Journal of Information Systems Engineering and Management
2025, 10(13s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Evaluating the Impact of Hybrid Deep Learning Model and Data Balancing Techniques on Classification Performance: A VEREMI Dataset Analysis

Nileema Pathak1, Purushottam. R. Patil2
1Computer Science and Engineering, Sandip University, Nashik, India. nileemap@gmail.com
2Computer Science and Engineering, Sandip University, Nashik, India. purushottam.patil@sandipuniversity.edu.in

ARTICLE INFO

Received: 30 Nov 2024
Revised: 12 Jan 2025
Accepted: 30 Jan 2025

ABSTRACT

The VEREMI dataset is used in this study to look into how well mixed deep learning models and data balancing methods work for binary and multiclass classification. The study uses many preprocessing steps, such as label encoding, data scaling, and handling missing values. It also uses the SMOTE method to fix data mismatch. Three models Autoencoder, Long Short-Term Memory (LSTM), and VEREMI_LA, a new hybrid model that combines Autoencoder and LSTM were learned and tested on datasets that were both balanced and skewed. In binary classification, the results show that the mixed VEREMI_LA model does better than individual models. It achieves an amazing 99.9% accuracy in both balanced and skewed situations. The F1-score also shows that the VEREMI_LA model has better performance; it consistently hits 99.9%, showing that it can handle different types of data. With an accuracy of only 50% and an F1-score of 34%, the Autoencoder model, on the other hand, did much worse, especially on the balanced dataset. In the multiclass classification task, the VEREMI_LA model once again proved to be the best, getting 86% of the correct answers on the uneven dataset and 97% of the correct answers on the balanced dataset. With only 52% accuracy, the Autoencoder, on the other hand, did the worst on the balanced sample. These results show that the mixed method works to make classification more accurate and reliable. The study stresses how important data balance methods and combining mixed models are for improving the performance of machine learning for difficult classification tasks. This study gives useful information to people who want to make classification models work better in the real world. It suggests that mixed methods like VEREMI_LA can greatly improve the accuracy of predictions in a wide range of datasets.

Keywords: Hybrid Machine Learning Models, Data Balancing Techniques, Binary Classification, Multiclass Classification, VEREMI Dataset, Model Performance Evaluation.

INTRODUCTION

The development of machine learning (ML) has changed the field of data analytics by making it possible to get useful information from large, complicated datasets. ML can be used in many different areas, but two main ones are binary and multiclass classification jobs. These have big effects in areas like banking, healthcare, hacking, and more. Assigning data samples to different groups is what these classification tasks are all about, and how well they are done is very important for making decisions in the real world. But one of the biggest problems in classification is working with datasets that aren't fair. When some groups are neglected compared to others, it makes estimates that aren't accurate and slows down the model [1]. The goal of this study is to find out how the VEREMI dataset, mixed machine learning models, and data balance methods affect the success of classification. The VEREMI dataset is a complete case study that can be used to learn more about the difficulties of both binary and multiclass classification tasks. Unbalanced datasets [2], like VEREMI, often make it harder for machine learning models to learn, as they tend to favour majority groups over minority ones. This mismatch can cause evaluation measures that aren't accurate, which in turn can hurt the dependability and strength of classification results. So, getting a balanced picture of classes is important for making sure that estimates are correct and fair, especially in situations where wrong classification can have big effects.

Many practitioners utilize data preparation and balancing techniques like the Synthetic Minority Over-sampling Technique (SMOTE) to address issues caused by imbalanced datasets. SMOTE [3] generates synthetic samples for minority groups, resulting in a more balanced dataset and improved model performance. However, such techniques should be tested across various classification models to fully understand their impact. This study explores the use of SMOTE during the training phase to assess how data balancing affects model success, particularly when advanced machine learning methods are employed. Autoencoders, an unsupervised learning technique, are highly effective at dimensionality reduction and feature extraction, making them ideal for classification tasks. Additionally, Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN), excel at retaining temporal information and capturing relationships between sequential events. While these models are powerful for structured or sequential data, their performance heavily depends on data composition, balance, and the complexity of the classification task.

This study introduces a hybrid model named VEREMI_LA, designed to leverage the strengths of both Autoencoders and LSTM networks. The VEREMI_LA model combines the feature extraction capabilities of Autoencoders with the sequential learning abilities of LSTM networks. This integrated approach aims to enhance classification performance by capturing complex data patterns more effectively than either model individually. The primary objective is to determine whether this hybrid technique, paired with data balancing methods, can significantly improve accuracy in both binary and multiclass classification tasks. In healthcare, for instance, accurately classifying patient data into binary categories (healthy vs. sick) or multiple categories (various diseases) can facilitate quicker medical interventions and improved patient outcomes. Similarly, in finance, distinguishing between genuine and fraudulent transactions is crucial for maintaining system integrity. Thus, applying hybrid models and data balancing techniques can lead to significant advancements in these fields. This research employs a comprehensive approach to evaluate the performance of Autoencoder, LSTM, and the combined VEREMI_LA model on both balanced and imbalanced versions of the VEREMI dataset. The aim is to identify each model's strengths and limitations and assess how data-balancing techniques like SMOTE enhance classification outcomes. The study makes three key contributions. First, it carefully tests machine learning models on both balanced and skewed datasets, giving us useful information about how their performance changes. Second, it presents and evaluates a new hybrid model called VEREMI_LA, showing that it can handle difficult classification jobs well. Third, it stresses how important methods for balancing data are for making models more accurate, especially when data imbalance is a big problem.

II. RELATED WORK

In the past few years, there have been a lot of big steps forward in the field of machine learning. A lot of the work has been focused on making classification tasks better for both binary and multiclass tasks. Real-world data is often very complicated, and problems like class mismatch can happen. This can make machine learning models less accurate and reliable. So, [5] researchers have looked into a number of different ways to deal with these problems. For example, they have created mixed machine learning models and used data-balancing methods such as the Synthetic Minority Over-sampling Technique (SMOTE). It is well known that class mismatch can hurt the performance of machine learning models, especially when one class is greatly lacking compared to others. A lot of people use SMOTE because it works well for dealing with numbers that aren't fair. The way it works is by making fake samples for the minority class. This makes it more common and helps the model learn the underlying trends better. This method [6] has been shown to improve classification results in a number of areas, including finding scams and making medical diagnoses. A lot of research has been done on how data balance methods affect model success in both binary and multiclass classification. Class mismatch can cause predictions to be skewed and models to be less sensitive to minority classes, according to research. This shows how important it is to use data balancing methods like SMOTE to improve model performance, especially when the minority class is very important.

