Electrical and Computer Engineering

EECE 5644 (Summer 2024)

Final Project Report

Title : Performance Analysis of Machine Learning Models in 5G Network Traffic Classification

Aniket Fasate

fasate.a@northeastern.edu

Ian Cummings

cummings.i@northeastern.edu

Abstract

This project investigates the effectiveness of various machine learning models in classifying 5G network traffic types, specifically enhanced Mobile Broad Band (eMBB), Ultra Reliable Low Latency Communication (URLLC), and massive Machine Type Communication (mMTC). Our objective is to analyze and determine the most efficient models for handling the diverse and dynamic nature of 5G traffic, with a focus on network slicing—a crucial technology for ensuring optimal network performance and resource utilization. We evaluated several machine learning models including Naïve Bayes, Random Forest, Logistic Regression, and Linear Discriminant Analysis, utilizing a dataset characterized by multiple key performance indicators. The findings indicate that deep learning models show considerable promise due to their accuracy and adaptability, potentially paving the way for enhanced 5G network management solutions. This research contributes foundational insights into the application of machine learning techniques for the future of network traffic classification and management.

Contents

1 Introduction

As graduate students in the Internet of Things (IoT) program, we focused our research on 5G networks, analyzing how various machine learning models classify different types of network traffic, specifically eMBB, mMTC, and URLLC. The aim of this project is to evaluate the performance of different models in this context and to gain a foundational understanding of 5G network slicing.

Our key findings indicate that deep learning models are particularly adept at managing the diverse and dynamic nature of 5G traffic. These models excel in accuracy and demonstrate significant potential for future network requirements. Our study lays the groundwork for more intelligent, machine learning-driven solutions that could transform 5G network management and set the stage for the next generation of digital communication. In our analysis, we assessed the performance of Naïve Bayes, Random Forest, Logistic Regression, and LDA models.

2 Problem Formulation

In 5G networks, network slicing plays a crucial role as it allows the creation of multiple virtual networks on a single physical infrastructure. These networks are designed to support a wide range of applications, each with distinct requirements for bandwidth, latency, reliability, and security. Network slicing ensures that each application has a customized portion of the network tailored to its specific needs.

This approach not only accommodates diverse network requirements but also enhances the efficiency of physical network utilization. Resources can be dynamically allocated to different slices based on demand, leading to better bandwidth utilization and cost reduction. To achieve efficient and rapid network slicing, it is essential to use fast and effective machine learning algorithms that can accurately classify network traffic and allocate the necessary network resources.

Furthermore, network traffic classification offers additional benefits, such as maintaining Quality of Service (QoS), improving security through attacker detection, streamlining billing and charging processes, and managing congestion. Therefore, addressing the challenges associated with network traffic classification in 5G networks is a critical problem that needs to be solved.

3 Related Work

In [1], the "DeepSlice" model is introduced, which utilizes neural networks and a random forest classifier to improve network slicing by predicting and managing network loads. The primary goal of this study is to ensure optimal resource allocation and high reliability, especially during network failures. The focus on using a Random Forest classifier and Deep Learning Neural Network for dynamic resource management in 5G networks provides valuable insights for our project on network traffic classification, making it an essential reference for our use case.

In [2], an approach to enhancing network security in 5G technologies is discussed. This paper addresses the challenges and vulnerabilities of network slicing, particularly concerning isolation and security against threats such as DDoS attacks. The research introduces 'Secure5G,' a neural network-based model designed to actively manage and secure network slices from potential cyber threats by dynamically analyzing traffic patterns to differentiate between legitimate and malicious requests. This model enhances the robustness of 5G networks by ensuring that each slice operates independently and securely, protecting against network vulnerabilities. The paper offers crucial insights into developing secure network architectures necessary to support the diverse and extensive demands of future 5G applications.

4 Data Processing

The dataset utilized in our study comprises 8 features or KPIs (Key Performance Indicators), which include: Use Case Type, LTE/5G UE Category, Technology Supported, Day, Time, QCI, Packet Loss Rate (Reliability), and Packet Delay Budget (Latency). These features are associated with multi-class labels (3 categories) representing the network slice types: enhanced Mobile Broad Band (eMBB), Ultra Reliable Low Latency Communication (URLLC), and massive Machine Type Communication (mMTC). The dataset consists of 466,739 samples with corresponding labels.

5 KPIs (Key Performance Indicators) (Inputs)

Use Case Type (8 Categories): Refers to the type of device utilizing the network. These categories include smartphones, IoT devices, Industry 4.0, AR/VR, healthcare, public safety, smart city, and home devices.

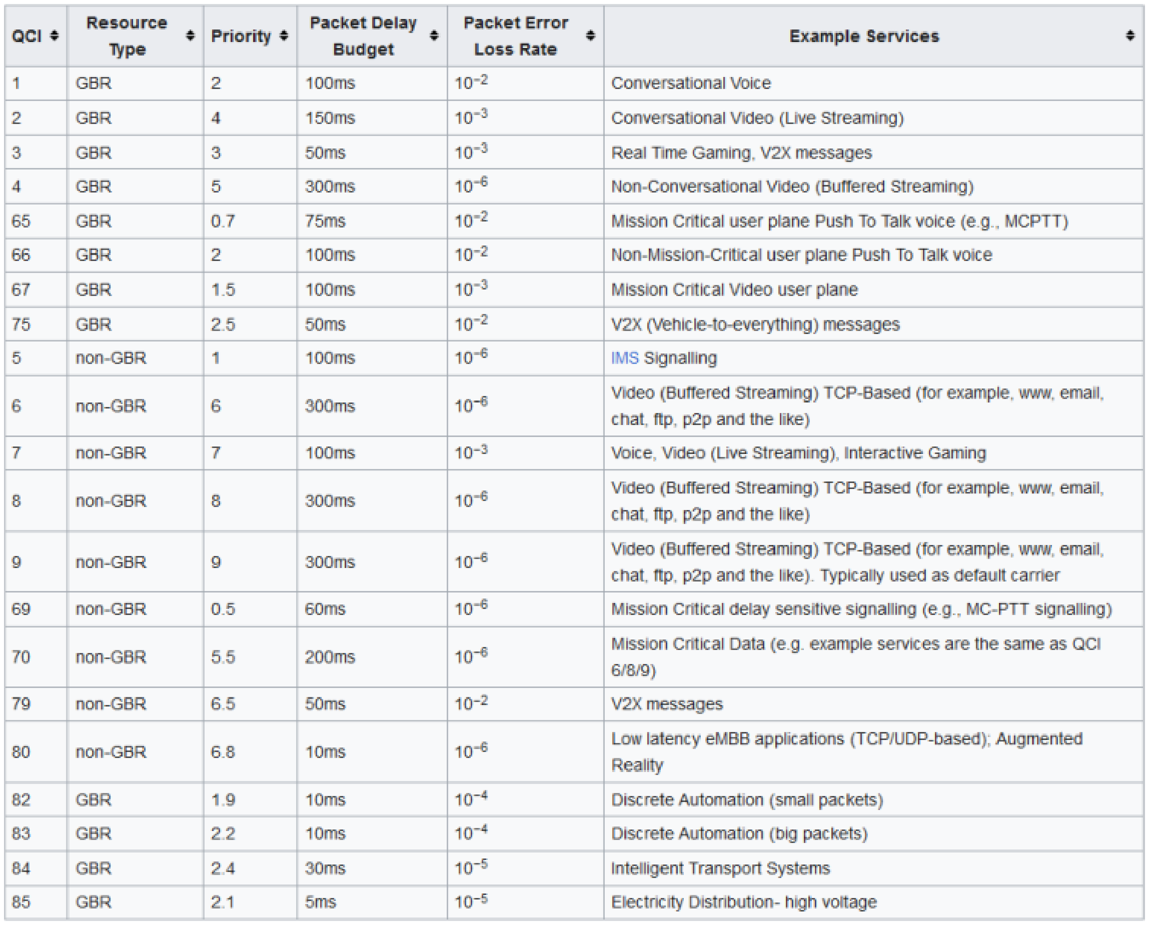
LTE/5G UE Category (23 Categories): Specifies the performance characteristics of a particular LTE device.

