Indian Institute of Technology, Dharwad

CS209 : Artificial Intelligence
And
CS214 : Artificial Intelligence Laboratory
Project Report :

Comprehensive Analysis of Synthetic Speech Detection Using RawNet2 Embeddings

Course Instructor:
Dr. Dileep A.D.

Mentor Name:
Rishith Sadashiv T N

Submitted by:
Yash Halbhavi [CS23BT069]
Sailaja [CS23BT063]
Dudekula Abbas [CS23BT044]
Bholendra Tripathi [225100008]

Contents

List of Figures

List of Tables

Abstract

This research presents a comprehensive analysis of synthetic speech detection using RawNet2 embeddings. We evaluate 13 different machine learning models on the AVSpoof 2019 LA dataset, focusing on both standard classification performance and techniques to address class imbalance. The study explores various data preprocessing methods including oversampling, undersampling, class weight balancing, and dimensionality reduction through PCA. Our results indicate that careful preprocessing significantly enhances detection performance, with ensemble methods like XGBoost and Random Forest demonstrating superior capability in identifying synthetic speech. This work contributes to the ongoing development of robust defenses against increasingly sophisticated audio spoofing attacks.

1 Introduction

The rapid advancements in AI-driven speech synthesis systems have led to increased risks of voice spoofing attacks. Modern text-to-speech (TTS) and voice conversion (VC) technologies can produce highly realistic synthetic speech that is increasingly difficult to distinguish from genuine human speech. This poses significant security concerns for voice authentication systems and creates avenues for potential fraud, misinformation, and identity theft.

Recent deep learning approaches have shown promising results in detecting synthetic speech. Among these, the RawNet2 architecture has demonstrated exceptional performance by directly processing raw waveform inputs rather than relying on engineered acoustic features. RawNet2 produces embeddings that capture subtle artifacts present in synthetic speech that may not be apparent in traditional spectral features.

This research explores the effectiveness of various machine learning algorithms in detecting synthetic speech using RawNet2 embeddings. We evaluate 13 different models ranging from traditional classifiers like Logistic Regression and Support Vector Machines to advanced ensemble methods and neural networks. Our analysis particularly focuses on addressing the common challenge of class imbalance in spoofing detection datasets, where genuine speech samples typically outnumber synthetic ones by a significant margin.

The objectives of this study are to:

- Evaluate the performance of multiple machine learning models on RawNet2 embeddings for synthetic speech detection
- Address class imbalance through various preprocessing techniques
- Analyze the impact of dimensionality reduction on model performance
- Provide comprehensive comparisons and insights into optimal model selection

The findings of this research contribute to the development of robust anti-spoofing systems that can reliably detect increasingly sophisticated synthetic speech attacks.

2 Data Handling

This section covers the preprocessing steps applied to the AVSpoof 2019 LA dataset and the extraction of RawNet2 embeddings that serve as features for our classification models.

2.1 Dataset Overview

The AVSpoof 2019 LA dataset contains three splits: training, validation, and testing. Each audio sample is categorized as either genuine human speech or synthetic speech generated by various TTS and VC systems. Table 1 shows the distribution of samples across these splits.

Table 1: Class Distribution Across Dataset Splits

Split	Genuine Speech	Synthetic Speech	Total Samples
Training	22,800	2,580	25,380
Validation	22,296	2,548	24,844
Testing	63,882	7,355	71,237

Table 2: Unique Values for Different Data Sets

Data Set	Unqiue Count	Total Count
Training Data	25380	25380
Validation Data	24844	24844
Testing Data	71237	71237

As evident from Table 1, there is a significant class imbalance in all three splits, with genuine speech samples outnumbering synthetic speech samples by approximately 9:1. This imbalance presents a challenge for classification models, as they may become biased toward the majority class. And from Table 2, we infer that all values are unique and no duplicates.

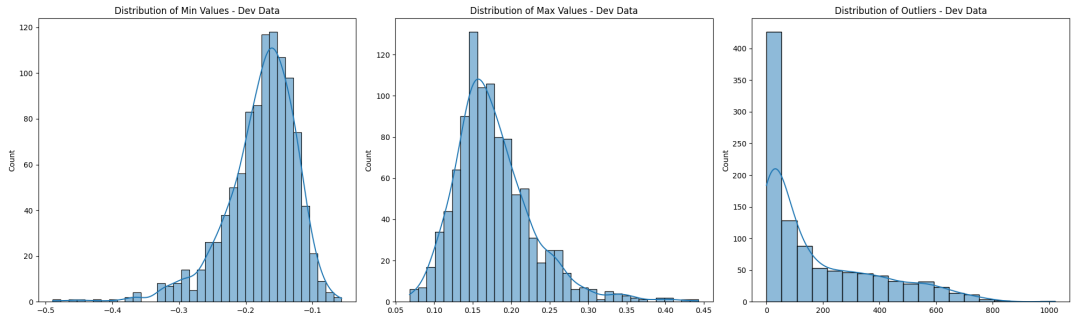


Figure 1: Distribution of Dev (Validation) Data

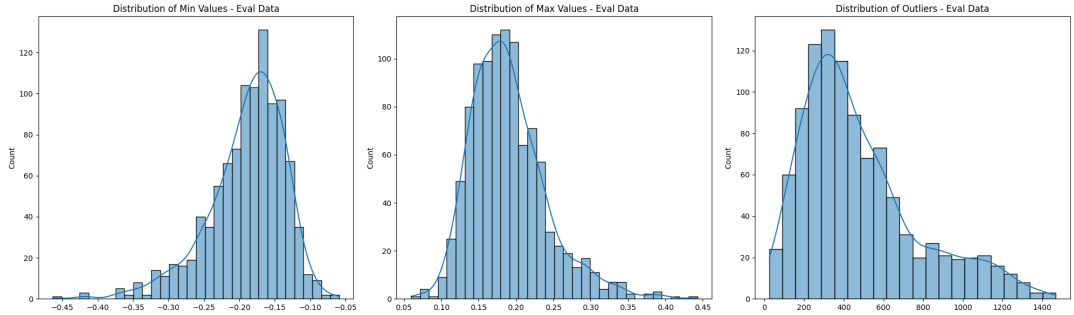


Figure 2: Distribution of Eval (Testing) Data

2.2 RawNet2 Embedding Extraction

RawNet2 is a deep neural network architecture designed to process raw audio waveforms directly. For this study, we utilized a pre-trained RawNet2 model to extract 1024-dimensional embeddings from each audio sample. These embeddings capture the subtle characteristics that differentiate genuine speech from synthetic speech.

The extracted embeddings serve as input features for all classification models evaluated in this study. The high dimensionality of these embeddings (1024 features) makes them potentially suitable for dimensionality reduction techniques like PCA, which we explore later in Section 4.

2.3 Initial Data Preparation

For our initial experiments, we used the standard train/validation/test splits provided in the dataset. The models were trained on the training set, hyperparameters were optimized using the validation set, and final performance was evaluated on the test set. No additional preprocessing beyond standard feature scaling was applied in this initial phase to establish baseline performance metrics.

For feature scaling, we applied standard normalization (z-score normalization) to ensure that all features have zero mean and unit variance:

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

where μ is the mean and σ is the standard deviation of the training data.

3 Techniques for Enhancing Model Performance

This section outlines several methodologies and techniques that can be applied to improve the performance of classification models. Emphasis is placed on both evaluation metrics and hyperparameter tuning strategies that address challenges such as class imbalance and overfitting.

3.1 Evaluation Metrics and Their Importance

While overall accuracy is a useful metric, it can be misleading when dealing with imbalanced datasets. In such scenarios, the F1-score is critical, especially for the minority class. To gain a comprehensive understanding of model performance, the following approaches are recommended:

- **Accuracy Metrics:** Provide a baseline understanding of the proportion of correct classifications.
- **F1-Score Metrics:** Combine precision and recall to deliver a balanced metric that is particularly sensitive to imbalanced classes.

3.2 ROC and Precision-Recall Analysis

For imbalanced classification tasks, it is beneficial to complement traditional ROC analysis with Precision-Recall (PR) curves:

- **ROC Curves:** Useful for visualizing the trade-off between true positive and false positive rates.
- **Precision-Recall Curves:** Provide clearer insights into the performance on the minority class. The area under the Precision-Recall curve (AUPRC) is a key indicator of a model's ability to detect rare events while managing false positive rates.

3.3 Hyperparameter Tuning for Enhanced Performance

Optimizing hyperparameters plays a crucial role in refining model behavior. The following parameters are among the most influential:

3.3.1 Random Forest: Tree Depth Adjustment

Altering the maximum depth of trees in a Random Forest model can significantly impact its capability to capture complex data structures. Key considerations include:

- **Increased Depth:** Enhances the model's ability to recognize intricate patterns.
- **Overfitting Concerns:** Excessive depth may lead to overfitting, particularly on the majority class.

3.3.2 Neural Network Architecture Design

The architecture of a neural network, including the number of layers and neurons, is pivotal for achieving optimal performance:

- **Deeper Architectures:** Can capture more abstract features, improving overall performance.
- **Regularization:** Techniques such as dropout are necessary to prevent overfitting as complexity increases.

3.3.3 Learning Rate Optimization

The learning rate is a critical hyperparameter for gradient-based methods and ensemble algorithms such as boosting:

- **Higher Learning Rates:** Can speed up convergence but may lead to suboptimal solutions or divergence.
- **Lower Learning Rates:** Provide a more stable convergence path but typically require more iterations.

3.4 Integrating the Techniques

A holistic approach to performance enhancement involves:

- Balancing multiple evaluation metrics (accuracy, F1-score, ROC, and PR curves) to guide improvements.
- Iteratively tuning hyperparameters and testing various model architectures.
- Continuously monitoring the impact of these adjustments, particularly in scenarios with class imbalance.

By systematically applying these techniques, practitioners can better tailor their models to the inherent challenges of their datasets, thereby improving both the robustness and effectiveness of their classification systems.

4 Extensive Data Handling

This section explores techniques to improve model performance through advanced preprocessing methods. We focus on addressing class imbalance and dimensionality reduction, which are critical challenges for our dataset.

4.1 Addressing Class Imbalance

We implemented four strategies to address the class imbalance issue:

4.1.1 Using Validation Data for Training

By combining the training and validation sets, we increased the number of samples available for training, particularly for the minority class (synthetic speech). This approach provides a larger and more diverse training set at the cost of having less data for hyperparameter tuning.

4.1.2 Oversampling using SMOTE

Synthetic Minority Over-sampling Technique (SMOTE) creates synthetic samples of the minority class by interpolating between existing samples. This increases the representation of synthetic speech in the training data without simply duplicating existing samples.

Algorithm 1 SMOTE Algorithm

1: **for** each sample x_i in minority class **do**
2: Find k nearest neighbors in minority class
3: Randomly select one of the k nearest neighbors x_j
4: Generate synthetic sample: $x_{new} = x_i + \lambda \times (x_j - x_i)$ where $\lambda \in [0, 1]$
5: **end for**

4.1.3 Undersampling

Random undersampling reduces the number of majority class samples (genuine speech) to create a balanced dataset. While this addresses class imbalance, it discards potentially useful information from the majority class.

We implemented random undersampling to match the number of genuine speech samples to the number of synthetic speech samples in the training set.

4.1.4 Class Weight Balancing

Class weight balancing assigns higher weights to the minority class during model training. This approach modifies the loss function to penalize misclassifications of the minority class more severely.

We assigned weights inversely proportional to class frequencies in the training data:

$$weight_{class} = \frac{n_{samples}}{n_{classes} \times n_{samples_{class}}} \tag{2}$$

4.2 Dimensionality Reduction Using PCA

Principal Component Analysis (PCA) was applied to reduce the dimensionality of RawNet2 embeddings. This not only addresses the curse of dimensionality but also potentially removes noise and redundancy in the features.

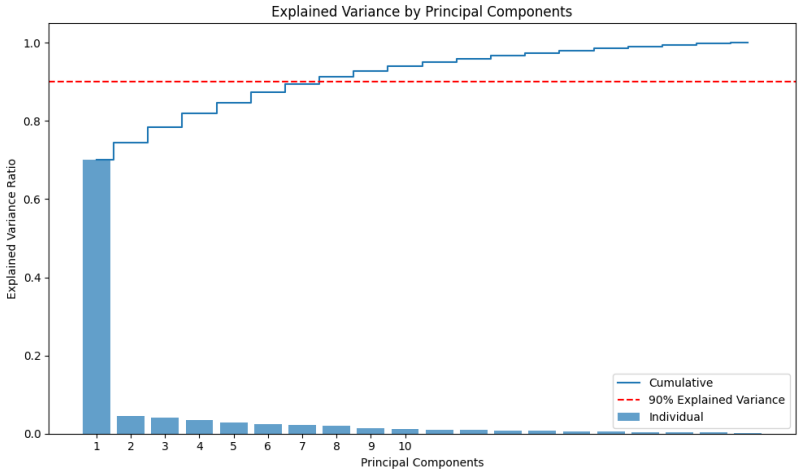


Figure 3: Distribution of PCA

The scree plot in Figure 3 shows the percentage of variance explained by each principal component. Based on this analysis, we selected the top 100 components that collectively explain approximately 95% of the variance in the data.

Figure 4: Visualization of Data in First Two Principal Components

Figure 4 shows the projection of samples onto the first two principal components. The separation between genuine and synthetic speech samples in this reduced space indicates that PCA preserves the discriminative information required for classification.

5 Models and their Results

This section showcases the results obtained from all 13 models without extensive preprocessing. Each model was trained on the RawNet2 embeddings extracted from the training set and evaluated on the test set. Standard scaling was applied to the features, but no class imbalance techniques were implemented at this stage.

5.1 Logistic Classification

5.1.1 Logistic Classification without processing

Logistic classification is a type of statistical model used to predict the probability of a binary outcome — i.e., one that has only two possible values, such as:0 or 1.It is a linear model.Despite its simplicity, it often serves as a strong baseline for binary classification tasks. For our implementation, we used L2 regularization with a regularization strength parameter C=1.0 and the 'lbfgs' solver, which is efficient for the dataset size.

Table 3: Performance Metrics

Training		Validation		Test (Evaluation)	
Accuracy	0.8985	Accuracy	0.8970	Accuracy	0.8889
F1 Score	0.0100	F1 Score	0.0062	F1 Score	0.0421
Recall	**0.0050**	Recall	**0.0031**	Recall	**0.0237**
EER	0.2772	EER	0.3097	EER	0.4180
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[22791 9]		[22277 19]		[63150 732]	
[2567 13]		[2540 8]		[7181 174]	

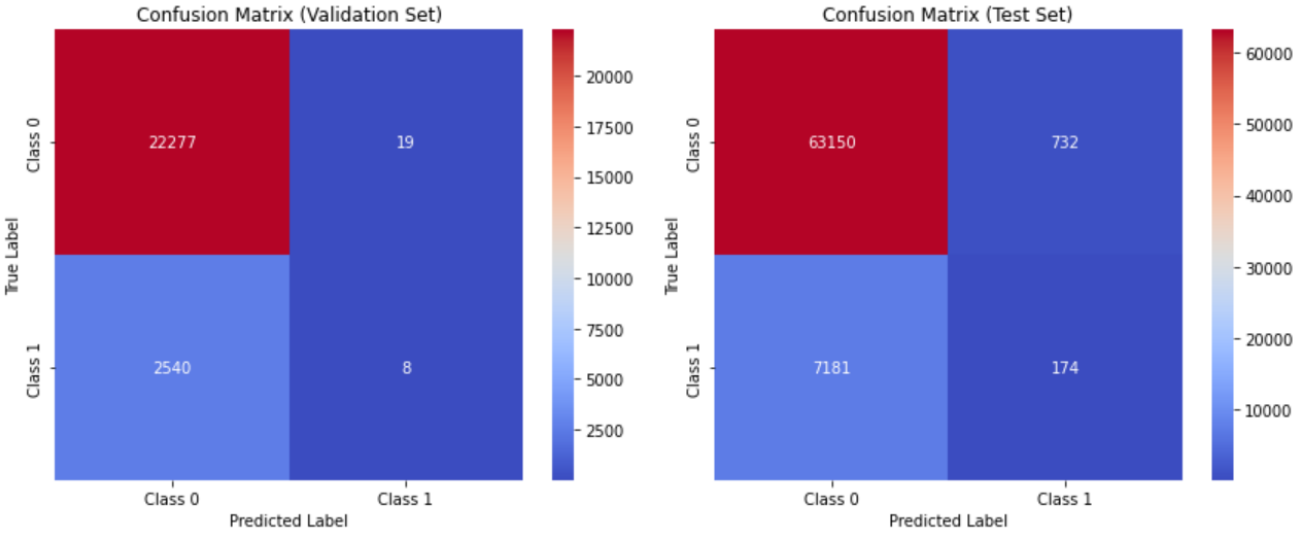


Figure 5: Visual Analysis of Confusion Matrices

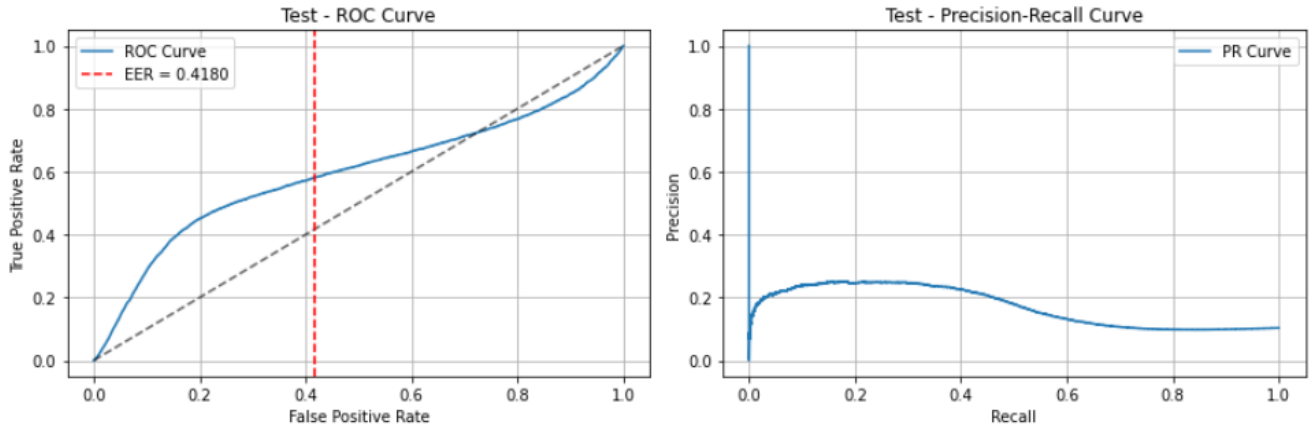


Figure 6: ROC and precision-recall Curve

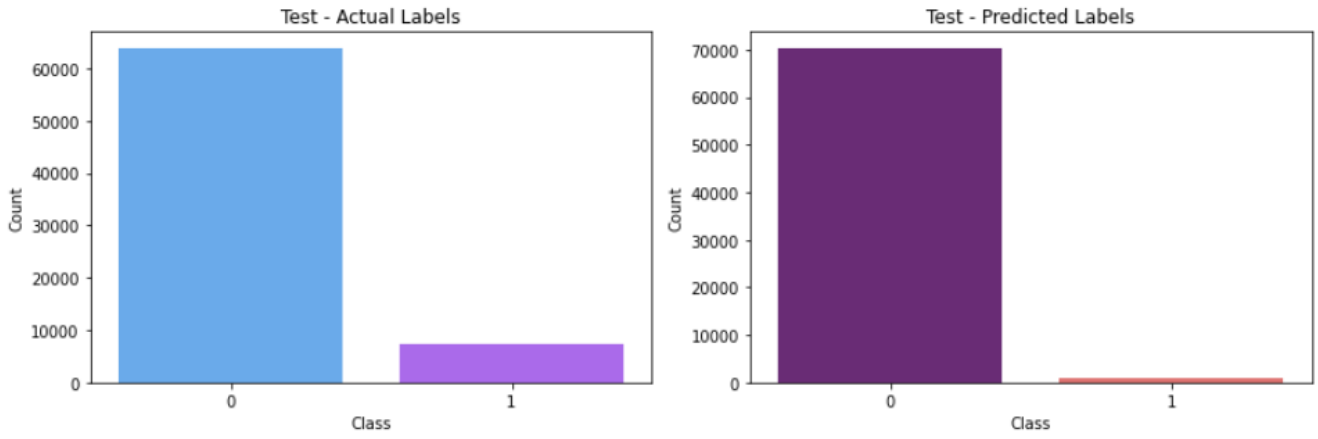


Figure 7: Actual vs Predicted

5.1.2 Logistic Classification (Processed)

Over Sampling(SMOTE)

Table 4: Performance Metrics (Training, Validation, Test)

Training		Validation		Test (Evaluation)	
Accuracy	0.7536	Accuracy	0.6992	Accuracy	0.4590
F1 Score	0.7673	F1 Score	0.3267	F1 Score	0.1982
Recall	0.8124	Recall	0.7115	Recall	0.6477
EER	0.2487	EER	0.2971	EER	0.4212
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[15843 6957]		[15559 6737]		[27931 35951]	
[4277 18523]		[735 1813]		[2591 4764]	

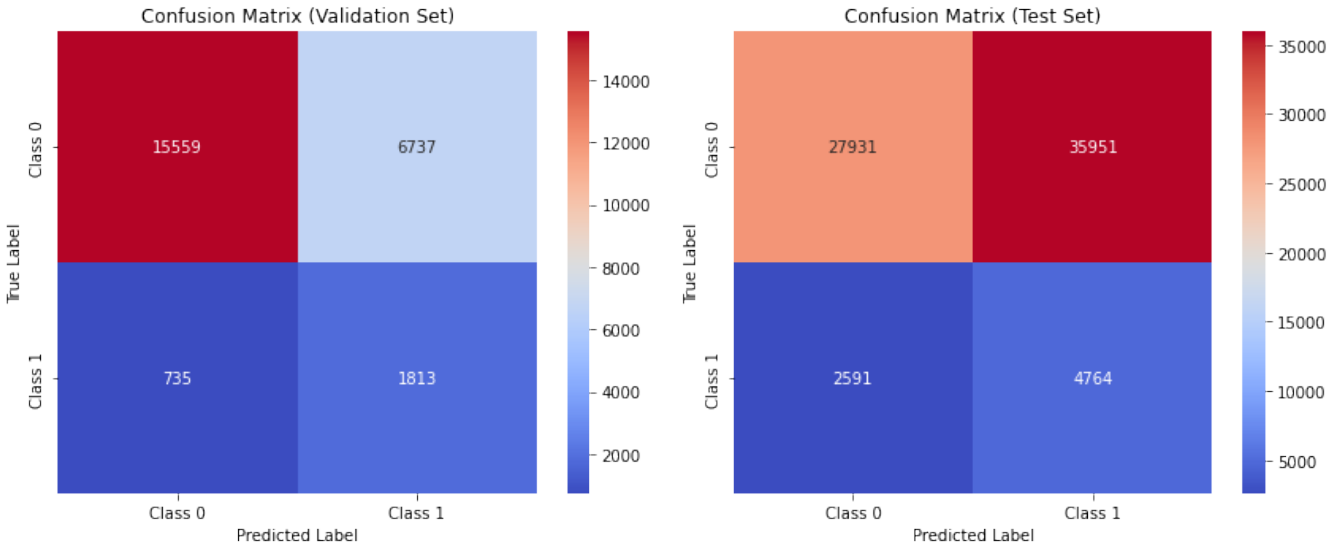


Figure 8: Visual Analysis of Confusion Matrices

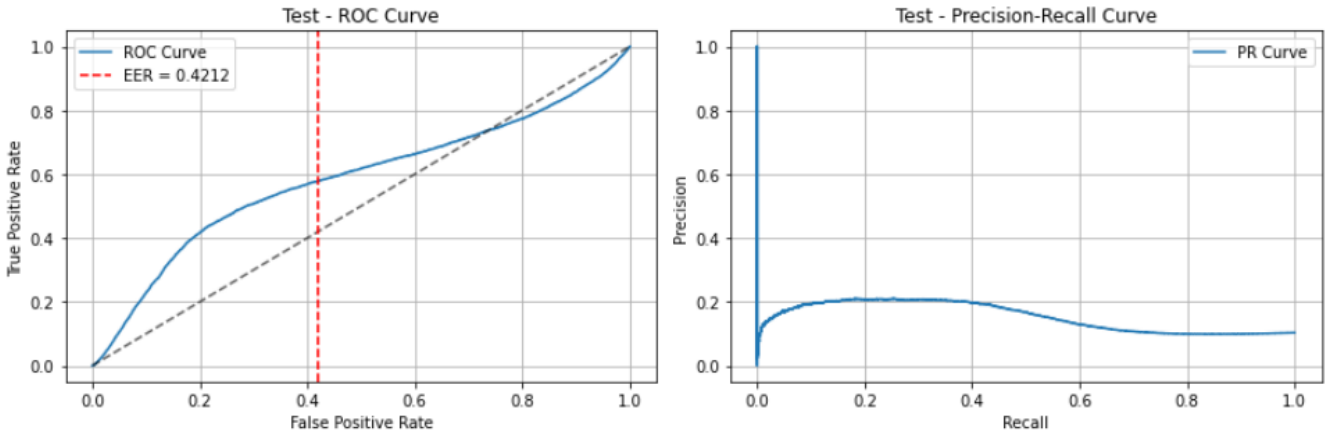


Figure 9: ROC and Precison-recall Curve

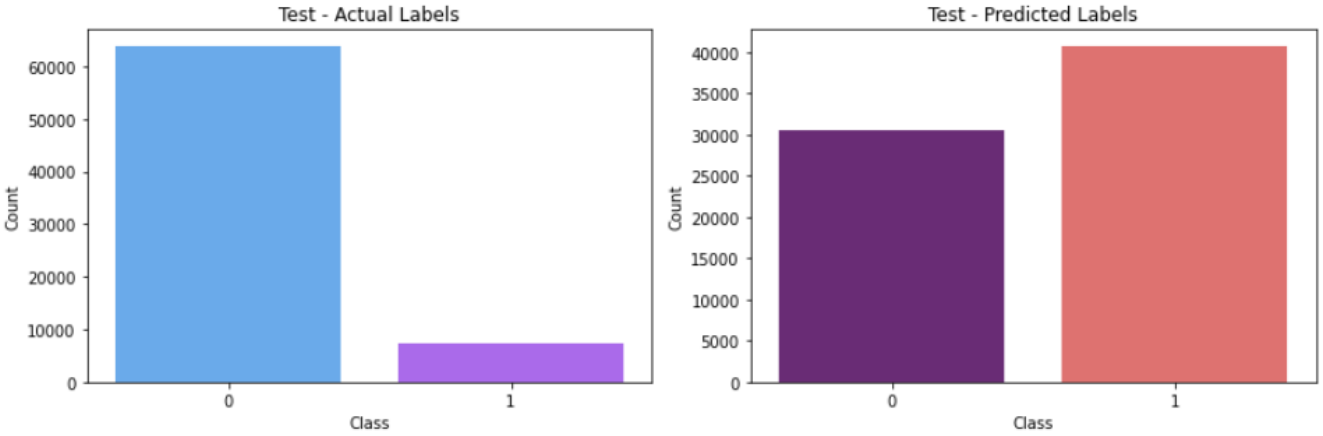


Figure 10: Actual vs predicted

Under Sampling	**label 0**	2580
	label 1	2580

Table 5: Performance Metrics (Training, Validation, Test)

Training		Validation		Test (Evaluation)	
Accuracy	0.7302	Accuracy	0.6553	Accuracy	0.5010
F1 Score	0.7449	F1 Score	0.3015	F1 Score	0.2072
Recall	0.7876	Recall	0.7253	Recall	0.6317
EER	0.2771	EER	0.3097	EER	0.4153
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[1736 844]		[14432 7864]		[31042 32840]	
[548 2032]		[700 1848]		[2709 4646]	

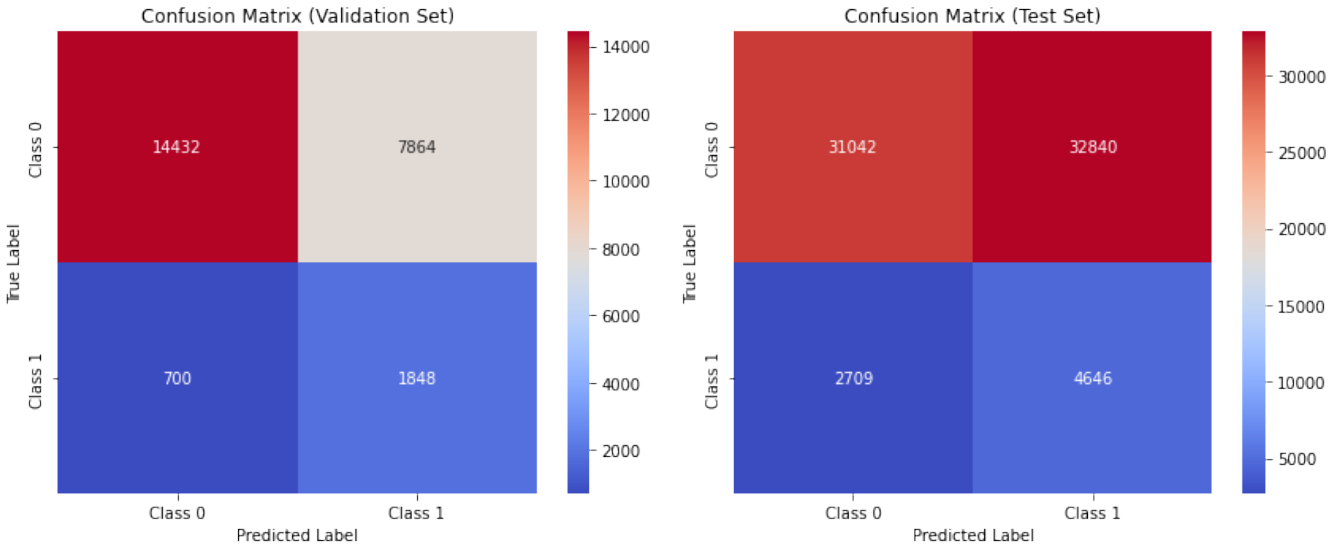


Figure 11: Visualization of Confusion matrices

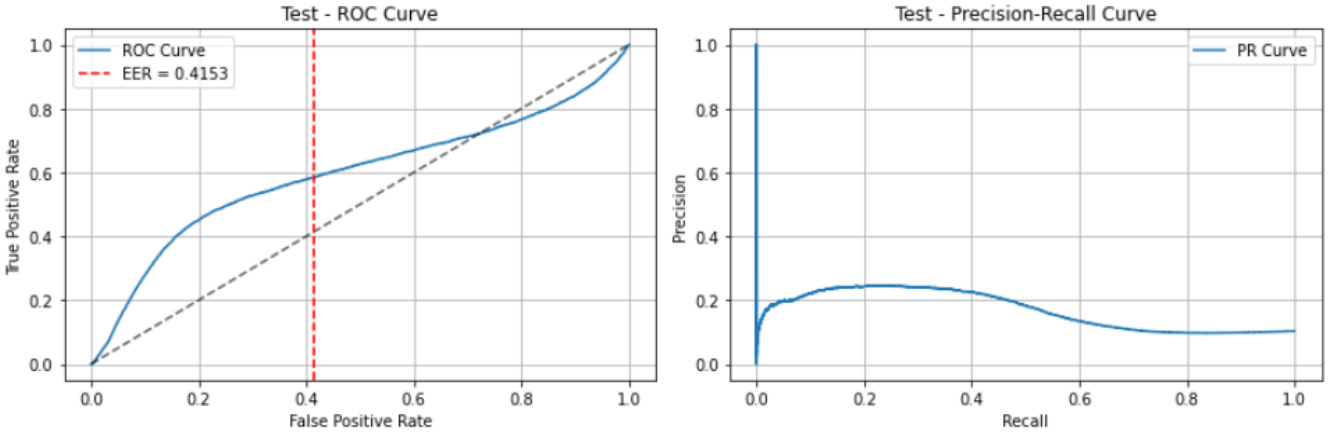


Figure 12: ROC and Precision-Recall Curve

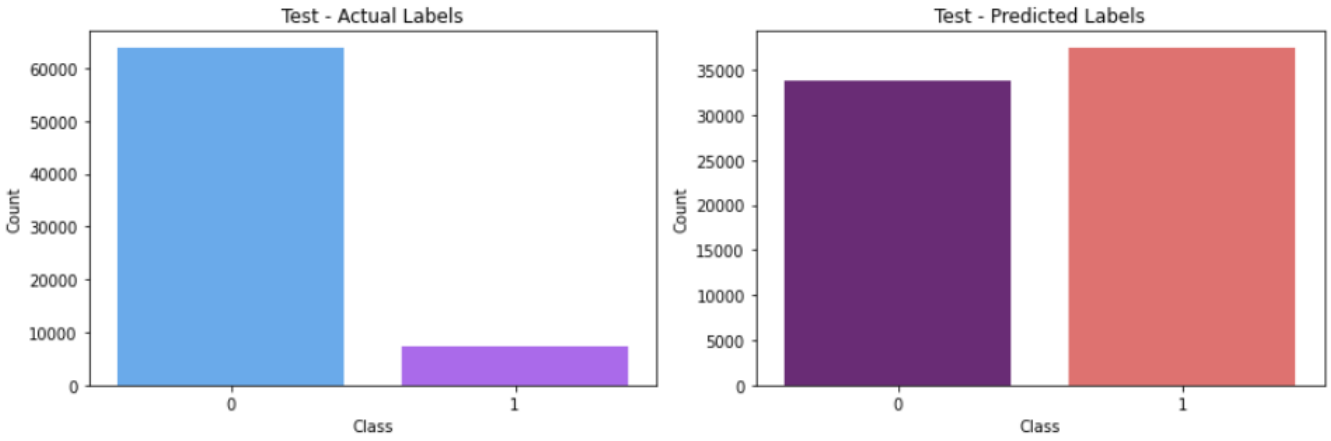


Figure 13: Actual vs predicted Labels

Weight Balanced

Table 6: Performance Metrics (Training, Validation, Test)