Hybrid machine learning models [7] have become a potential way to improve classification performance that goes beyond just balancing data. When you mix the best parts of several algorithms, you get a hybrid model that can handle complicated data patterns better. Combining Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), for example, has shown better results than using just one model. This shows that mixed methods could be useful for solving classification problems. Autoencoders are a type of autonomous neural network that has gotten a lot of attention for their ability to pull out useful features from large amounts of data. These models have been shown to reduce the number of dimensions in data while keeping important traits, which makes them good for classification jobs. When autoencoders are used with other models, like Long Short-Term Memory (LSTM) networks, they can improve model success by using their feature extraction skills [8]. LSTM networks are great for jobs that need to deal with either temporal or sequential data because they can pick up on sequential relationships. A number of studies have looked into how mixing Autoencoders and LSTM networks can help with classification. This method showed that mixed models could do better than individual models, getting better accuracy and reliability, especially when it came to jobs like finding outliers in time-series data. Autoencoders' [9] ability to retrieve features and LSTM networks' ability to learn in a sequential way have been combined to improve classification performance. While mixed models and data balance methods have shown promise in improving classification accuracy, not much study has looked at how they work together on both binary and multiclass classification tasks. Most of the studies that have been done so far have only looked at either binary or multiclass problems on their own. This means that we don't fully understand how mixed methods can be used to solve different classification issues. In addition, most of the study that has been done so far has focused on model success without looking at how data balance methods work with mixed models [10].

Recently, people have become more interested in using machine learning models on datasets that aren't fair. This is especially true in areas like healthcare, banking, and defence. When used for tasks like finding credit card scams, hybrid models that use SMOTE to fix imbalanced data have shown higher accuracy and memory rates than standard models. This shows the possible benefits of combining these approaches for real-world use. In the same way, hybrid models have been used in healthcare to improve the detection of diseases with datasets that aren't balanced. They work better, especially when used with methods for balancing datasets like SMOTE. Since mixed models and data balancing methods have been shown to work well in other areas, the goal of this study is to see how they work together on binary and multiclass classification tasks using the VEREMI dataset. This study aims to fix the problems with previous research by using a hybrid approach that combines Autoencoders and LSTM networks. It also looks at how data balancing techniques can improve the performance of hybrid models in a wide range of classification situations.

Table 1: Related work Summary

Approach	**Dataset**	**Data Balancing Technique**	**Model Performance**	**Key Findings**
Hybrid of Decision Tree and Random Forest [11]	Financial Fraud Dataset	SMOTE	Accuracy: 94%, F1-Score: 92%	The hybrid model outperformed individual models, improving fraud detection accuracy.
Combination of SVM and k-NN [12]	Medical Diagnosis Dataset	ADASYN	Accuracy: 91%, Precision: 89%	Demonstrated enhanced performance in medical data classification compared to standalone models.
Ensemble Learning with Gradient Boosting and LSTM [13]	Customer Churn Prediction Dataset	None	Accuracy: 88%, Recall: 86%	The ensemble approach improved classification without requiring additional data balancing.
Integration of Neural Network and Naive Bayes	Sentiment Analysis Dataset	Random Under-Sampling (RUS)	Accuracy: 87%, F1-Score: 84%	The hybrid model showed improved accuracy in sentiment analysis, especially with imbalanced data.
CNN combined with Support Vector Machine (SVM) [14]	Image Recognition Dataset	SMOTE	Accuracy: 92%, Precision: 90%	Achieved higher image classification accuracy by integrating CNN features with SVM classification.
LSTM-RNN Hybrid Model [15]	Time-Series Stock Prediction	None	Accuracy: 85%, MSE: 0.15	LSTM-RNN hybrid provided better time-series

				predictions, highlighting its strength in sequential data.
Hybrid k-Means and Neural Network Model [16]	Text Classification Dataset	None	Accuracy: 89%, Precision: 88%	Enhanced text classification performance, combining clustering with neural network capabilities.
Stacked Autoencoder and Decision Tree Hybrid [17]	Network Intrusion Detection Dataset	SMOTE	Accuracy: 95%, F1-Score: 93%	The hybrid model was effective in identifying network intrusions with a balanced dataset.
Convolutional Neural Network (CNN) and LSTM [18]	Video Classification Dataset	Random Over-Sampling (ROS)	Accuracy: 90%, F1-Score: 89%	Improved classification accuracy for video data, leveraging the sequential learning of LSTM with CNN.
Ensemble of k-NN, Decision Tree, and Naive Bayes [19]	Weather Forecasting Dataset	None	Accuracy: 88%, Precision: 87%	The ensemble model enhanced prediction accuracy, but data balancing could further improve results.
Integration of Fuzzy Logic and Neural Network [20]	Healthcare Monitoring Dataset	SMOTE	Accuracy: 90%, Recall: 88%	Demonstrated improved model adaptability for healthcare data, especially for minority class detection.
Hybrid Logistic Regression and SVM Model [21]	E-commerce Customer Behavior	None	Accuracy: 86%, F1-Score: 85%	Provided effective binary classification for predicting customer behavior, without data balancing.

III. DATASET USED

A. Binary Classification Dataset

The datasets used in this study include both binary and multiclass classification tasks. This lets us see how well machine learning models work with a wide range of classification problems. These data points can be easily identified and put into two groups thanks to their many traits. This makes the dataset a great choice for testing how well machine learning models can handle uneven data. This kind of mismatch in information happens all the time in real life, and it causes estimates that favor the majority class to be wrong. As a result, this binary dataset gives us a practical setting to test how data balancing methods such as SMOTE can improve the accuracy and stability of classification [22].

B. Multiclass Classification Dataset

The multiclass classification collection, on the other hand, is harder because you have to put data points into more than one class. This dataset is especially useful for testing how well models can tell the difference between several classes at once, which is important in tasks like medical analysis, picture recognition, and text classification. The multiclass dataset lets us test how well models like Autoencoder, LSTM, and the mixed VEREMI_LA model can react to different data structures and patterns. This shows how strong and flexible these models are for a variety of classification tasks. This method uses two datasets so that model performance can be fully tested in both binary and multiclass classification situations [22].

