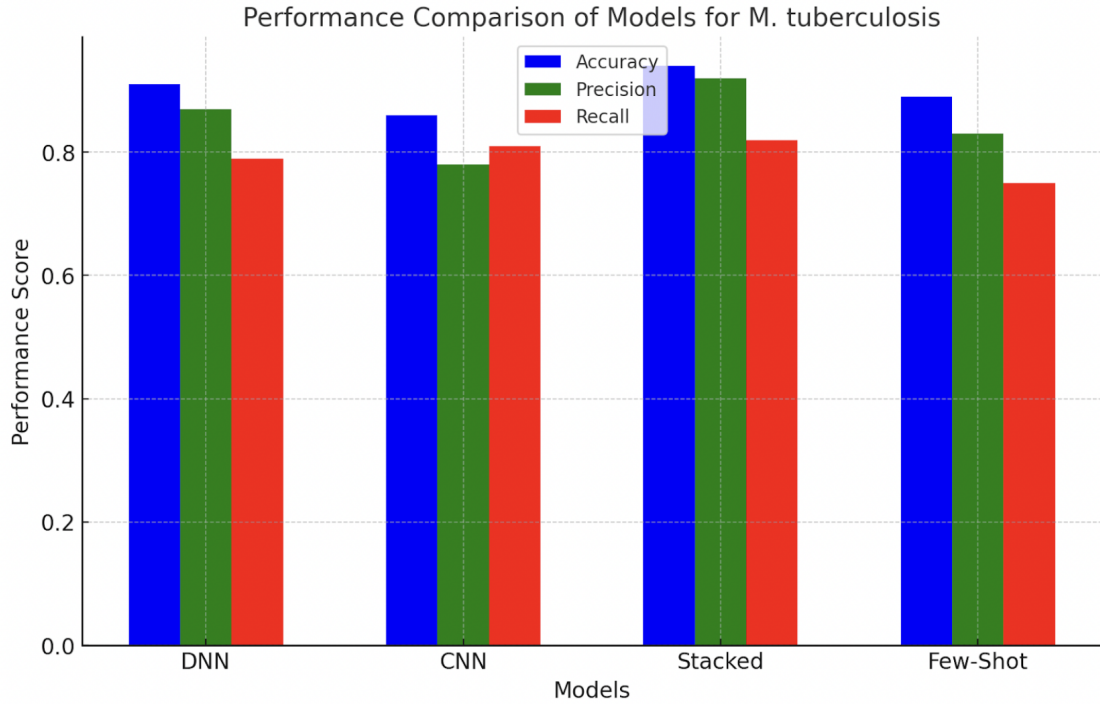


Figure 5.16: modelcomparison graph for MT

5.3 Salmonella Enterica

5.3.1 CNN

Antibiotic	Accuracy	Precision	Recall
Ampicillin	0.93	0.79	0.88
Cefotaxime	0.94	0.56	0.90
Chloramphenicol	0.99	0.89	0.97
Ciprofloxacin	0.99	0.91	0.95
Gentamicin	0.96	0.59	0.81
Streptomycin	0.93	0.72	0.63
Tetracycline	0.92	0.78	0.69

Table 5.9: performance Metrics

confusion Matrix analysis

For the Salmonella enterica, the confusion matrices of the CNN model in predicting

the antibiotic resistance in predicting antibiotic resistance to different antibiotics (ampicillin, cefotaxime, chloramphenicol, ciprofloxacin, gentamicin, streptomycin, and tetracycline), reveals good prediction performance of both true positive and true negative counts. Indeed, the model performs very well for ciprofloxacin and chloramphenicol, in which case only a few of the false classifications were introduced, achieving near-perfect classification accuracy. The model correctly classifies most susceptible and resistant cases for other antibiotics, such as ampicillin, cefotaxime and tetracycline, but has slightly more false negatives, some resistant cases are

misclassified as susceptible. Gentamicin and streptomycin also have a more favourable classification but have higher false positives, implying a little susceptible case is misclassified as resistant. The CNN model can distinguish antibiotic resistance patterns well in general, however, class balance refinement, feature refinement or tuning of the threshold can further improve the accuracy of prediction, especially for antibiotics with higher misclassification rates.