Training		Validation		Test (Evaluation)	
Accuracy	0.6928	Accuracy	0.6900	Accuracy	0.4669
F1 Score	0.3467	F1 Score	0.3245	F1 Score	0.1983
Recall	0.8019	Recall	0.7261	Recall	0.6387
EER	0.2648	EER	0.2983	EER	0.4211
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[15515 7285]		[15292 7004]		[28560 35322]	
[511 2069]		[698 1850]		[2657 4698]	

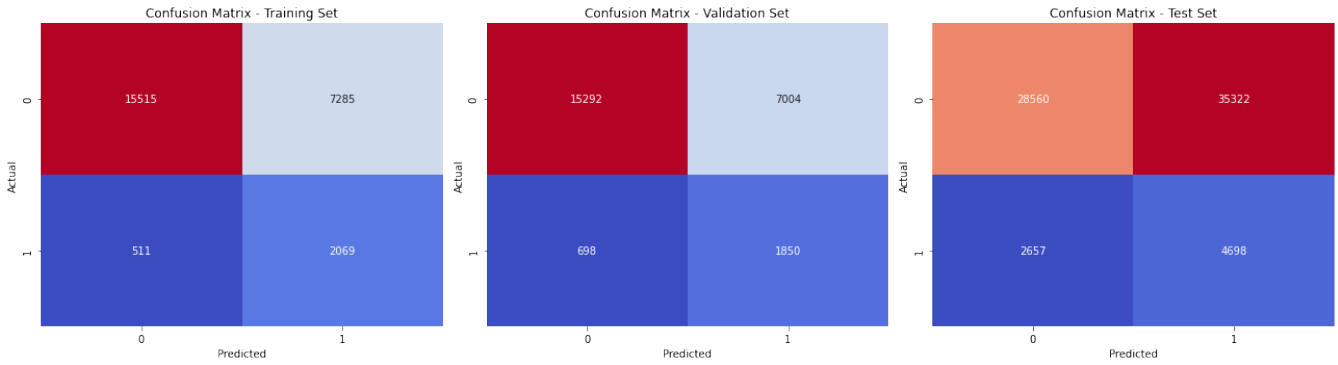


Figure 14: Visualization of Confusion Matrices

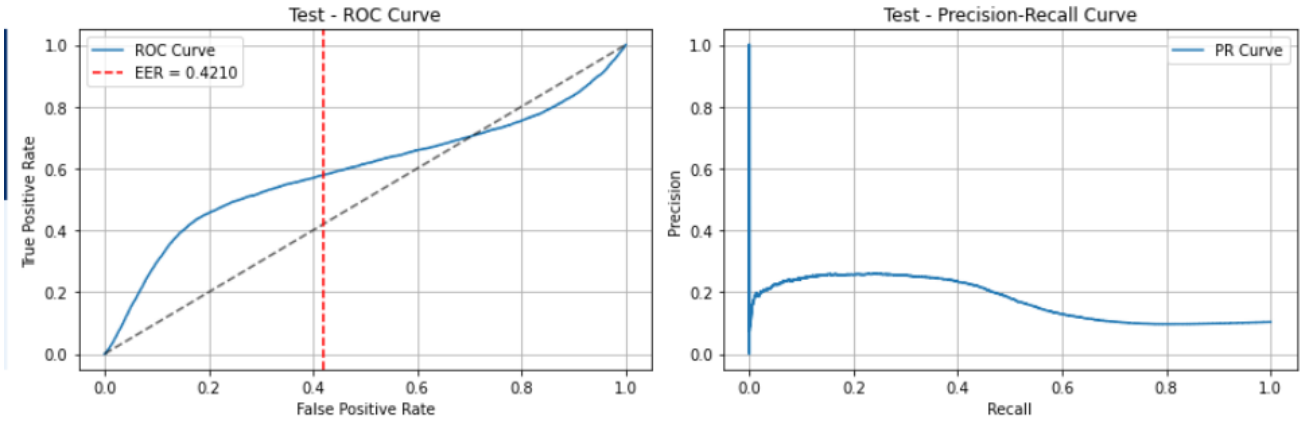


Figure 15: ROC and Precision-Recall Curve

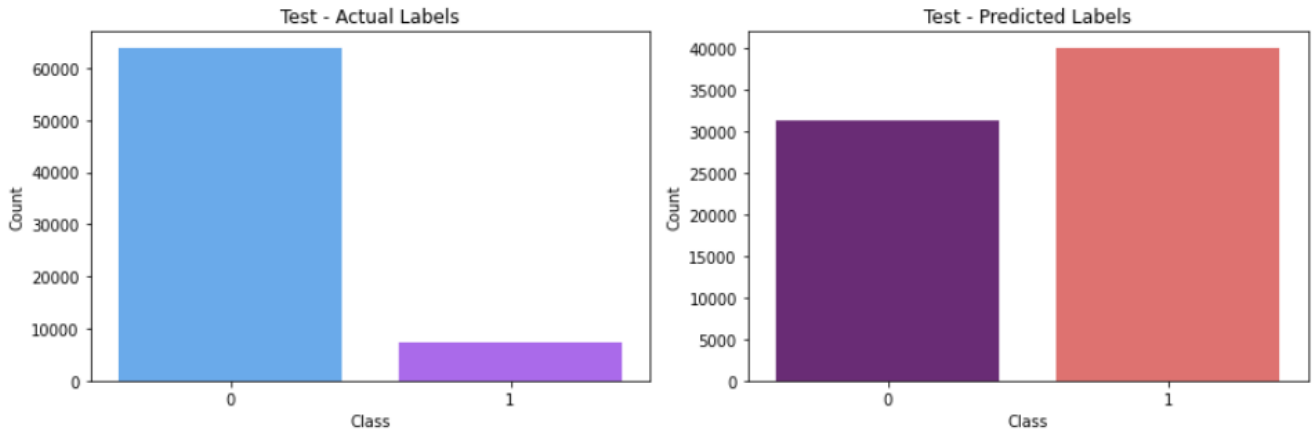


Figure 16: Actual vs Predicted

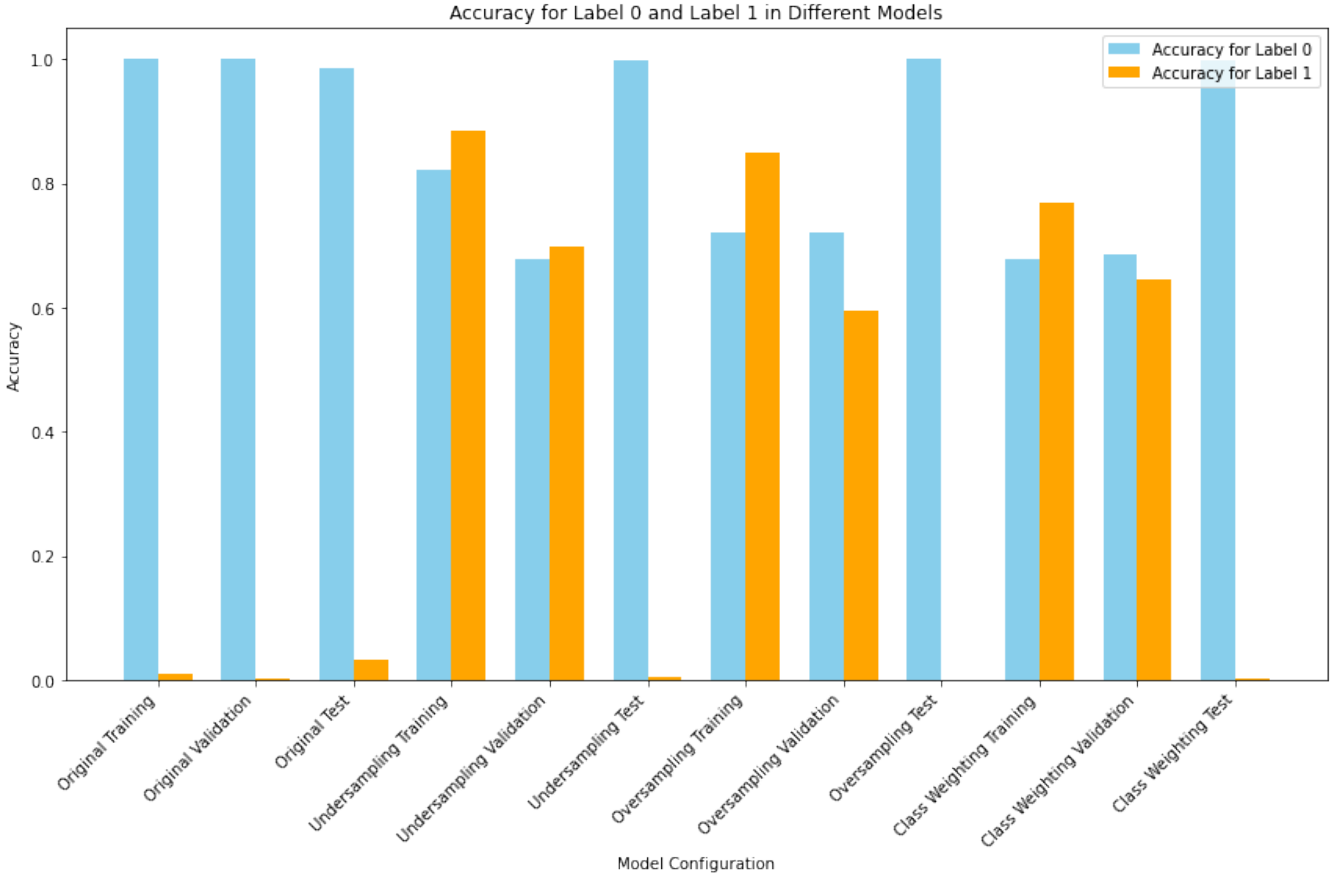


Figure 17: Accuracy for label 0 and 1

5.1.3 Conclusion

- Recall is extremely low across all sets (Training: 0.5%, Validation: 0.31%, Test: 2.37%).
- Indicates a heavily imbalanced dataset — the model is biased toward the majority class (likely negative) and fails to identify positive instances.
- High accuracy (˜89%) is misleading because most predictions are for the dominant class.

5.1.4 Under-sampling

- Improved minority class detection.
- Trade-off: Accuracy and precision dropped.

- EER is still rising. Model generalization is not great — although it detects positives better, it's misclassifying many negatives as positives on the test set.

5.1.5 Over-sampling (SMOTE)

- Technique used to handle imbalanced datasets by synthetically generating new samples for the minority class instead of duplicating them.
- SMOTE helps the model learn generalizable patterns for the minority class better than random under-sampling.
- Slight drop in F1 and accuracy due to potential overfitting to synthetic samples.

5.1.6 Class Weight Balancing

- A technique used to handle class imbalance during model training.
- Tells the model to penalize misclassification of the minority class more heavily, helping it learn patterns from both classes more effectively.
- Generalization across validation and test sets suggests the model isn't overfitting.
- Accuracy and F1 score decreased, especially on test data (Accuracy: 46.69%, F1: 0.1983), due to an increase in false positives (many majority class samples predicted as positive).

Logistic regression is a linear model. It assumes that there is a linear relationship between the input features and the log-odds of the target class.
In many real-world datasets, the relationship between the features and the target class may be non-linear, meaning that a straight line cannot effectively separate the classes.
It may fail to capture complex patterns that distinguish the two classes. This can lead to a poor fit and low performance, particularly for the minority class (Label 1 in the dataset)

5.2 Decision Tree

The F1 score, Recall & Precision calculated are weighted

5.2.1 Original Dataset

Table 7: Decision Tree Performance on Original Data

Training		Validation		Test	
Accuracy	0.863	Accuracy	0.739	Accuracy	0.577
Precision	0.941	Precision	0.836	Precision	0.831
Recall	0.863	Recall	0.739	Recall	0.577
F1 Score	0.885	F1 Score	0.779	F1 Score	0.660
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[19343 3457]		[17453 4843]		[37327 26555]	
[12 2568]		[1650 898]		[3574 3781]	

5.2.2 SMOTE Oversampled Dataset

Table 8: Decision Tree Performance on SMOTE Data

Training		Validation		Test	
Accuracy	0.904	Accuracy	0.670	Accuracy	0.686
Precision	0.907	Precision	0.844	Precision	0.796
Recall	0.904	Recall	0.670	Recall	0.686
F1 Score	0.904	F1 Score	0.733	F1 Score	0.735
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[19638 3162]		[15383 6913]		[47976 15906]	
[1202 21598]		[1286 1262]		[6443 912]	

5.2.3 Undersampled Dataset

Table 9: Decision Tree Performance on Undersampled Data

Training		Validation		Test	
Accuracy	0.555	Accuracy	0.612	Accuracy	0.817
Precision	0.556	Precision	0.836	Precision	0.806
Recall	0.555	Recall	0.612	Recall	0.817
F1 Score	0.553	F1 Score	0.688	F1 Score	0.812
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[1601 979]		[13931 8365]		[57811 6071]	
[1319 1261]		[1275 1273]		[6973 382]	

5.2.4 PCA Dataset

Table 10: Decision Tree Performance with PCA

Training		Validation		Test	
Accuracy	0.874	Accuracy	0.742	Accuracy	0.814
Precision	0.937	Precision	0.830	Precision	0.810
Recall	0.874	Recall	0.742	Recall	0.814
F1 Score	0.892	F1 Score	0.780	F1 Score	0.812
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[19725 3075]		[17657 4639]		[57469 6413]	
[131 2449]		[1770 778]		[6811 544]	

	Train EER	**Dev EER**	**Test EER**
Original	0.0900	0.4300	0.4500
Oversampling (SMOTE + PCA)	0.1087	0.4047	0.5678
Undersampling	0.4415	0.4343	0.5236
PCA (100 PCs)	0.1031	0.4350	0.5246

Table 11: Equal Error Rate (EER) for Decision Tree

5.2.5 Model Performance Visualization

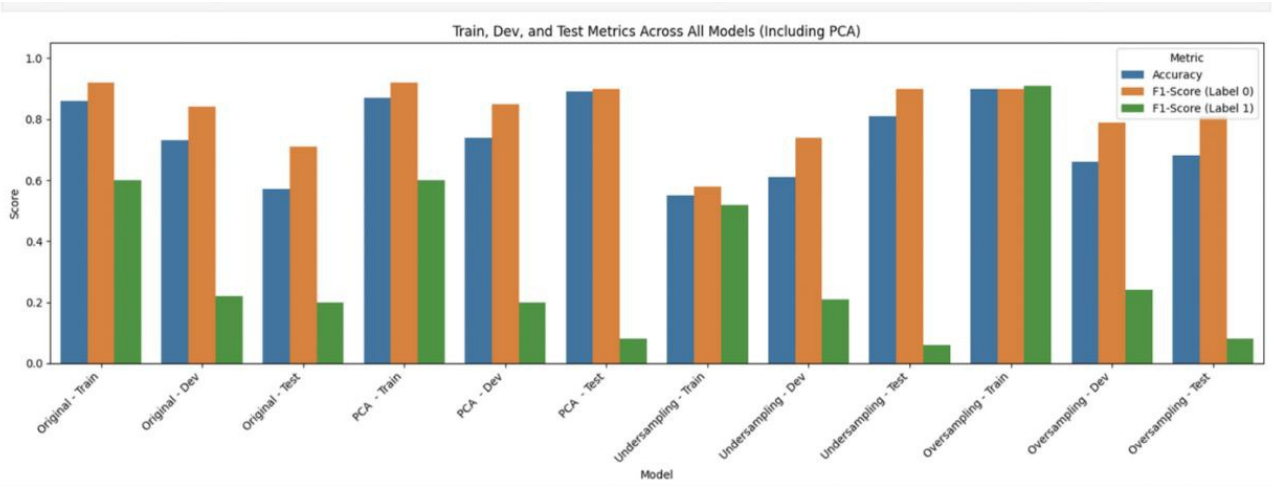


Figure 18: ACCURACY,F1scores

5.2.6 Decision Tree Classification

5.2.7 Original Dataset

- **Training:** High accuracy (86.33%) but overfitting; Test accuracy drops to 57.70%.
- **Test:** Poor performance on both classes ($F1_1 = 0.20$).
- **Key Issue:** Overly complex tree captures noise in training data.

5.2.8 PCA

- **Training/Test:** PCA improves generalization; Test accuracy rises to 81.43%.
- **Test Confusion Matrix:** Predicts class 0 well ($F1_0 = 0.90$) but fails on class 1 ($F1_1 = 0.08$).
- **Conclusion:** Dimensionality reduction reduces overfitting but does not address class imbalance.

5.2.9 Oversampled (SMOTE) + PCA

- **Training:** High accuracy (90.42%) due to synthetic data.
- **Test:** Poor generalization; accuracy 68.62%, $F1_1 = 0.08$.
- **Conclusion:** Overfitting on synthetic patterns harms real-world performance.

5.2.10 Undersampled

- **Test:** Improved accuracy (81.68%) but still fails on class 1 ($F1_1 = 0.06$).
- **Conclusion:** Undersampling discards critical data, limiting minority-class learning.

5.2.11 Key Observations

1. **Overfitting on Original Data:** The unpruned tree achieves high training accuracy (86.33%) but memorizes noise, leading to a drastic drop in Test accuracy (57.70%) and high Test EER (0.6596).
2. **PCA Improves Generalization but Not Recall:** Reducing dimensionality to the top principal components raises Test accuracy to 81.43%, yet the $F1_1$ remains at 0.08, indicating that PCA captures majority-class variance but loses minority-specific signals.
3. **Synthetic Overfitting with SMOTE + PCA:** Although balanced by SMOTE, the combination with PCA leads to a Test accuracy of only 68.62% and $F1_1$ of 0.08, suggesting the model learns interpolation artifacts rather than genuine minority-class patterns.
4. **Data Loss in Undersampling:** Undersampling boosts Test accuracy to 81.68% but still yields $F1_1 = 0.06$, showing that removing majority examples deprives the tree of context needed to form robust splits.

5.2.12 Conclusion

- Good at modeling complex patterns and class imbalance.
- Strong training performance but tends to overfit.
- Moderate generalization; validation and test performance drops slightly.
- Useful for interpretation and feature importance analysis.

5.3 Random Forest

Random Forest is an ensemble learning method that constructs multiple decision trees and aggregates their outputs to perform classification (or regression). For classification tasks, the final prediction is typically obtained via majority voting.

5.3.1 How it Works on a Dataset

1. **Bootstrap Sampling**: For each decision tree T_i, a random sample *with replacement* is drawn from the training dataset. This is known as **bagging** (Bootstrap Aggregating).
2. **Random Feature Selection**: At each node of a tree, instead of considering all features, only a randomly selected subset of features is evaluated to determine the best split. This promotes model diversity and reduces correlation between trees.
3. **Tree Construction**: Each tree is grown independently using the sampled data and selected features. Trees are typically grown to their maximum depth without pruning.
4. **Prediction**: For classification, each tree $T_i(x)$ predicts a class label for input x, and the final predicted class $\hat{y}$ is the one with the majority vote:

$$\hat{y} = \text{mode}(T_1(x), T_2(x), \ldots, T_m(x))$$

where m is the number of trees in the forest.

5.3.2 Advantages

- Reduces variance compared to a single Decision Tree.
- Less prone to overfitting due to randomization and ensemble averaging.
- Handles missing data and noisy datasets effectively.
- Robust to class imbalance when combined with techniques such as class weighting or SMOTE (Synthetic Minority Over-sampling Technique).
- Performs well on high-dimensional datasets and non-linear problems.

5.3.3 Important Notes

Random Forest combines the strengths of multiple decision trees to improve generalization. While each tree may be a high-variance, low-bias estimator, their combination through bagging stabilizes the prediction and results in a lower overall variance. **The F1 score, Recall & Precision calculated are weighted**

5.3.4 Original Dataset

Table 12: Random Forest Performance on Original Data

Training		Validation		Test	
Accuracy	1.000	Accuracy	0.897	Accuracy	0.757
Precision	1.000	Precision	0.805	Precision	0.858
Recall	1.000	Recall	0.897	Recall	0.757
F1 Score	1.000	F1 Score	0.848	F1 Score	0.796
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[22800 0]		[22296 0]		[50262 13620]	
[0 2580]		[2548 0]		[3666 3689]	

5.3.5 SMOTE Oversampled Dataset

Table 13: Random Forest Performance on SMOTE Data

Training		Validation		Test	
Accuracy	1.000	Accuracy	0.883	Accuracy	0.897
Precision	1.000	Precision	0.847	Precision	0.822
Recall	1.000	Recall	0.888	Recall	0.897
F1 Score	1.000	F1 Score	0.859	F1 Score	0.848
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[22800 0]		[21821 475]		[63858 24]	
[0 22800]		[2300 248]		[7350 5]	

5.3.6 Undersampled Dataset

Table 14: Random Forest Performance on Undersampled Data

Training		Validation		Test	
Accuracy	1.000	Accuracy	0.678	Accuracy	0.731
Precision	1.000	Precision	0.875	Precision	0.847
Recall	1.000	Recall	0.678	Recall	0.731
F1 Score	1.000	F1 Score	0.741	F1 Score	0.776
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[2580 0]		[15055 7241]		[48770 15112]	
[0 2580]		[752 1796]		[4041 3314]	

5.3.7 PCA Dataset

Table 15: Random Forest Performance with PCA

Training		Validation		Test	
Accuracy	1.000	Accuracy	0.898	Accuracy	0.897
Precision	1.000	Precision	0.908	Precision	0.804
Recall	1.000	Recall	0.898	Recall	0.897
F1 Score	1.000	F1 Score	0.849	F1 Score	0.848
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[22800 0]		[22296 0]		[63882 0]	
[0 2580]		[2545 3]		[7355 0]	

	Train EER	Dev EER	Test EER
Original	0.0000	0.3651	0.3414
Oversampling (SMOTE)	0.0000	0.2900	0.3432
Undersampling	0.0000	0.3155	0.3652
PCA (100 PCs)	0.0000	0.3355	0.4436

Table 16: Equal Error Rate (EER) for Random Forest

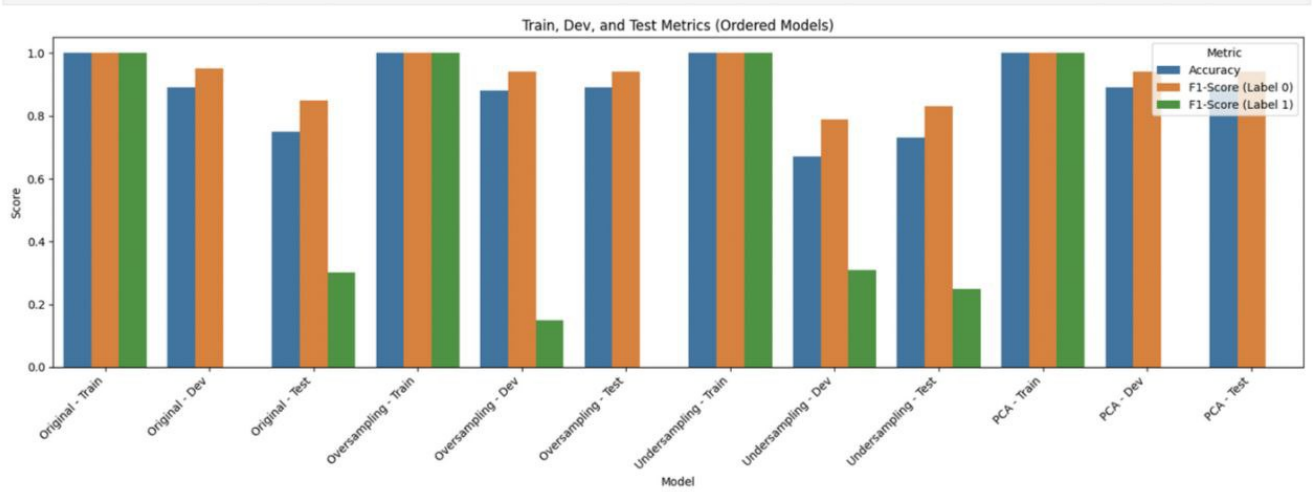


Figure 19: ACCURACY,F1scores

5.3.8 Random Forest Classification

5.3.9 Original Dataset

- **Training:** Perfect accuracy (100%) due to overfitting.
- **Test:** Best overall minority-class performance among models ($F1_1 = 0.30$).
- **Confusion Matrix:** Correctly predicts 36.66% of class 1 instances.
- **Observation:** Despite overfitting on training, Random Forest generalizes better than Naive Bayes or a single Decision Tree for class 1.

5.3.10 Oversampled (SMOTE)

- **Training:** Perfect accuracy (100%) on synthetic data.
- **Test:** $F1_1$ drops sharply to 0.001.
- **Conclusion:** Aggressive overfitting to synthetic patterns makes the model nearly useless for minority prediction on real-world data.

5.3.11 Undersampled

- **Test:** Moderate accuracy (73.11%) and $F1_1$ improves slightly to 0.25.
- **Confusion Matrix:** Captures 33.14% of class 1 samples.
- **Conclusion:** Balancing the training set via undersampling allows the forest to better learn class 1 patterns, though at the expense of total predictive power.

5.3.12 PCA

- **Test:** High overall accuracy (89.68%) but zero recall on class 1 ($F1_1 = 0.00$).
- **Conclusion:** Dimensionality reduction with PCA filters out critical minority class signals, leading to complete failure on class 1.

5.3.13 Key Observations

1. **Best Minority Recall Among Baselines:** The original Random Forest achieves Test $F1_1 = 0.30$ and $Recall_1 = 36.7\%$, the best among all models evaluated.

2. **SMOTE Causes Extreme Overfitting:** Perfect training performance is misleading, as the model memorizes interpolated samples and fails to generalize ($F1_1 = 0.001$).

3. **Undersampling Enables Class 1 Learning:** Reducing majority dominance leads to Test $F1_1 = 0.25$ and $Recall_1 = 33.1\%$, albeit with reduced overall accuracy.

4. **PCA Filters Out Class 1 Signal:** Top principal components mostly reflect majority-class variance, making PCA unsuitable for preserving class 1 structure.

5.3.14 Conclusion

- Best performance across all datasets.
- Robust to overfitting due to ensemble learning.
- Effective with PCA and SMOTE for handling imbalance and dimensionality.
- Less interpretable but highly reliable in practice.