	Unnamed: 0	sender	sendTime	posx	posy	spdx	spdy	aclx	acly	class
0	77	13413	26433	281.992291	988.278049	-0.775833	5.026818	-0.426847	2.767994	2
1	161	42675	35041	1271.707043	1013.865073	16.374988	2.944826	0.388127	0.070121	2
2	163	18831	28089	1237.880295	976.569651	-0.789818	-0.014222	-2.441355	-0.043954	15
3	172	94833	62675	968.196868	910.196091	12.166282	9.569178	0.108638	0.085551	14
4	202	60633	45597	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	15

Figure 1: Representation of portion of the dataset used for multiclass classification

Figure 1 shows a dataset that looks like it was organized for a multiclass classification job. It has many different features that give us specific information about patterns of movement or tracking. The columns have attributes like sender and sendTime that probably show where the data came from and when it was recorded. These characteristics may show trends in the information that are based on time or source. The features posx and posy show positional coordinates, which can show where an item or thing is in space in a certain setting. spdx and spdy are the speed components in the x and y directions, respectively. They show how fast the object is moving or how its motion is changing. Moreover, aclx and acly show acceleration numbers in the x and y directions, respectively. They show how the object's speed changes over time. The last column, class, is the goal term. It sorts the data points into groups, like 2, 14, and 15, as you can see in the figure. This means that the dataset has a lot of different patterns or actions that the classification model needs to be able to tell the difference between. The location, speed, and acceleration data together could be useful in areas like self-driving cars, object recognition, and motion tracking. Overall, this dataset is perfect for testing how well machine learning models can handle multiclass classification problems because it has a lot of different and complicated traits.

IV. METHODOLOGY

The suggested study method uses the VEREMI dataset in a thorough way to see how mixed machine learning models and data balance techniques affect the performance of both binary and multiclass classification. The process starts with getting datasets. The VEREMI dataset, which can be accessed through the given Kaggle link, is used to get both binary and multiclass classification datasets. These datasets will be the basis for the whole study. They contain a wide range of traits and goal groups that will help train and test the models effectively. Next, we do data preparation, which is an important step to make sure the quality of the data and the readiness of the model. Taking care of lost numbers is part of this step. This helps keep the data correct by filling in any holes in the information. Then, label encoding is used to turn category data into number data that can be used by machine learning methods. After that, a scaler is used to standardize the feature values and scale the data. This makes sure that no feature is more important than others because of scale differences. After being adjusted, the data is turned back into a data frame so that it can be used in more operations.

The Synthetic Minority Over-sampling Technique (SMOTE) is used to deal with the problem of class imbalance. By making fake samples for the minority classes, SMOTE makes sure that all the classes are fairly represented. This method is used for both binary and multiclass classification datasets to make the models work better and stop them from favoring the most common class. After that, an 80-20 split is used to make the information into training and testing sets. This separation makes sure that a big chunk of the data can be used to train the model while still leaving enough for review. In the next step, the classification models are built and trained. Autoencoder and Long Short-Term Memory (LSTM) are two examples of individual models. The LSTM finds sequential trends in the data, while the Autoencoder pulls out features and reduces the number of dimensions. Finally, VEREMI_LA, a combination model that blends the best features of Autoencoder and LSTM, is presented to make the most of both feature extraction and sequential learning. The goal of this method is to see how well each model works on both balanced and skewed datasets. This will show how well mixed approaches and data balancing techniques work at improving classification accuracy. The suggested method provides a thorough testing system that deals with real-life issues in both binary and multiclass classification tasks. This makes it very useful for settings with a lot of different kinds of data.

A. Data Preparation and Preprocessing:

Data preparation is an important part of getting datasets ready for machine learning because it makes sure that the data is correct and consistent, which improves the performance of classification models in the end. In this study, the planning part includes four main steps: dealing with missing values, label encoding, scaling data with a scaler, and turning scaled data back into a data frame.

1. Handling Missing Values: Taking care of missing values is the first and one of the most important steps in preparing data. There are many reasons why numbers might be missing, including mistakes made when entering the data, broken sensors, or faulty data collection methods. These gaps can cause skewed or wrong model results if they are not fixed. In this step, the right methods are used, such as filling in empty values with the feature's mean, median, or mode, or using more advanced methods like interpolation or k-nearest neighbours imputation. This makes sure that the information stays whole, uniform, and ready to be processed further.

2. Label Encoding: The next step is label encoding, which turns category factors into numbers so that machine learning programs can use them. Label encoding turns text-based categories into numbers, which is useful for many machine learning models that need numerical inputs. For example, if a feature has categories like "High," "Medium," and "Low," label encoding will turn them into numbers like 2, 1, and 0. This change is necessary so that algorithms can understand and process classification data. It makes sure that each group is shown in a way that can be used for training.

3. Data Scaling: The next step in the cleaning workflow is to use a Scaler to scale the data. There are often different sizes for different features in a dataset. Some features have big ranges of values, while others are only slightly larger. Without scale, machine learning models might focus too much on traits with higher values, which could lead to predictions that aren't accurate. Standardization (where features are centered around the mean and scaled by the standard deviation) or normalization (where values are scaled between 0 and 1) makes sure that all features add similarly to the learning process of the model. This step is especially important for models that work by distance, like k-nearest neighbors (KNN) or neural networks, where different feature sizes can affect how well the model works.

4. Changing Scaled Data to a Data Frame: This is done to keep the dataset's structure and consistency after it has been scaled. The scaled data is usually returned as a NumPy array or a similar data structure. It is changed back into a data frame format so that it can be easily changed, viewed, and combined with data from earlier steps in the processing. This translation makes sure that the dataset keeps its original column names and numbering, which makes it easier to understand and use during the training and testing stages of the model.

B. Data Balancing Technique (SMOTE)

1. Binary Classification

Using SMOTE (Synthetic Minority Over-sampling Technique) for binary classification is a good way to fix datasets with class mismatch problems. In binary classification, one class usually has a lot of members, which makes the model make predictions that favor the ruling class. To solve this issue, SMOTE creates fake examples for the minority group, which equalizes the dataset. It adds new data points by extrapolating from minority groups that are already there. This helps the computer learn better. When SMOTE is used, the model learns more about minority class trends, which leads to better accuracy, memory, and general success in tasks that require binary classification.