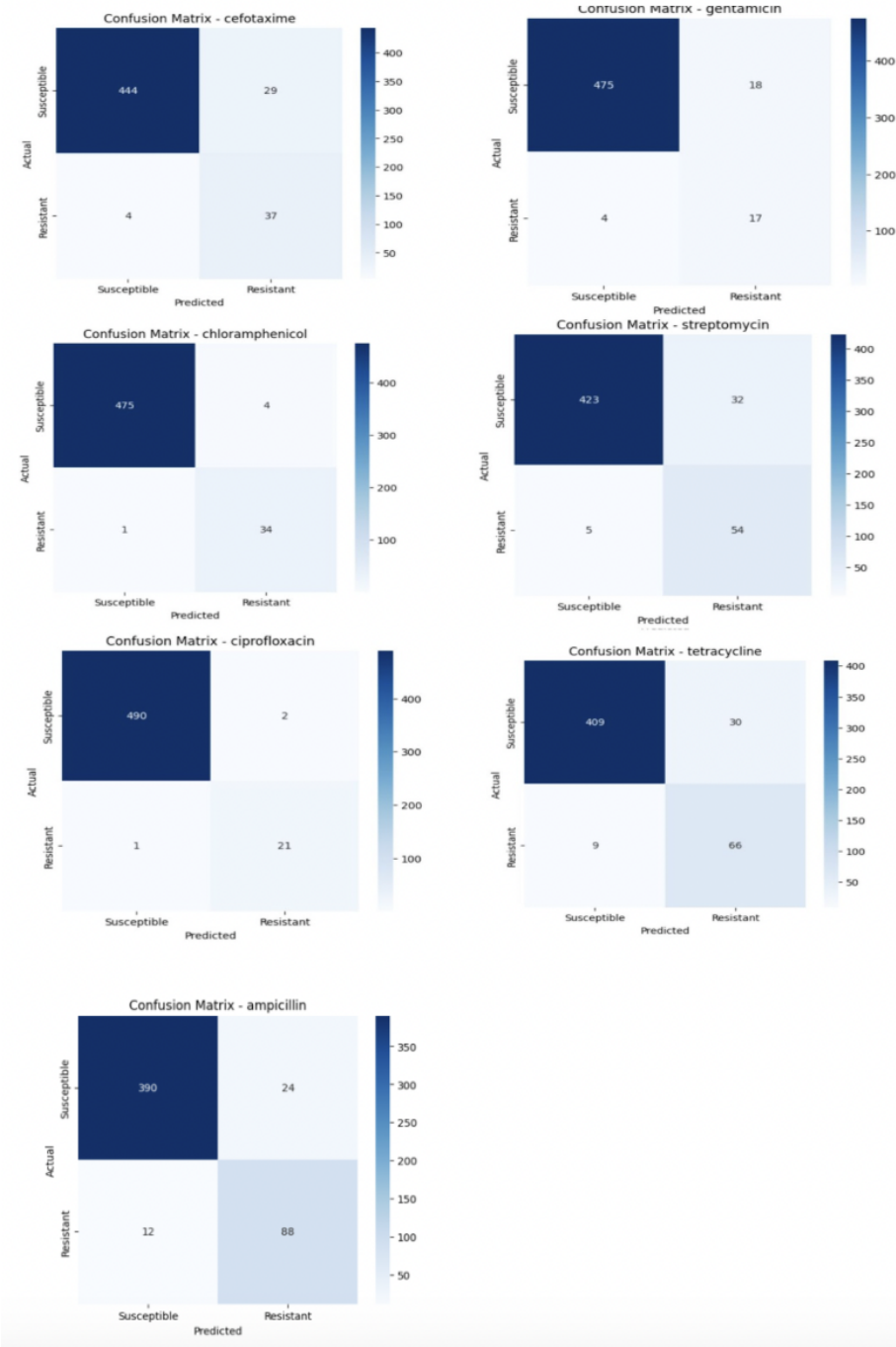


Figure 5.17: Confusion Matrix for CNN model salmonella Enterica

Train and test validation result

Salmonella enterica antibiotic resistance prediction to multiple antibiotics (ampicillin, cefotaxime, chloramphenicol, ciprofloxacin, gentamicin, streptomycin, and tetracycline) through the training and validation accuracy graphs of the used CNN model demonstrates the learning of the model with the rapid improvement at the early stage and the plateauing at the end. Ciprofloxacin and chloramphenicol, achieve near-perfect accuracy, signifying a good for feature extraction, whereas ampicillin, cefotaxime, and tetracycline display more initial fluctuations and are followed by high accuracy. A steadier learning curve of Streptomycin and Gentamicin suggests that the classification task might slightly be more complex. CNNs seem more suitable for that task, especially in antibiotics where spatial genomic patterns are important, compared to ‘standard’ deep neural networks. While some cases seem to have minor indications of overfitting, it’s possible that with some more regularization (e.g., dropout or data augmentation) generalization can be slightly improved. In general, the CNN model is able to predict antimicrobial resistance very well and therefore is a promising tool for antimicrobial resistance classification.

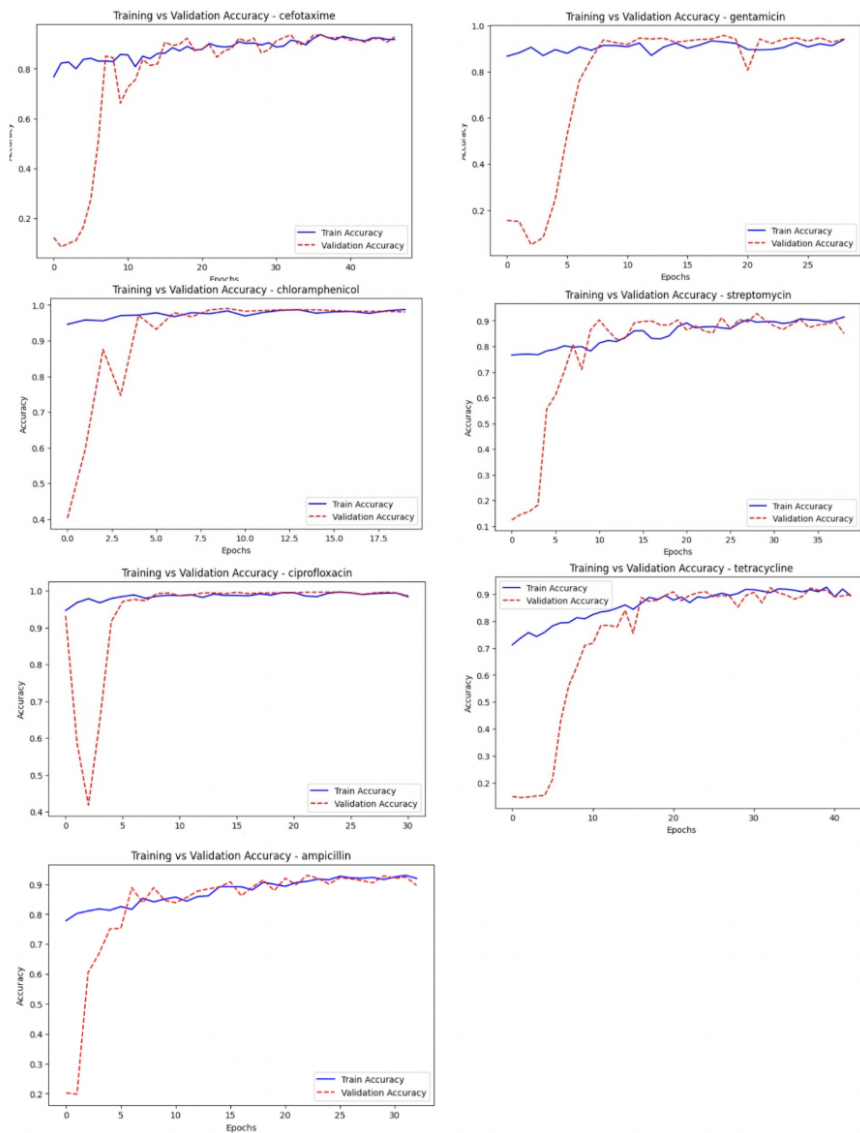


Figure 5.18: Train and validation accuracy graph for CNN model salmonella Enterica