5.4 Support Vector Machine

5.4.1 Original Dataset

Table 17: SVM Matrices - Original dataset

Set (Kernel)	Accuracy	Precision	Recall	F1-score
Train (Linear/Poly/RBF)	0.9005	1.0000	0.0000	0.0000
Train (Sigmoid)	0.8337	0.1461	0.1386	0.1423
Dev (Linear/Poly/RBF)	0.8976	1.0000	0.0000	0.0000
Dev (Sigmoid)	0.8253	0.1528	0.1552	0.1540

5.4.2 With Sampling Techniques (Train Set)

5.4.3 Oversampling (SMOTE)

Table 18: PCA Results

Accuracy	Precision	Recall	F1-score
0.6846	0.2018	0.7112	0.3143

Figure 20: SVM Oversampling – Figures 1–6

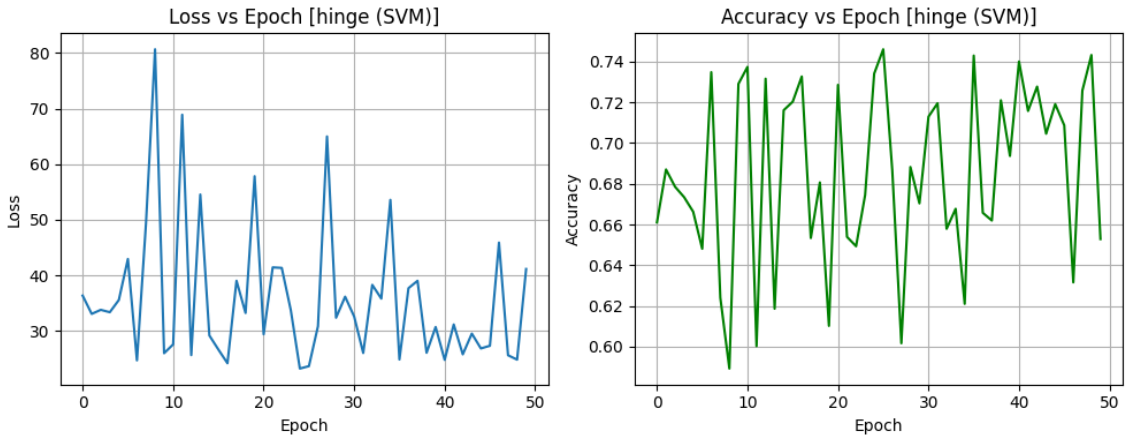


Figure 21: Oversampling hinge(SVM)

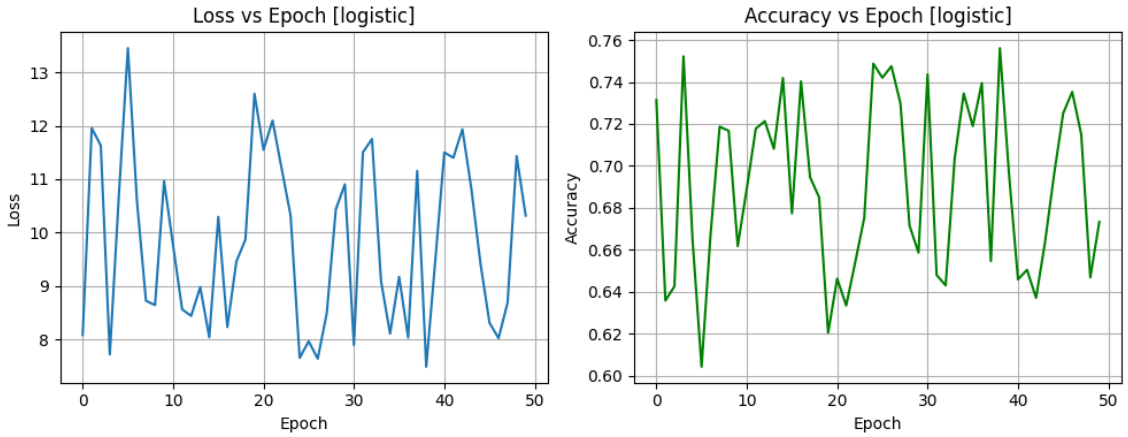


Figure 22: Oversampling (logistic)

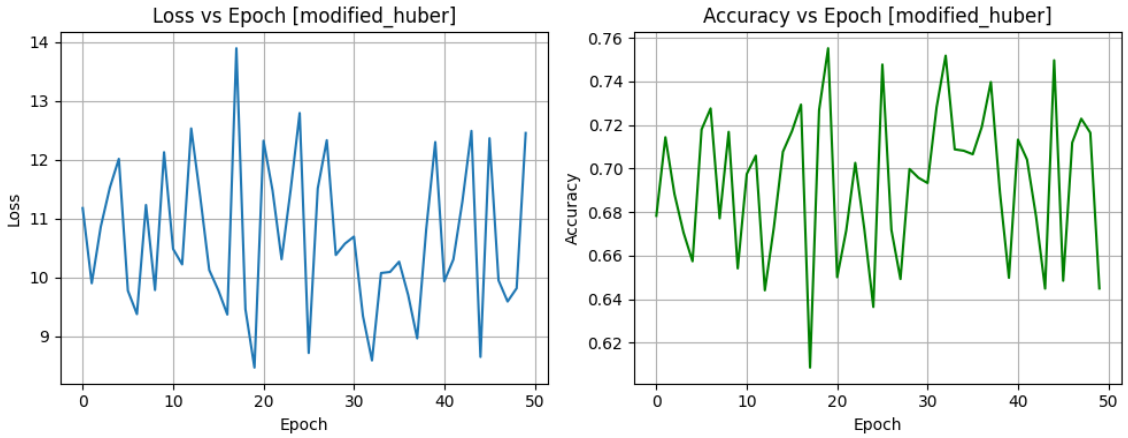


Figure 23: Oversampling (modifier-huber)

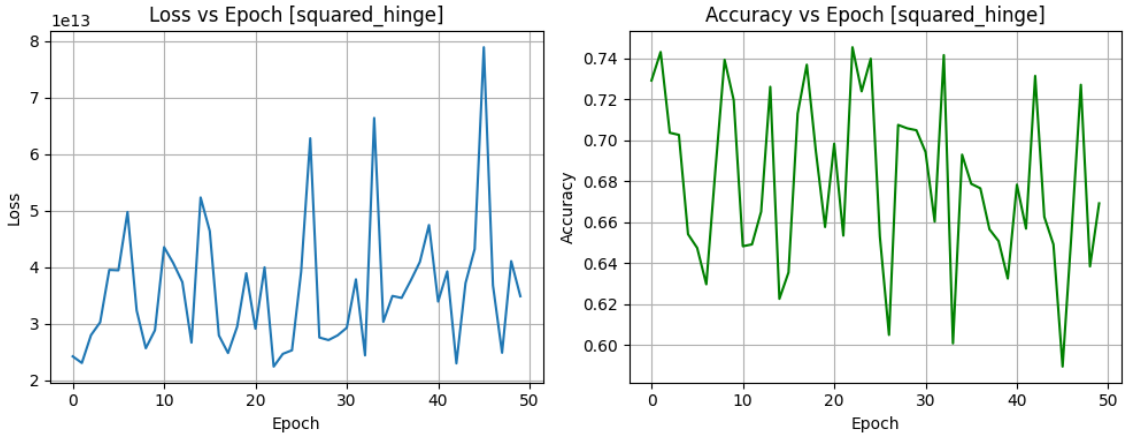


Figure 24: Oversampling (squared-hinge)

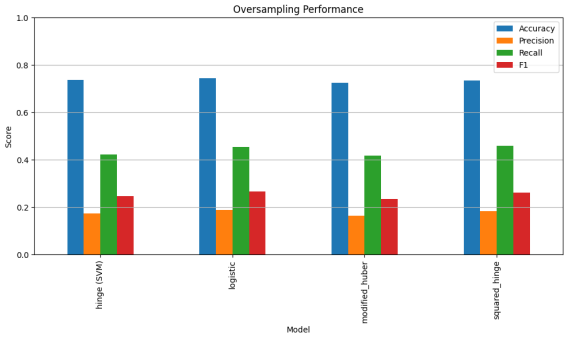


Figure 25: Oversampling Performance

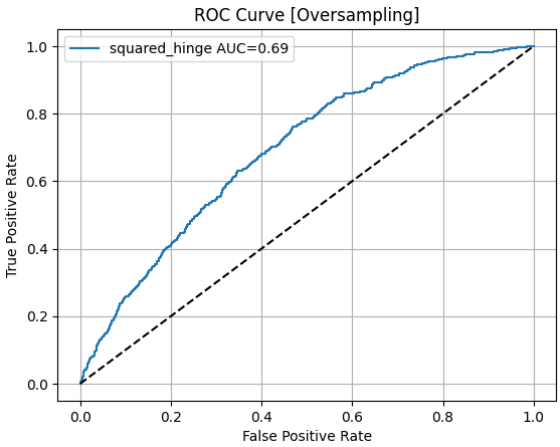


Figure 26: ROC curve for Oversampling

5.4.4 Undersampling

Table 19: PCA Results

Accuracy	Precision	Recall	F1-score
0.5952	0.1727	0.7868	0.3832

Figure 27: SVM Undersampling – Figures 7–12

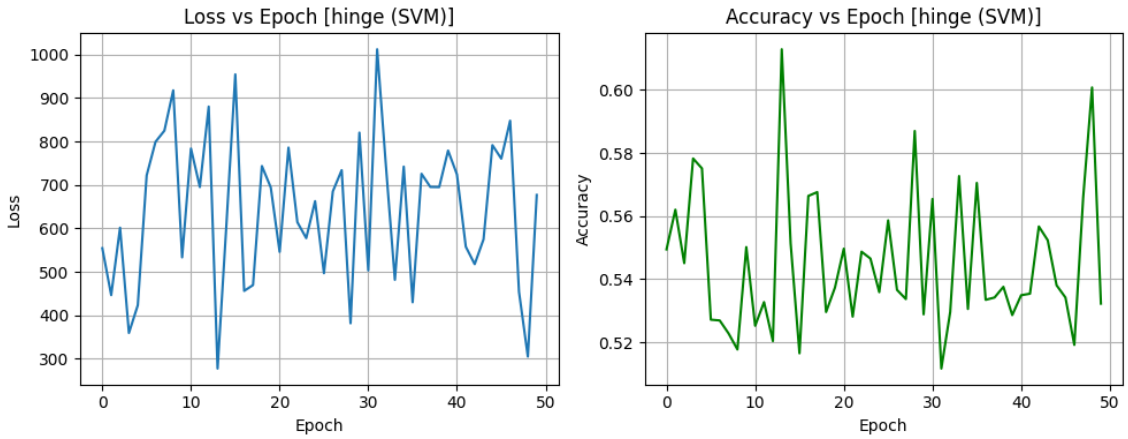


Figure 28: Undersampling hinge(SVM)

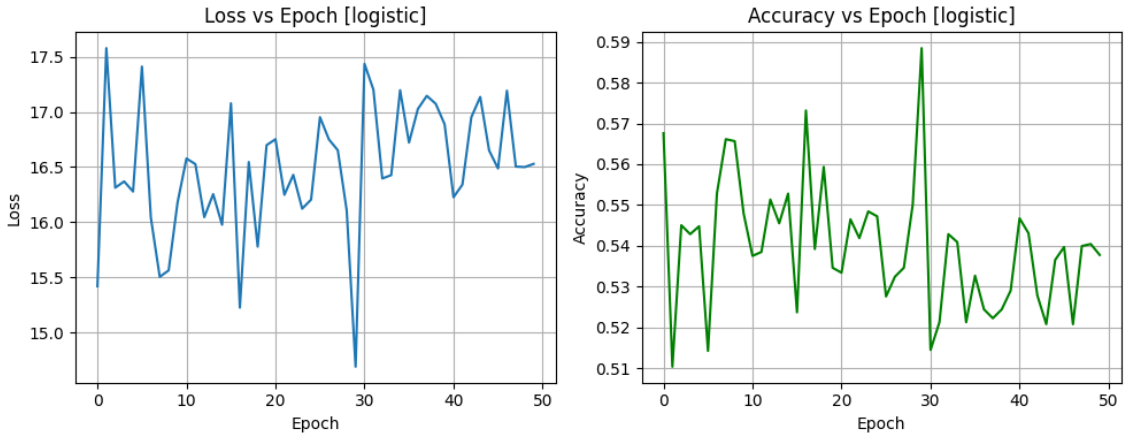


Figure 29: Undersampling (logistic)

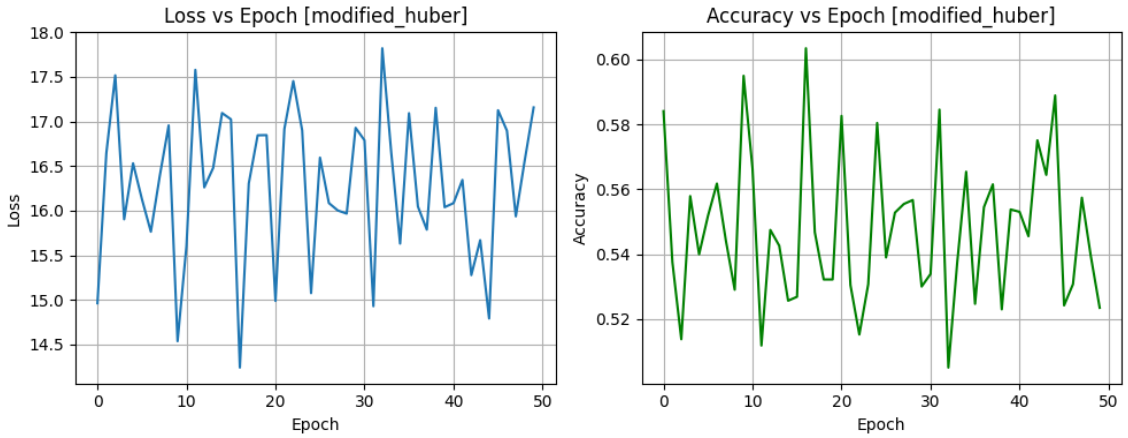


Figure 30: Undersampling (modifier-huber)

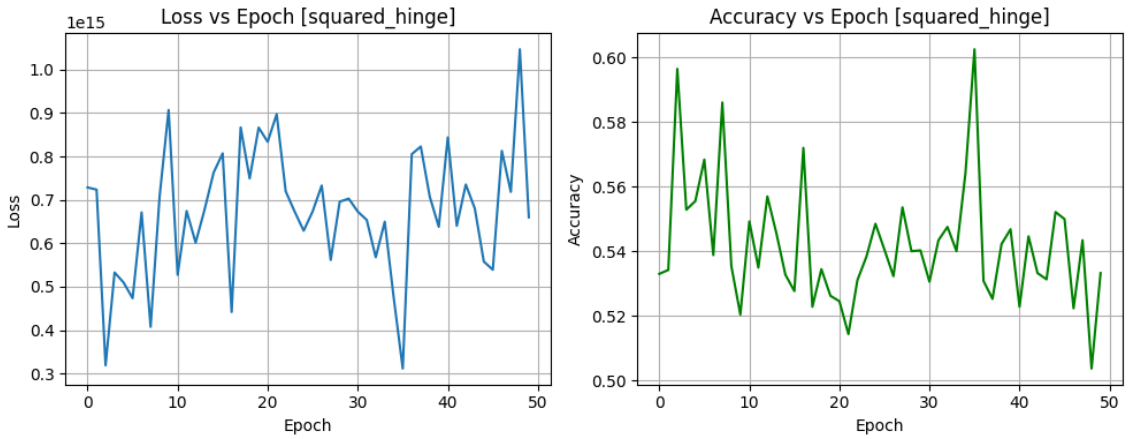


Figure 31: Undersampling (squared-hinge)

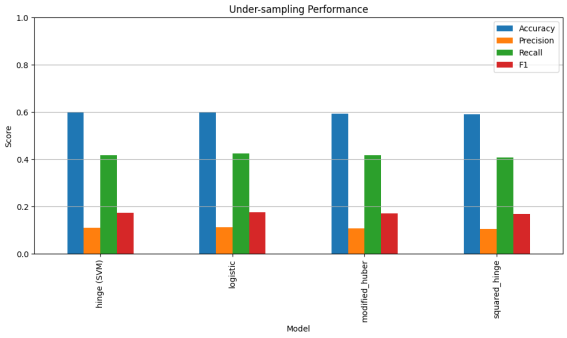


Figure 32: Undersampling performance

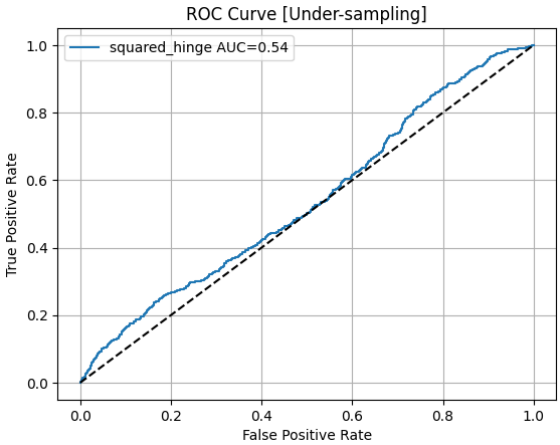


Figure 33: ROC curve undersampling

5.4.5 Class Weight Biasing

Table 20: PCA Results

Accuracy	Precision	Recall	F1-score
0.6411	0.1908	0.7810	0.3067

Figure 34: SVM Class Weight Biasing – Figures 13–18

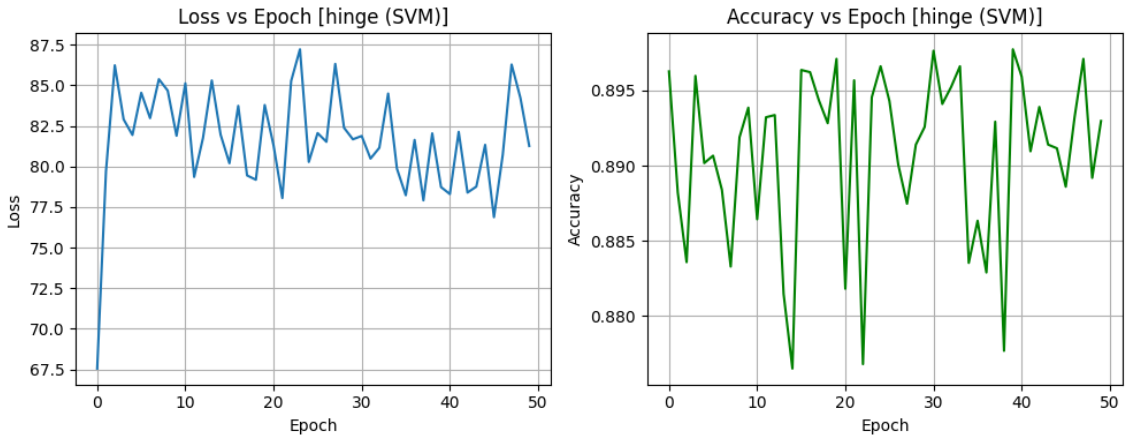


Figure 35: SVM Class Weight Biasing - hinge(SVM)

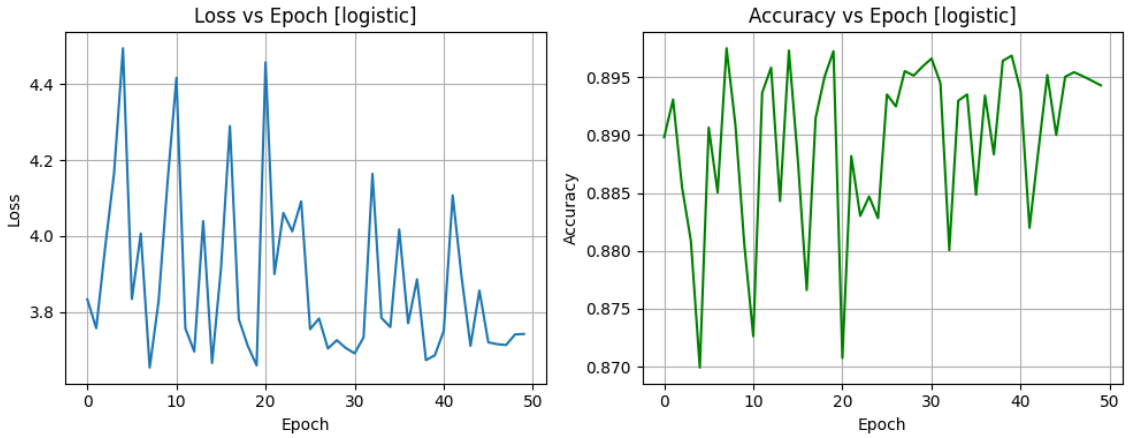


Figure 36: SVM Class Weight Biasing- logistic

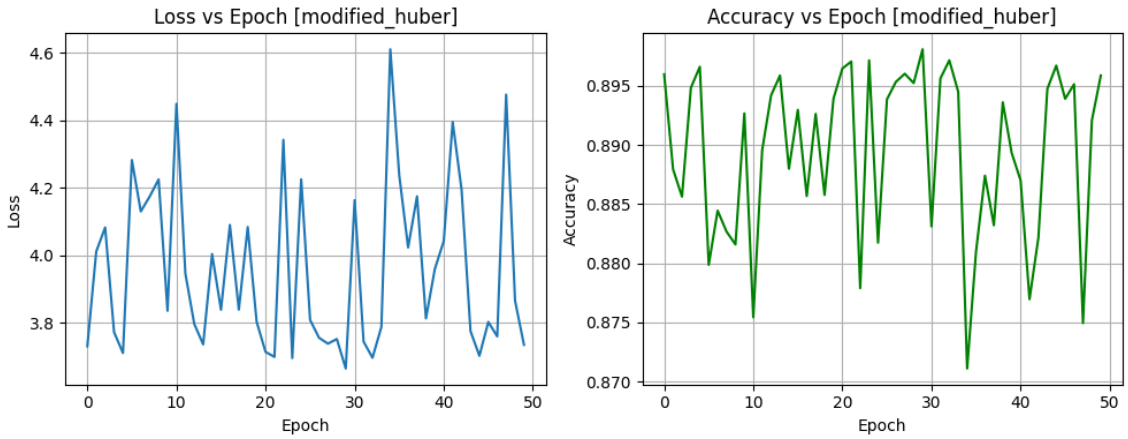


Figure 37: SVM Class Weight Biasing - modifier-huber

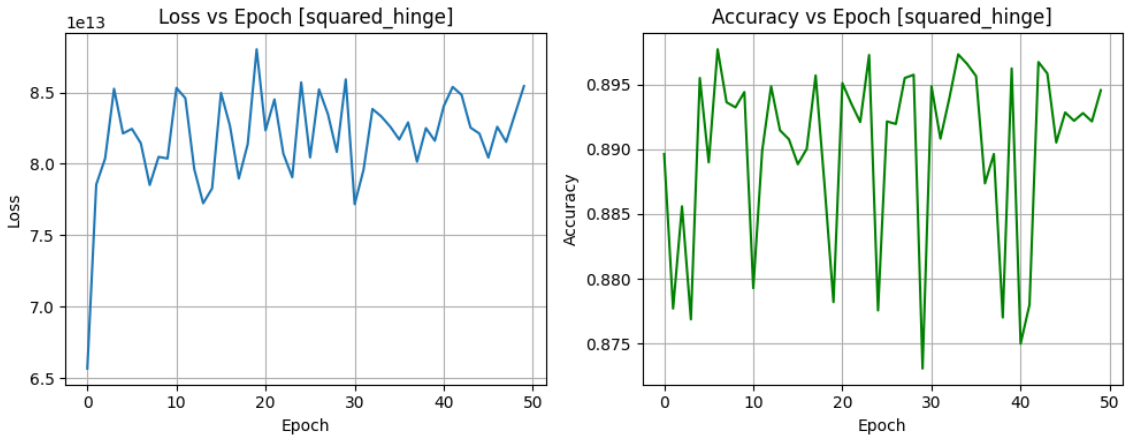


Figure 38: SVM Class Weight Biasing - squared-hinge

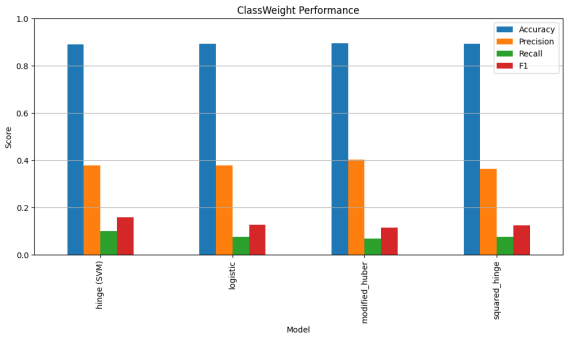


Figure 39: ClassWeight performance

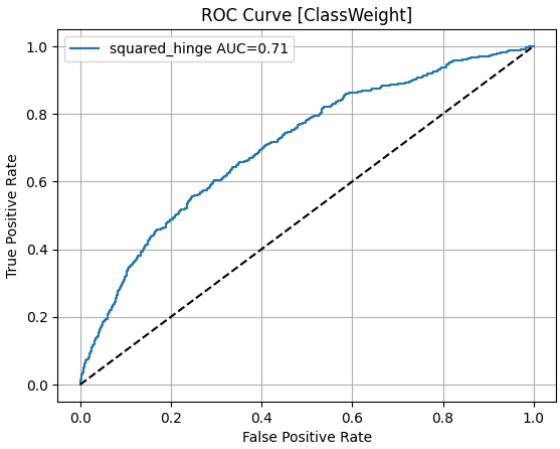


Figure 40: ROC curve classweight

5.4.6 PCA

Table 21: PCA Results

n components	Accuracy	Precision	Recall	F1-score
50,100,200	0.8983	1.0000	0.0000	0.0000

Figure 41: SVM PCA – Figures 19–24

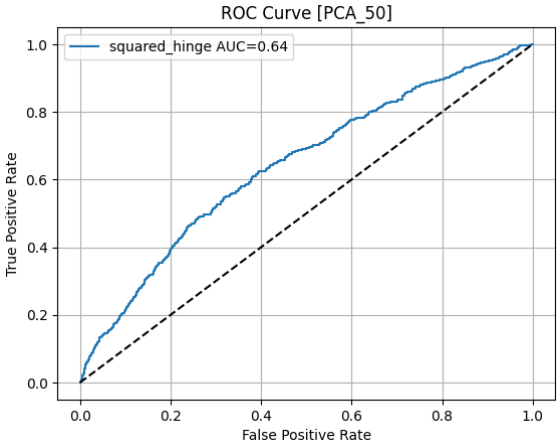


Figure 42: ROC curve - PCA 50

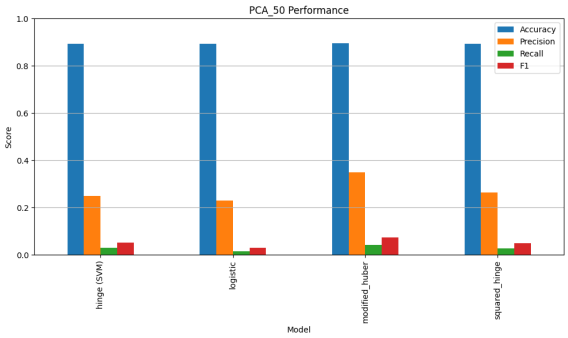


Figure 43: PCA-50 performance

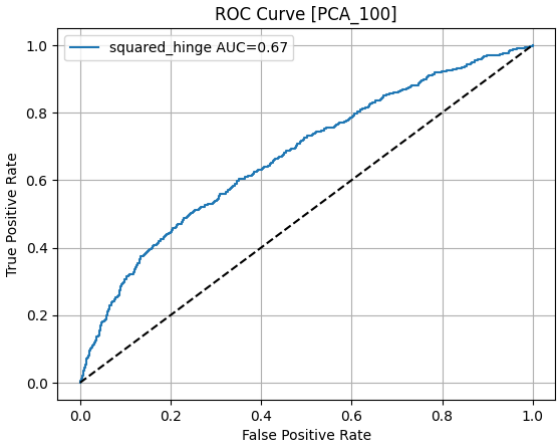


Figure 44: ROC curve - PCA100

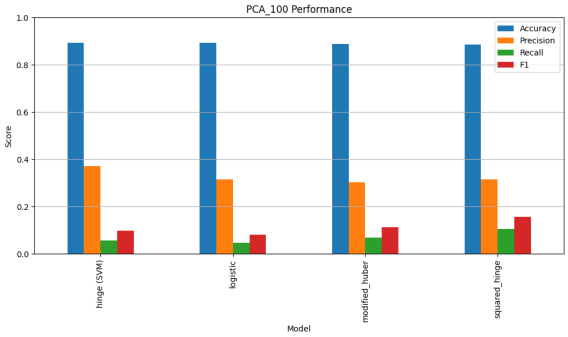


Figure 45: PCA-100 performance

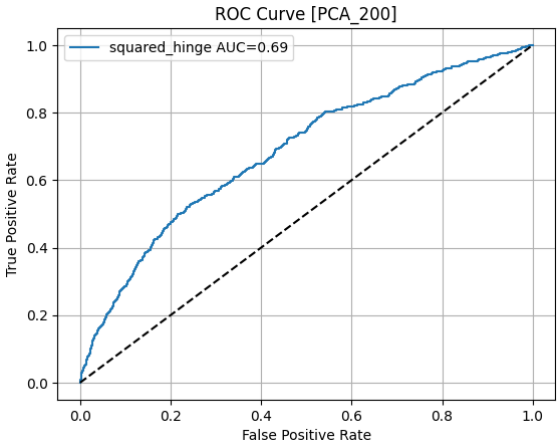


Figure 46: ROC curve - PCA200

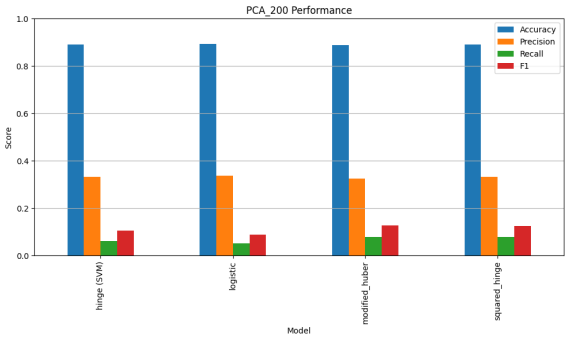


Figure 47: PCA-200 performance

5.4.7 SVM Confusion Matrices

Table 22: SVM Confusion Matrix – Undersampling

Actual \ Predicted	Fake	Real
Fake	13433	6835
Real	1125	5039

Table 23: SVM Confusion Matrix – Oversampling

Actual \ Predicted	Fake	Real
Fake	15010	5258
Real	2732	3432

Table 24: SVM Confusion Matrix – Class Weight

Actual \ Predicted	Fake	Real
Fake	14207	6104
Real	1906	4258

5.5 Naive Bayes

Naive Bayes classifiers apply Bayes' theorem with strong independence assumptions between features. Despite these simplifying assumptions, they often perform well in practice. **The F1 score, Recall & Precision calculated are weighted**

We implemented a Gaussian Naive Bayes classifier, which assumes that features follow a normal distribution.

5.5.1 Original Dataset

Table 25: Naive-Bayes Performance on Original Data

Training		Validation		Test	
Accuracy	0.638	Accuracy	0.644	Accuracy	0.897
Precision	0.843	Precision	0.841	Precision	0.804
Recall	0.638	Recall	0.644	Recall	0.897
F1 Score	0.709	F1 Score	0.713	F1 Score	0.848
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[14861 7939]		[14737 7559]		[63882 0]	
[1251 1329]		[1279 1269]		[7355 0]	

5.5.2 SMOTE Oversampled and PCA100 Dataset

Table 26: Naive-Bayes Performance on SMOTE Data

Set	Accuracy	Precision	Recall	F1 Score
Training	0.684	0.685	0.684	0.684
Validation	0.660	0.850	0.660	0.725
Test	0.897	0.804	0.897	0.848

- **Training Set Confusion Matrix:**

$$\begin{bmatrix} 15148 & 7652 \\ 6738 & 16062 \end{bmatrix}$$

- **Dev Set Confusion Matrix:**

$$\begin{bmatrix} 14996 & 7300 \\ 1158 & 1390 \end{bmatrix}$$

- **Test Set Confusion Matrix:**

$$\begin{bmatrix} 63882 & 0 \\ 7355 & 0 \end{bmatrix}$$

5.5.3 Undersampled and PCA100 Dataset

Table 27: Naive-Bayes Performance on Undersampled Data

Set	Accuracy	Precision	Recall	F1 Score
Training	0.600	0.600	0.600	0.600
Validation	0.607	0.854	0.607	0.684
Test	0.892	0.805	0.892	0.846

- **Training Set Confusion Matrix:**

$$\begin{bmatrix} 1536 & 1044 \\ 1022 & 1558 \end{bmatrix}$$

- **Dev Set Confusion Matrix:**

$$\begin{bmatrix} 1536 & 1044 \\ 1022 & 1558 \end{bmatrix}$$

- **Test Set Confusion Matrix:**

$$\begin{bmatrix} 63573 & 309 \\ 7351 & 4 \end{bmatrix}$$

Sampling	Train EER	Dev EER	Test EER
Original	0.4264	0.4264	0.6494
Oversampling (SMOTE)	0.3233	0.3390	0.6430
Undersampling	0.4000	0.3830	0.6800

Table 28: Equal Error Rate (EER) for Naive Bayes

5.5.4 Model Performance Visualization

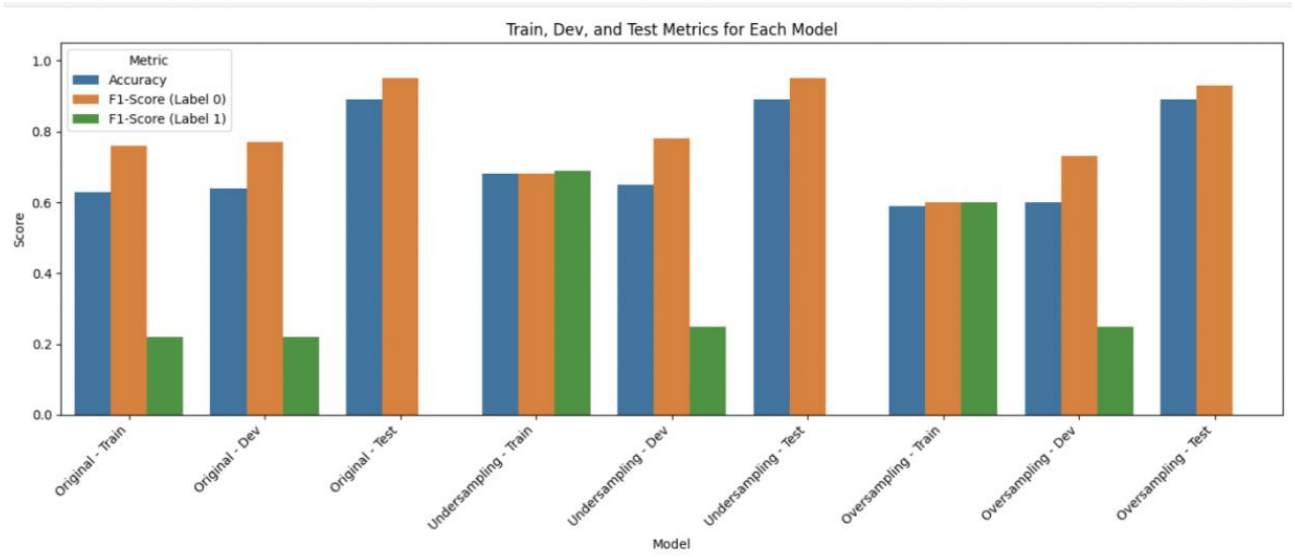


Figure 48: ACCURACY,F1 scores

5.5.5 Original Dataset

- **Training:** Moderate accuracy (63.79%), poor F1 for class 1 (0.22). Confusion matrix shows 7,939 class 0 and 1,251 class 1 samples misclassified.

- **Validation:** Slight improvement (64.42% accuracy), but class 1 F1 remains low (0.22).
- **Test:** High accuracy (89.65%) achieved by predicting only class 0 ($F1_1 = 0$).
- **Key Issue:** Severe bias toward majority class (label 0).

5.5.6 Oversampled (SMOTE)

- **Training:** Balanced predictions (68.44% accuracy), but synthetic data fails to generalize.
- **Validation/Test:** Still predicts only class 0 ($F1_1 = 0$).
- **Conclusion:** SMOTE introduces noise without improving minority-class detection.

5.5.7 Undersampled

- **Training/Validation:** Reduced accuracy (~60%) due to data loss.
- **Test:** High accuracy (89.24%) but predicts only class 0 ($F1_1 = 0$).
- **Conclusion:** Undersampling degrades overall performance without resolving imbalance.