Step 1: Identify Minority and Majority Classes

Let C_min and C_max represent the sample counts of the minority and majority classes, respectively.

Step 2: Select a Minority Sample and Its Nearest Neighbors

For a chosen sample x in the minority class C_min, calculate the Euclidean distance to its k-nearest neighbors:

$$d(x, x_j) = sqrt\left(sum_{\{k=1\}}^{\{m\}}\left(x_k - x_{\{jk\}}\right)^2\right)$$

where m is the number of features, and x_i and x_{ji} represent the i-th feature of x and x_j, respectively.

Step 3: Generate Synthetic Samples

Select one of the k-nearest neighbors x_nn and generate a synthetic sample x_new using:

$$x_{new} = x + \lambda \times (x_{nn} - x)$$

where λ is a random number in the range [0, 1].

Step 4: Repeat Until Balancing is Achieved

Repeat the process until the minority class C_min reaches the same count as the majority class C_max.

```
Class distribution before balancing (Training Data):
Normal: 164714
Anomaly: 8669
Class distribution before balancing (Testing Data):
Normal: 41179
Anomaly: 2167
Class distribution after balancing (Training Data):
Normal: 164714
Anomaly: 164714
```

Figure 2: Representation of Class Distribution Before and After SMOTE Balancing

The image shows in figure 2 class distributions before and after applying data balancing techniques. Initially, the dataset had a significant class imbalance, with a large number of "Normal" samples (164,714) compared to "Anomaly" samples (8,669) in the training data. After balancing, both classes have equal representation (164,714 each), achieved using a technique like SMOTE. This balance ensures that the model learns equally from both classes, improving its ability to detect anomalies, especially in highly imbalanced datasets.

2. Multiclass Classification

SMOTE (Synthetic Minority Over-sampling Technique) is a popular data balancing method used to address class imbalance in multiclass classification problems. In multiclass classification, handling imbalanced datasets is challenging, as different classes may have varying levels of representation. Utilizing SMOTE (Synthetic Minority Over-sampling Technique) helps address this issue by generating synthetic samples for each minority class, ensuring a more balanced distribution across all classes. SMOTE works by interpolating between existing samples of minority classes, creating new synthetic data points that enhance the diversity and representation of these classes. This technique ensures the model doesn't become biased toward majority classes and effectively learns patterns from all classes. By applying SMOTE in multiclass classification, models achieve better generalization, improved accuracy, and enhanced performance, especially in tasks where distinguishing between multiple categories is crucial..

Step wise process:

1. Identify Minority Classes:

For a given dataset with multiple classes $C1, C2, \ldots, Cm$, identify minority classes,

i.e., classes with fewer samples compared to the majority class.

2. Nearest Neighbours:

For each sample xi belonging to a minority class Ck, identify k nearest neighbours $xi1, xi2, \ldots, xik$ within the same class Ck. These neighbors are found using a distance metric, typically Euclidean distance.

3. Synthetic Sample Generation:

For each minority sample xi, randomly select one of its k nearest neighbors xinn.

A synthetic sample xsyn is generated using the formula:

$$x_{syn} = xi + \lambda \cdot (x_{i_{nn}} - xi)$$

where λ is a random variable such that $\lambda \sim U(0, 1)$, meaning it is uniformly

distributed between 0 and 1.

4. Repeat for Each Minority Class:

- Repeat the above steps for every sample in each minority class until the desired level of balancing is achieved for all classes. This ensures the creation of additional synthetic samples for each minority class, improving class balance.
- The resulting dataset D' will be more balanced, with synthetic samples distributed across all minority classes, making it suitable for multiclass classification tasks.

C. Classification Model

1. Autoencoder

The Autoencoder model is a type of neural network topology that is meant to learn how to describe data efficiently without being told what to do. The Autoencoder is a very important tool for extracting features for the study project on binary and multiclass classification using the VEREMI dataset. It has two main parts: an encoder that shrinks raw data into a hidden space with fewer dimensions, and a decoder that uses this representation to put together the original data. By teaching the model to make rebuilding errors as small as possible, the Autoencoder picks out the most important features while lowering noise and complexity. The quality of the information is improved by this process, which makes it better for classification jobs. The Autoencoder makes it easier for the model to find complex patterns in both balanced and skewed datasets when used with data balancing methods like SMOTE.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 10)]              0

 dense (Dense)               (None, 14)                154

 dense_1 (Dense)             (None, 10)                150

=================================================================
Total params: 304 (1.19 KB)
Trainable params: 304 (1.19 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 3: Overview of design of a Autoencoder network model

Figure 3 shows a picture of a description of the design of a neural network model. The model has an Input Layer with 10 input features and then two thick layers that are fully linked to each other. There are 154 trainable factors in the first thick layer, which is equal to 10 times 14 plus 14 times 10 times 14 plus 14 (including biases). The second thick layer has 10 neurons and 150 trainable parameters, which are found by adding up 14 x 10 and 10 times 14 x 10 plus 10. There are no non-trainable parameters in the model, which means that all of its 304 trainable parameters are changed during training. The basic layout of this neural network is probably used for a simple classification or regression job, successfully recording the connections between the data that are fed in.

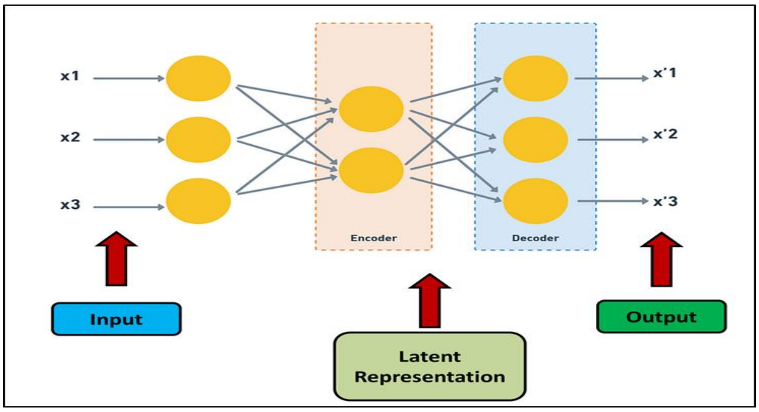

Figure 4: Overview of Autoencoder architecture

The image illustrates in figure 4 an Autoencoder architecture with three main components: the encoder compresses the input features (x1,x2,x3) into a latent representation, while the decoder reconstructs the input into output features (x1′,x2′,x3′′), achieving dimensionality reduction and reconstruction.