5.3.2 DNN

Antibiotic	Accuracy	Precision	Recall
Ampicillin	0.93	0.86	0.78
Cefotaxime	0.95	0.65	0.85
Chloramphenicol	0.93	0.89	0.97
Ciprofloxacin	0.92	0.90	0.85
Gentamicin	0.96	0.50	0.81
Streptomycin	0.93	0.64	0.80
Tetracycline	0.93	0.73	0.79

Table 5.10: performance Metrics

confusion Matrix analysis

Confusion matrices of a DNN model for the classification of ampicillin, cefotaxime, chloramphenicol, ciprofloxacin, gentamicin, streptomycin, and tetracycline resistance for *Salmonella enterica* show antibiotic resistance prediction performance of a DNN. There are as many confusion matrices as there are true positives, true negatives, false positives and false negatives of resistant cases misclassified as resistant and of susceptible cases misclassified as resistant. In addition, the model performs well across most antibiotics as the true negative rate is high and represents accuracy in classifying susceptible cases. However, importantly the model is nearly perfect at predicting ciprofloxacin and chloramphenicol, with practically no false classifications. There however appear to be relatively more false negative than false positive results for antibiotics such as ampicillin, streptomycin and tetracycline where there might be the potential for not correctly identifying resistance cases. Since the incorrect classification of resistant samples as susceptible could turn into a major antibiotic treatment recommendation, more model optimization such as better data balancing, feature engineering, or hyperparameter optimization might be required. To summarize, these results showed that the DNN model performed well, but more refinements may further strengthen the reliability of the prediction in antibiotics where the classification of resistance is more difficult.

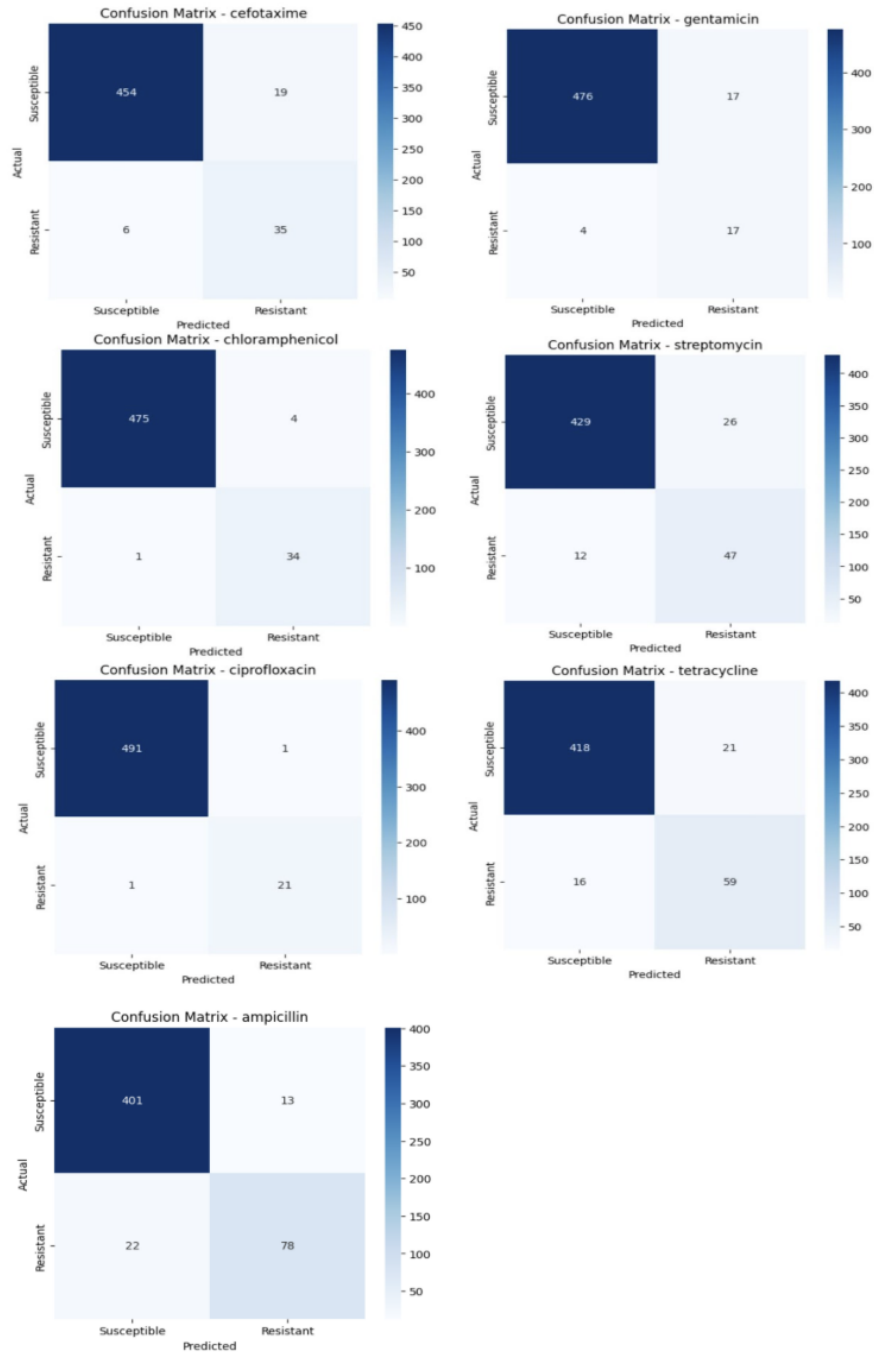


Figure 5.19: confusion matrix for DNN model salmonella Enterica

Train and test validation result

These figures are presented as DNN graphs where we show the training and validation accuracy trends for different antibiotics for an antibiotic resistance prediction model. While the rate of improvement of the training accuracy differs, it always improves. Validation accuracy decreases and in many cases, it remains higher but very volatile due to potential instability or variable data. Rifampin and isoniazid have lower overall accuracy and, although their combination prevents overfitting altogether, they still require richer optimizations to improve score accuracy and prevent underfitting. The validation accuracy of ethambutol and streptomycin shows sudden fluctuation may be

sensitive to data distribution or insufficient regularization. Index validation accuracy surpasses training for ofloxacin and capreomycin but experiences the least stable trends. While many models can generalize well, there are some for which a lower loss would justify tuning hyperparams, increasing the amount of data seen during training (whether in training or on the held-out data) or applying regularization techniques in the pursuit of robustness. Overall the models perform well, however further refinement is required to predict antibiotic resistance stability, reduce overfitting and improve generalization.