Technology Supported (2 Categories): Indicates whether the device supports LTE/5G or IoT (including LTE-M and NB-IoT).

Day (7 Categories): Denotes the day of the week.

Time (23 Categories): Represents the time of day.

QCI (QoS Class Identifier): This metric is used by networks to ensure that a specific user equipment (UE) is allocated sufficient resources to meet the required Quality of Service (QoS). The QCI values, which range as illustrated in the table below, correspond to different KPI values.



QCI	Resource Type	Priority	Packet Delay Budget	Packet Error Loss Rate	Example Services
1	GBR	2	100ms	10^{-2}	Conversational Voice
2	GBR	4	150ms	10^{-3}	Conversational Video (Live Streaming)
3	GBR	3	50ms	10^{-3}	Real Time Gaming, V2X messages
4	GBR	5	300ms	10^{-6}	Non-Conversational Video (Buffered Streaming)
65	GBR	0.7	75ms	10^{-2}	Mission Critical user plane Push To Talk voice (e.g., MCPTT)
66	GBR	2	100ms	10^{-2}	Non-Mission-Critical user plane Push To Talk voice
67	GBR	1.5	100ms	10^{-3}	Mission Critical Video user plane
75	GBR	2.5	50ms	10^{-2}	V2X (Vehicle-to-everything) messages
5	non-GBR	1	100ms	10^{-6}	IMS Signalling
6	non-GBR	6	300ms	10^{-6}	Video (Buffered Streaming) TCP-Based (for example, www, email, chat, ftp, p2p and the like)
7	non-GBR	7	100ms	10^{-3}	Voice, Video (Live Streaming), Interactive Gaming
8	non-GBR	8	300ms	10^{-6}	Video (Buffered Streaming) TCP-Based (for example, www, email, chat, ftp, p2p and the like)
9	non-GBR	9	300ms	10^{-6}	Video (Buffered Streaming) TCP-Based (for example, www, email, chat, ftp, p2p and the like). Typically used as default carrier
69	non-GBR	0.5	60ms	10^{-6}	Mission Critical delay sensitive signalling (e.g., MC-PTT signalling)
70	non-GBR	5.5	200ms	10^{-6}	Mission Critical Data (e.g. example services are the same as QCI 6/8/9)
79	non-GBR	6.5	50ms	10^{-2}	V2X messages
80	non-GBR	6.8	10ms	10^{-6}	Low latency eMBB applications (TCP/UDP-based); Augmented Reality
82	GBR	1.9	10ms	10^{-4}	Discrete Automation (small packets)
83	GBR	2.2	10ms	10^{-4}	Discrete Automation (big packets)
84	GBR	2.4	30ms	10^{-5}	Intelligent Transport Systems
85	GBR	2.1	5ms	10^{-5}	Electricity Distribution- high voltage

Figure 1: Standardized characteristics as defined in the 3GPP TS 23.203 standard ”Policy and Charging Control Architecture” (Source: Wikipedia, 3GPP Specifications)

Since the QCI values capture the Packet Loss Rate and Packet Delay Budget, these are not considered as inputs to the model.

6 Slice Categories (Outputs)

enhanced Mobile Broad Band(eMBB): This category is aimed at delivering much faster data speeds and greater capacity than current 4G networks. It is designed to support applications requiring high data rates over wide coverage areas, such as 4K streaming, virtual reality (VR), rural and remote areas, and densely populated environments like stadiums and concert halls.

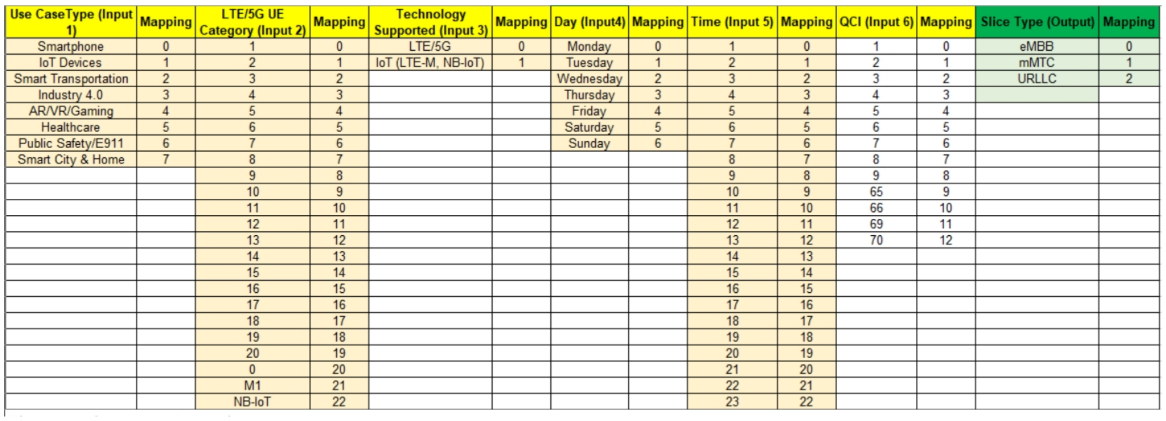
Ultra Reliable Low Latency Communication (URLLC): This category is essential for extremely reliable and low-latency communications, critical for mission-critical and low-latency applications. The latency targets can be as low as 1 millisecond. Examples include autonomous vehicles, remote surgery, biomedical applications, and industrial automation.

massive Machine Type Communication (mMTC): This category is tailored to provide wide-area coverage and focuses on connecting numerous low-energy, low-cost devices that transmit small amounts of data infrequently. Examples include IoT devices and smart agriculture.