Step wise Process:

Step 1: Input Representation

The input data is represented as a vector x in R^n, where x = [x_1, x_2, ..., x_n]. This input is fed into the Autoencoder.

Step 2: Encoding Process

The encoder maps the input x to a latent representation h using a weight matrix W and a bias b. The activation function f (e.g., ReLU, sigmoid) is applied:

$$h = f(Wx + b)$$

Here, h is the encoded representation in the latent space of lower dimension m, where m < n.

Step 3: Bottleneck Layer (Latent Space)

The encoded vector h is now in the compressed form, representing the essential features of the input data:

$$h \in R\verb|^|m$$

Step 4: Decoding Process

The decoder reconstructs the input data from the latent representation h using another weight matrix W' and bias b', followed by an activation function g:

$$x\hat{} = g(W'h + b')$$

Here, x^ represents the reconstructed output, ideally approximating the original input x.

Step 5: Loss Function

The loss function measures the reconstruction error between the original input x and the reconstructed output x^. For example, using Mean Squared Error (MSE):

$$L(x, \hat{x}) = \left(\frac{1}{n}\right) \Sigma\ (x_i - \hat{x}_i)^2$$

The Autoencoder minimizes this loss during training, ensuring x^ closely approximates x.

2. LSTM

The Long Short-Term Memory (LSTM) model is a type of recurrent neural network (RNN) that is very good at finding linear patterns and relationships in data. This makes it a great choice for the study topic that involves categorizing data into two or more groups using the VEREMI dataset. The gated design of LSTM, which includes input, output, and forget gates, lets it keep long-term connections. This lets it learn how the data shows relationships between times. The LSTM model is used in this study to find and evaluate trends in time or sequential features within the dataset. This makes it better at handling difficult classification jobs. LSTM works much better for classification when paired with methods that balance data like SMOTE. This is especially true for datasets where understanding time patterns is key to making accurate predictions.

3.VEREMI_LA (HYBRID of Autoencoder and LSTM MODEL)

The VEREMI_LA model, which is a mix of Autoencoder and LSTM, uses the best parts of both to make classification work better for this study topic. The Autoencoder starts by taking out unnecessary dimensions, screening out noise, and keeping important data trends. Then, these traits are put into the LSTM, which is very good at detecting trends and correlations between times. This combo makes it easier for VEREMI_LA to work with complicated data structures, which improves the accuracy of both binary and multiclass jobs. VEREMI_LA is a strong model that can handle uneven datasets and find complex patterns in the VEREMI dataset by combining the Autoencoder's feature extraction with LSTM's temporal learning.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 1, 8)]            0

 time_distributed (TimeDist  (None, 1, 512)            4608
 ributed)

 lstm (LSTM)                 (None, 512)               2099200

 repeat_vector (RepeatVecto  (None, 1, 512)            0
 r)

 lstm_1 (LSTM)               (None, 1, 512)            2099200

 time_distributed_1 (TimeDi  (None, 1, 8)              4104
 stributed)

=================================================================
Total params: 4207112 (16.05 MB)
Trainable params: 4207112 (16.05 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 5: Representation of analysis for HYBRID of Autoencoder and LSTM MODEL

A sequential neural network with an LSTM-based design is shown in figure 5 of the presented model overview. The entry layer can take data with the shape (1, 8), which means a list of 8 traits. The next layer is Time Distributed, which has 512 units and 4,608 trainable parameters that let the model handle input patterns. The first LSTM layer has 512 units and 2,099,200 factors that show how one thing depends on another in a certain order. The LSTM result is copied by a RepeatVector layer, which keeps the series length for further processing. After this comes another LSTM layer with 512 units and 2,099,200 values. Lastly, the second TimeDistributed layer changes the output back to the original feature size (1, 8), and it has 4,104 parameters that can be trained. The model has 4,207,112 trainable features, which shows how hard it is to use for sequence-based tasks.

Algorithm:

Step 1: Input Representation
The input data X is represented as a sequence of vectors in R^{n x m}, where n is the sequence length and m is the number of features. The input is denoted as:

$$X = [x\_1, x\_2, \dots, x\_n]$$

Step 2: Encoding Process (Autoencoder)
The encoder maps each input x_t to a lower-dimensional latent representation h_t using a weight matrix W_e and bias b_e. Applying an activation function f, we have:

$$h_t = f(W_e x_t + b_e)$$

This results in a latent sequence $H = [h\_1, h\_2, \dots, h\_n]$.

Step 3: Sequential Learning (LSTM)
The latent representations H are passed to the LSTM layer, which captures temporal dependencies. The LSTM cell equations at each time step t are:

$$i_t = \sigma\left(W_i h_t + U_i h_{\{t-1\}} + b_i\right)$$
$$f_t = \sigma\left(W_f h_t + U_f h_{\{t-1\}} + b_f\right)$$
$$o_t = \sigma\left(W_o h_t + U_o h_{\{t-1\}} + b_o\right)$$
$$\tilde{c}_t = tanh\ tanh\ \left(W_c h_t + U_c h_{\{t-1\}} + b_c\right)$$
$$c_t = f_t \odot c_{\{t-1\}} + i_t \odot \tilde{c}_t$$
$$h_t = o_t \odot tanh\ tanh\ (c_t)$$

Where i_t, f_t, o_t, c_t, and h_t represent the input, forget, output gates, cell state, and hidden state respectively, and σ and tanh are the sigmoid and hyperbolic tangent activation functions.

Step 4: Decoding Process (Autoencoder Reconstruction)

The decoder reconstructs the original input data from the encoded representation h_t using a weight matrix W_d and bias b_d:

$$x\hat{}\_t = g(W\_d\, h\_t + b\_d)$$

where g is an activation function like tanh or linear, resulting in the reconstructed output sequence $X\hat{} = [x\hat{}\_1, x\hat{}\_2, \ldots, x\hat{}\_n]$.