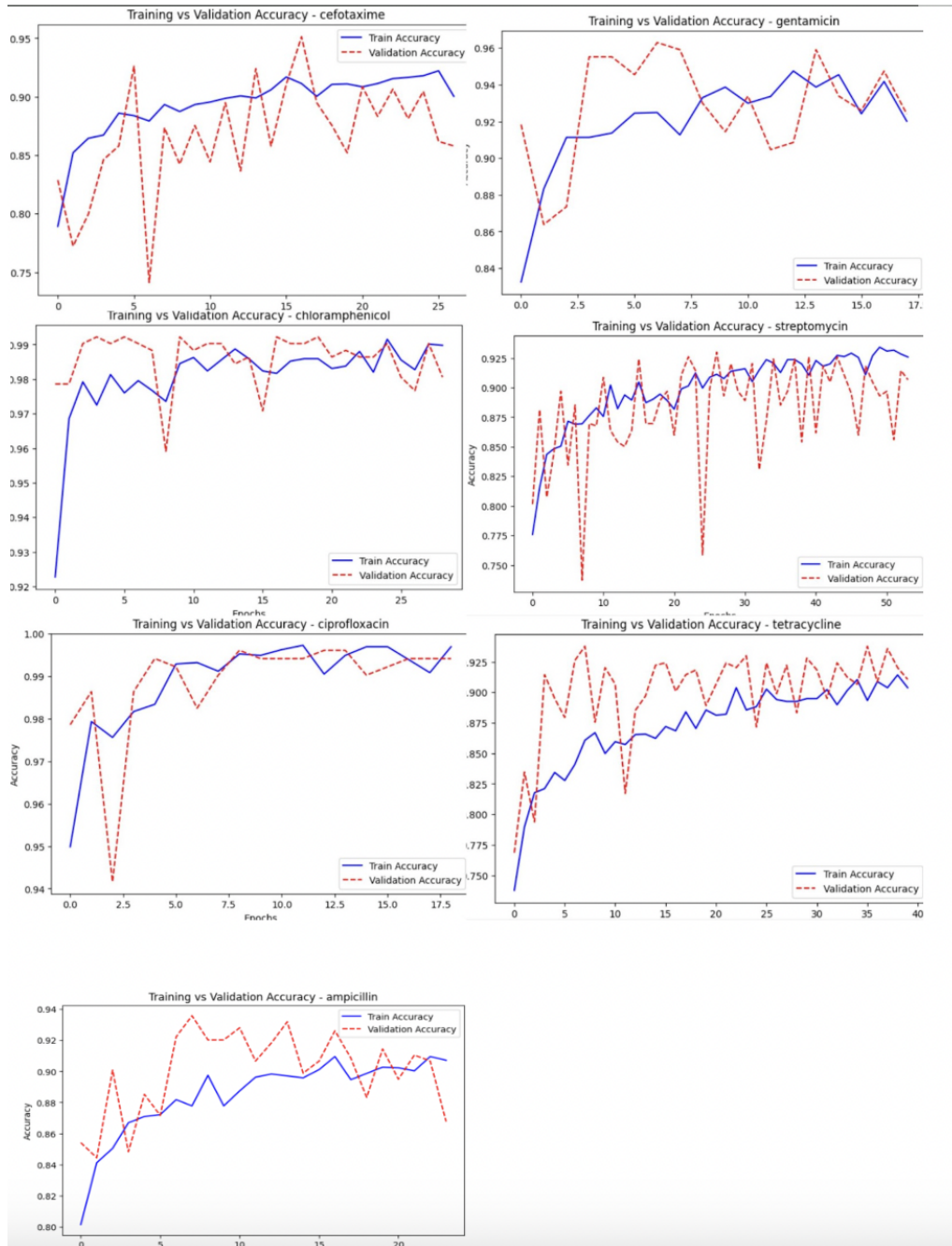


Figure 5.20: Train and validation accuracy DNN model salmonella Enterica

5.3.3 Stack Ensemble

Antibiotic	Accuracy	Precision	Recall
Ampicillin	0.94	0.87	0.79
Cefotaxime	0.95	0.72	0.63
Chloramphenicol	0.92	0.97	0.89
Ciprofloxacin	0.99	1.00	0.82
Gentamicin	0.98	0.93	0.67
Streptomycin	0.94	0.84	0.61
Tetracycline	0.94	0.85	0.71

Table 5.11: performance Metrics

Confusion Matrix analysis

A stacked model is used in predicting antibiotic resistance for *Salmonella enterica* across several antibiotics (ampicillin, cefotaxime, chloramphenicol, ciprofloxacin, gentamicin, streptomycin, tetracycline) and a significant improvement in prediction accuracy where false positive and false negatives are reduced as compared standalone models. For chloramphenicol and ciprofloxacin, the model performs well, avoiding errors with almost all susceptible and resistant instances. Cefotaxime and gentamicin are misclassified slightly more, with some resistant cases being classified as susceptible. Ampicillin and tetracycline are classified rather evenly, but with almost no errors, while streptomycin is classified moderately in either category. In all, the stacked model combines multiple points of view and hence provides better generalization for predictive robustness. Although the remaining misclassification rates may be due to further hyperparameter tuning or a more refined set of feature selection techniques, further accuracy may be realized to usefully predict resistance.