Correlation Matrix between Inputs and Outputs: The correlation matrix reveals that 'Day' and 'Time' inputs contribute the least to the output.

7 Processing

1. The dataset is imported into the Python script using the pandas library.
2. The dataset undergoes cleaning by removing empty columns, and all values are converted to string type.
3. Given that all feature and label categories are in string format, they need to be transformed into numerical categories.



Use CaseType (Input 1)	Mapping	LTE/5G UE Category (Input 2)	Mapping	Technology Supported (Input 3)	Mapping	Day (Input4)	Mapping	Time (Input 5)	Mapping	QCI (Input 6)	Mapping	Slice Type (Output)	Mapping
Smartphone	0	1	0	LTE/5G	0	Monday	0	1	0	1	0	eMBB	0
IoT Devices	1	2	1	IoT (LTE-M, NB-IoT)	1	Tuesday	1	2	1	2	1	mMTC	1
Smart Transportation	2	3	2			Wednesday	2	3	2	3	2	URLLC	2
Industry 4.0	3	4	3			Thursday	3	4	3	4	3		
AR/VR/Gaming	4	5	4			Friday	4	5	4	5	4		
Healthcare	5	6	5			Saturday	5	6	5	6	5		
Public Safety/E911	6	7	6			Sunday	6	7	6	7	6		
Smart City & Home	7	8	7					8	7	8	7		
		9	8					9	8	9	8		
		10	9					10	9	65	9		
		11	10					11	10	66	10		
		12	11					12	11	69	11		
		13	12					13	12	70	12		
		14	13					14	13				
		15	14					15	14				
		16	15					16	15				
		17	16					17	16				
		18	17					18	17				
		19	18					19	18				
		20	19					20	19				
		0	20					21	20				
		M1	21					22	21				
		NB-IoT	22					23	22				

Figure 2: Category Mapping

4. After completing the mapping, the dataset is divided into features (X) and labels (y),

which are saved as two separate CSV files. These files are then loaded for model training and analysis.

5. The dataset comprises 466,739 samples, each with 8 features, along with 466,739 corresponding labels.

Code: hthttps://github.com/AniketFasate27/EECE5644-Intro$_to_ML/tree/main/Project$.

6. The dataset, loaded from the CSV file 'processed_data_without_packet_loss_delay.csv', includes pre-processed telecommunications data with one-hot encoding applied. Initial steps involved removing columns with no information (empty or unnamed columns) to maintain the dataset's quality and relevance.

Code: https://github.com/AniketFasate27/EECE5644-Intro$_to_ML/tree/main/Project$.

8 Experiments

Random Forest Classification

The Random Forest Classifier is an ensemble learning algorithm that combines the outputs of multiple decision trees to produce a final output value. This classifier includes several decision trees and utilizes bagging and feature bagging to ensure these trees are uncorrelated. Key parameters for the classifier include:

1. n_estimators: The number of decision trees in the forest.
2. max_depth: The maximum depth of the trees.
3. min_samples_split: The minimum number of samples required to split a node.
4. min_samples_leaf: The minimum number of samples required to be a leaf node.

For our project, we used the Random Forest Classifier from scikit-learn, tuned the hyperparameters using Grid Search, and evaluated the model's performance based on accuracy, weighted F1 score, ROC (One-vs-Rest), and ROC-AUC scores.

Data Loading and Processing:

1. Features (X_data.csv) and labels (y_data.csv) were loaded.
2. As previously mentioned, Packet Loss Rate and Packet Delay Budget values were excluded from the features.
3. To ensure computational efficiency during the grid search and focus on a manageable subset of data, a random sample of 100,000 entries was selected from the dataset.

Hyperparameter Tuning:

1. To identify the optimal parameters for the Random Forest Classifier that yield the highest accuracy, Grid Search with Cross-Validation was performed using GridSearchCV from the scikit-learn library. The parameters considered were n_estimators, max_depth, min_samples_split, and min_samples_leaf, with a 5-fold cross-validation.
2. Grid Search exhaustively tests all combinations of model parameters on the dataset and returns the best-performing model and its parameters. The grid search was conducted on the data subset of 100,000 samples, which was split into a 70:30 ratio for training and testing.

8.1 Model Training:

1. After identifying the best-performing model, its parameters are used for training. The complete dataset (466,739 samples) is divided into training and test sets using a 70:30 split. The 'stratify' parameter is employed to maintain the class distribution within both the training and test sets. The model is then trained on the entire dataset.

Model Performance Metrics:

Accuracy, weighted F1 score, ROC-AUC (One-vs-Rest) score, and ROC Curves are calculated and plotted.

1. **Weighted F1 Score:** The F1 score is computed individually for each class, and their weighted average is calculated, with each class's F1 score weighted by the number of true instances of that class.

2. **ROC-AUC (One-vs-Rest):** The ROC-AUC score is calculated for each class against the rest, and their weighted average is computed.

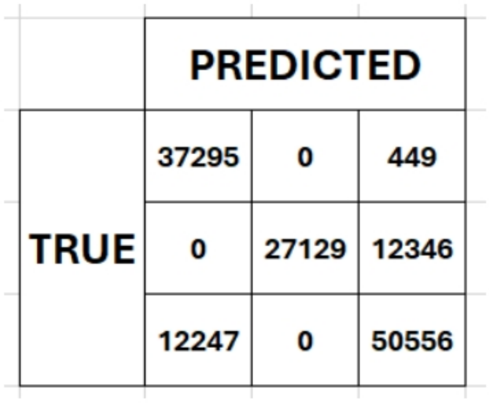
3. During the train-test split, stratification was applied to check for improvements in accuracy. Due to the minimal class imbalance, this adjustment did not significantly enhance the model's performance.

8.2 Results

1. Optimum Model Parameters:

- max_depth = 5
- min_samples_leaf = 1
- min_samples_split = 2
- n_estimators = 9