5.5.8 Key Observations

1. **Severe Majority-Class Bias:** Across original, SMOTE, and undersampled variants, the model almost exclusively predicts class 0 on unseen data (Test $F1_1 = 0.00$). Even though SMOTE temporarily boosts Training $F1_1$ ($\approx$0.69), these synthetic examples fail to transfer to Validation/Test, indicating that the minority-class feature distributions are not well captured by the simple likelihood estimates of Naive Bayes.
2. **Equal Error Rate (EER) Stability:** The EER remains roughly constant (~0.42) on both Training and Validation for the original data, suggesting that the trade-off between false positives and false negatives is not improved by naive oversampling or undersampling.
3. **Feature Independence Breakdown:** The near-zero recall on class 1 implies that important feature correlations (which Naive Bayes ignores) are critical for distinguishing the minority class. In practice, correlated features (e.g., combined behavioral signals) may be essential, so consider models that can capture interactions (e.g., tree-based methods with feature crosses).

5.5.9 Conclusion

- Fast and simple with decent performance on training and validation sets.
- Severely biased towards the majority class in the test set.
- Sampling techniques (SMOTE, undersampling) do not significantly improve its generalization.
- Assumption of feature independence limits performance on real-world data.

5.6 k-Nearest Neighbors (k-NN)

5.6.1 Original Dataset

- Train: Accuracy = 0.9139, Precision = 0.7297, Recall = 0.2438, F1-score = 0.3655
- Dev: Accuracy = 0.9073, Precision = 0.6540, Recall = 0.2033, F1-score = 0.3102
- Eval: Accuracy = 0.9168, Precision = 0.6728, Recall = 0.3774, F1-score = 0.4836

Table 29: With Sampling Techniques (Train Set)

quantity	Train	Dev	Eval
Accuracy	0.9139	0.9073	0.9168
Precision	0.7297	0.6540	0.6728
Recall	0.2438	0.2033	0.3774
F1-score	0.3655	0.3102	0.4836

5.6.2 With Sampling Techniques (Train Set)

Table 30: With Sampling Techniques (Train Set)

quantity	Oversampling (SMOTE)	Undersampling	Class Bias (Real)
Accuracy	0.6089	0.7638	0.9229
Precision	0.5611	0.7113	0.8724
Recall	1.0000	0.8880	1.0000
F1-score	0.7188	0.7899	0.9318

5.6.3 PCA Results

Table 31: PCA Results

quantity	n=50	n=100	n=200
Accuracy	0.9136	0.9137	0.9145
Precision	0.7669	0.7653	0.7572
Recall	0.2155	0.2174	0.2345
F1-score	0.3365	0.3387	0.3581

5.6.4 KNN Confusion Matrices

Oversampling

Table 32: KNN Confusion Matrix – Oversampling

Actual \ Predicted	Fake	Real
Fake	10981	8697
Real	0	7584

Undersampling

Table 33: KNN Confusion Matrix – Undersampling

Actual \ Predicted	Fake	Real
Fake	13700	6978
Real	845	5795

Class Bias (Real)

Table 34: KNN Confusion Matrix – Real Class Bias

Actual \ Predicted	Fake	Real
Fake	7883	12839
Real	0	6640

PCA (n = 50)

Table 35: KNN Confusion Matrix – PCA (50 components)

Actual \ Predicted	Fake	Real
Fake	22631	169
Real	2024	556

Dev Set (All Samples)

Table 36: KNN Confusion Matrix – Dev Set

Actual \ Predicted	Fake	Real
Fake	22022	274
Real	2030	518

5.6.5 Visual Confusion Matrices

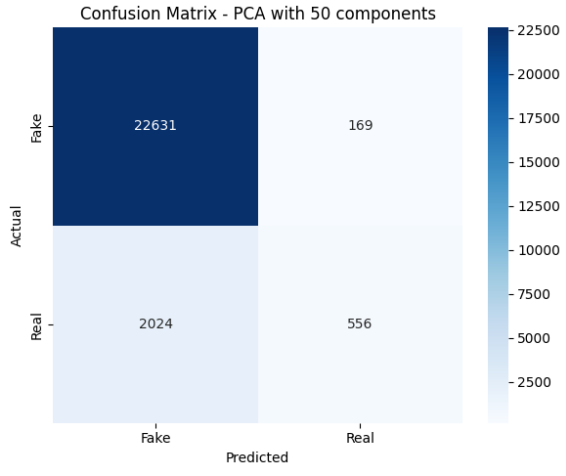


Figure 49: KNN Confusion Matrix – PCA (50 components)

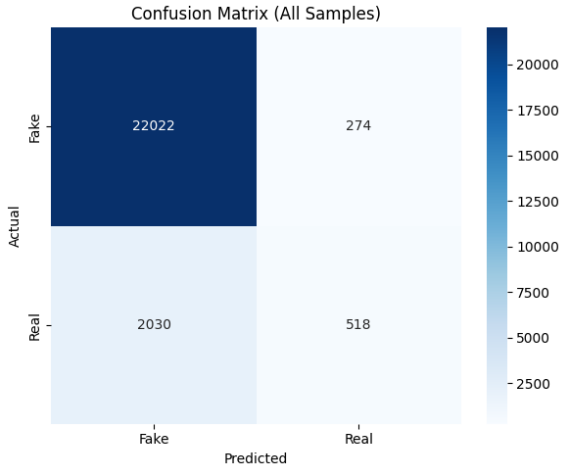


Figure 50: KNN Confusion Matrix – Dev Set

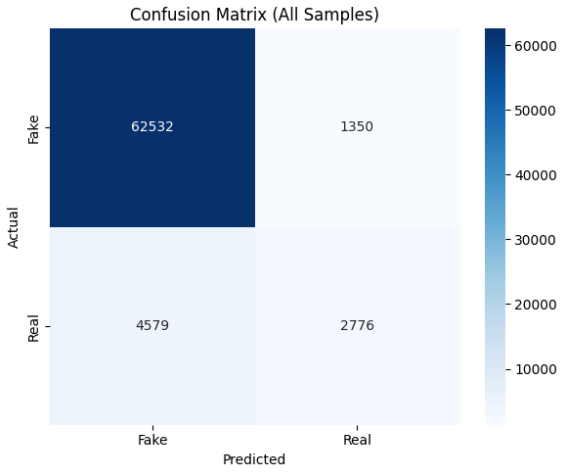


Figure 51: KNN Confusion Matrix – Eval Set

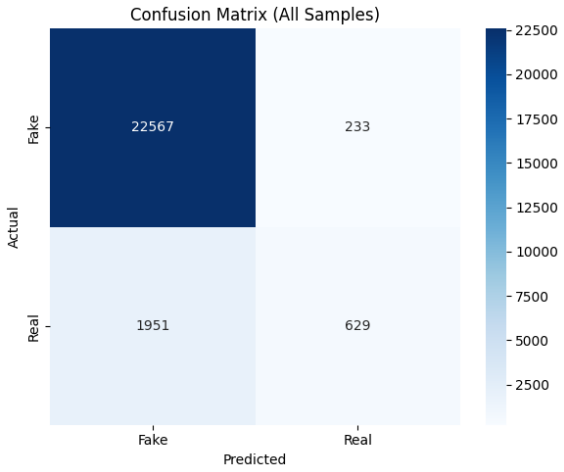


Figure 52: KNN Confusion Matrix – Train Set

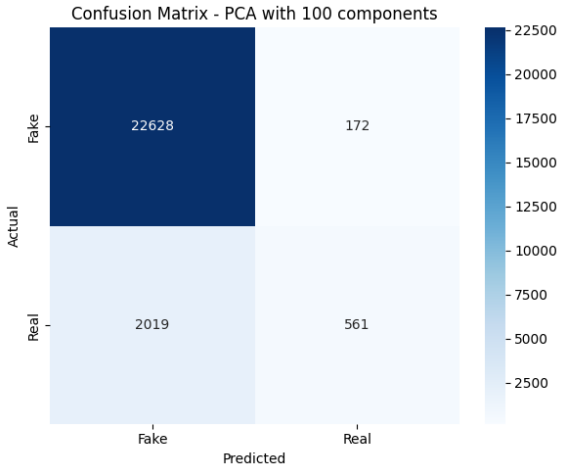


Figure 53: KNN Confusion Matrix – PCA (100 components)

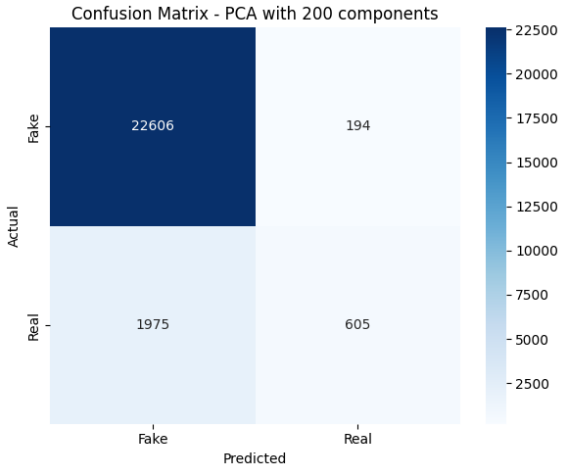


Figure 54: KNN Confusion Matrix – PCA (200 components)

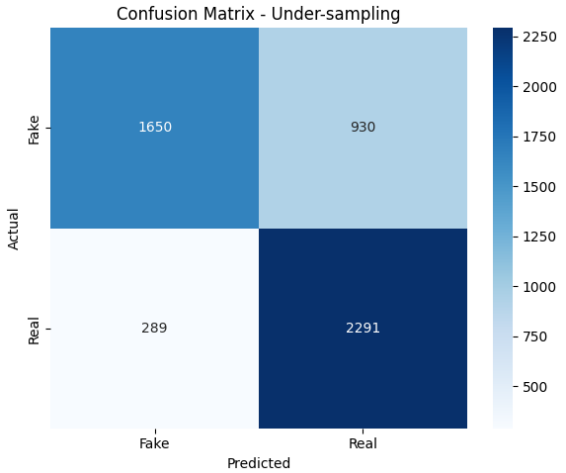


Figure 55: KNN Confusion Matrix – Under-sampling

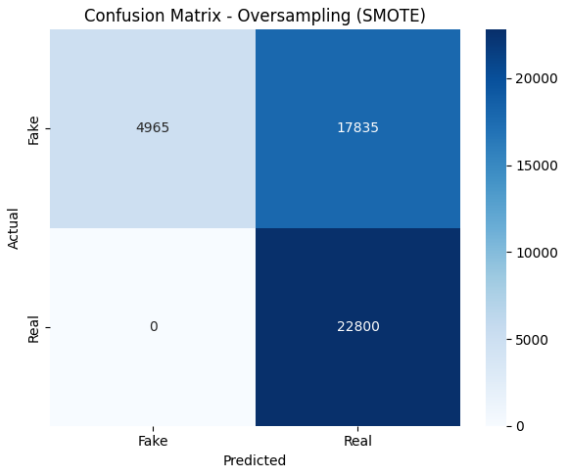


Figure 56: KNN Confusion Matrix – Over-sampling

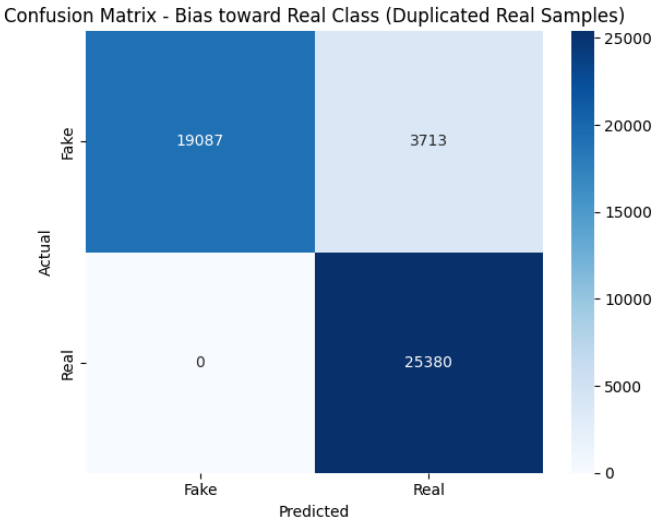


Figure 57: KNN Confusion Matrix – Biased (real one assigned more value)

5.7 Single-layer Neural Network

5.7.1 Single Layer Neural Network without processing

A shallow neural network consists of an input layer, one or two hidden layers, and an output layer.

Table 37: Performance Metrics

Single Layer Perceptron	
Accuracy	0.5013
Precision	0.1240
Recall	**0.6318**
F1 Score	0.2074
Confusion Matrix:	
[31066	32816]
[2708	4647]

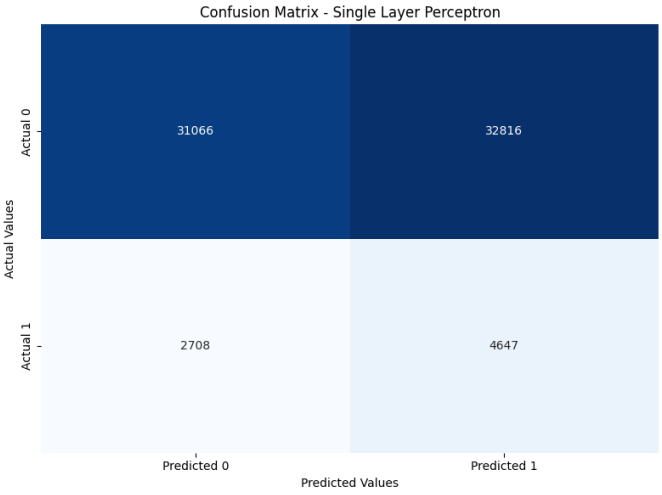


Table 38: Visual Analysis of Confusion Matrices

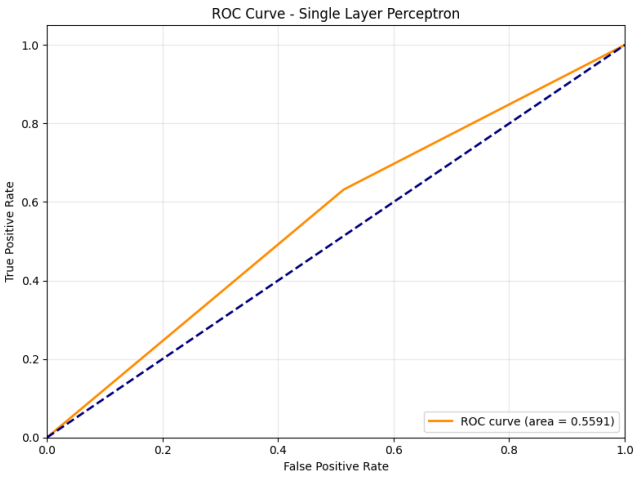


Figure 58: ROC Curve

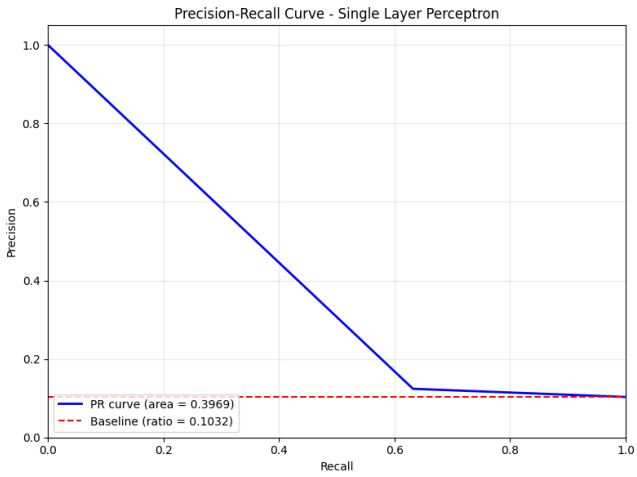


Figure 59: Precision recall Curve

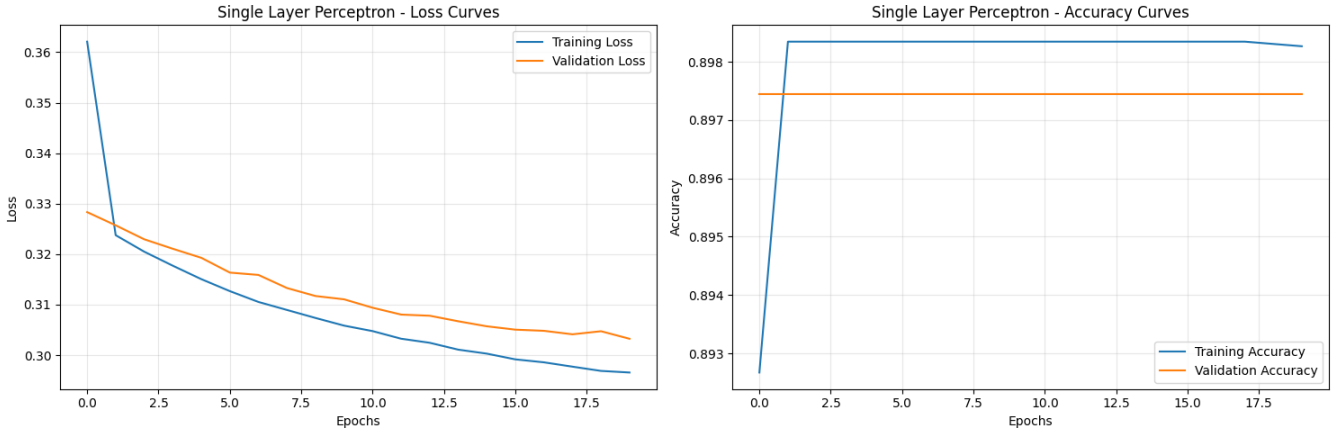


Figure 60: Accuracy vs Epochs

5.7.2 Single Layer Neural Network (Processed)

The network used ReLU activation functions for the hidden layer, a sigmoid activation for the output layer, and was trained using the Adam optimizer with a learning rate of 0.001.

The Loss function was Binary-Cross Entropy (Suitable for unbalanced class representation)

Class/Metric	Precision	Recall	F1-Score	Support
0	0.91	0.33	0.48	63882
1	0.11	0.71	0.19	7355
Accuracy	–	–	0.37	71237
Macro Avg	0.51	0.52	0.34	71237
Weighted Avg	0.82	0.37	0.45	71237

Table 39: Regularized - Classification Report

Table 40: Performance Metrics

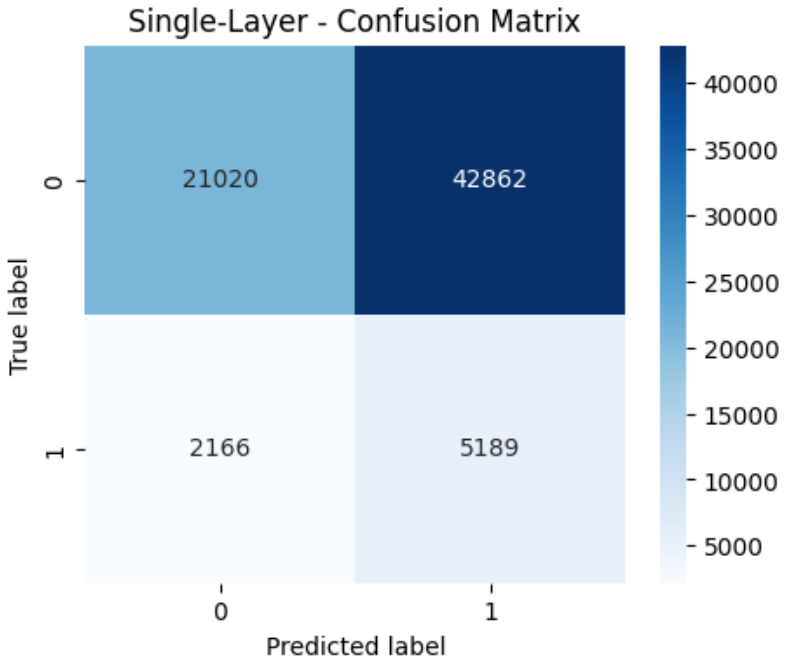


Table 41: Visual Analysis of Confusion Matrices

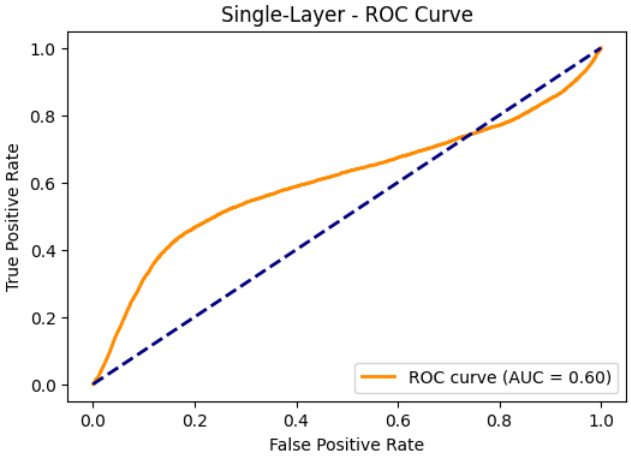


Figure 61: ROC Curve

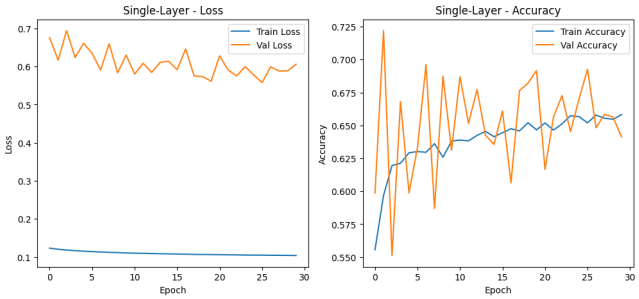


Figure 62: Precision recall Curve

OverSampling with PCA n = 50

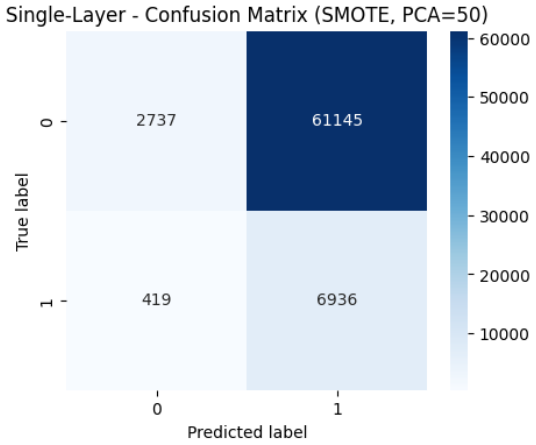


Figure 63: Single-Layer - Confusion Matrix (SMOTE, PCA=50

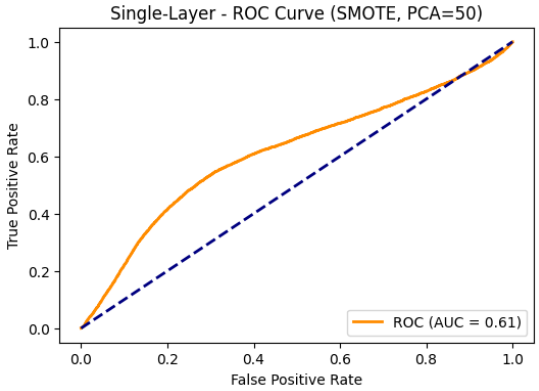


Figure 64: ROC Curve SMOTE PCA = 50

Class/Metric	Precision	Recall	F1-Score	Support
0	0.87	0.04	0.08	63882
1	0.10	0.94	0.18	7355
Accuracy	–	–	0.14	71237
Macro Avg	0.48	0.49	0.13	71237
Weighted Avg	0.79	0.14	0.09	71237

Table 42: Single-Layer-PCA 50-SMOTE - Classification Report

OverSampling with PCA n = 100

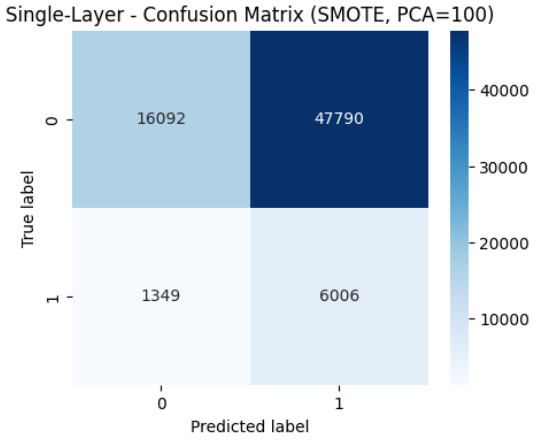


Figure 65: Single-Layer - Confusion Matrix (SMOTE, PCA=100

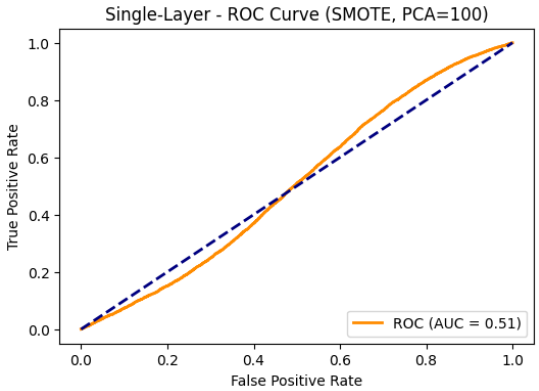


Figure 66: ROC Curve SMOTE PCA = 100

OverSampling with PCA n = 200

Class/Metric	Precision	Recall	F1-Score	Support
0	0.92	0.25	0.40	63882
1	0.11	0.82	0.20	7355
Accuracy	–	–	0.31	71237
Macro Avg	0.52	0.53	0.30	71237
Weighted Avg	0.84	0.31	0.38	71237

Table 43: Single-Layer-PCA 100-SMOTE - Classification Report

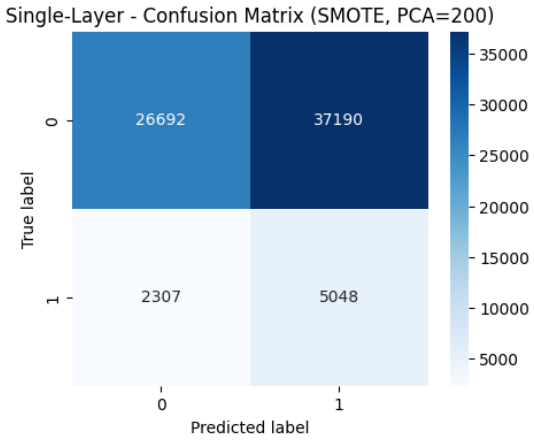


Figure 67: Single-Layer - Confusion Matrix (SMOTE, PCA=200

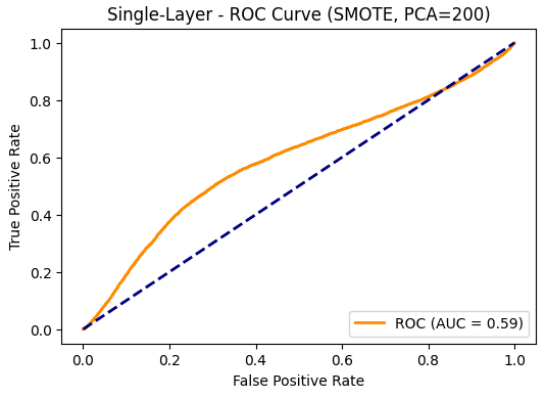


Figure 68: ROC Curve SMOTE PCA = 200

Class/Metric	Precision	Recall	F1-Score	Support
0	0.92	0.42	0.57	63882
1	0.12	0.69	0.20	7355
Accuracy	–	–	0.45	71237
Macro Avg	0.52	0.55	0.39	71237
Weighted Avg	0.84	0.45	0.54	71237

Table 44: Single-Layer-PCA 200-SMOTE - Classification Report

5.7.3 Conclusion

The experiments with the single-layer neural network reveal several important insights regarding model performance under various processing regimes.

- **Baseline Performance:** Without additional processing, the shallow network yields moderate overall performance with acceptable accuracy (approximately 50%), yet it suffers from significant class imbalance. The confusion matrix and associated metrics indicate that the model struggles to identify positive instances reliably, as reflected in the relatively low recall for class 1.
- **Processed Model Improvements:** When processing is introduced—specifically by employing ReLU activations, sigmoid output, and training with the Adam optimizer using a binary cross-entropy loss—the performance metrics shift. Although recall

for the minority class improves (up to 71% for class 1), the precision remains very low (around 11%), indicating that many negative samples are incorrectly classified as positives. This trade-off points directly to the challenges posed by a highly imbalanced dataset.

- **Impact of Oversampling with SMOTE and PCA:** Implementing SMOTE to synthetically balance the training set, coupled with dimensionality reduction via PCA, shows varying effects on performance based on the number of principal components:
  - **PCA with 50 Components:** The network achieves a drastic imbalance with recall for class 1 very high (94%), yet at the cost of drastically reduced precision and overall accuracy.
  - **PCA with 100 Components:** This configuration demonstrates a modest improvement in both metrics, where overall accuracy and the F1 score see a slight boost, but the recall and precision trade-offs remain prominent.
  - **PCA with 200 Components:** Further increasing the dimensionality helps to improve the overall F1 score and accuracy compared to the lower PCA configurations, yet the network still faces difficulty balancing precision and recall due to intrinsic data imbalance.
- **Overall Findings:** These results confirm that while preprocessing steps such as activation function tuning, optimizer adjustment, and synthetic oversampling can boost recall for the minority class, they often result in a low precision. This underscores the continuing challenge of addressing class imbalance. Future work might explore alternative regularization techniques, more sophisticated data augmentation strategies, or advanced architectures to better reconcile this precision-recall trade-off.

Overall, although the network is able to enhance minority class recognition through both processing and oversampling techniques, each method introduces specific trade-offs, emphasizing the need for a more balanced approach to mitigate the inherent bias towards the majority class.

5.8 Multi-Layer Neural Network

5.8.1 Multi Neural Network without processing

Deep neural networks contain multiple hidden layers, allowing them to learn more complex representations. Our deep neural network consisted of 2 hidden layers with [512, 128] neurons, respectively.

The network used ReLU activations for all hidden layers, dropout regularization (rate=0.3) after each hidden layer to prevent overfitting, and batch normalization to stabilize training.

Table 45: Performance Metrics

Multi Layer Perceptron

Accuracy	0.4383
Precision	0.1124
Recall	**0.6442**
F1 Score	0.1915

Confusion Matrix:
[26485 37397]
[2617 4738]

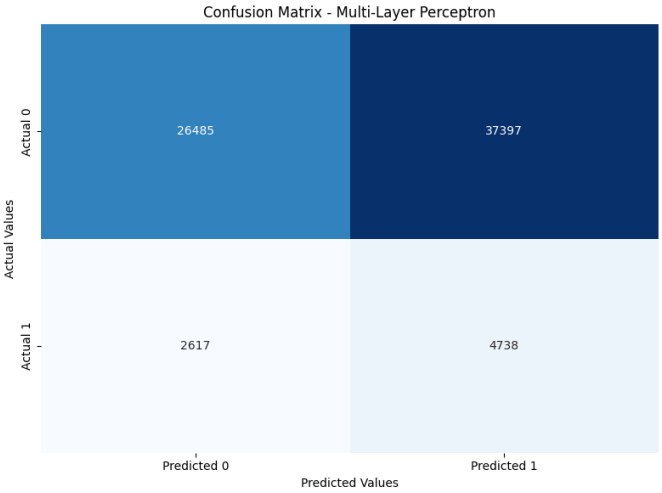


Table 46: Visual Analysis of Confusion Matrices

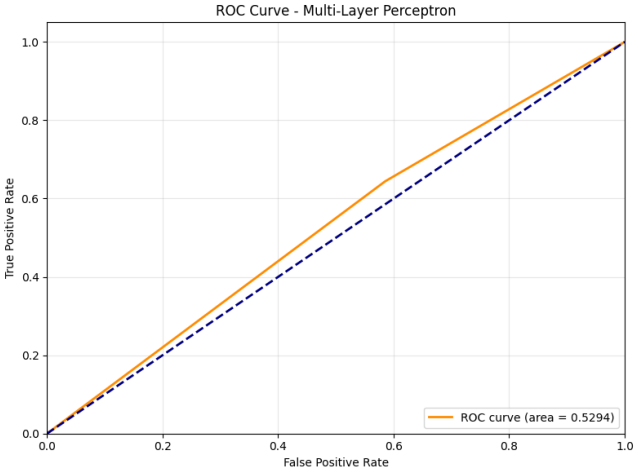


Figure 69: ROC Curve

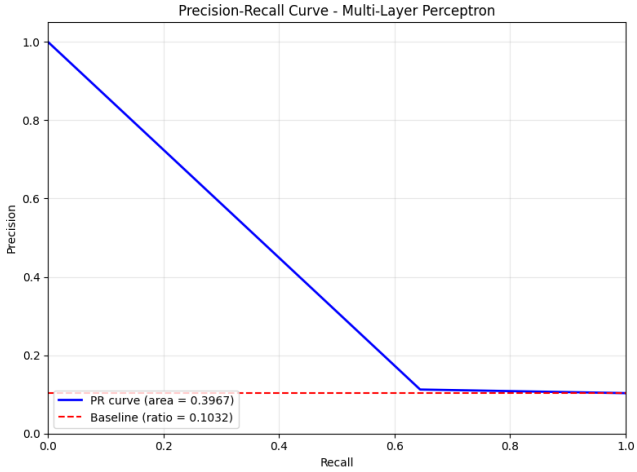


Figure 70: Precision recall Curve

Figure 71: Accuracy vs Epochs

5.8.2 Multi Neural Network (Processed)

The network used ReLU activation functions for the hidden layer, a sigmoid activation for the output layer, and was trained using the Adam optimizer with a learning rate of 0.001.