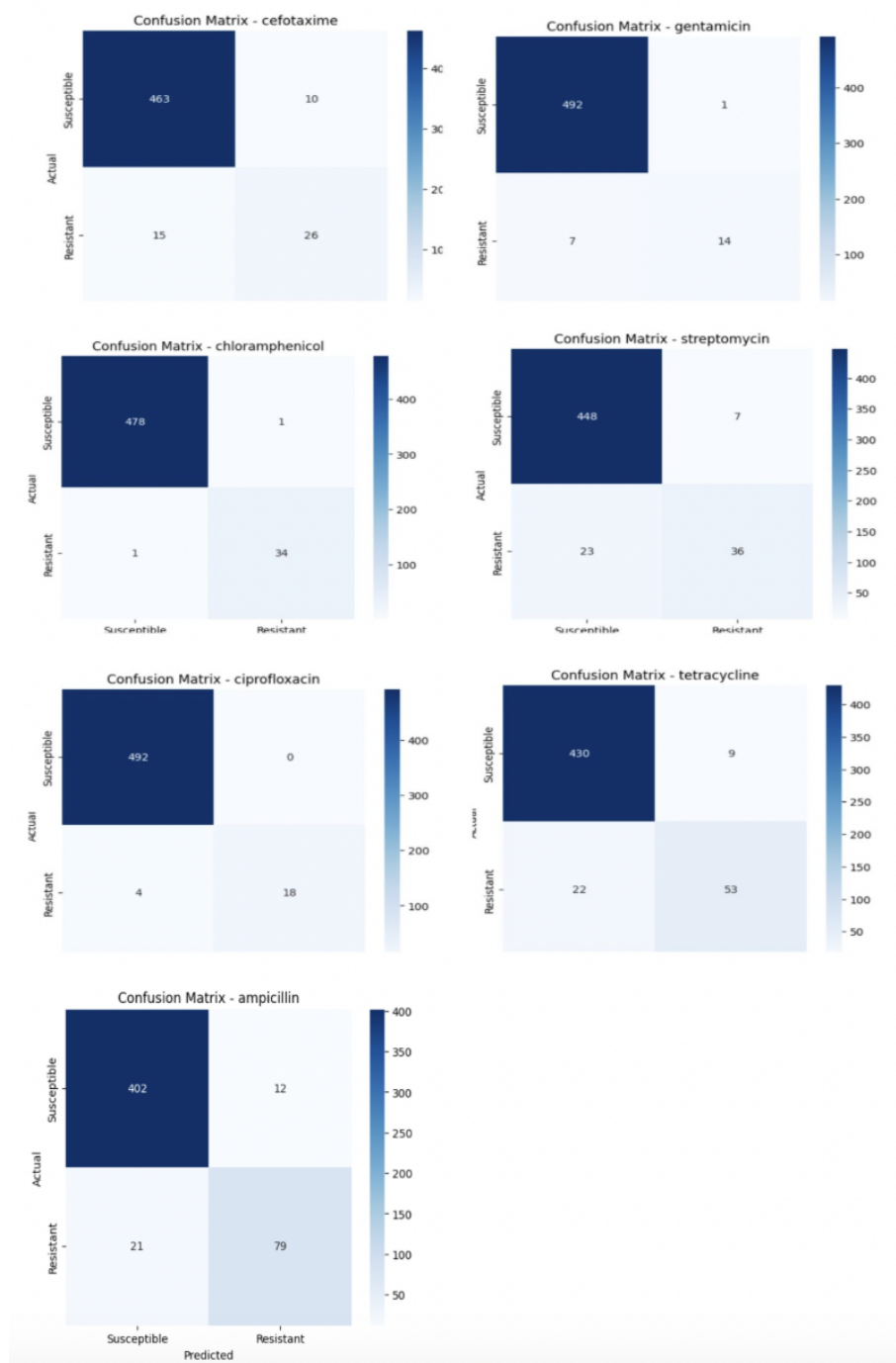


Figure 5.21: confusion matrix for stack ensemble model salmonella Enterica

5.3.4 Prototypical Network

Table 5.12: Model Performance Metrics for Antibiotic Resistance Prediction

Antibiotic	Accuracy	Precision	Recall	F1-Score
Ampicillin	0.96	0.93	0.99	0.96
Cefotaxime	0.97	0.95	1.00	0.97
Chloramphenicol	0.89	0.85	0.83	0.84
Ciprofloxacin	0.85	0.87	0.80	0.83
Gentamicin	0.98	0.98	0.99	0.98
Streptomycin	0.94	0.91	0.99	0.95
Tetracycline	0.96	0.95	0.97	0.96

Confusion Matrix analysis

In *Salmonella enterica* antibiotic resistance prediction, prototypical modelling achieves very strong classification performance on multiple antibiotics according to their confusion matrices. True positive, and true negative values are high while false positive and false negative values are relatively low, which shows that the model is robust. The model has high precision and recall for the antibiotics gentamicin, cefotaxime, and ampicillin, with the number of false negatives and false positives remaining below 20 in most cases. On the other hand, ciprofloxacin and chloramphenicol have higher false positive rates, 70 resistant samples were wrongly classified as non-resistant by ciprofloxacin, whereas 41 false positives were made by chloramphenicol. This indicates the possibility of trouble in differentiating the borderline cases or in the issue of data imbalance in some antibiotics. Despite variations in misclassification rates for each antibiotic based on the type of antibiotic and distribution of the dataset, the model is still reliable in predicting antibiotic resistance to *Salmonella enterica* with high overall classification accuracy.