2. Accuracy: 82.12%
3. F1 Score (Weighted): 0.8195
4. ROC-AUC Score: 0.94695
5. Confusion Matrix:



	PREDICTED		
TRUE	37295	0	449
	0	27129	12346
	12247	0	50556

Figure 3: Confusion Matrix

6. ROC Curve

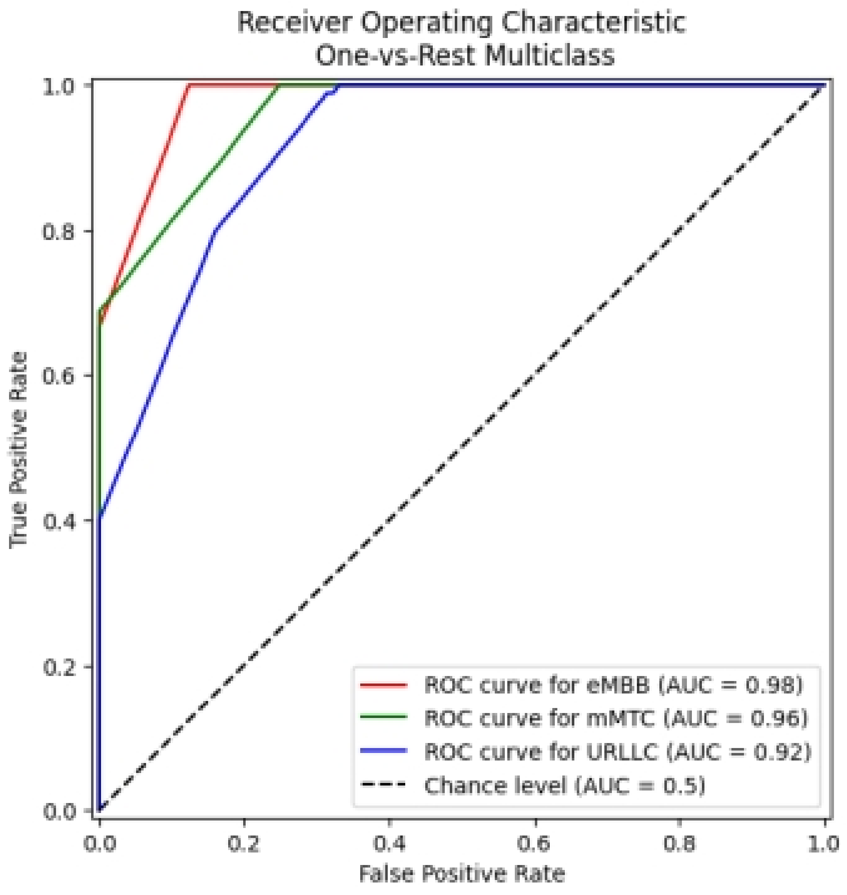


Figure 4: Updated Data Points

8.2.1 Train RMSE vs Test RMSE:

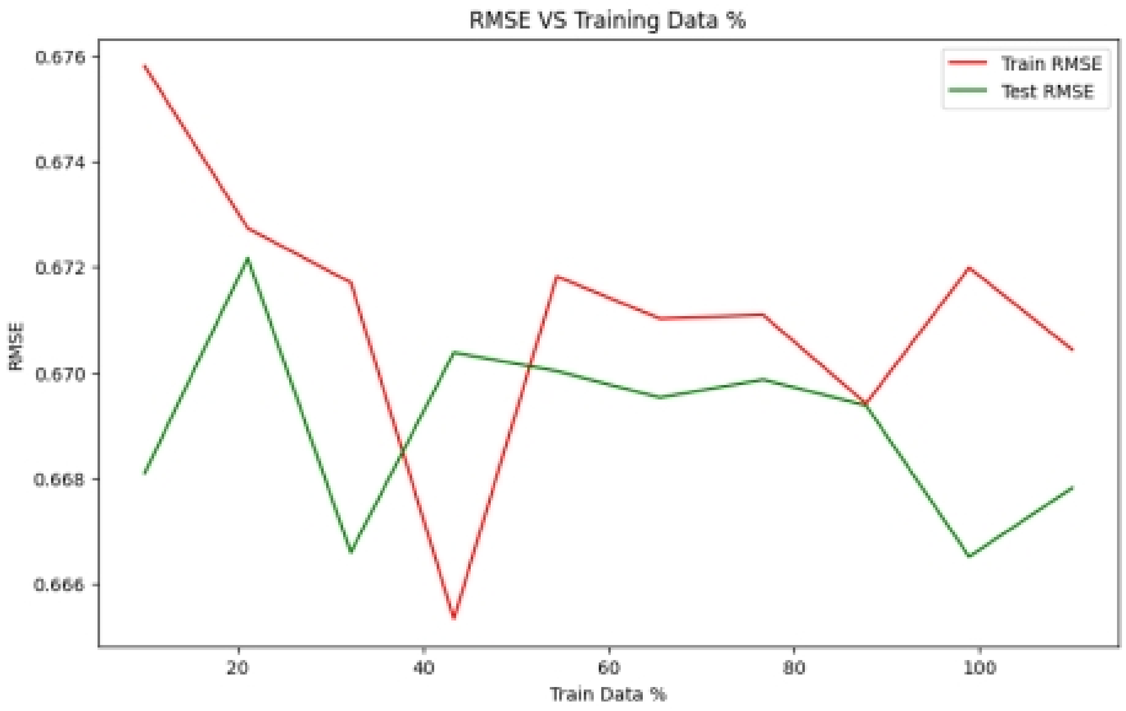


Figure 5: Train RMSE vs Test RMSE:

Observations from the Graph:

- **Decrease in RMSE:** There is a general downward trend in both training and testing RMSE as the percentage of training data increases, indicating improved model performance with more data.
- **Variability:**Despite the presence of variability in the graph, the magnitude of these fluctuations is relatively low. Stratification Impact: The stratification of the dataset has enhanced the model's stability.

Code: https://github.com/AniketFasate27/EECE5644-Intro$_{t}o_{M}L/tree/main/Project$.
Code: https://github.com/AniketFasate27/EECE5644-Intro$_{t}o_{M}L/tree/main/Project$.

9 Logistic Regression Classification

This report details the application of Logistic Regression, a machine learning classification algorithm, to predict network slice types within a telecommunications dataset. The importance of this task lies in the efficient categorization of network traffic, which is essential for maintaining service quality and effective resource management. Logistic Regression was selected due to its interpretability and effectiveness in both binary and multi-class classification scenarios.

For our analysis, we employed the Logistic Regression classifier from Scikit-learn, tuning the hyperparameters using methods such as RandomizedSearchCV. The model's performance was assessed using various metrics.

9.1 Methodology

Logistic Regression was chosen for its simplicity, efficiency, and interpretability. It is well-regarded for its performance in classification tasks and was adapted to handle a multi-class scenario using the one-vs-rest (OvR) strategy. This strategy involves fitting a separate classifier for each class against all other classes, enabling the extension of Logistic Regression's binary classification capabilities to our multi-class problem.

For model optimization, 'RandomizedSearchCV' was utilized to explore a predefined parameter space and determine the optimal regularization strength (C) and solver type. The chosen metric for validation was accuracy, and a three-fold cross-validation approach was implemented to ensure the robustness of the model.