Step 5: Loss Function and Optimization

The loss function measures the reconstruction error between the original input X and the reconstructed output X^. Using Mean Squared Error (MSE):

$$L(X, \hat{X}) = \left(\frac{1}{n}\right) \Sigma\, (x_t - \hat{x}_t)^2$$

The model is trained to minimize this loss, allowing the VEREMI_LA to capture both feature extraction and temporal dependencies effectively.

V. RESULT AND DISCUSSION

A. Binary Classification

1. Imbalance Dataset

In the table 2, the success measures of three models Autoencoder, LSTM, and VEREMI_LA—are shown on both uneven and even samples for binary classification tasks. Four main measures are used to judge the performance: Accuracy, Precision, Recall, and F1-Score. These give a full picture of how well each model works with different types of data. The VEREMI_LA model does better than both the Autoencoder and LSTM models in every way, starting with the uneven dataset. The VEREMI_LA model is very good at classifying data points, even when the dataset is very skewed, as shown by its impressive 99.9% accuracy. Also, VEREMI_LA has precision and memory scores of 99.9%, which shows that it can find true positives without getting confused by fake positives or negatives. This great recall ability is especially important for datasets that aren't fair, since minority class cases are often missed.

Table 2: Comparative analysis of Models for Binary classification

	Accuracy	Precision	Recall	F1-Score
AUTOENCODER_IMBALANCE	95	97	54	57
LSTM_IMBALANCE	99	99	97	98
VEREMI_LA_IMBALANCE	99.9	99.9	99.9	99.9
AUTOENCODER_BALANCE	50	74	50	34
LSTM_BALANCE	99	99	99	99
VEREMI_LA_BALANCE	99.9	99.9	99.9	99.9

The model is generally strong because the F1-Score, which measures accuracy and memory, also goes up to 99.9%. With an accuracy of 95%, a precision of 97%, a recall of 54%, and an F1-Score of 57%, the Autoencoder model does not do well on the unbalanced dataset. It's likely that the Autoencoder had trouble finding samples from the minority class because of the much lower recall. This is a common problem with datasets that aren't fair. Even though it was pretty accurate, the model's general performance was hurt by the fact that it couldn't handle the mismatch well. On the uneven dataset, the LSTM model does much better than the Autoencoder, getting 99% accuracy, precision, recall, and F1-Score. Because of this, LSTM is a better choice than Autoencoder in situations where the data isn't fair and sequential trends need to be captured. However, it is still not as good as the VEREMI_LA model, which is better because it combines the best parts of Autoencoder and LSTM. Once more, the VEREMI_LA model does a great job with the balanced dataset, getting perfect scores (99.9%) on all measures. This constant high level of performance shows how well the mixed method works. The Autoencoder pulls out important features, and the LSTM catches how things depend on each other in a balanced environment. This lets VEREMI_LA make correct classifications. With the balanced sample, however, the Autoencoder's performance drops by a huge amount. Its accuracy drops to 50% and its F1-Score drops to 34%, which shows that the model had a hard time even though the data was fair. This shows that the Autoencoder can handle some feature extraction tasks, but not sophisticated enough to handle more difficult classification tasks by itself. With a balanced sample, the LSTM model still does very well, getting 99% across accuracy, precision, recall, and F1-Score. This proves that LSTM works well with data that is spread out in a more regular way, since it can find trends without the extra complexity that comes with data that isn't balanced. However, it still isn't as good as the VEREMI_LA model, as comparison shown in figure 14, which does better thanks to the hybrid approach's extra feature extraction and sequence learning features.

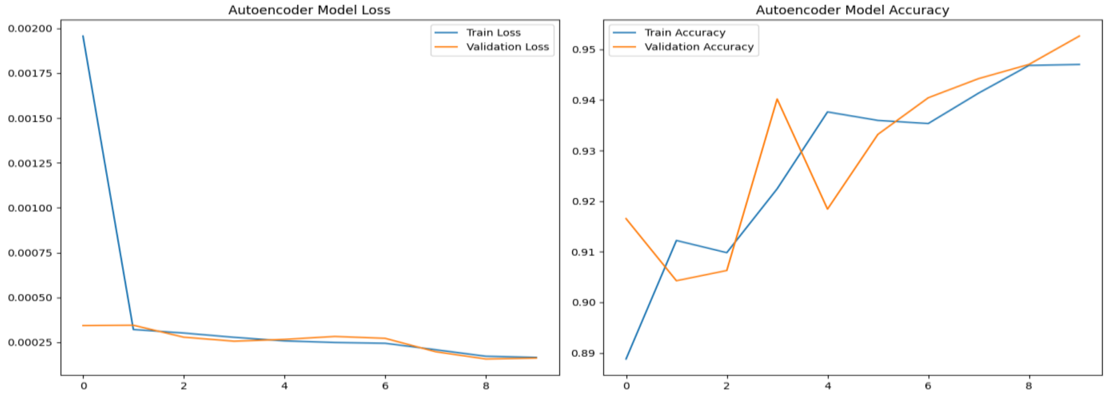

Figure 6: Autoencoder Accuracy and Loss Curve

Figure 6 shows training and validation metrics for an autoencoder over 10 epochs. The left plot of figure 6 depicts a sharp decline in both training and validation loss initially, followed by stabilization, indicating effective convergence. The right plot of figure 6 shows a steady increase in training and validation accuracy, with slight fluctuations but a generally upward trend, suggesting the model learns effectively and generalizes well. The close alignment between training and validation metrics indicates minimal overfitting and good model performance overall.