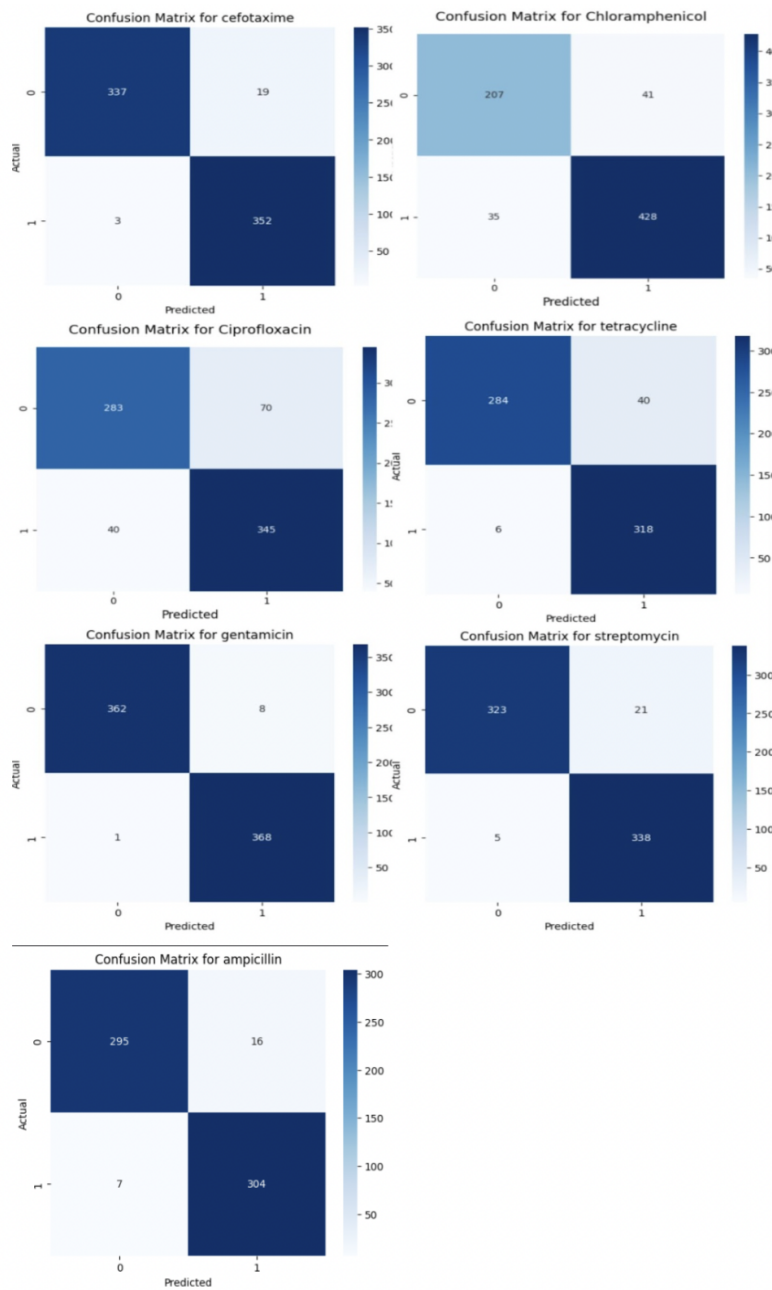


Figure 5.22: confusion matrix for Prototypical network model salmonella Enterica

Train and test validation result

Training versus validation accuracy graphs for various antibiotics in the prototypical model created for *Salmonella enterica* show consistently high accuracy over all antibiotics after 200 epochs, with accuracy of the validation, especially in cases of high training accuracy, comparable or greater than training accuracy, which suggests that effective learning was achieved and that overfitting is minimal. With very few exceptions in ciprofloxacin (98.2%) and chloramphenicol (99.0%), both accuracies converge to above 99% where the predictive capability is strong and with very little variance. Conversely, the training accuracy of antibiotics like ampicillin and tetracycline increases gradually while the validation accuracy shows little fluctuations, perhaps owing to the complexity of the dataset or class imbalances. Finally, based on the overall trend for all antibiotics seen across all the leveraging models, it appears that the model was able to learn antibiotic resistance patterns rapidly in the first 50 epochs and then stabilise. Though some slight dips may indicate the model's sensitivity to some changes in the data, validation accuracy peaking above training accuracy in most cases points to effective generalization. The performance of this prototypical model to predict antimicrobial resistance level demonstrates that the model is a robust framework for antimicrobial resistance classification for *Salmonella enterica* among different antibiotics.