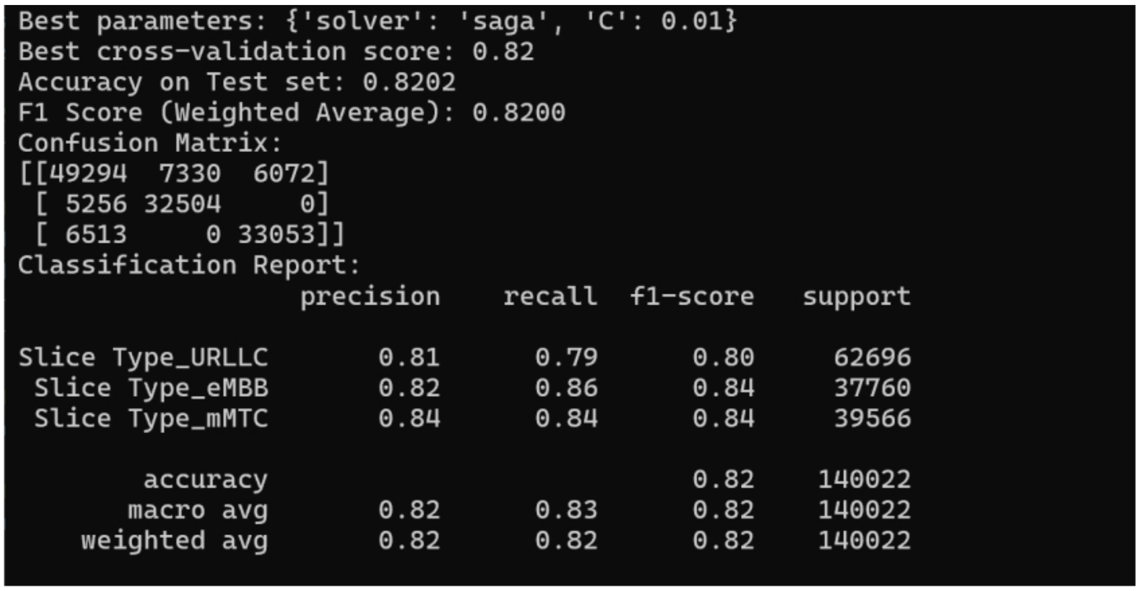
9.2 Training/Test Split:

The dataset was divided into training and test sets using a 70:30 ratio. This split ensures a substantial amount of data is available for learning while retaining a representative subset for unbiased evaluation. The 'random_state' parameter was set to 42 to guarantee that the results are reproducible across different runs.

9.3 Model Training:

The Logistic Regression model was initialized with a maximum of 10,000 iterations (‘max_iter‘) to ensure convergence. The ‘multi_class‘ parameter was set to ’ovr’ to employ a one-vs-rest approach, which is suitable for multi-class classification.

9.4 Model Evaluation



```
Best parameters: {'solver': 'saga', 'C': 0.01}
Best cross-validation score: 0.82
Accuracy on Test set: 0.8202
F1 Score (Weighted Average): 0.8200
Confusion Matrix:
[[49294  7330  6072]
 [ 5256 32504     0]
 [ 6513     0 33053]]
Classification Report:
                  precision    recall  f1-score   support

Slice Type_URLLC       0.81      0.79      0.80     62696
 Slice Type_eMBB       0.82      0.86      0.84     37760
 Slice Type_mMTC       0.84      0.84      0.84     39566

        accuracy                           0.82    140022
       macro avg       0.82      0.83      0.82    140022
    weighted avg       0.82      0.82      0.82    140022
```

Figure 6: Model Evaluation

9.5 ROC Curve and AUC:

The ROC curve is a graphical representation that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The AUC score provides a single measure of the model’s ability to distinguish between classes, irrespective of the classification threshold set. In this analysis, separate ROC curves were plotted for each class:

- Slice Type_URLLC: AUC of 0.93
- Slice Type_eMBB: AUC of 0.98

- Slice Type_mMTC: AUC of 0.98

These high AUC values indicate excellent model performance in distinguishing each class from the others, particularly for eMBB and mMTC, where the AUC scores are nearly perfect.

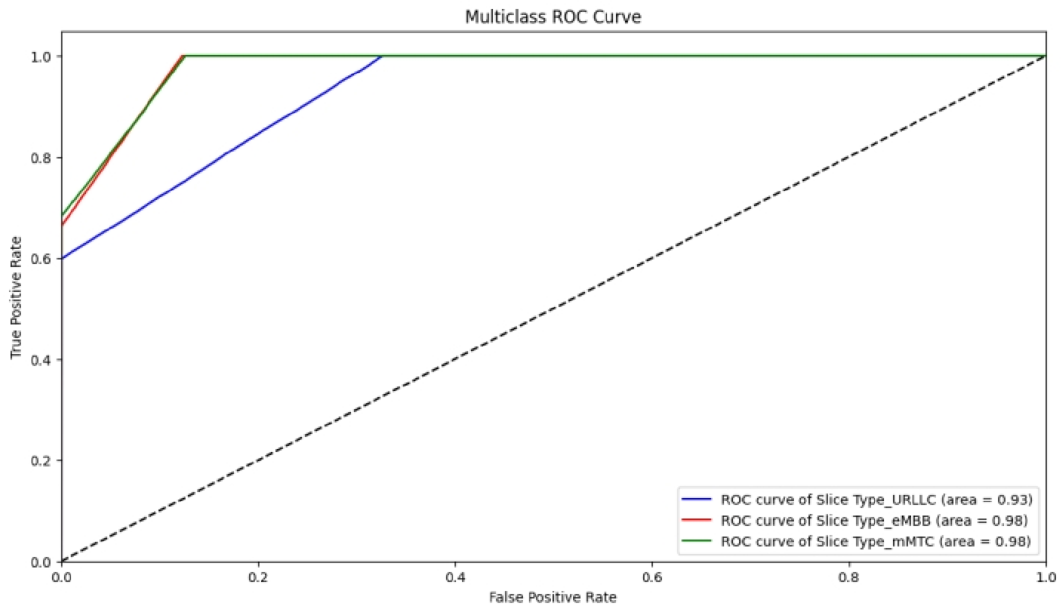


Figure 7: False Positive Rate

9.5.1 Accuracy:

The model achieved an accuracy of 82.02% on the test set. This high accuracy reflects the model's effectiveness in correctly predicting the class labels across the dataset, indicating strong generalization performance.

9.5.2 F1 Score (Weighted Average)

The F1 score, which is the harmonic mean of precision and recall, was calculated using a weighted average to account for any class imbalance within the dataset, resulting in a score of 0.8200. This metric is particularly useful in scenarios with imbalanced classes, as it does not inflate the overall score based on high performance on an abundant class.