The Loss function was Binary-Cross Entropy (Suitable for unbalanced class representation)

Class/Metric	Precision	Recall	F1-Score	Support
0	0.89	0.21	0.34	63882
1	0.10	0.78	0.18	7355
Accuracy	–	–	0.27	71237
Macro Avg	0.50	0.50	0.26	71237
Weighted Avg	0.81	0.27	0.33	71237

Table 47: Regularized - Classification Report

Table 48: Performance Metrics

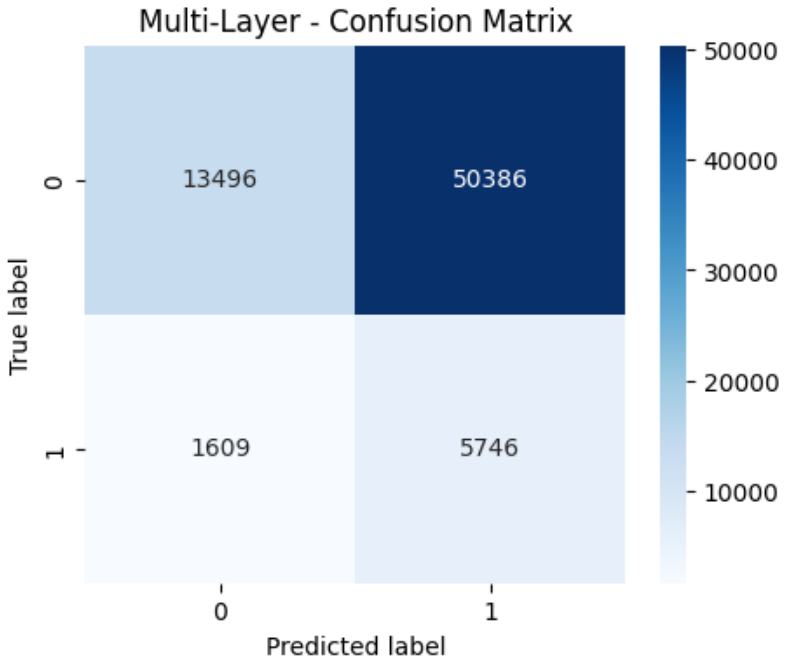


Table 49: Visual Analysis of Confusion Matrices

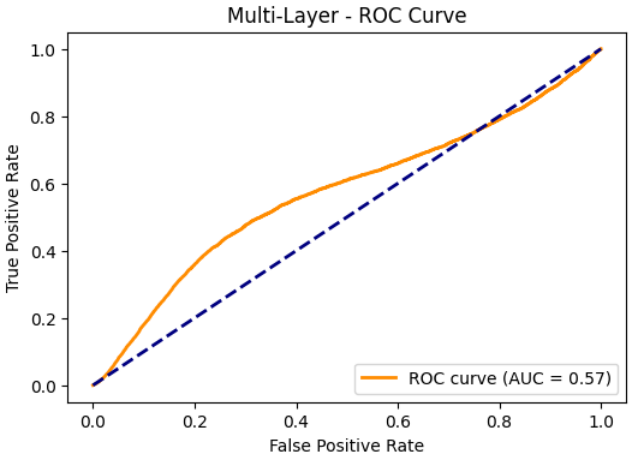


Figure 72: ROC Curve

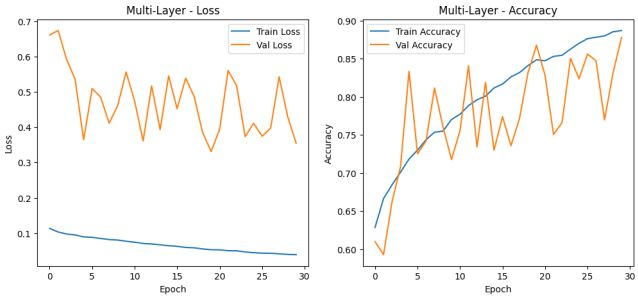


Figure 73: Precision recall Curve

OverSampling with PCA n = 50

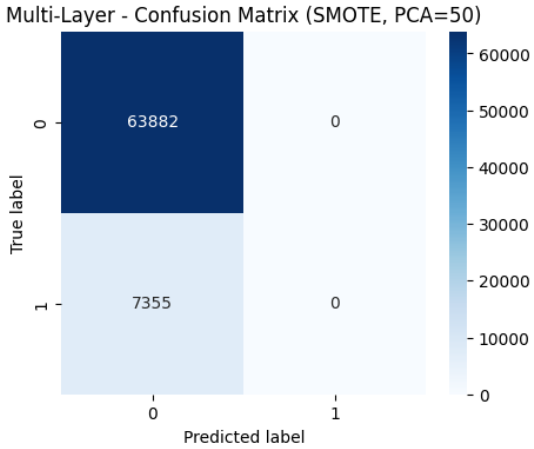


Figure 74: Multi-Layer - Confusion Matrix (SMOTE, PCA=50

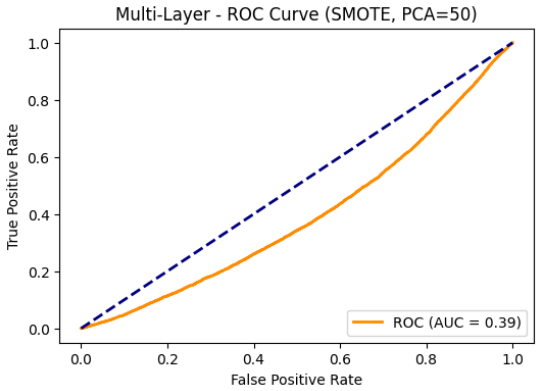


Figure 75: ROC Curve SMOTE PCA = 50

Class/Metric	Precision	Recall	F1-Score	Support
0	0.90	1.00	0.95	63882
1	0.00	0.00	0.00	7355
Accuracy	–	–	0.90	71237
Macro Avg	0.45	0.50	0.47	71237
Weighted Avg	0.80	0.90	0.85	71237

Table 50: Multi-Layer-PCA 50-SMOTE - Classification Report

OverSampling with PCA n = 100

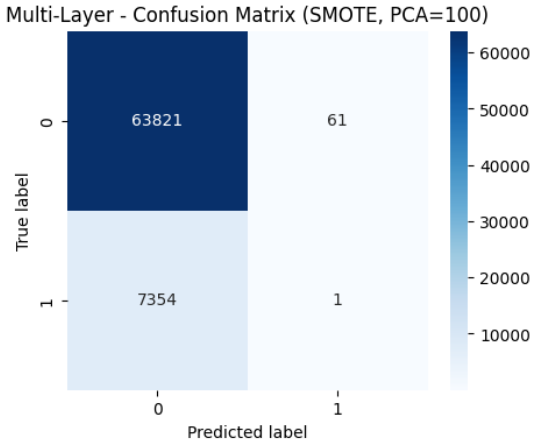


Figure 76: Multi-Layer - Confusion Matrix (SMOTE, PCA=100

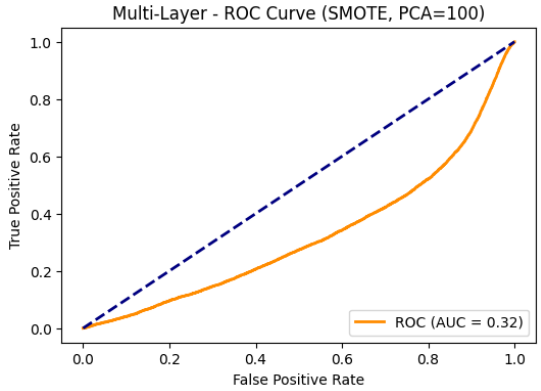


Figure 77: ROC Curve SMOTE PCA = 100

Class/Metric	Precision	Recall	F1-Score	Support
0	0.90	1.00	0.95	63882
1	0.02	0.00	0.00	7355
Accuracy	–	–	0.90	71237
Macro Avg	0.46	0.50	0.47	71237
Weighted Avg	0.81	0.90	0.85	71237

Table 51: Multi-Layer-PCA 100-SMOTE - Classification Report

OverSampling with PCA n = 200

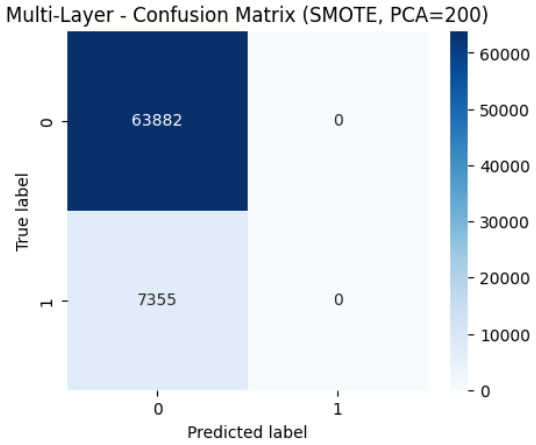


Figure 78: Multi-Layer - Confusion Matrix (SMOTE, PCA=200

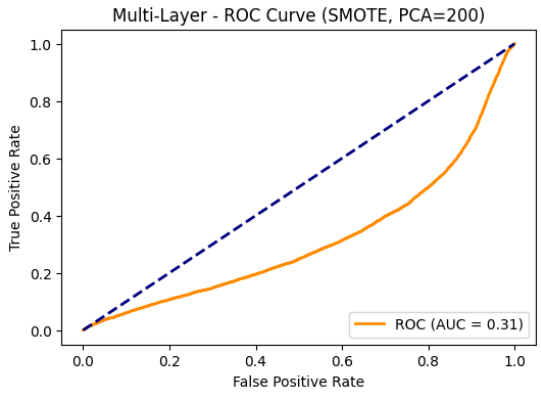


Figure 79: ROC Curve SMOTE PCA = 200

5.8.3 Conclusion

The experiments on the multi-layer neural network highlight several important observations regarding the impact of network complexity and data processing on performance.

- **Baseline Model Observations:** In its unprocessed form, the deep neural network—with two hidden layers containing 512 and 128 neurons respectively—demonstrates

Class/Metric	Precision	Recall	F1-Score	Support
0	0.90	1.00	0.95	63882
1	0.00	0.00	0.00	7355
Accuracy	–	–	0.90	71237
Macro Avg	0.45	0.50	0.47	71237
Weighted Avg	0.80	0.90	0.85	71237

Table 52: Multi-Layer-PCA 200-SMOTE - Classification Report

an overall accuracy of approximately 43.83%. While the recall for the minority class is moderately high (64.42%), the precision remains low at 11.24%, indicating that the network frequently misclassifies negative instances as positive. The corresponding confusion matrix and ROC/precision-recall curves further emphasize these limitations.

- **Effects of Processing and Regularization:** With the application of processing techniques (ReLU activations, dropout regularization at a rate of 0.3, batch normalization, and the use of the Adam optimizer with a binary cross-entropy loss function), the network's performance shifts. Despite improvements in balancing the learning process, the processed model still exhibits similar challenges found in the baseline, as indicated by the classification report where overall accuracy drops to 27%. This suggests that even with regularization, the inherent imbalance remains a major factor affecting precision and overall model reliability.
- **Oversampling with SMOTE Coupled with PCA:** Incorporating SMOTE for oversampling—together with dimensionality reduction via PCA—produces varied results depending on the number of principal components:
  - **PCA with 50 Components:** The network achieves very high accuracy (approximately 90%), as nearly all samples are classified as the majority class. However, the metrics indicate that the network fails entirely to identify the minority class (with a precision, recall, and F1-score of 0 for class 1).
  - **PCA with 100 Components:** Similar trends are observed where overall accuracy remains high but the network again struggles to recover any meaningful signal from the minority class. This configuration yields a slight variation in reported metrics, yet still emphasizes the dominance of the majority class.
  - **PCA with 200 Components:** Increasing the dimensionality through PCA leads to metrics that closely mirror those observed with 50 and 100 components; a persistent pattern emerges where the model overwhelmingly classifies samples into the majority class, resulting in high accuracy at the cost of completely neglecting the minority class.
- **Key Takeaways:** The multi-layer neural network's performance reaffirms the challenges inherent in dealing with imbalanced datasets. Even advanced regularization

and oversampling approaches may improve global metrics such as accuracy; however, they often do so by simply reinforcing the bias toward the majority class. The inability to accurately detect the minority class remains a critical limitation. These findings underscore the need for alternative strategies—potentially involving more sophisticated data augmentation, cost-sensitive learning, or architectural modifications—to effectively balance precision and recall.

Overall, while the multi-layer structure offers greater capacity for capturing complex relationships in data, addressing dataset imbalance continues to be the pivotal challenge. Future work should focus on integrating advanced imbalance mitigation techniques to achieve a more balanced performance across all classes.

5.9 Deep Neural Network

5.9.1 Deep Neural Network without processing

Table 53: Performance Metrics

Deep Neural Network	
Accuracy	0.3997
Precision	0.1088
Recall	**0.6697**
F1 Score	0.1872
Confusion Matrix:	
[23545	40337]
[2429	4926]

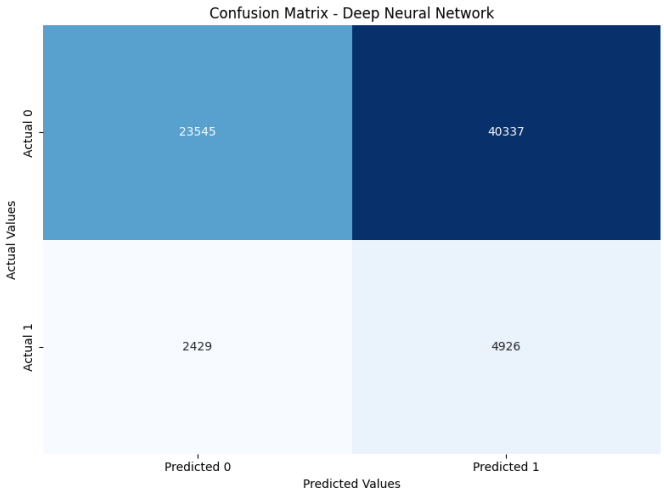


Table 54: Visual Analysis of Confusion Matrices

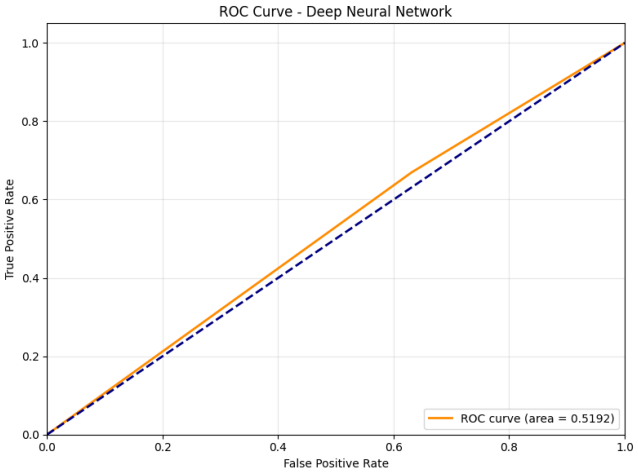


Figure 80: ROC Curve

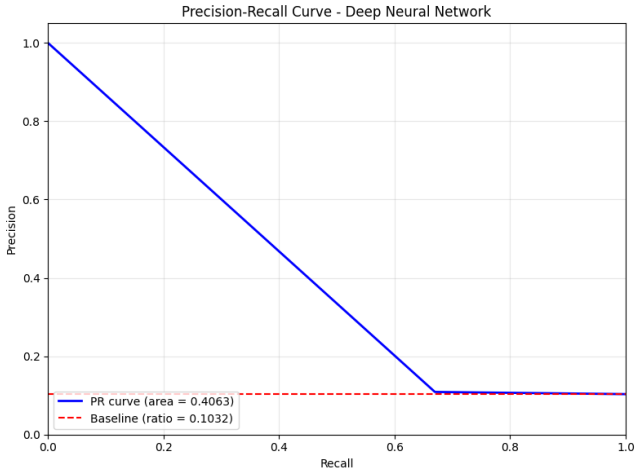


Figure 81: Precision recall Curve

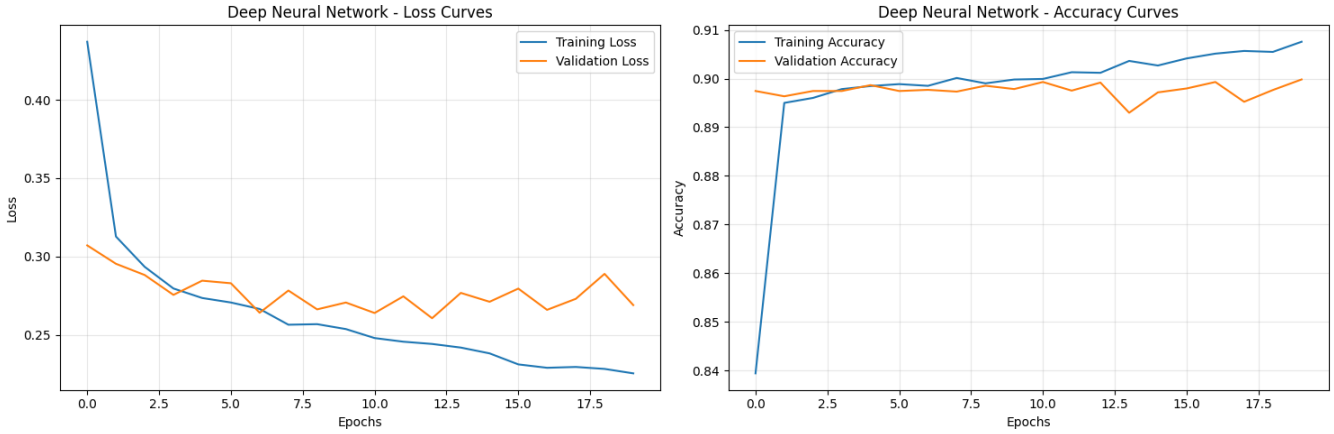


Figure 82: Accuracy vs Epochs

5.9.2 Deep Neural Network (Processed)

The network used ReLU activation functions for the hidden layer, a sigmoid activation for the output layer, and was trained using the Adam optimizer with a learning rate of 0.001. The Loss function was Binary-Cross Entropy (Suitable for unbalanced class representation)

Class/Metric	Precision	Recall	F1-Score	Support
0	0.89	0.23	0.37	63882
1	0.10	0.74	0.18	7355
Accuracy	–	–	0.29	71237
Macro Avg	0.49	0.49	0.27	71237
Weighted Avg	0.81	0.29	0.35	71237

Table 55: Regularized - Classification Report

Table 56: Performance Metrics

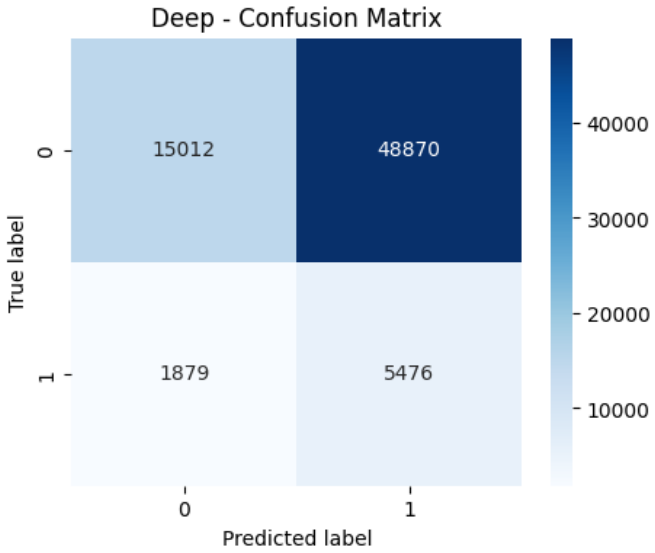


Table 57: Visual Analysis of Confusion Matrices

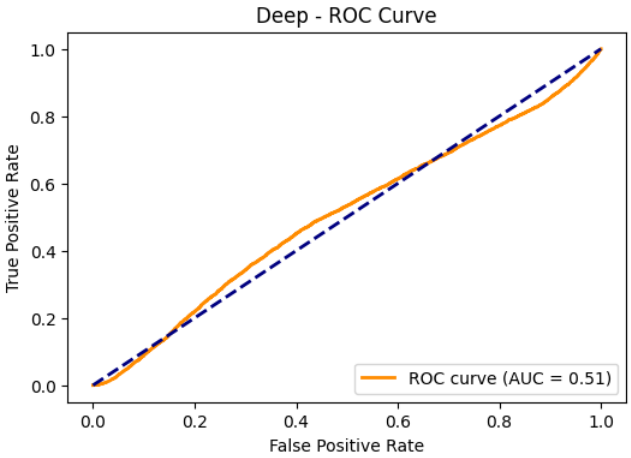


Figure 83: ROC Curve

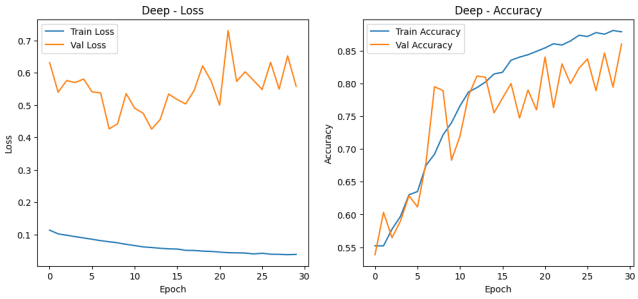


Figure 84: Precision recall Curve

5.10 Artifical Neural Network

5.10.1 Regularized Artificial Neural Network without processing

Table 58: Performance Metrics

Artificial Neural Network

Accuracy	0.8968
Precision	**0.0000**
Recall	**0.0000**
F1 Score	**0.0000**

Confusion Matrix:
[63882 0]
[7355 0]

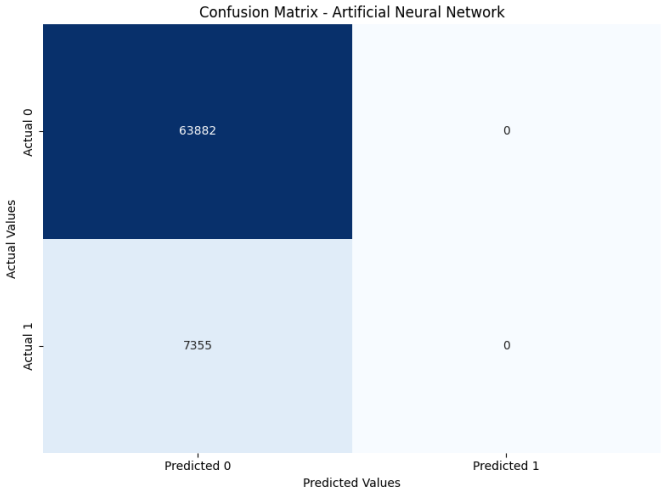


Table 59: Visual Analysis of Confusion Matrices

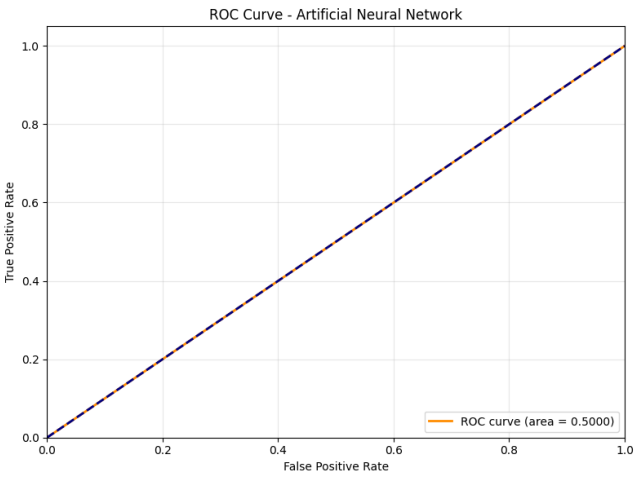


Figure 85: ROC Curve

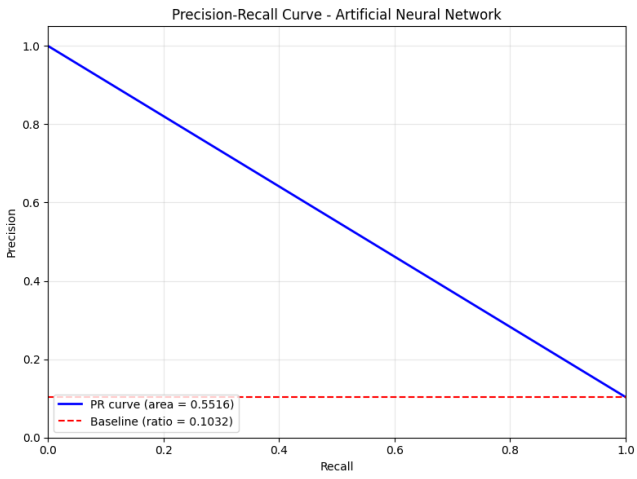


Figure 86: Precision recall Curve

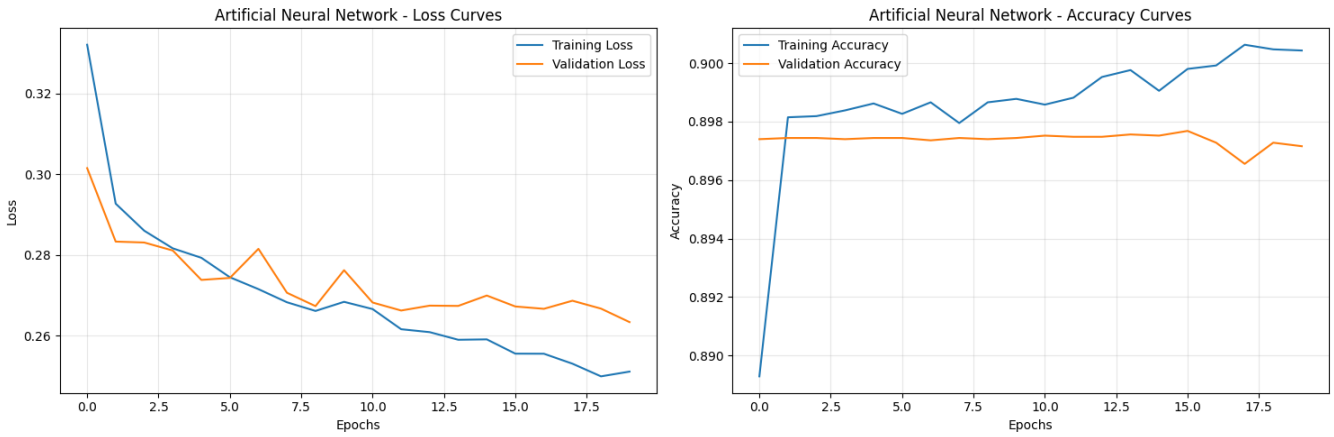


Figure 87: Accuracy vs Epochs

5.10.2 Regularized Artificial Neural Network (Processed)

The network used ReLU activation functions for the hidden layer, a sigmoid activation for the output layer, and was trained using the Adam optimizer with a learning rate of 0.001.

The Loss function was Binary-Cross Entropy (Suitable for unbalanced class representation)

Class/Metric	Precision	Recall	F1-Score	Support
0	0.90	1.00	0.95	63882
1	0.00	0.00	0.00	7355
Accuracy	–	–	0.90	71237
Macro Avg	0.45	0.50	0.47	71237
Weighted Avg	0.80	0.90	0.85	71237

Table 60: Regularized - Classification Report

Table 61: Performance Metrics

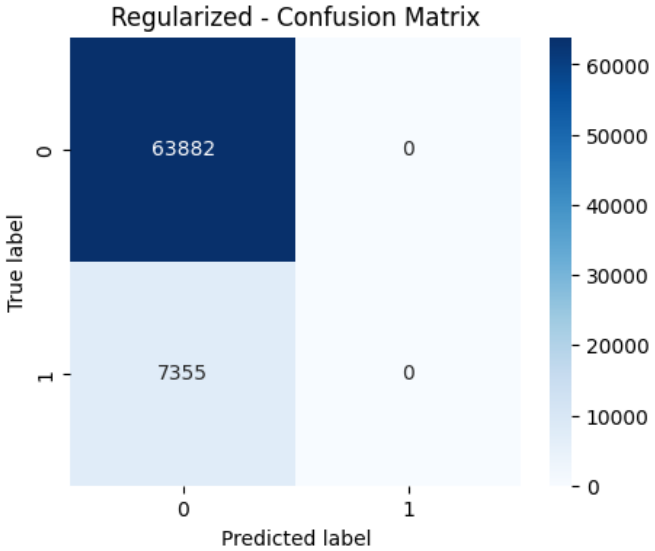


Table 62: Visual Analysis of Confusion Matrices

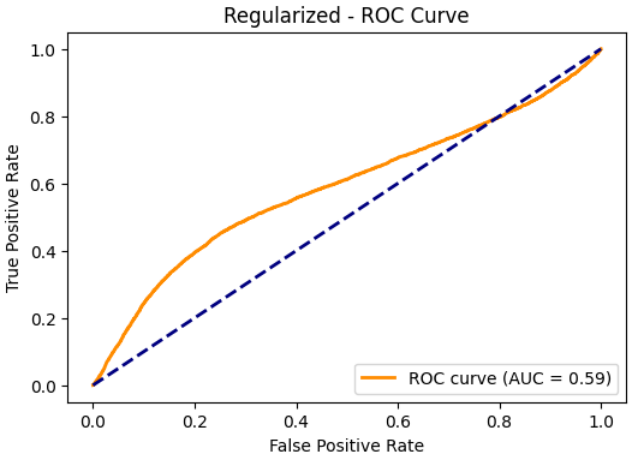


Figure 88: ROC Curve

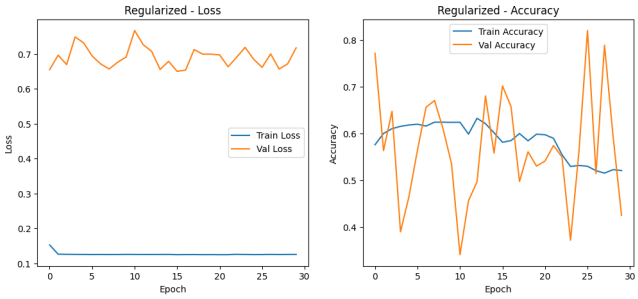


Figure 89: Precision recall Curve

5.11 Conclusion for DNN & ANN

Despite performing PCA for dimensionality reduction and applying data balancing techniques, the results of both models remained unsatisfactory. In the case of the Deep Neural Network, although the processed model showed some improvement in recall, the overall performance was hindered by low precision and F1 scores—indicating that the model struggled to effectively differentiate between classes. Similarly, the Regularized Artificial Neural Network, while achieving high accuracy on the dominant class, failed to detect any instances of the minority class, as reflected by precision, recall, and F1 scores of zero for that class. These outcomes suggest that even with preprocessing efforts, the inherent data imbalance led to models that predominantly predicted one class, lacking the discriminative power necessary for robust classification.