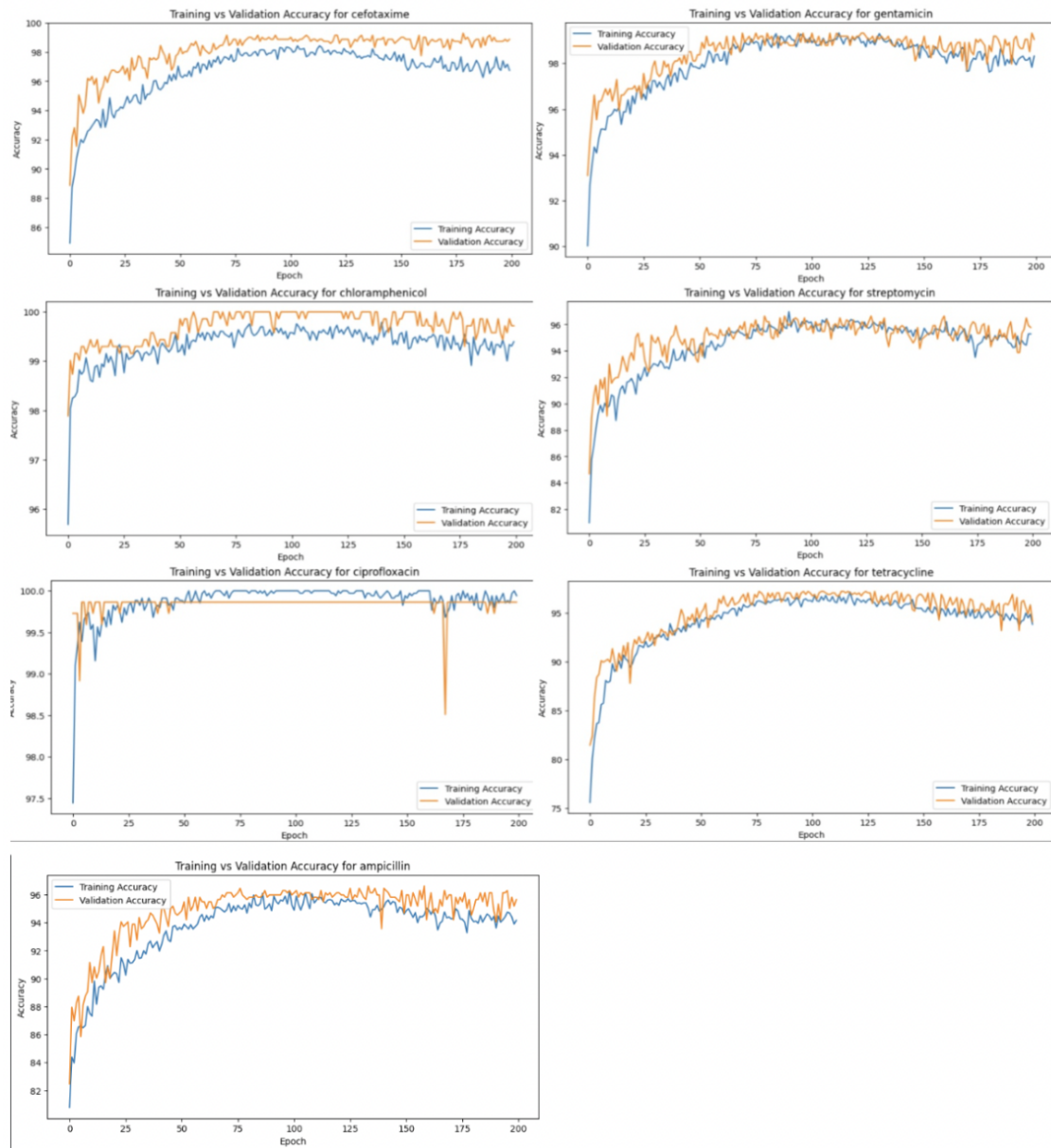


Figure 5.23: Train and validation accuracy Prototypical network model salmonella Enterica

5.3.5 Model comparison

The team tested the Salmonella dataset through Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), Stacked Models (XGBoost + LGBM + Logistic Regression + Random Forest), and Few-Shot Learning Models (Prototypical Networks). Every model demonstrated different performance results because its results depended on its accuracy, precision, recall and generalization abilities. The Few-Shot Model topped all models by achieving 97.89% for Cefotaxime antibiotic detection and 98.91% for Gentamicin antibiotic detection. It showed 98.39% Gentamicin precision and 95.45% Ciprofloxacin precision alongside above 99% recall rates for Cefotaxime and Gentamicin. The Few-Shot Model proves its excellence by detecting antibiotic resistance cases best with only a small amount of data before showing false results. The Stacked Model or XGBoost LGBM LR and Random Forest ensemble achieved 99.61% Chloramphenicol and 99.22% Ciprofloxacin accuracy but struggled to identify resistant forms of Streptomycin and Cefotaxime with recall ratios at 61.02% and 63.41%. The model detects most susceptible cases yet fails to identify many resistant strains. The lower numbers indicate that the model has classification problems with an unbalanced strain distribution because it identifies too many resistant bacteria. The CNN system generates dependable results because it precisely tests samples of Ampicillin (78.5% precision recording 88% of samples correctly) and Ciprofloxacin (95.45% recall with an accuracy of 99.41%). The model's perception of spatial nature enabled it to analyze multi-layered resistance patterns effectively in genetic readings. The system did not perform as well against Streptomycin at 61.02% recall and Tetracycline at 70.67% recall. The DNN Model showed good overall results but proved less reliable when measuring precision and recall numbers. The model could not properly detect all cases of antibiotic resistance when using Cefotaxime and Streptomycin despite the test. Testing samples that belong to the susceptible group always yielded correct results because the model focused more on those cases when it achieved high accuracy.

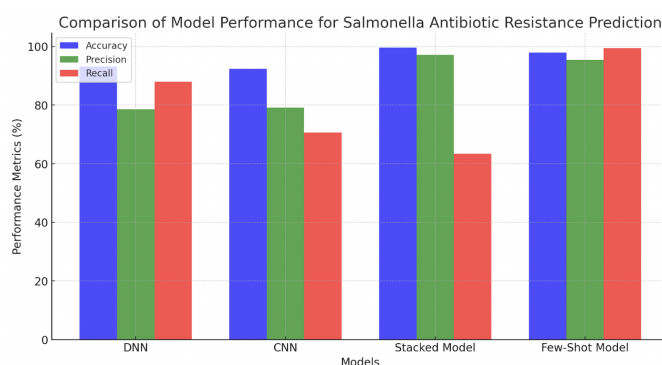


Figure 5.24: Model comparison for salmonella Enterica

Model Ranking (Best to Worst)

1. The Few-Shot Learning Model (Prototypical Networks) wins overall because it gives the best test results and detects resistance more accurately than other models.

2. This stacked model uses XGBoost LGBM Logistic Regression and Random Forest algorithms to produce the highest precision results which makes it perfect to identify non-resistant bacteria with high accuracy.
3. The CNN model delivers a consistent performance while recognizing resistance in more antibiotics except showing unreliable results in Streptomycin and Tetracycline tests.
4. The DNN demonstrated mixed outcomes within its performance range as it struggled to identify samples properly because of its fitting problems and unequal test data distribution.

Chapter 6

Conclusion

This paper outlines a systematic approach for the prediction of the AMR by analyzing microbial genomes and applying the predictive machine algorithms. First there is a step of extracting bacterial genome sequences, followed by converting of contig sequences into the shotgun scaffold sequences and their alignment with the reference genomes. The study therefore lays a robust data pipeline for AMR detection, involving k-mer analysis, mutation profiling and identification of AMR genes from CARD database. Using such genotype matrix and phenotypic information helps in realizing mechanisms of bacterial resistance and for the identification of the AMR profile of a given model. This has ensured that there is an integration of bioinformatics and AI in the workflow to arrive at correct predictions of antibiotic resistance from genomic sequences. This supports the role of machine learning in the further enhancement of antibiotic use and reducing the menace of AMR in the global setting.