9.5.3 Confusion Matrix:

The confusion matrix offers detailed insights into the distribution of predictions across actual labels, pinpointing areas where the model excelled and identifying where misclassifications were more frequent:

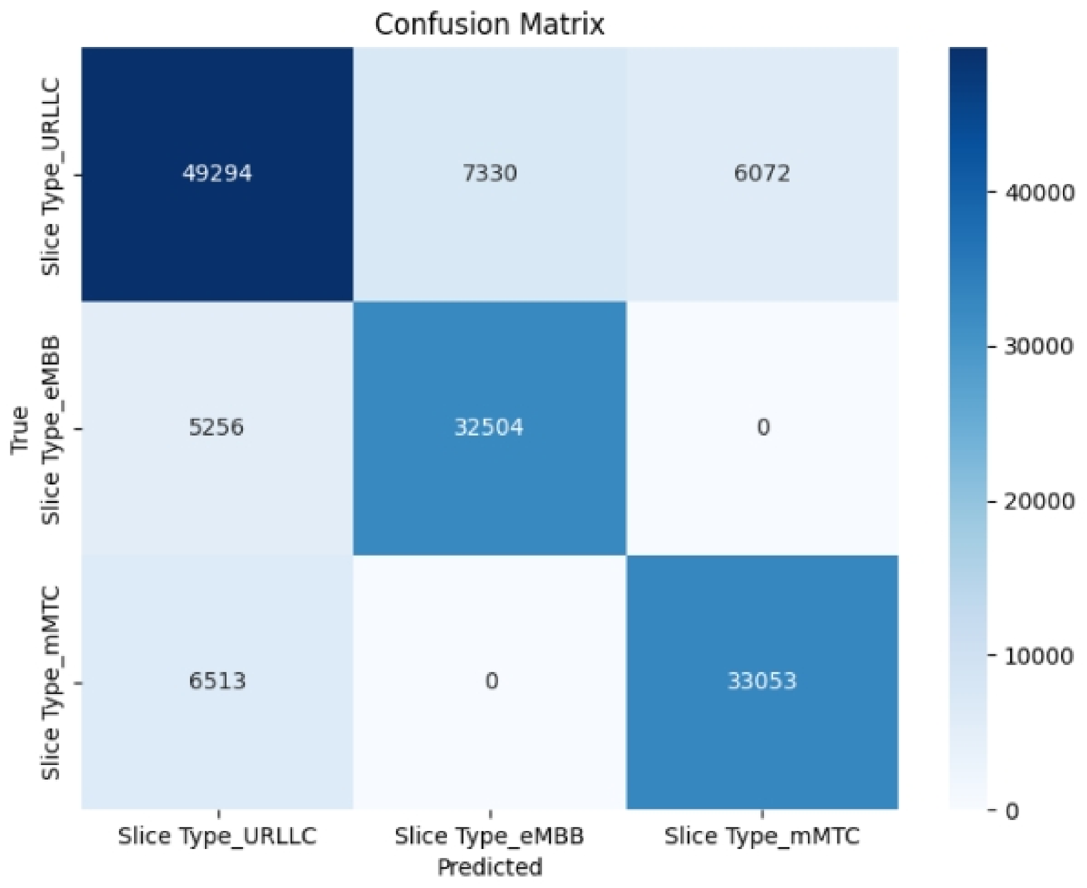


Figure 8: Confusion Matrix

Analysis of the Confusion Matrix

The model exhibits high precision in correctly identifying eMBB, with no misclassifications into mMTC. Significant confusion is observed between URLLC and mMTC, indicating overlapping features or insufficient differentiation by the model. Although mMTC is well-distinguished from eMBB with no misclassifications, there is potential for improvement in distinguishing it from URLLC.

Analysis of the Confusion Matrix

- The model exhibits high precision in correctly identifying eMBB, with no misclassifications into mMTC.
- Significant confusion is observed between URLLC and mMTC, indicating overlapping features or insufficient differentiation by the model.
- Although mMTC is well-distinguished from eMBB with no misclassifications, there is potential for improvement in distinguishing it from URLLC.

Classification Report

The classification report provides a comprehensive analysis of precision, recall, and F1 scores for each class:

- **Slice Type_URLLC:** Precision of 81%, Recall of 79%, and an F1-score of 80%.
- **Slice Type_eMBB:** Precision of 82%, Recall of 86%, and an F1-score of 84%.
- **Slice Type_mMTC:** Precision of 84%, Recall of 84%, and an F1-score of 84%.

These metrics indicate consistent and effective performance across all classes, with the model showing particularly strong results for the eMBB and mMTC classes, as evidenced by their high recall and F1 scores. The slight improvement in recall for the eMBB class from previous results and the balanced precision and recall for the mMTC class highlight the model's enhanced capability in correctly classifying these slice types.

Conclusion

The logistic regression model exhibited strong predictive capabilities across the three network slice types, as evidenced by its high accuracy and excellent AUC scores for each class. The detailed analysis of the confusion matrix and classification metrics from the latest run, which utilized the 'saga' solver and a regularization strength of 0.01, indicates focused improvements, particularly in differentiating between URLLC and mMTC, where cross-misclassifications were previously notable.

Code: https://github.com/AniketFasate27/EECE5644-Intro$_{t}o_{M}L/tree/main/Project$

10 Naive Bayes Classification

The Naive Bayes Classifier is a probabilistic model that assumes the independence of predictor variables given the class label. Despite this assumption, Naive Bayes often performs very well, particularly in high-dimensional datasets. It is commonly used for tasks such as text classification, spam detection, and sentiment analysis.

10.1 Data Loading and Processing

The dataset, stored in `Processed.csv`, was loaded to obtain the features and labels necessary for model training. To streamline computational efficiency and focus on key predictors, features related to 'Packet Loss Rate' and 'Packet Delay Budget' were excluded.

10.2 Model Training

The Naive Bayes model was trained using the `GaussianNB` algorithm from the scikit-learn library, which assumes that feature likelihoods follow a Gaussian distribution. The data was split into training (70%) and testing (30%) sets to validate the model's performance on unseen data.

10.3 Model Performance Metrics

The model's performance was assessed using several key metrics to gauge its predictive accuracy:

- **Accuracy:** The Naive Bayes classifier achieved an accuracy of 82%, indicating a high proportion of correct predictions on the test dataset.

- **ROC-AUC Score:** The ROC-AUC scores were outstanding for each class (URLLC = 0.93, eMBB = 0.94, mMTC = 0.98), highlighting the model's strong classification abilities.
- **F1 Score (Weighted Average):** The F1 score, representing the harmonic mean of precision and recall, was computed using a weighted average to address any class imbalance, resulting in a score of 0.847.
- **Confusion Matrix:** The confusion matrix showed high true positive rates for each class with minimal misclassifications, indicating the model's reliability.