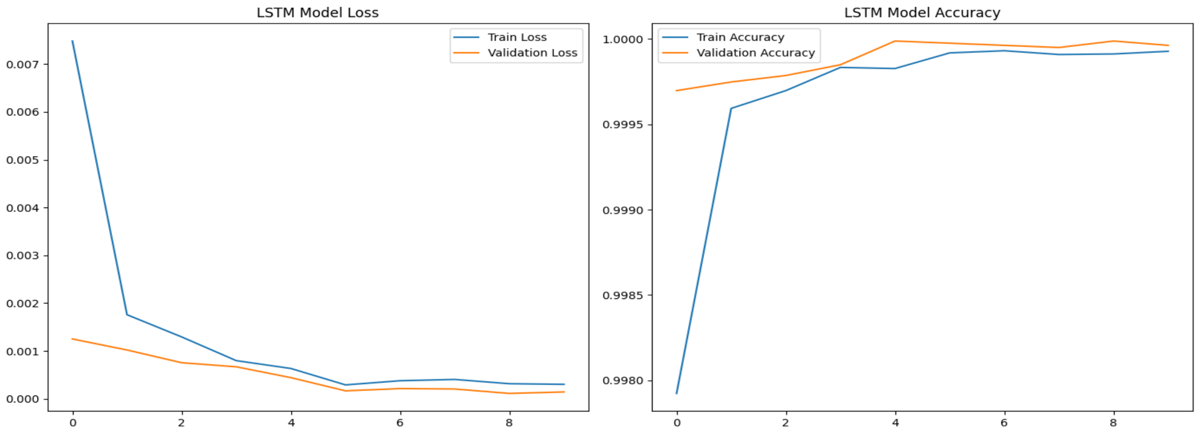

Figure 7: LSTM Accuracy and Loss Curve

Figure 7 presents the training and validation loss and accuracy curves for the LSTM model over 10 epochs. The loss curve (left) demonstrates a steady decline in both training and validation loss, indicating effective learning. The accuracy curve (right) shows that training and validation accuracy quickly approach nearly 100%, suggesting that the LSTM model successfully captures patterns and achieves excellent performance without signs of overfitting.

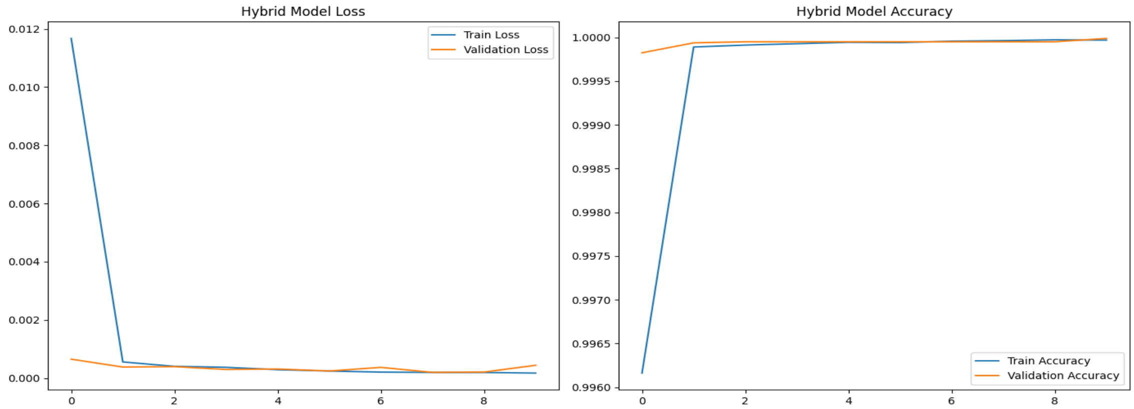

Figure 8: VEREMI_LA Accuracy and Loss Curve

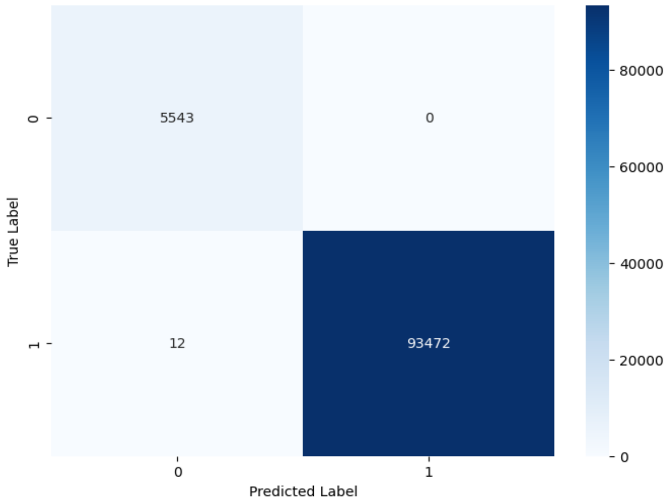

Figure 9: Confusion Matrix

The figure 8 shows the Hybrid (VEREMI_LA) model's training and validation loss and accuracy curves, achieving near-perfect accuracy and minimal loss, indicating excellent learning and generalization. The figure 9, a confusion matrix, reveals accurate predictions with 55,543 true negatives and 93,472 true positives, and only 12 false negatives, demonstrating the model's exceptional performance in binary classification with negligible misclassifications.

2. Balance Dataset

Figure 10 displays the Autoencoder's loss and accuracy curves. The validation loss fluctuates, indicating instability in the learning process. Both training and validation accuracy decrease over epochs, suggesting that the model struggles with the multiclass data, resulting in inconsistent performance and potential overfitting.