Bibliography

- [1] M. Boetzer, C. V. Henkel, H. J. Jansen, D. Butler, and W. Pirovano, “Scaffolding pre-assembled contigs using sspace,” *Bioinformatics*, vol. 27, no. 4, pp. 578–579, Dec. 2010, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btq683. eprint: https://academic.oup.com/bioinformatics/article-pdf/27/4/578/48866383/bioinformatics_27_4_578.pdf. [Online]. Available: <https://doi.org/10.1093/bioinformatics/btq683>.
- [2] J. Davies and D. Davies, “Origins and evolution of antibiotic resistance,” *Microbiology and Molecular Biology Reviews*, vol. 74, no. 3, pp. 417–433, 2010. DOI: 10.1128/mmbr.00016-10. [Online]. Available: <https://doi.org/10.1128/mmbr.00016-10>.
- [3] H. K. Allen, “Antibiotic resistance gene discovery in food-producing animals,” *Current Opinion in Microbiology*, vol. 19, pp. 25–29, 2014, Ecology and industrial microbiology • Special Section: Novel technologies in microbiology, ISSN: 1369-5274. DOI: <https://doi.org/10.1016/j.mib.2014.06.001>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S136952741400068X>.
- [4] G. Arango-Argoty, E. Garner, A. Pruden, and et al., “Deeparg: A deep learning approach for predicting antibiotic resistance genes from metagenomic data,” *Microbiome*, vol. 6, p. 23, 2018. DOI: 10.1186/s40168-018-0401-z. [Online]. Available: <https://doi.org/10.1186/s40168-018-0401-z>.
- [5] L. Li, X. Yin, and T. Zhang, “Tracking antibiotic resistance gene pollution from different sources using machine-learning classification,” *Microbiome*, vol. 6, p. 93, 2018. DOI: 10.1186/s40168-018-0480-x. [Online]. Available: <https://doi.org/10.1186/s40168-018-0480-x>.
- [6] J. H. Yang, S. N. Wright, M. Hamblin, *et al.*, “A white-box machine learning approach for revealing antibiotic mechanisms of action,” *Cell*, vol. 177, no. 6, 1649–1661.e9, 2019. DOI: 10.1016/j.cell.2019.04.016. [Online]. Available: <https://doi.org/10.1016/j.cell.2019.04.016>.
- [7] D. Aytan-Aktug, P. T. L. C. Clausen, V. Bortolaia, F. M. Aarestrup, and O. Lund, “Prediction of acquired antimicrobial resistance for multiple bacterial species using neural networks,” *mSystems*, vol. 5, no. 1, 10.1128/msystems.00774–19, 2020. DOI: 10.1128/msystems.00774-19. eprint: <https://journals.asm.org/doi/pdf/10.1128/msystems.00774-19>. [Online]. Available: <https://journals.asm.org/doi/abs/10.1128/msystems.00774-19>.
- [8] M. R. Szal, V. Stebliankin, K. Mathee, and G. Narasimhan, “Causal inference in microbiomes using intervention calculus,” *bioRxiv*, 2020. DOI: 10.1101/2020.02.28.970624. eprint: <https://www.biorxiv.org/content/early/2020/03/03/>

2020.02.28.970624.full.pdf. [Online]. Available: <https://www.biorxiv.org/content/early/2020/03/03/2020.02.28.970624>.

- [9] J. Stokes, K. Yang, K. L. Swanson, *et al.*, “A deep learning approach to antibiotic discovery,” *Cell*, vol. 181, no. 2, pp. 475–483, 2020. DOI: 10.1016/j.cell.2020.04.001.
- [10] M. N. Anahtar, J. H. Yang, and S. Kanjilal, “Applications of machine learning to the problem of antimicrobial resistance,” *Journal of Clinical Microbiology*, vol. 59, no. 7, e01260–20, 2021. DOI: 10.1128/jcm.01260-20. [Online]. Available: <https://journals.asm.org/doi/10.1128/jcm.01260-20>.
- [11] J. L. Espinoza, C. L. Dupont, A. O’Rourke, *et al.*, “Predicting antimicrobial mechanism-of-action from transcriptomes: A generalizable explainable artificial intelligence approach,” *PLOS Computational Biology*, vol. 17, no. 3, pp. 1–25, Mar. 2021. DOI: 10.1371/journal.pcbi.1008857. [Online]. Available: <https://doi.org/10.1371/journal.pcbi.1008857>.
- [12] G. Feretzakis, A. Sakagianni, E. Loupelis, *et al.*, “Machine learning for antibiotic resistance prediction: A prototype using off-the-shelf techniques and entry-level data to guide empiric antimicrobial therapy,” *Healthc Inform Res*, vol. 27, no. 3, pp. 214–221, Jul. 2021. DOI: 10.4258/hir.2021.27.3.214.
- [13] M. Pascucci, G. Royer, J. Adamek, and *et al.*, “Ai-based mobile application to fight antibiotic resistance,” *Nature Communications*, vol. 12, p. 1173, 2021. DOI: 10.1038/s41467-021-21187-3. [Online]. Available: <https://doi.org/10.1038/s41467-021-21187-3>.
- [14] M. J. Ellington, O. Ekelund, F. M. Aarestrup, *et al.*, “Machine learning for antimicrobial resistance prediction: Current practice, limitations, and clinical perspective,” *Clinical Microbiology Reviews*, vol. 35, no. 3, e00179–21, 2022. DOI: 10.1128/cmr.00179-21. [Online]. Available: <https://journals.asm.org/doi/10.1128/cmr.00179-21>.
- [15] H. Yoo, B. A. Sokhansanj, J. R. Brown, and G. Rosen, “Predicting antimicrobial resistance using large language models,” *arXiv*, 2023. DOI: 10.48550/arxiv.2401.00642. eprint: arxiv.2401.00642. [Online]. Available: <https://doi.org/10.48550/arxiv.2401.00642>.