10.4 Results

- **Accuracy:** 0.82
- **ROC AUC Score (One-vs-Rest):** 0.95
- **F1 Score Weighted:** 0.8477
- **Confusion Matrix:**

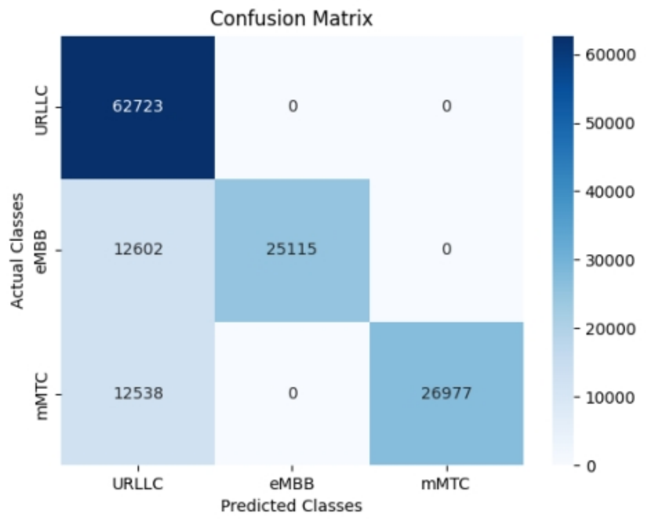


Figure 9: Confusion Matrix

10.5 ROC Curve

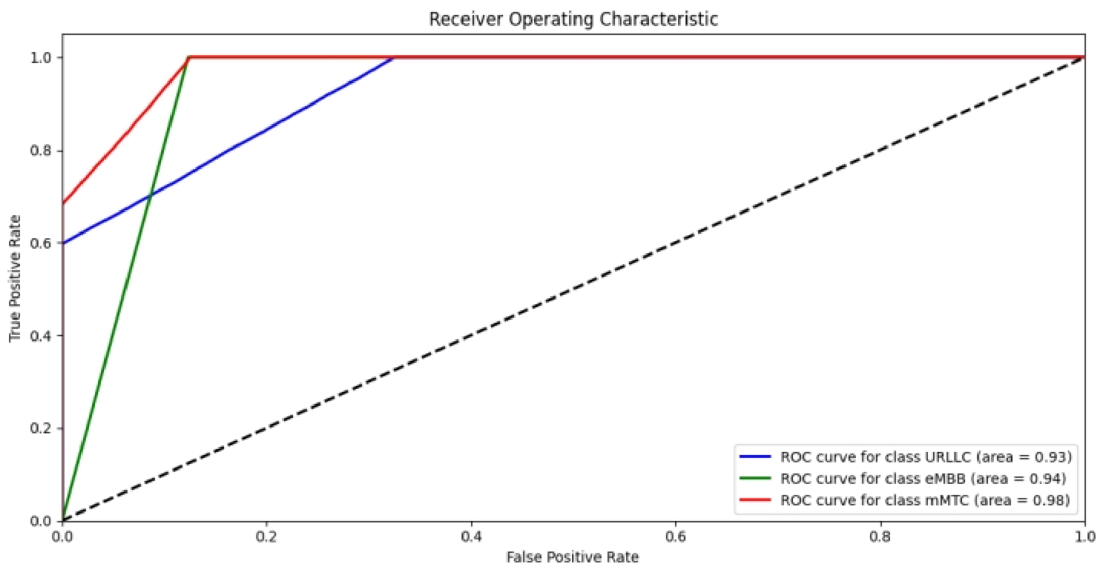


Figure 10: ROC Curve

Code: https://github.com/AniketFasate27/EECE5644-Intro$_to_ML/tree/main/Project$

11 Linear Discriminant Analysis (LDA) Classification

To mitigate overfitting and reduce computational costs, features are projected onto a lower-dimensional space with high class separability using the Linear Discriminant Analysis (LDA) classification and dimensionality reduction technique. LDA is particularly useful when there are differences in within-class frequencies and generally performs more reliably than other methods.

11.1 Data Loading and Processing

The dataset was loaded from **processed_data_without_packet_loss_delay.csv**, which contains the pre-processed data necessary for model training. As previously mentioned, the Packet Loss Rate and Packet Delay Budget values were excluded from the features.

11.2 Model Training

The model was trained using the `LinearDiscriminantAnalysis` class from the scikit-learn library. Principal Component Analysis (PCA) was initially employed to visualize the data distribution in a two-dimensional space before applying LDA. LDA was then used to identify a subspace that optimizes class separation after establishing a reasonable class separation.

11.3 Model Performance Metrics

- **Accuracy:** The percentage of correct predictions on the test set provided a quick measure of the model's performance.
- **F1 Score:** Given the imbalanced class distribution, F1 scores were calculated for each class and then weighted to provide a single score that accounts for both precision and recall.
- **ROC-AUC:** ROC-AUC scores were calculated for each class against the others, and their weighted average was computed.
- **Confusion Matrix:** This was analyzed to understand how well the model performed across various classes, providing a detailed breakdown of correct and incorrect classifications for each class.

11.4 Results

- **Optimum Model Parameters:** LDA did not require the same level of parameter optimization as other classifiers due to its simplicity.
- **Accuracy:** 0.8202
- **F1 Score (Weighted Average):** 0.81