5.12 AdaBoost

AdaBoost (Adaptive Boosting) is an ensemble learning algorithm that combines multiple weak learners (typically decision stumps) into a strong classifier.

5.12.1 How AdaBoost Works

- Starts with equal weights assigned to all training examples.
- Trains a weak learner (usually a simple decision tree with depth=1).
- Increases the weights of misclassified examples so the next learner focuses more on the “hard” cases.
- Repeats this process for several iterations.
- Final prediction is a weighted vote of all weak learners.

- In Each Iteration a weak classifier is trained.
- Error is calculated.
- The model gives higher weight to misclassified instances.
- The final model is a combination of all the weak classifiers with different weights.

5.12.2 AdaBoost without processing

- AdaBoost on original data performs poorly for the minority class.
- High accuracy is misleading and doesn't reflect real performance.
- The model completely ignores the positive class in the test set (recall = 0).
- Indicates strong bias toward the majority class and lack of generalization for minority detection.

Training		Validation		Test (Evaluation)	
Accuracy	0.9022	Accuracy	0.8914	Accuracy	0.8968
F1 Score	0.2051	F1 Score	0.1054	F1 Score	0.0000
Recall	**0.1240**	Recall	**0.0624**	Recall	**0.0000**
EER	0.2158	EER	0.3139	EER	0.5423
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[22579 221]		[21986 310]		[63882 0]	
[2260 320]		[2389 159]		[7355 0]	

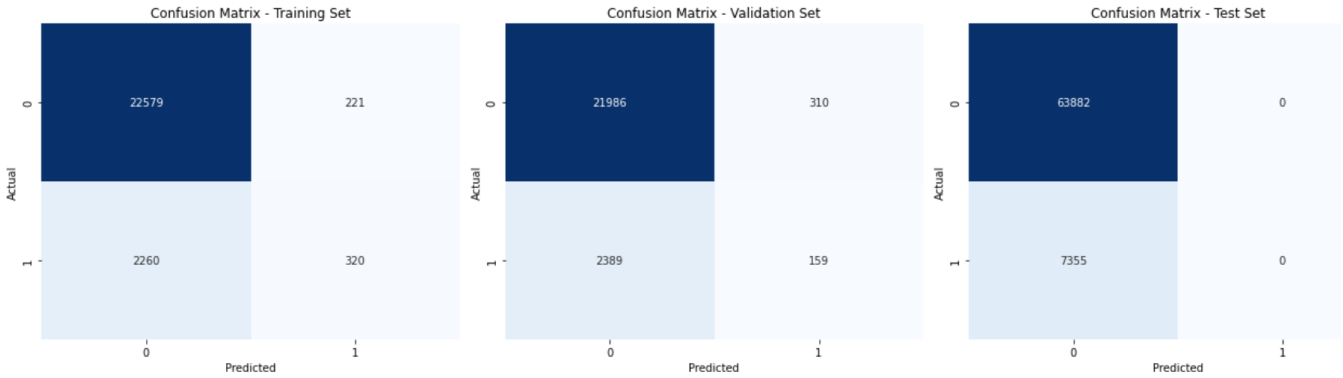


Figure 90: Visualization of the AdaBoost model performance

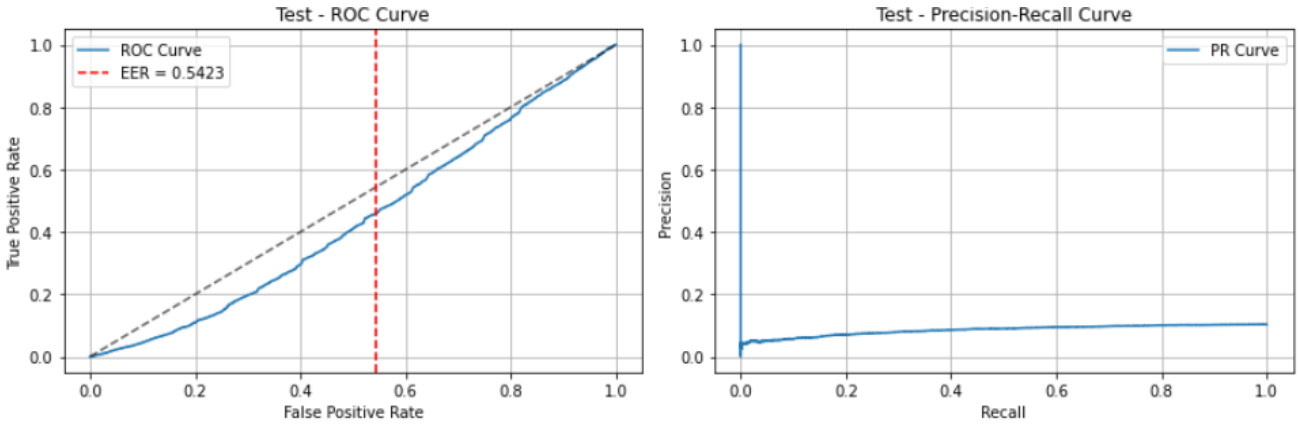


Figure 91: ROC and Precision-Recall Curve

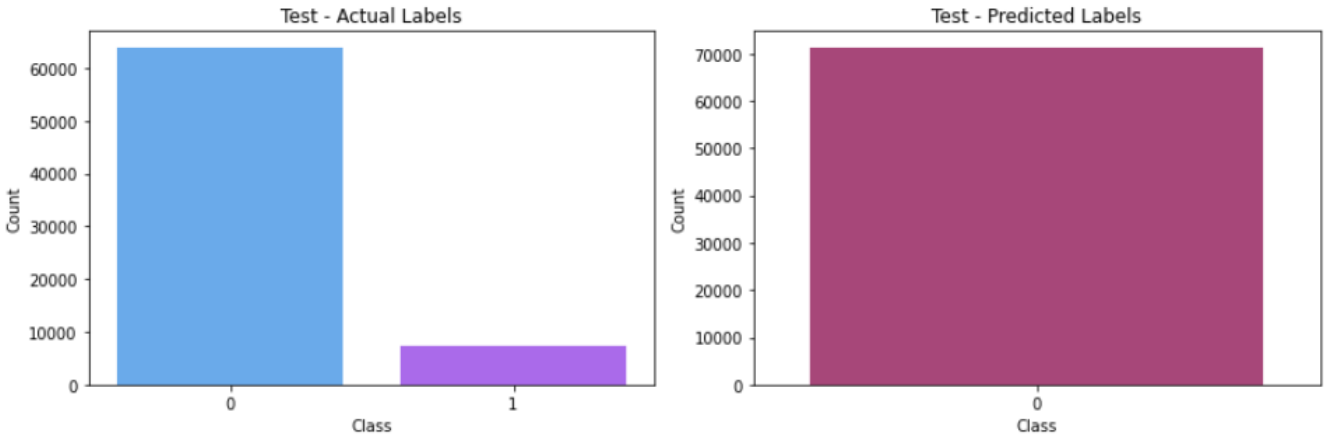


Figure 92: Actual vs Predicted

5.12.3 AdaBoost (Processed)

Over Sampling

- The AdaBoost model showed best performance on the training set and decent results on the validation set, especially in terms of recall.
- However, on the test set, the model failed to generalize and detect the minority class, with a recall of just 0.03% and an F1 score near zero.
- The synthetic data led to overfitting, causing the model to misclassify real minority instances.

Table 63: Performance Metrics (Training, Validation, Test)

Training		Validation		Test (Evaluation)	
Accuracy	0.7444	Accuracy	0.6148	Accuracy	0.8945
F1 Score	0.7764	F1 Score	0.2719	F1 Score	0.0005
Recall	0.8876	Recall	0.7013	Recall	0.0003
EER	0.2502	EER	0.3476	EER	0.5766
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[13708 9092]		[13488 8808]		[63716 166]	
[2563 20237]		[761 1787]		[7353 2]	

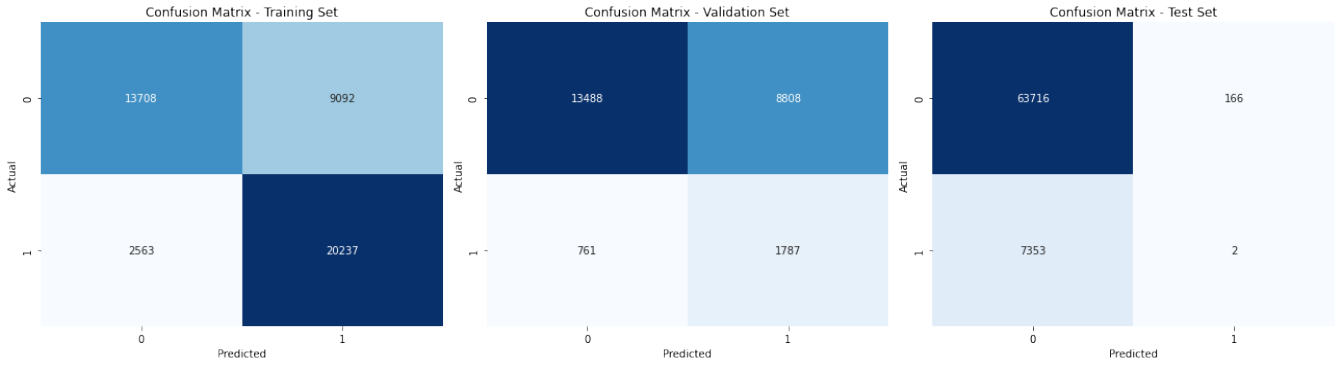


Figure 93: Visualization of Confusion matrices

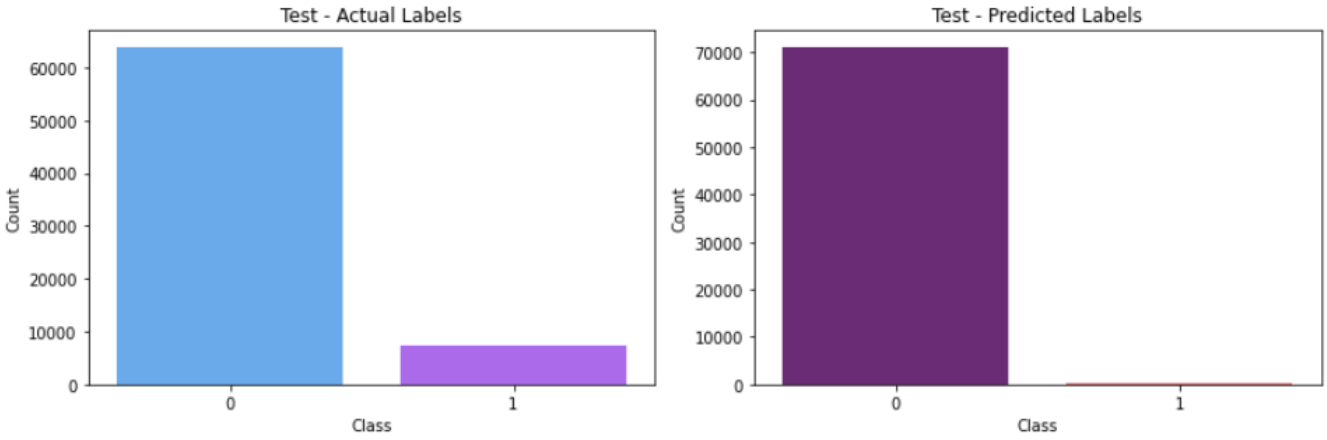


Figure 94: Actual vs predicted

Under sampling

- Undersampling balances the dataset by removing random samples from the majority class.
- The model failed to generalize to the test data, achieving a recall of just 1.33% and an F1 score of 0.0245.
- This suggests that the model, while effective on the balanced training data, is still biased toward the majority class when faced with the original test distribution.
- The high test accuracy is misleading, as it reflects mostly correct predictions on the majority class.
- Undersampling may have removed valuable information, leading to weak performance on unseen data.

Label 0 2580
Label 1 2580

Table 64: Performance Metrics (Training, Validation, Test)

Training		Validation		Test (Evaluation)	
Accuracy	0.7207	Accuracy	0.6464	Accuracy	0.8906
F1 Score	0.7347	F1 Score	0.2859	F1 Score	0.0245
Recall	0.7733	Recall	0.6903	Recall	0.0133
EER	0.2787	EER	0.3377	EER	0.4505
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[1724 856]		[14299 7997]		[63346 536]	
[585 1995]		[789 1759]		[7257 98]	

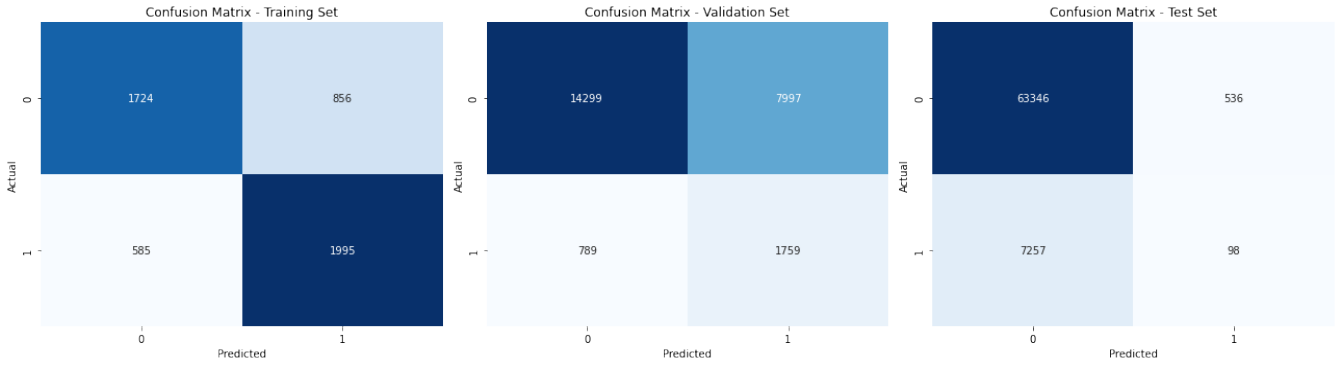


Figure 95: Visualization of Confusion matrices

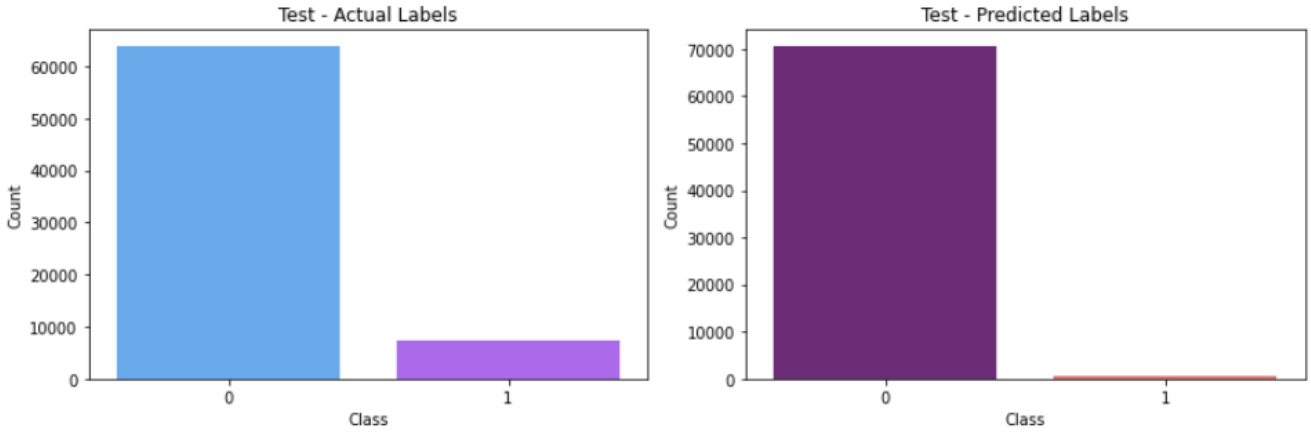
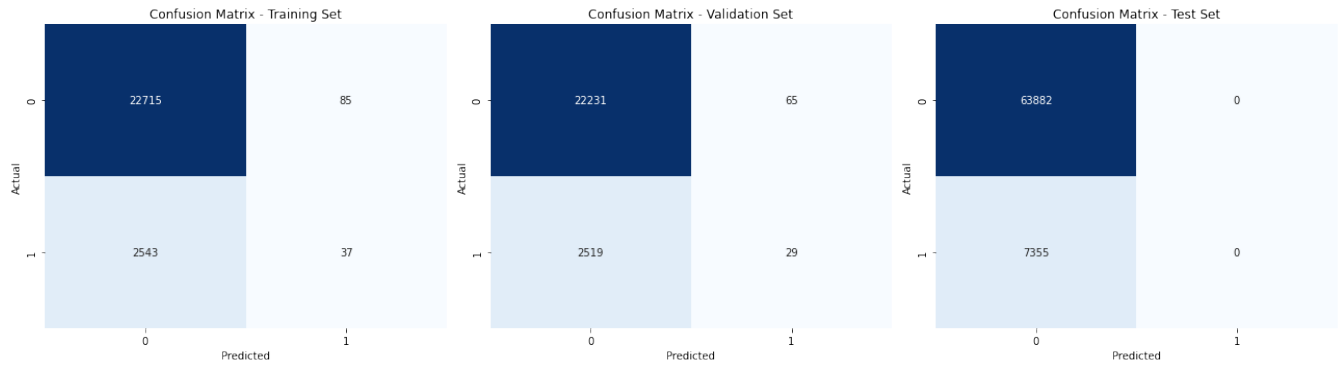


Figure 96: Actual vs predicted

PCA reduction(n=50)

Table 65: Performance Metrics (Training, Validation, Test)

Training		Validation		Test (Evaluation)	
Accuracy	0.8965	Accuracy	0.8960	Accuracy	0.8968
F1 Score	0.0274	F1 Score	0.0220	F1 Score	0.0000
Recall	0.0143	Recall	0.0114	Recall	0.0000
EER	0.2947	EER	0.3270	EER	0.4570
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[22715 85]		[22231 65]		[63882 0]	
[2543 37]		[2519 29]		[7355 0]	



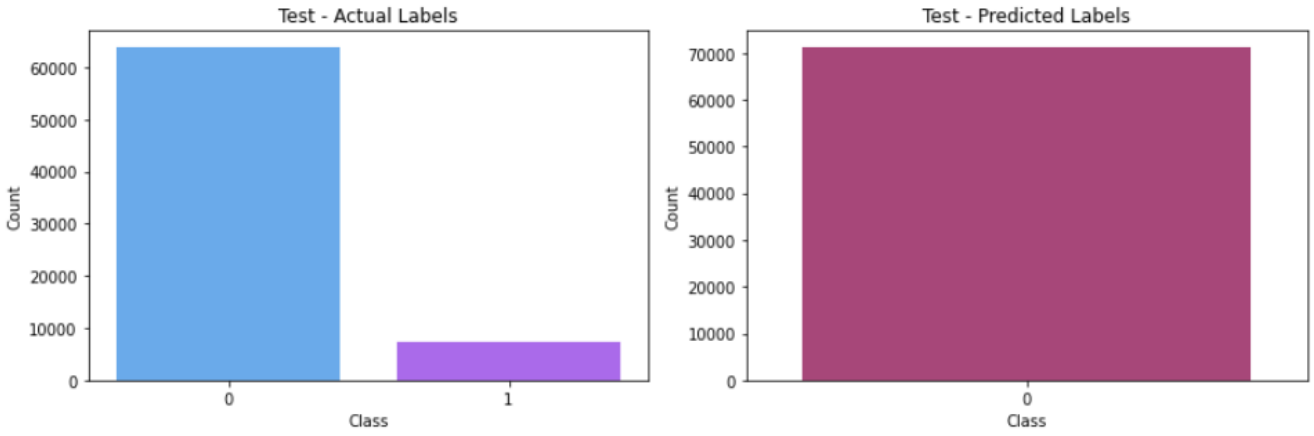


Figure 97: Actual vs predicted

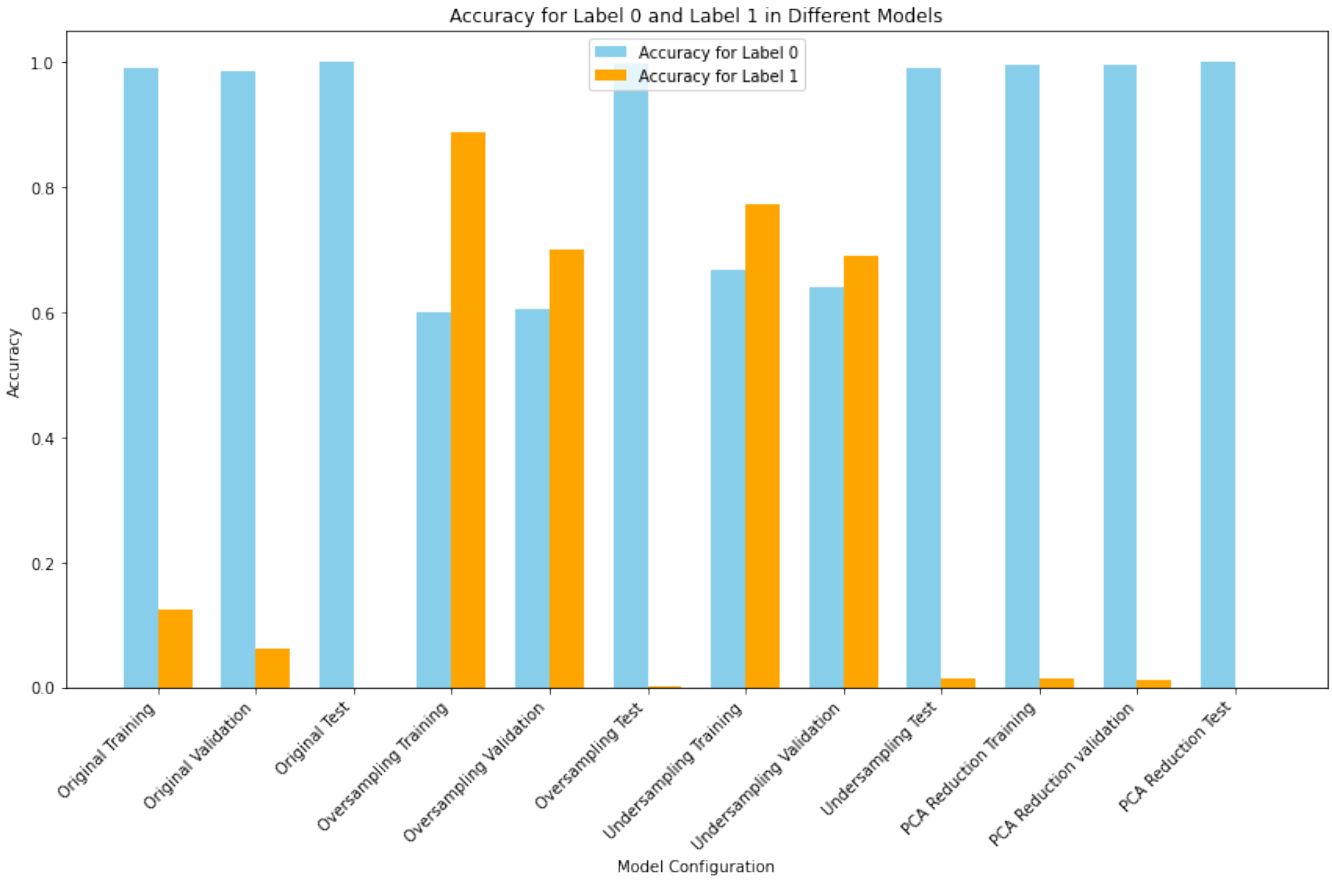


Figure 98: Acuuracy for label 0 and 1

5.13 Gradient Boosting Machine (GBM)

5.13.1 Gradient Boosting Machine without processing

Gradient Boosting builds an ensemble of weak prediction models, typically decision trees, in a stage-wise manner. Each new model is trained to correct the errors of the combined existing models.

Our GBM implementation used 50 estimators with a maximum depth of 4 and a learning rate of 0.1.

Table 66: Performance Metrics (Training, Validation, Test)

Training		Validation		Test (Evaluation)	
Accuracy	0.8993	Accuracy	0.8975	Accuracy	0.8881
F1 Score	0.0177	F1 Score	0.0024	F1 Score	0.0545
Recall	0.0089	Recall	0.0012	Recall	0.0313
EER	0.2379	EER	0.3258	EER	0.3624
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[22800 0]		[22295 1]		[63033 849]	
[2557 23]		[2545 3]		[7125 230]	

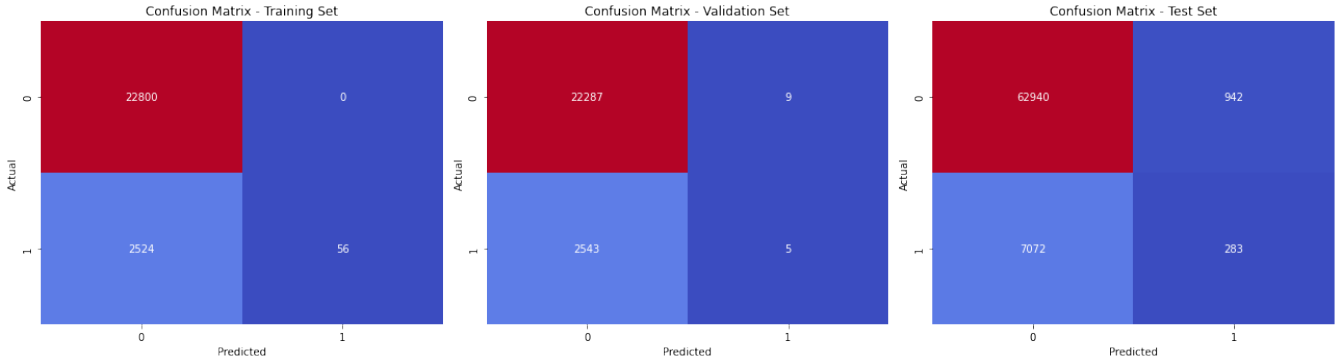


Figure 99: Visual Analysis of Gradient Boost Confusion Matrices

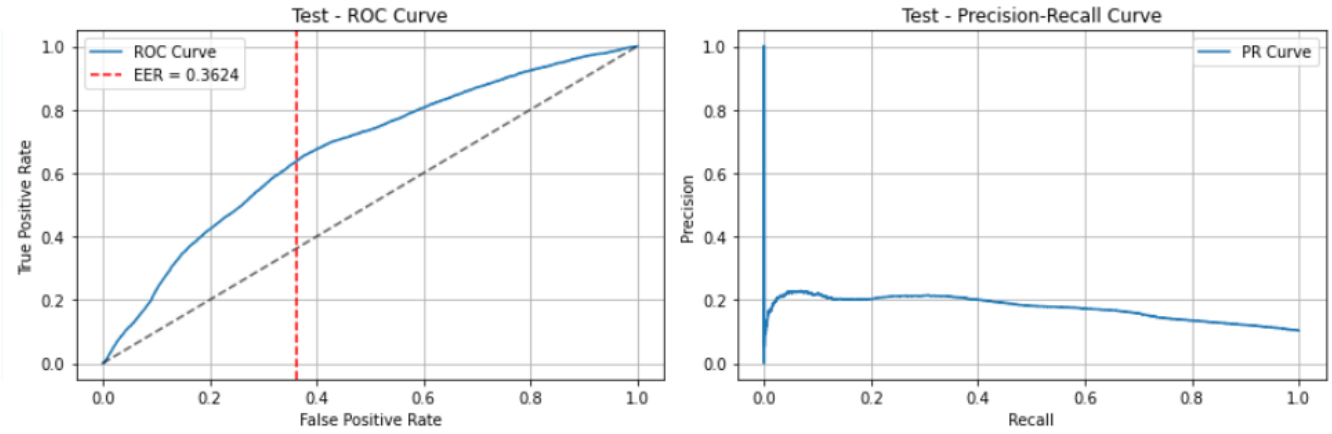


Figure 100: ROC and Precision-Recall Curve

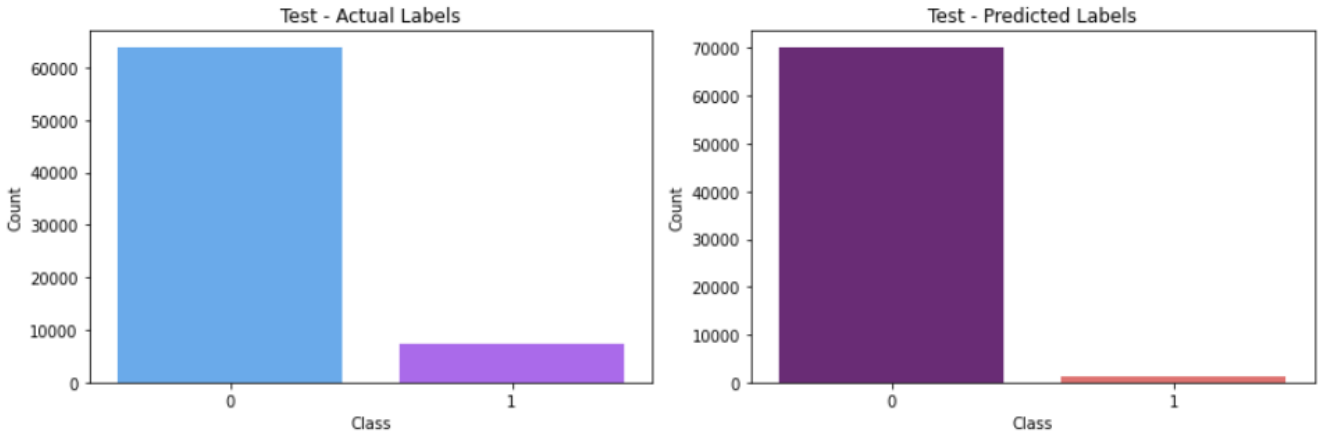


Figure 101: Actual vs Predicted labels

5.13.2 Gradient Boosting Machine (Processed)

Under Sampling

label 0	2580
label 1	2580

Table 67: Performance Metrics (Training, Validation, Test)

Training		Validation		Test (Evaluation)	
Accuracy	0.8529	Accuracy	0.6793	Accuracy	0.8945
F1 Score	0.8573	F1 Score	0.3088	F1 Score	0.0103
Recall	0.8837	Recall	0.6986	Recall	0.0053
EER	0.1469	EER	0.3258	EER	0.3959
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[2121 459]		[15096 7200]		[63683 199]	
[300 2280]		[768 1780]		[7316 39]	

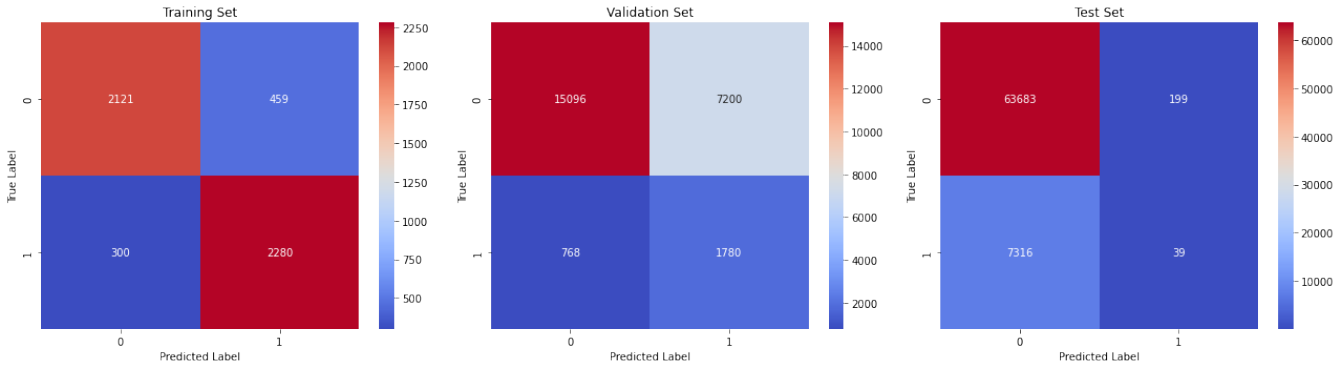


Figure 102: Visualization of Confusion matrices

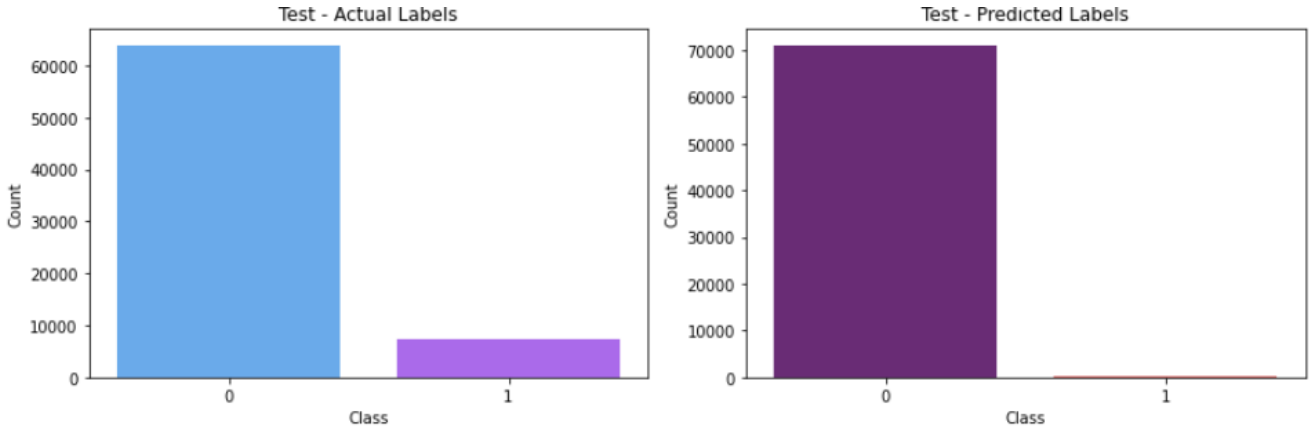


Figure 103: Actual vs Predicted

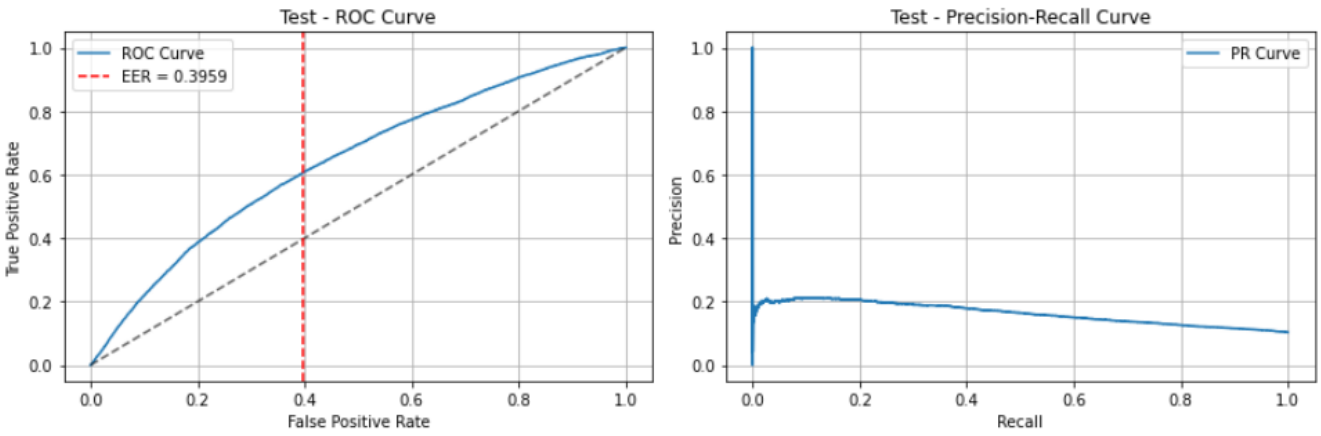


Figure 104: ROC and Precision-Recall Curve

Observation: After applying **undersampling**, the training performance for the minority class significantly improved (F1 Score: 0.8573, Recall: 0.8837), indicating that the model learned to detect the minority class better. However, poor performance on the validation and test sets (Test F1 Score: 0.0103, Recall: 0.0053) suggests that the model is **overfitting** to the undersampled training data and fails to generalize to the original distribution.

Over Sampling(SMOTE)

label 0	22800
label 1	22800

Table 68: Performance Metrics (Training, Validation, Test)

Training		Validation		Test (Evaluation)	
Accuracy	0.7848	Accuracy	0.7073	Accuracy	0.8967
F1 Score	0.7979	F1 Score	0.2944	F1 Score	0.0003
Recall	0.8499	Recall	0.5954	Recall	0.0001
EER	0.2205	EER	0.3323	EER	0.5981
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[16408 6392]		[16054 6242]		[63877 5]	
[3422 19378]		[1031 1517]		[7354 1]	

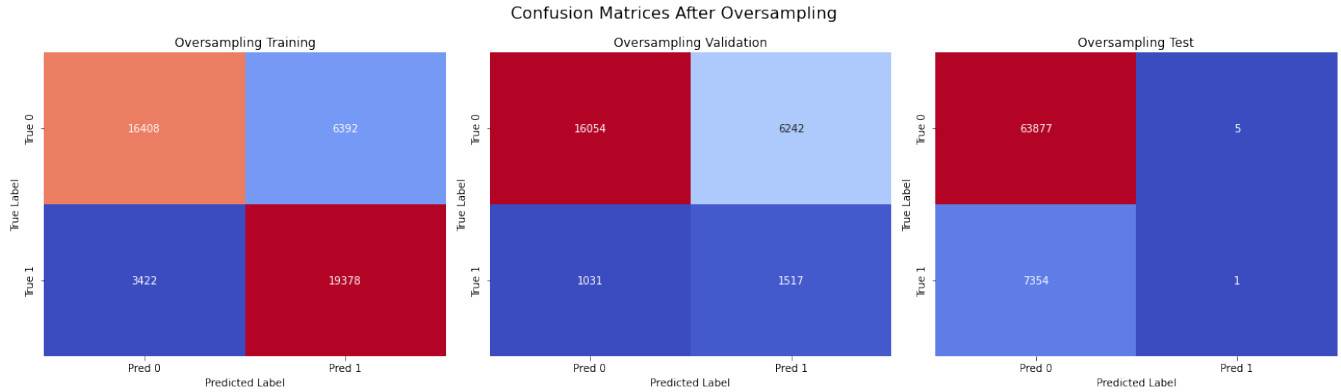


Figure 105: Visualization of Confusion matrices

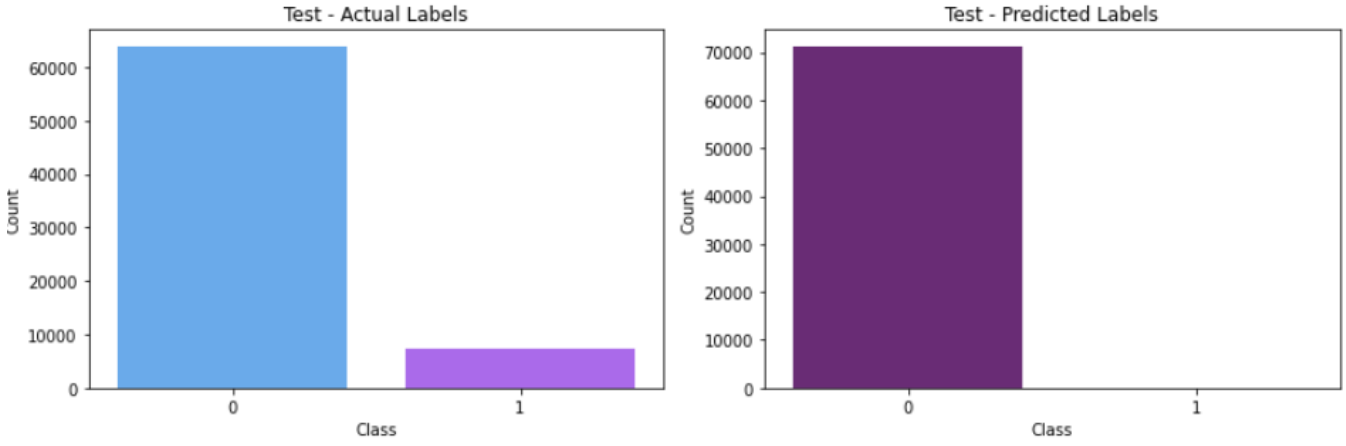


Figure 106: Actual vs predicted

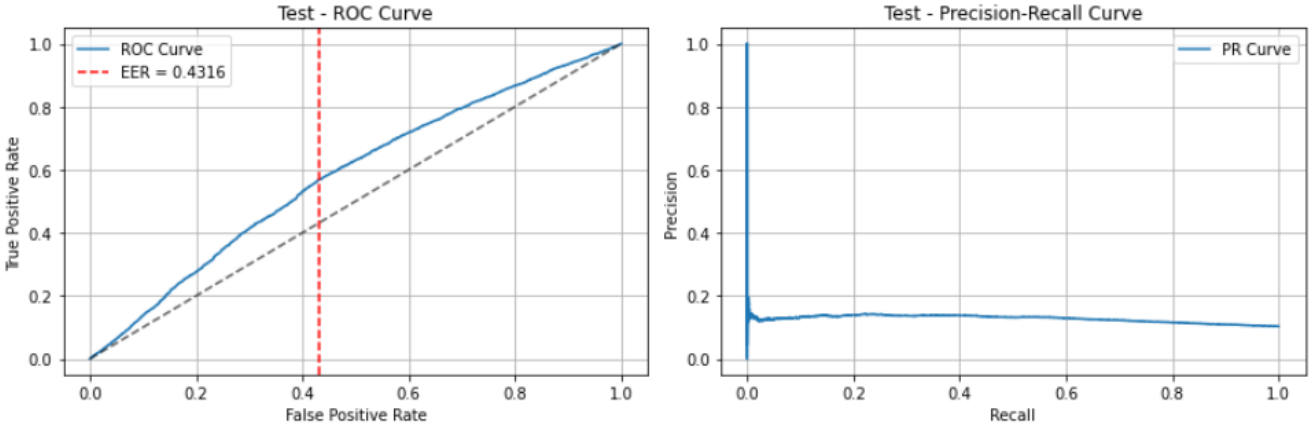


Figure 107: ROC and Precision-Recall Curve

SMOTE generates new samples for the minority class by interpolating between existing ones. If the original minority class data is limited, noisy, or not diverse, the generated samples may not accurately reflect the true distribution of the class. This can lead the model to **overfit** on synthetic patterns rather than learning generalizable features.

Class Weighting

label 0 22800
label 1 2580

Table 69: Performance Metrics (Training, Validation, Test)

Training		Validation		Test (Evaluation)	
Accuracy	0.6881	Accuracy	0.6805	Accuracy	0.8960
F1 Score	0.3336	F1 Score	0.2928	F1 Score	0.0035
Recall	0.7682	Recall	0.6448	Recall	0.0018
EER	0.2819	EER	0.3347	EER	0.4697
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[15481 7319]		[15263 7033]		[63815 67]	
[598 1982]		[905 1643]		[7342 13]	

Weight balancing improves learning on the minority class in the training set, but if the features aren't generalizable (i.e., they don't capture real differences between classes), the model may not perform well on unseen test samples. Overfitting to Minority Class in Training

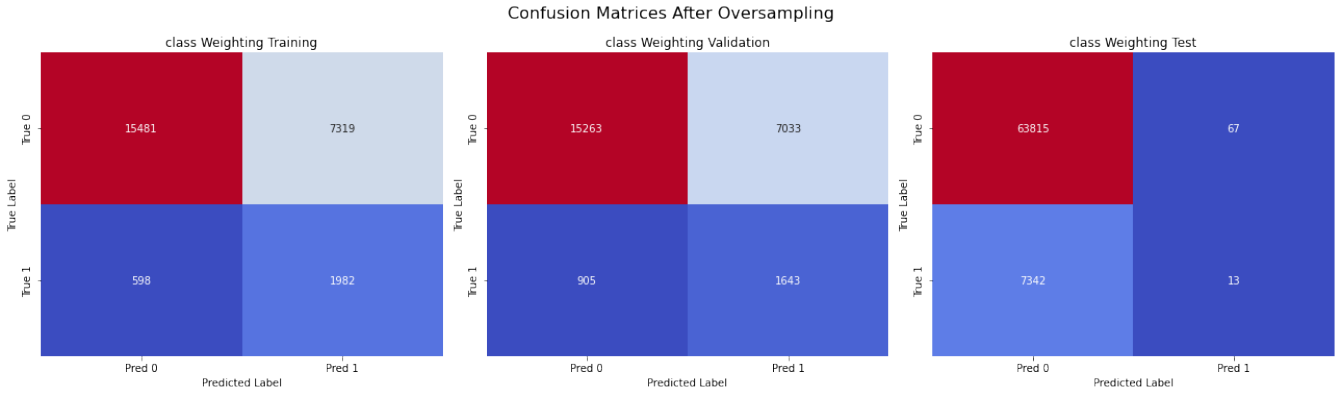


Figure 108: Visualization of Confusion matrices

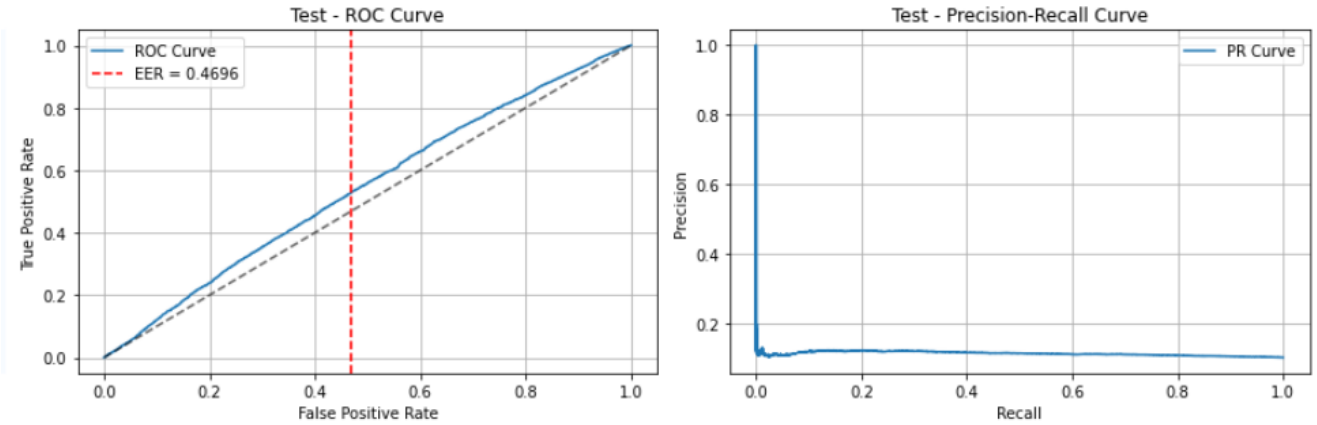


Figure 109: ROC and Precision-recall Curve

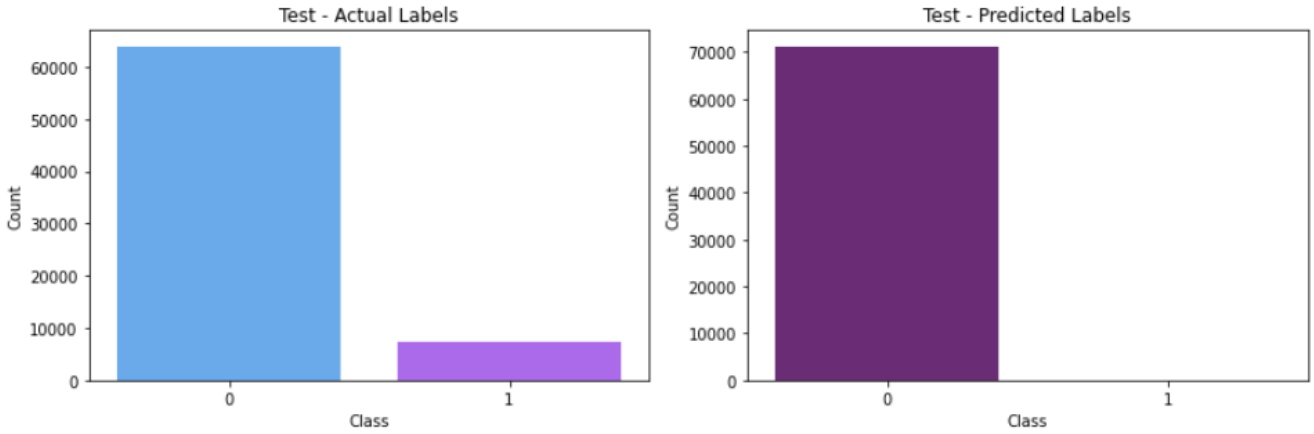


Figure 110: Actual vs Predicted labels Curve

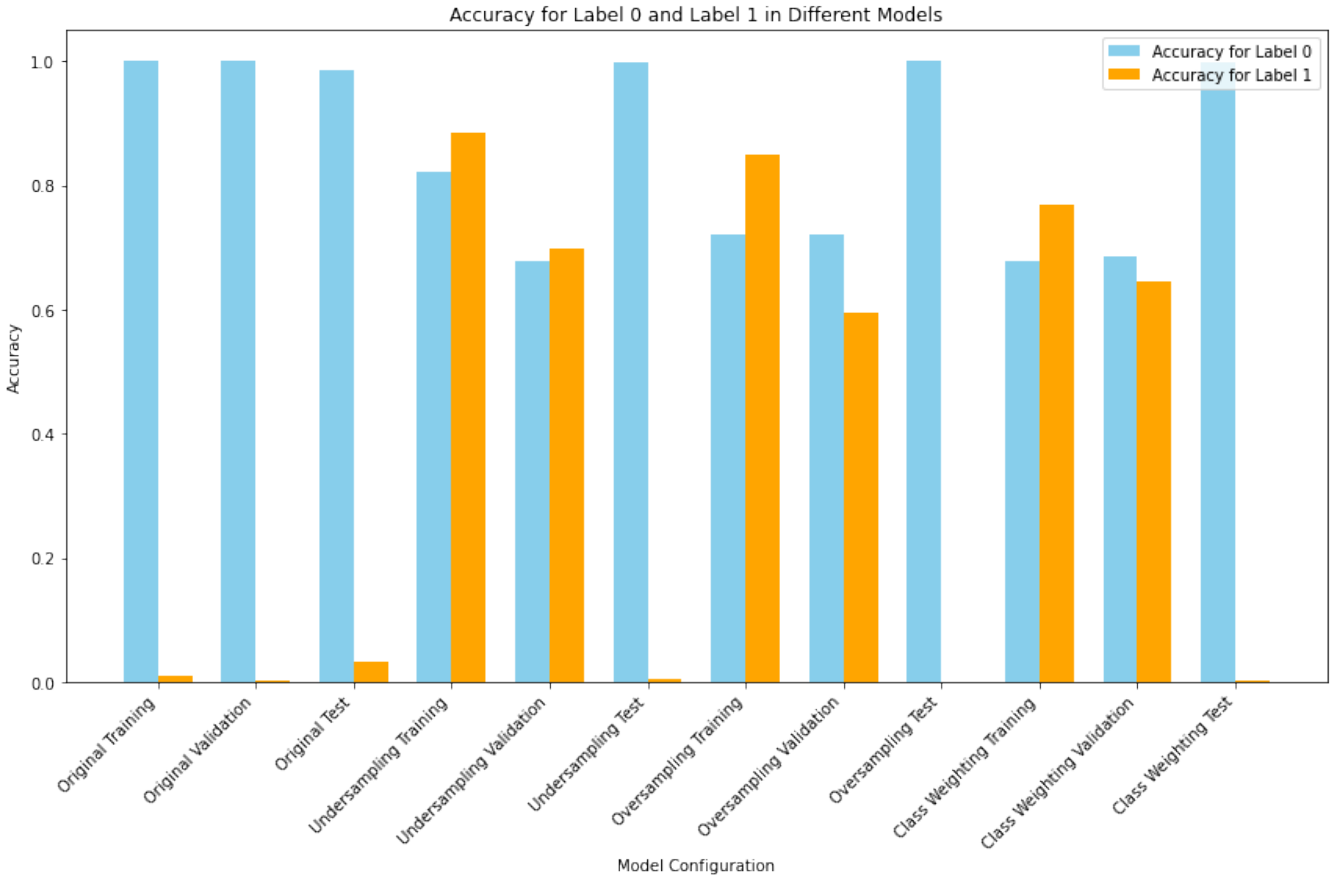


Figure 111: Accuarcy

5.14 XGBoost

5.14.1 XGBoost without processing

XGBoost is an optimized distributed gradient boosting library designed for efficiency and performance. It implements machine learning algorithms under the Gradient Boosting framework with additional regularization to prevent overfitting. We configured XGBoost with 100 estimators, maximum depth of 4, learning rate of 0.1, and L2 regularization parameter of 1.0.

Training		Validation		Test (Evaluation)	
Accuracy	0.9065	Accuracy	0.8974	Accuracy	0.8968
F1 Score	0.1492	F1 Score	0.0109	F1 Score	0.0000
Recall	**0.0806**	Recall	**0.0055**	Recall	**0.0000**
EER	0.1403	EER	0.2947	EER	0.3880
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[22800 0]		[22282 14]		[63882 0]	
[2372 208]		[2534 14]		[7355 0]	

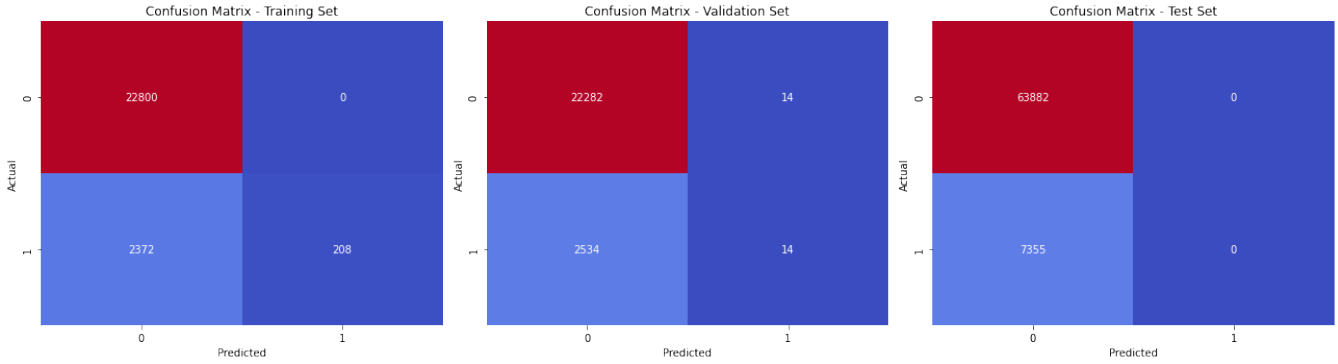


Figure 112: Visual Analysis of XGBOOST Confusion Matrices

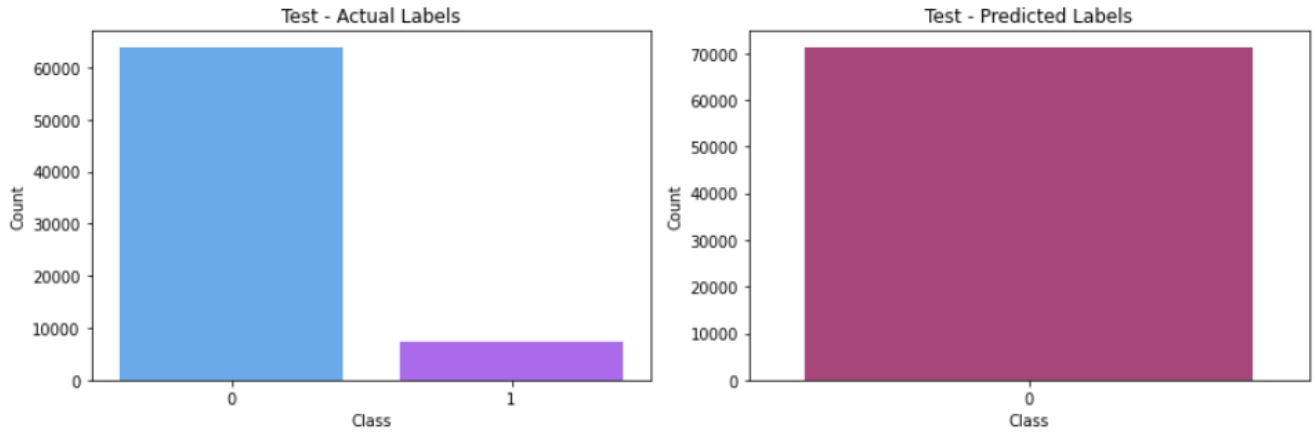


Figure 113: Actual vs Predicted Labels

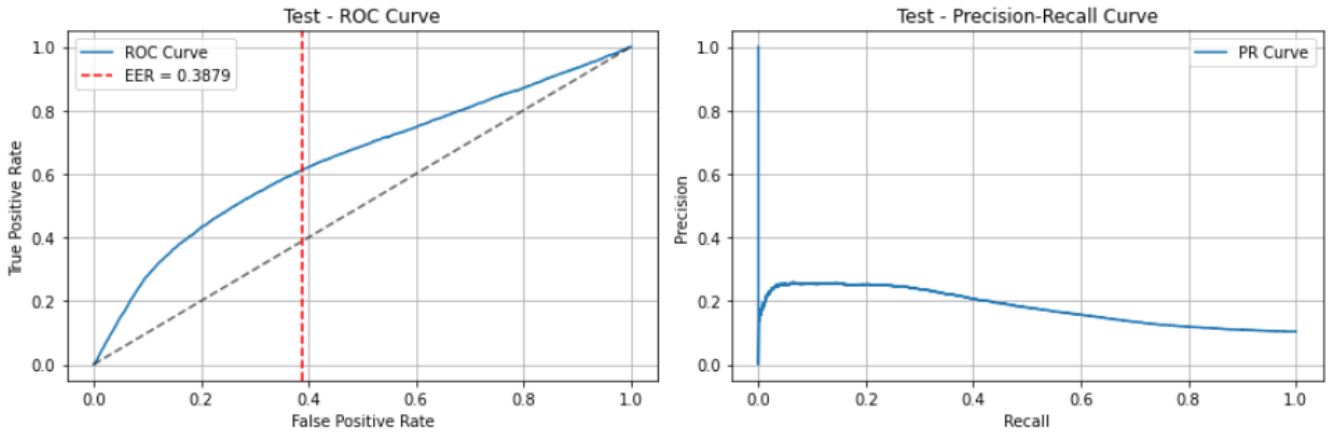


Figure 114: ROC and Precision-Recall Curve

5.14.2 XGBoost (Processed)

Oversampling (SMOTE)

Training		Validation		Test (Evaluation)	
Accuracy	0.8872	Accuracy	0.7829	Accuracy	0.8968
F1 Score	0.8922	F1 Score	0.3238	F1 Score	0.0000
Recall	0.9334	Recall	0.5067	Recall	0.0000
EER	0.1073	EER	0.3090	EER	0.5878
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[19174 3626]		[18160 4136]		[63882 0]	
[1519 21281]		[1257 1291]		[7355 0]	

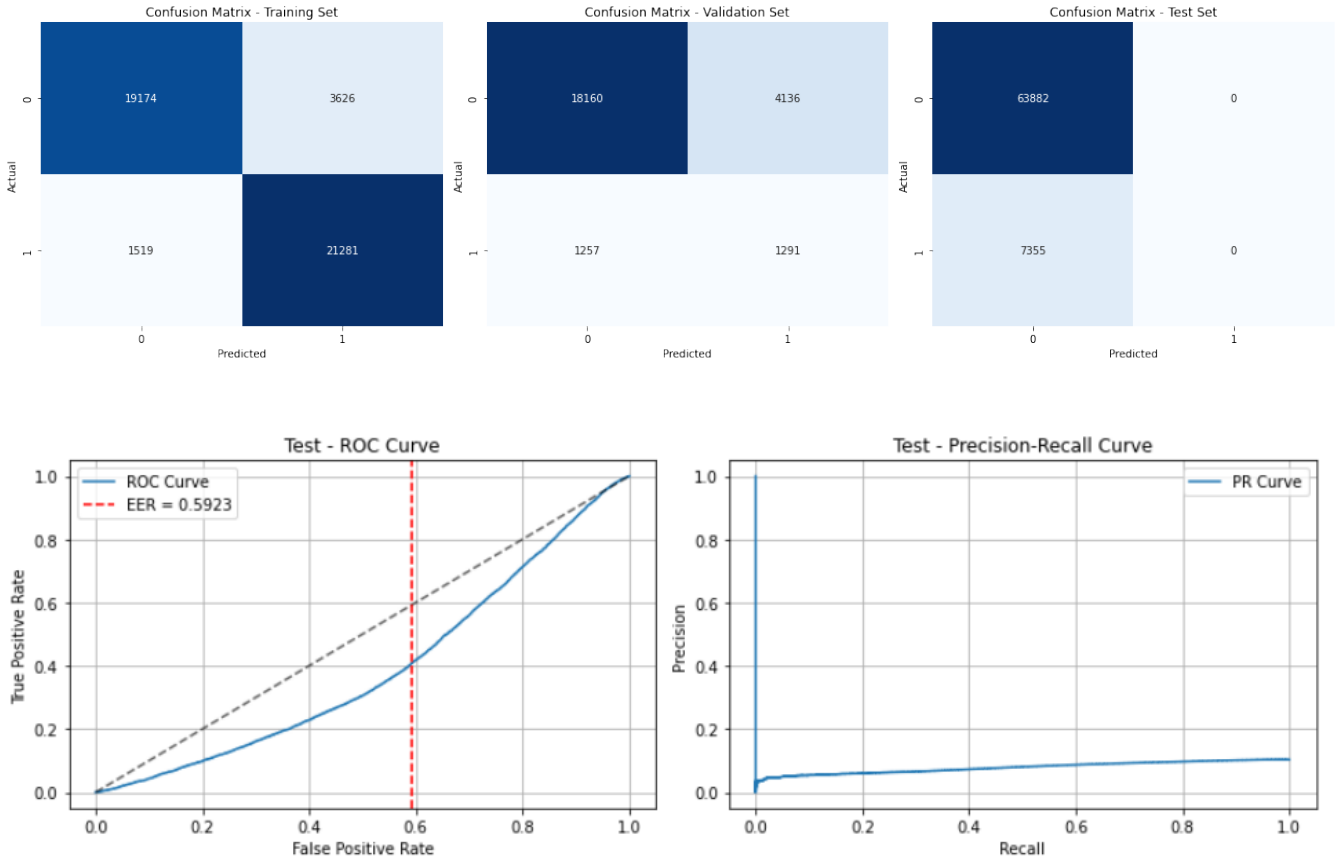


Figure 115: ROC and Precision-recall Curve

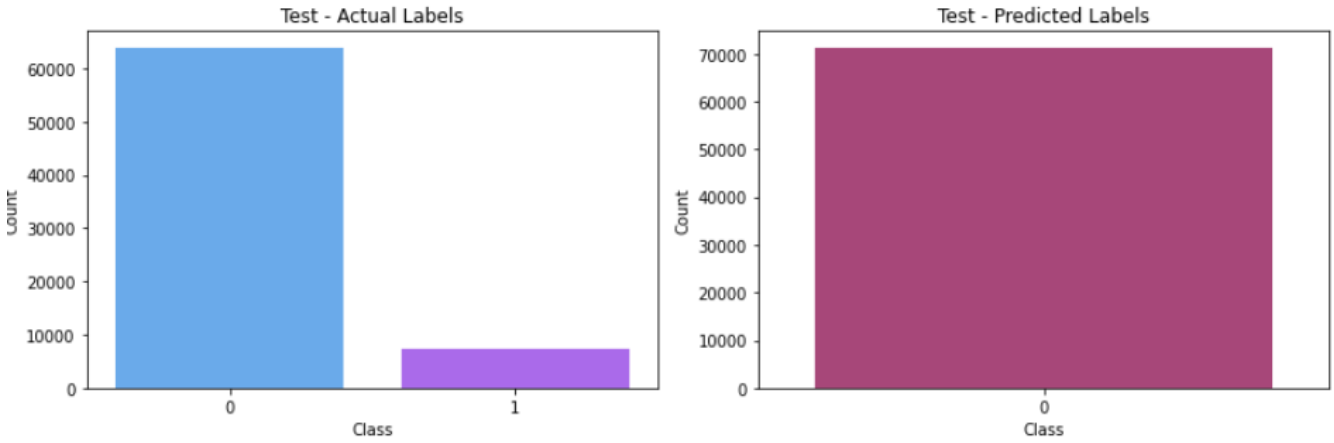


Figure 116: Actual vs predicted Labels

Undersampled

Training		Validation		Test (Evaluation)	
Accuracy	0.9269	Accuracy	0.6953	Accuracy	0.8914
F1 Score	0.9283	F1 Score	0.3227	F1 Score	0.0323
Recall	0.9457	Recall	0.7076	Recall	0.0175
EER	0.0705	EER	0.3010	EER	0.3670
Confusion Matrix:		Confusion Matrix:		Confusion Matrix:	
[2343 237]		[15472 6824]		[63375 507]	
[140 2440]		[745 1803]		[7226 129]	