11.5 Confusion Matrix

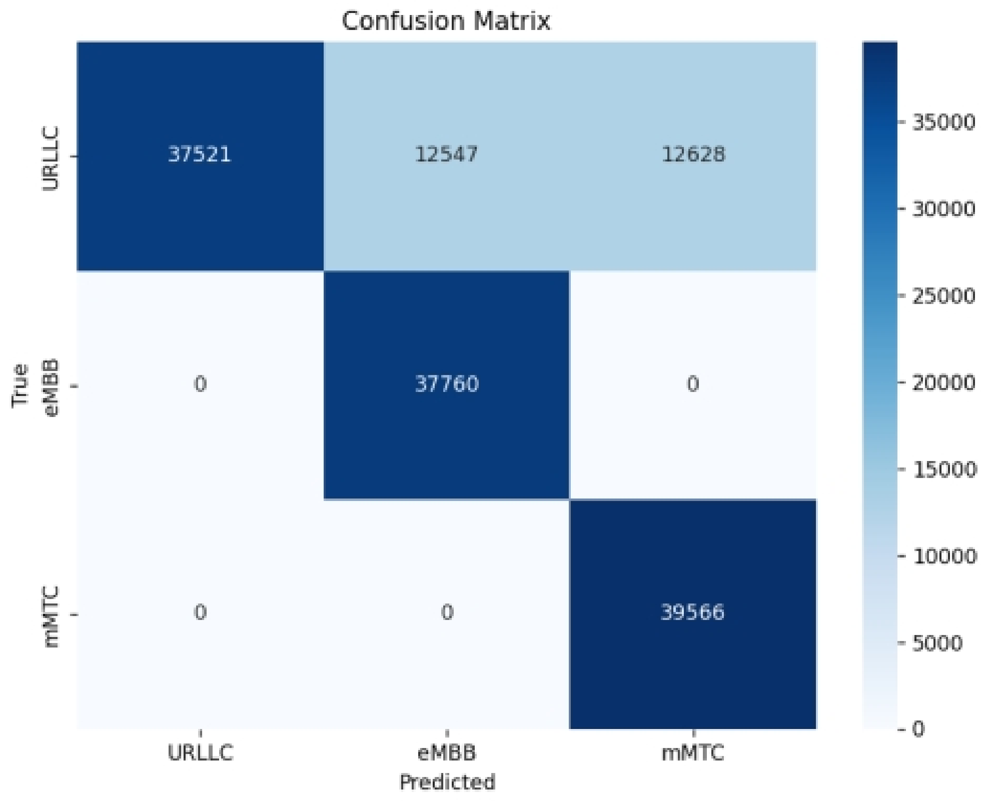


Figure 11: Confusion Matrix

11.6 Projection After LDA

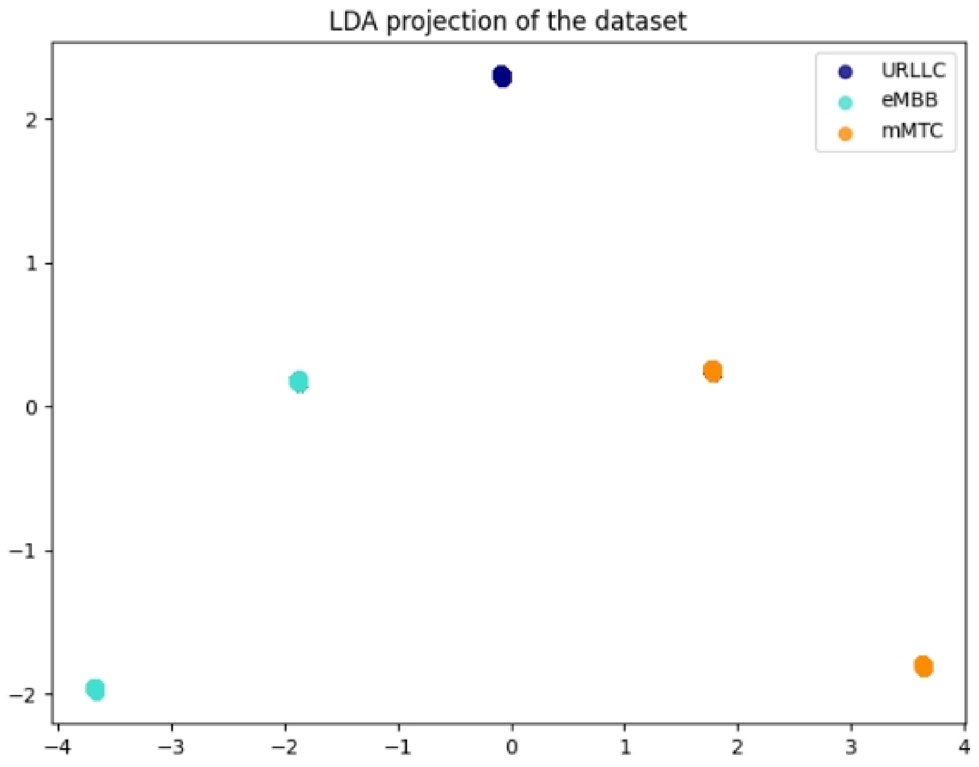


Figure 12: Projection After LDA

11.7 ROC Curve

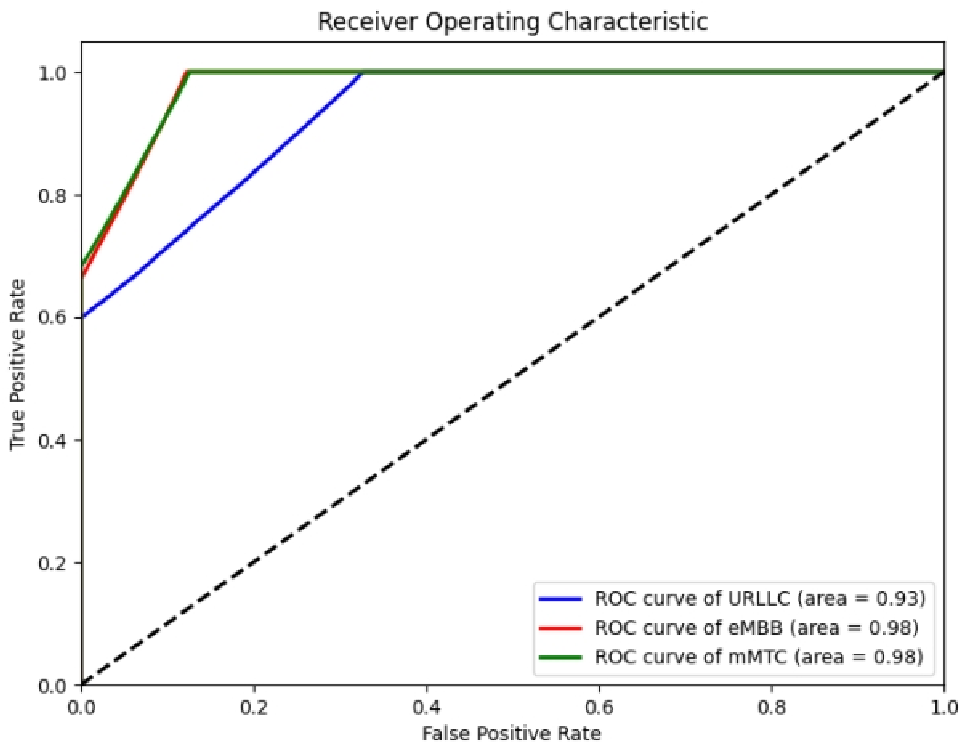


Figure 13: ROC Curve

12 Conclusion

The following table includes the performance of all the models we have used in this project.

[HTML]FFFF00 **Classifier Used**	**Accuracy**	**F1 score**	**ROC AUC score**
Random Forest	82.12%	0.8195	0.9469
Naive Bayes	82%	0.84766	0.95
Logistic Regression	82.02%	0.8200	0.958
Linear Discriminant Analysis	82.02%	0.8100	0.95

13 Future Work

In this study, we applied basic machine learning classifiers to the dataset and assessed their performance. According to our analysis, the Random Forest Classifier outperformed other ML models, while the Deep Learning Neural Network achieved the highest overall performance. Moving forward, exploring Neural Networks further could potentially yield even higher accuracy. Given the limited availability of datasets specific to this topic, the dataset used in this project may not be comprehensive enough, as it lacks a sufficient number of Key Performance Indicators (KPIs) and does not provide a high level of accuracy. Therefore, creating our own datasets using simulators and Software-Defined Networks (SDNs) could be beneficial.

14 References

1. A. Thantharate, R. Paropkari, V. Walunj and C. Beard, *"DeepSlice: A Deep Learning Approach towards an Efficient and Reliable Network Slicing in 5G Networks"* 2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON), New York, NY, USA, 2019, pp. 0762-0767, doi: 10.1109/UEMCON47517.2019.8993066.

2. A. Thantharate, R. Paropkari, V. Walunj, C. Beard and P. Kankariya, *"Secure5G: A Deep Learning Framework Towards a Secure Network Slicing in 5G and Beyond"* 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2020, pp. 0852-0857, doi: 10.1109/CCWC47524.2020.9031158.

3. Scikit.learn API documentation