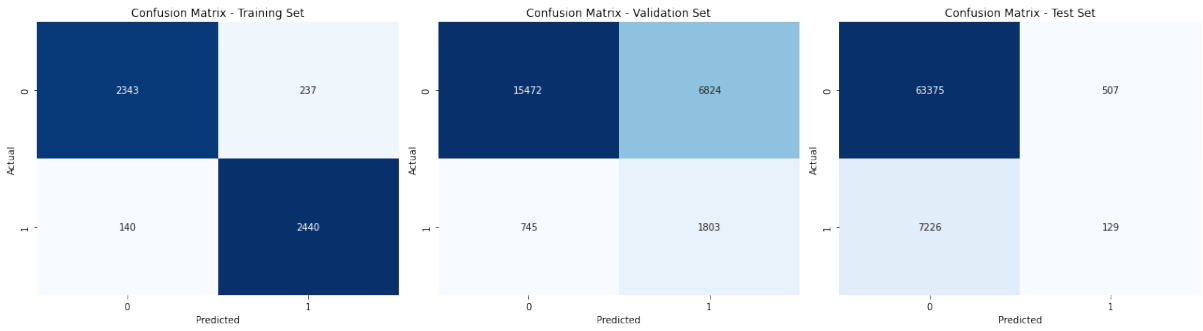


Figure 117: Visualizations for Validation and Test Sets (Undersampled)

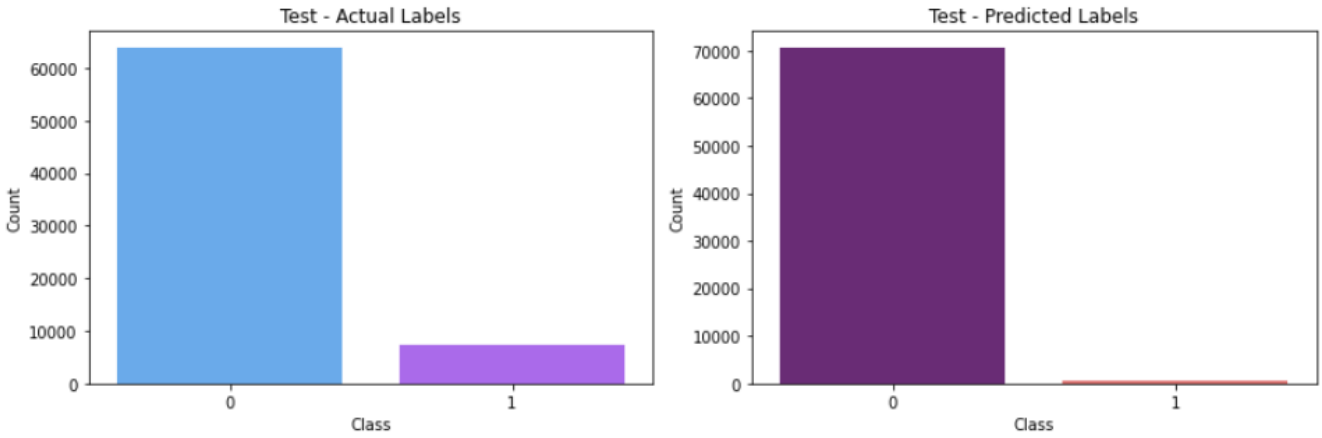


Figure 118: Actual vs Predicted

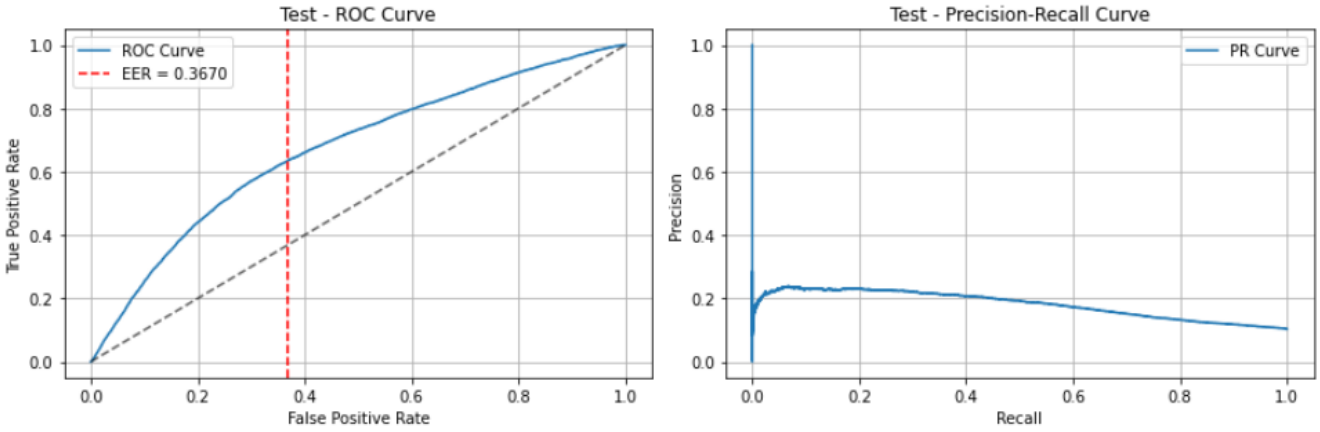


Figure 119: ROC and Precision-recall Curve

PCA REDUCTION(N=50)

Training	
Accuracy	0.8990
F1 Score	0.0131
Recall	0.0066
EER	0.2007
Confusion Matrix:	
[22800 0]	
[2563 17]	

Validation	
Accuracy	0.8974
F1 Score	0.0000
Recall	0.0000
EER	0.3151
Confusion Matrix:	
[22296 0]	
[2548 0]	

Test (Evaluation)	
Accuracy	0.8968
F1 Score	0.0000
Recall	0.0000
EER	0.4759
Confusion Matrix:	
[63882 0]	
[7355 0]	

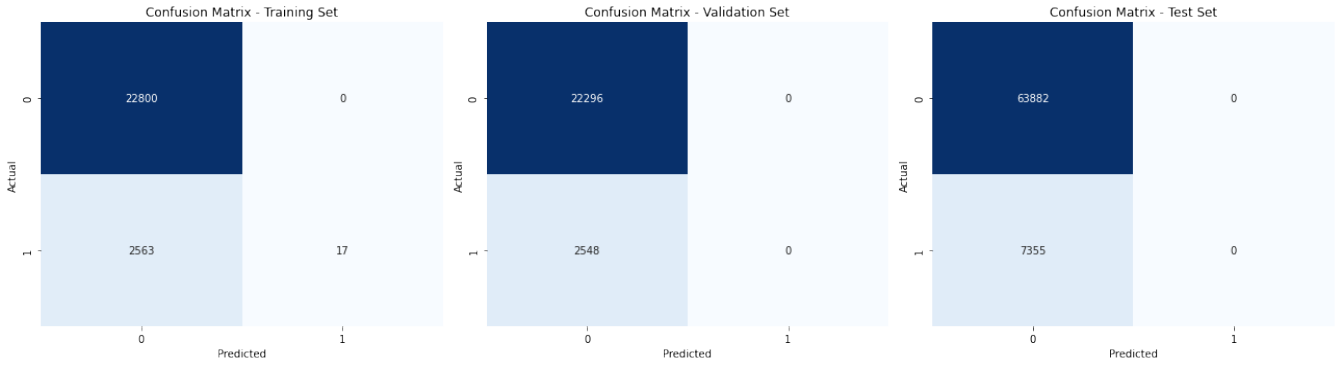


Figure 120: Confusion Matrices

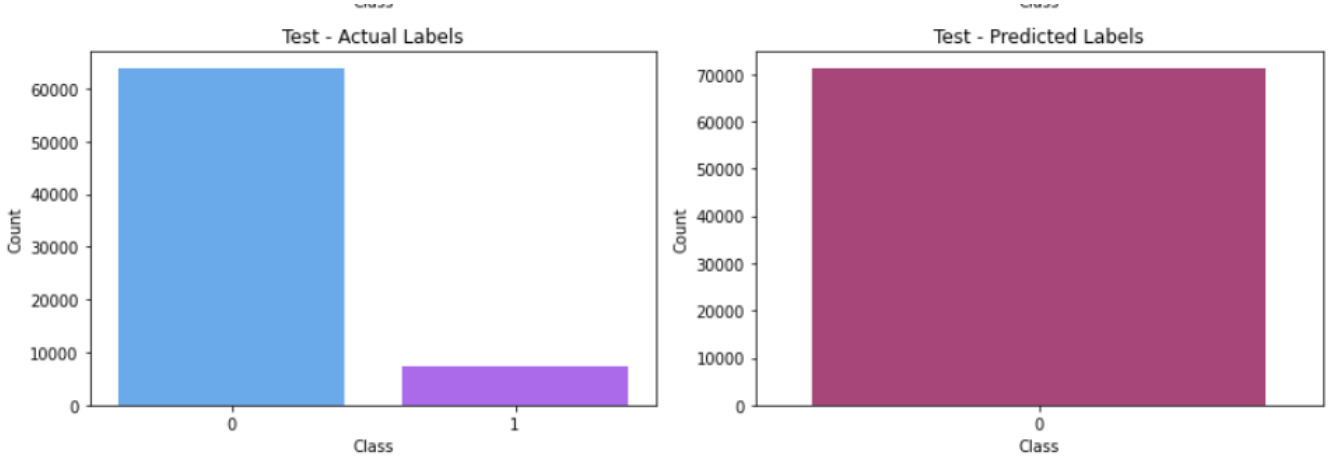


Figure 121: Actual vs predicted

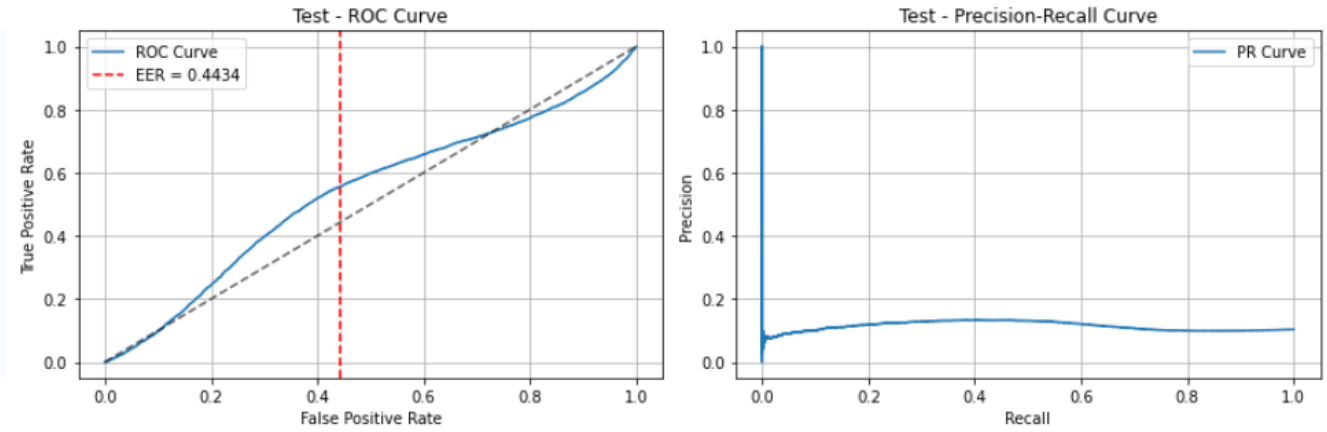


Figure 122: ROC and Precision-recall Curve

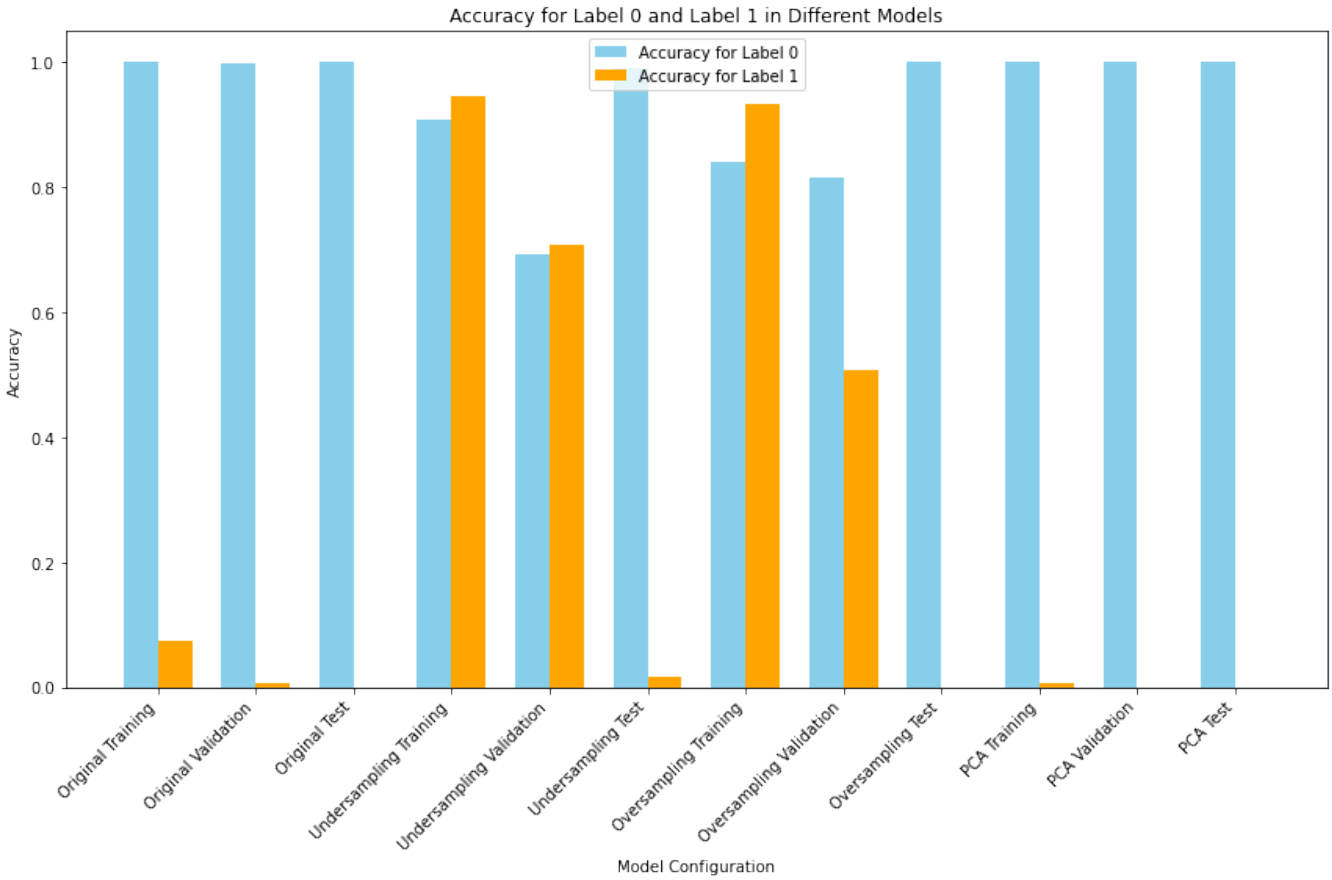


Figure 123: Accuracy for Label 0 and Label 1

XGBoost (Extreme Gradient Boosting) is a machine learning algorithm based on the gradient boosting framework. It builds a series of decision trees, where each new tree improves on the predictions made by the previous ones, making it highly effective.

How it Works

- Start with a weak model.
- Calculate the error (residual): Compute the error, or residual, which is the difference between the predicted values and the actual target values.
- Train a new decision tree to predict this residual.
- Add the new tree's predictions to the previous ones to improve overall performance.
- Repeat the process for multiple iterations or until model performance stops improving.

Original Dataset (XGBoost)

- The XGBoost model shows high accuracy on the training set but performs poorly on both the validation and test sets, with low F1 Score and Recall.
- The model is biased towards predicting the negative class.
- High training accuracy, but poor generalization to validation and test sets.

- Low Recall and F1 Score: The model fails to detect the positive class effectively.

Undersampling + XGBoost
Training

- Recall: 0.9457 — Excellent recall, indicating strong detection of the positive class.
- EER: 0.0705 — Low Equal Error Rate, showing good balance.

Validation

- Recall: 0.7076 — Improved recall, but still room for improvement.
- EER: 0.3010 — Higher EER than training, showing decreased performance.

Test

- Recall: 0.0175 — Extremely low recall, model struggles on unseen data.
- EER: 0.3670 — High EER, indicating many false predictions.

Conclusion: The model overfits the training data and generalizes poorly.
SMOTE + XGBoost

- Strong training performance: Accuracy = 0.8872, F1 Score = 0.8922, Recall = 0.9334.
- Good learning from the balanced training set.
- Validation performance drops, indicating overfitting.
- On the test set, the model fails to identify the positive class (Recall = 0, F1 = 0), showing it is not robust to class imbalance.

PCA (N=50) + XGBoost

- Applying PCA before training led to a significant decline in model performance.
- Validation and test sets both show Recall = 0 and F1 Score = 0 — positive class is completely ignored.
- Confusion matrices confirm no true positives detected.
- High EER values (Test: 0.4759) highlight imbalance in predictions.

6 Processed Results After Preprocessing

This section presents the results obtained from all models after applying preprocessing techniques. For each model, we compare the performance across different preprocessing methods: original (no preprocessing), oversampling with SMOTE, undersampling, and PCA dimensionality reduction.

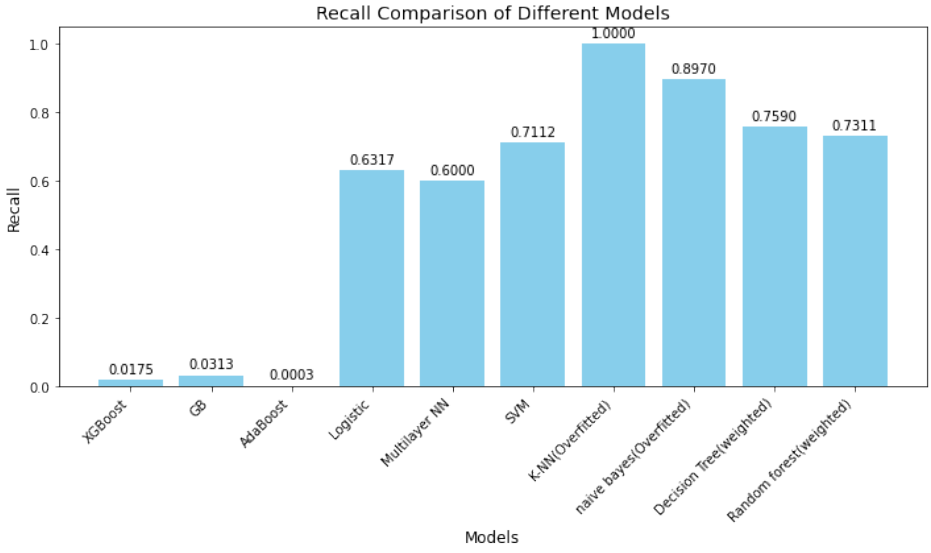


Figure 124: Recall Comparision across all models

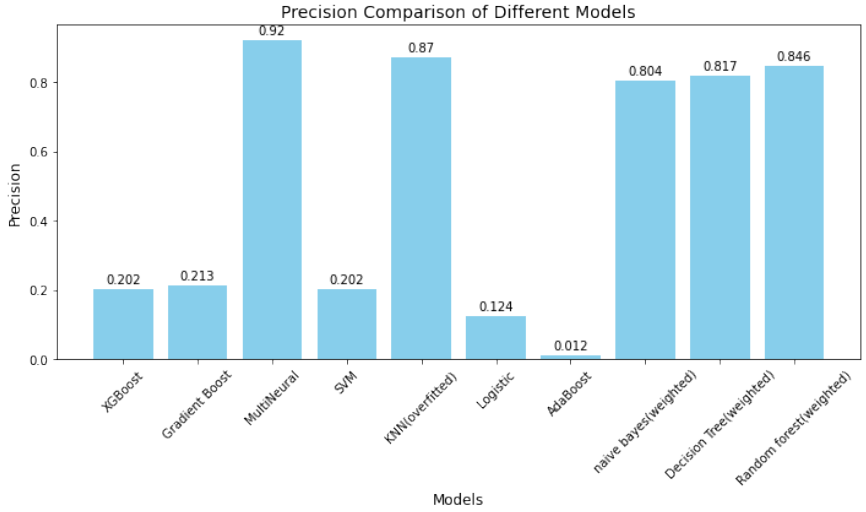


Figure 125: Precision Comparision across all models

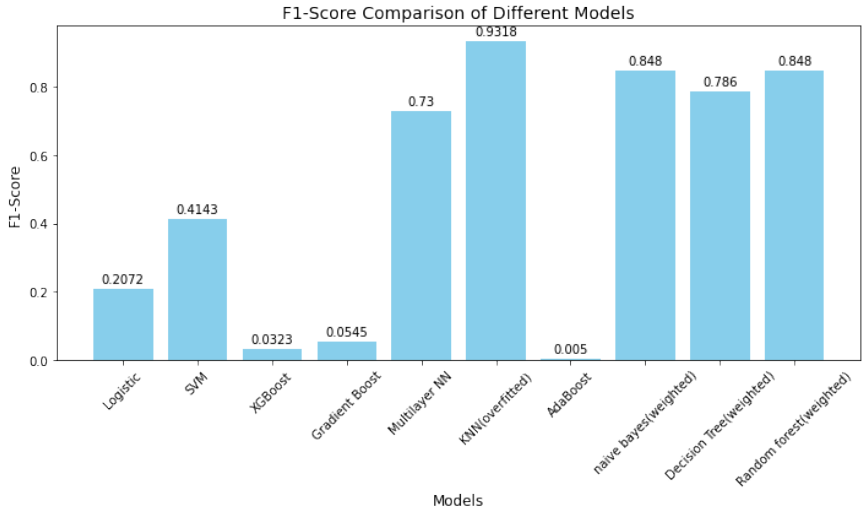


Figure 126: F1-Score Comparision across all models

After looking at the plots for Precision, Recall, and F1-score, we can infer the follow-

ing:

- XG-Boost and Gradient Boost didnt perform well for this dataset.
- Using Accuracy as a parameter in a Unbalanced classification is not a good option, as it tends to overfit and predict the majority class
- Logistic, Multi-layer NN, Naive Bayes and Random Forest have given good results as they were able to handle the noise and redundant data properly, additionally were able to detect the differences and similarities between the class 0 and class 1.
- Please note the values for Naive Bayes, Decision Tree and Random Forest are weighted so the comparision is not accurate.

6.1 Comparative Analysis of Preprocessing Techniques

This section provides a comprehensive description of the impact of different preprocessing techniques on model performance, explains how each preprocessing method affects accuracy across all models,

6.1.1 Oversampling vs. Undersampling

Oversampling with SMOTE generally outperforms random undersampling for most models. This suggests that the synthetic samples created by SMOTE provide valuable information that helps models better recognize synthetic speech patterns. In contrast, random undersampling, while equalizing class distribution, discards potentially useful information from the majority class.

The synthetic samples created also tend to overfit, and thereby making the model to just predict minority class.

6.1.2 Effect of PCA

PCA dimensionality reduction shows mixed results across different models. Linear models like Logistic Regression and linear SVM benefit significantly from PCA, likely due to the removal of noisy and redundant features. In contrast, tree-based models and neural networks, which can implicitly select relevant features, show less improvement or even slight degradation with PCA.

6.1.3 Combined Preprocessing Approaches

The combination of PCA followed by SMOTE oversampling yields the overall average results, particularly for linear models and shallow neural networks. But for NN and Random Forest there wasn't much improvement.

7 Discussion and Insights

This section discusses key findings and insights derived from our extensive experiments with different models and preprocessing techniques.

7.1 Model Performance Analysis

Even after experimenting with a wide array of classification models (*Logistic Regression, Random Forest, Decision Tree, Naïve Bayes, SVM with various kernels, Neural Networks – from single-layer to deep architectures using the Adam optimizer and cross-entropy loss, XGBoost, AdaBoost, KNN, Gradient Boosting*), the observed **accuracy** plateaued around 50% for distinguishing *Real* vs. *Fake* speech.

Additional methods were also explored:

- **Hyperparameter Tuning**: Systematic grid and random searches did not yield substantial improvements.
- **Threshold Adjustments**: Adjusting decision thresholds to optimize for precision or recall offered negligible gains.
- **Imbalance Handling**: Oversampling the *Fake* class, undersampling the *Real* class, and employing *weighted loss functions* to tackle the 9:1 imbalance ratio also failed to significantly improve results.
- **Loss and Optimizer Variations**: Switching among cross-entropy loss, focal loss, and experimenting with Adam, SGD, and other optimizers did not solve the underlying performance issues.

Potential Reasons for Limited Performance:

1. **Embedding Suitability:** Despite using 1024-dimensional embeddings from RawNet2, the features may not provide sufficient class separation for *Real* vs. *Fake* speech. High dimensionality can lead to the "curse of dimensionality" if many features are noisy or redundant.
2. **Fine-Tuning Deficit:** The pretrained RawNet2 model may not be fully aligned with our specific dataset. Fine-tuning (or domain adapting) the embedding layers could be essential to capture task-relevant characteristics.
3. **Data Quality and Label Consistency:** Noisy or inconsistent labels, as well as data distribution shifts (e.g., variations in recording conditions), may degrade model performance.
4. **Over/Underfitting Challenges:** Despite attempts at regularization and class rebalancing, the model may still overfit to majority-class cues or underfit due to excessive downsampling of the *Real* samples.

7.2 Feature Importance Analysis

To better understand why the models struggle, a deeper look into how the classifier weights or ranks the embeddings is crucial: **Interpretation of Embeddings:** If many of the 1024 embedding dimensions from RawNet2 have minimal variance or do not differentiate the two classes, they may hamper the decision boundaries.

7.3 Error Analysis

A confusion matrix frequently reveals critical insights:

- **Type I vs. Type II Errors**: Determine if the system predominantly misclassifies *Fake* as *Real* or vice versa. This helps in refining strategies (e.g., adjusting decision thresholds or applying focal loss).
- **Minority-Class Misclassifications**: With a 9:1 imbalance, a naive classifier predicting primarily *Real* could still achieve misleading accuracy. Monitoring precision, recall, and F1 score for the minority class is essential.

Common patterns in misclassified samples include:

- High-quality synthetic speech from advanced TTS systems that closely mimics natural speech characteristics
- Genuine speech with unusual acoustic properties (background noise, microphone artifacts, etc.)
- Short audio samples with limited phonetic content

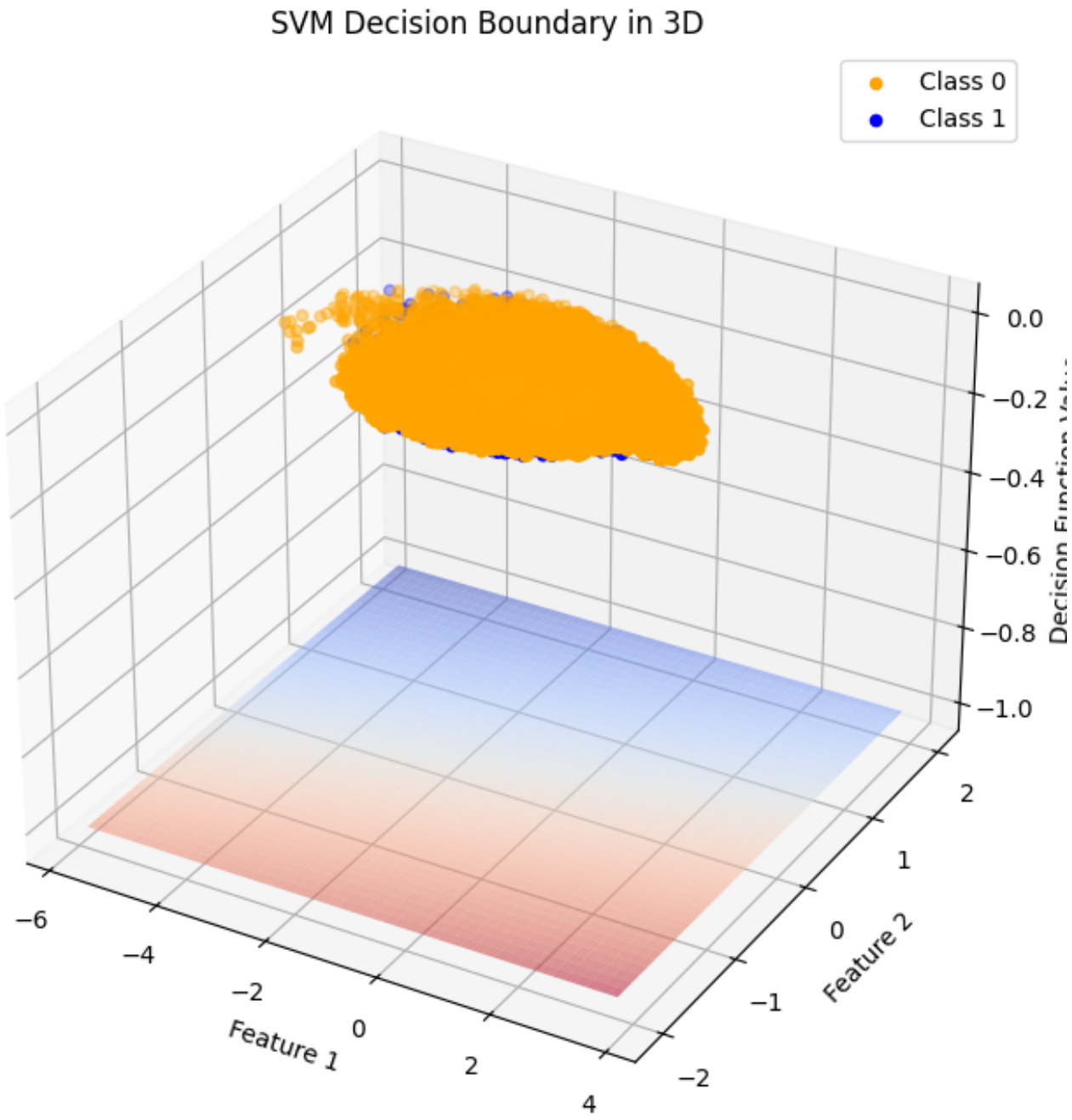


Figure 127: SVM Decision Boundary for Top 2 Features

As the Figure 127 shows, the top two features after applying PCA, have such high overlapping. Making it difficult for simple classifications to differentiate.

7.4 Computational Efficiency

- **Training Time:** The average runtime for most models was around 20-30 minutes, whereas the Neural Network models took approximately 30-40 minutes.
- **High-Dimensional Computation:** High-dimensional embeddings (1024 dimensions) can increase computation time in tree-based and distance-based algorithms. Dimensionality reduction or embedding compression might reduce the computational footprint.

- **Hardware Constraints:** Models that rely on large batch sizes or complex ensemble architectures may require more GPU/CPU resources to converge effectively.

8 Conclusion

In summary, despite:

- Trying numerous classification algorithms,
- Exhaustive hyperparameter tuning and threshold adjustments,
- Addressing class imbalance through sampling and weighted losses,
- Experimenting with different optimizers and loss functions,

the system's performance remained around 50% accuracy with modest F-1 Score, precision, and recall. Possible root causes include suboptimal embedding representations, insufficient fine-tuning for the task(due to computational limitations), and potentially problematic data distribution or label quality.

Going forward, **adapting the embeddings** specifically for *Real* vs. *Fake* classification and exploring more **robust imbalance strategies** (such as advanced data augmentation, focal loss, and dynamic thresholding) could yield better results. The success of the **AASIST** and **AASIST-L** methods emphasizes the need for domain-aware embedding adaptation and sophisticated handling of class imbalance.