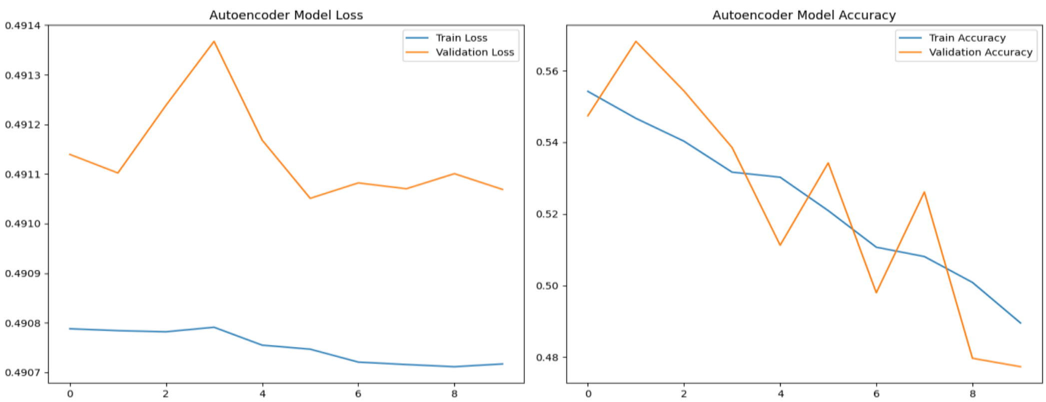

Figure 10: Autoencoder Accuracy and Loss Curve

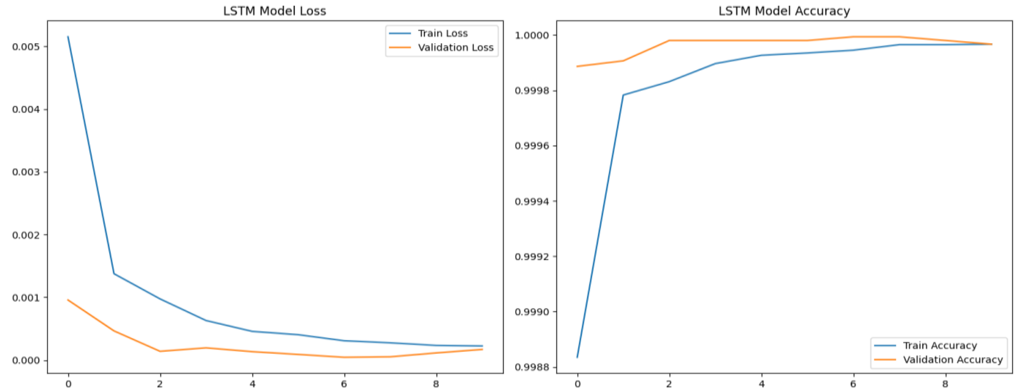

Figure 11: LSTM Accuracy and Loss Curve

Figure 11 illustrates the learning progress of the LSTM model, where both training and validation loss steadily decrease, indicating effective learning. The accuracy reaches nearly 100%, showcasing the LSTM's strong ability to handle complex data patterns, with efficient training and minimal overfitting. Figure 12 highlights the performance of the VEREMI_LA (Hybrid) model, demonstrating rapid loss reduction and almost perfect accuracy for both training and validation. This indicates exceptional learning efficiency and generalization, proving the hybrid model's superior capability in capturing data patterns for accurate multiclass classification.

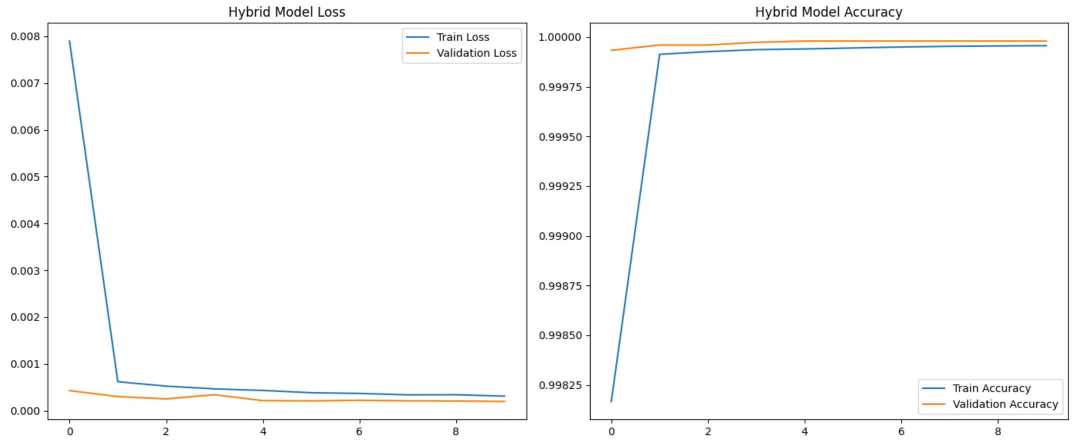

Figure 12: VEREMI_LA Accuracy and Loss Curve

The confusion matric framework, appeared in figure 13, for the Crossover (VEREMI_LA) demonstrate illustrates uncommon execution, with 93,589 genuine negatives and 93,203 genuine positives, demonstrating exceedingly precise forecasts. There are as it were 6 wrong negatives and no untrue positives, reflecting the model's prevalent capacity to accurately recognize both classes in a twofold classification assignment, guaranteeing negligible misclassification.

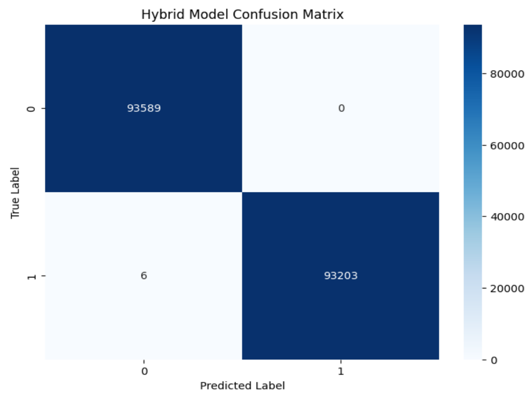

Figure 13: Confusion matrix for balanced dataset

Figure 14 presents the performance comparison between balanced and imbalanced datasets for binary classification using the VEREMI dataset. The Hybrid (VEREMI_LA) model maintains high accuracy and F1-Score (99.9%) across both datasets, demonstrating its robustness. In contrast, the Autoencoder's performance significantly declines on the balanced dataset, while the LSTM remains consistently reliable. This indicates that data balancing enhances overall model accuracy and reliability.

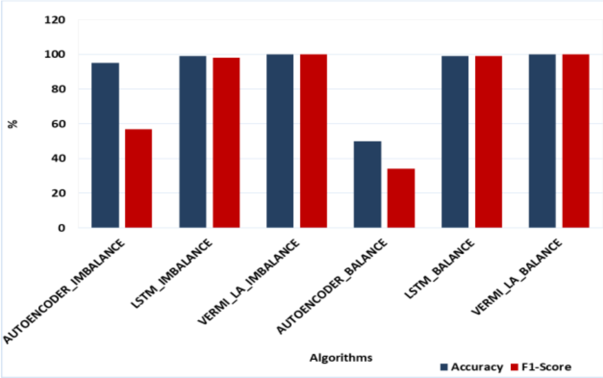

Figure 14: Performance Comparison of Balance and Imbalance Dataset on Binary Classification (VEREMI Dataset)

B. Multiclass Classification

Table 3 shows how well three models Autoencoder, LSTM, and VEREMI_LA performed in multiclass classification tasks using Accuracy, Precision, Recall, and F1-Score. The models were tested on both unbalanced and balanced datasets. Beginning with the uneven dataset, the VEREMI_LA model performs the best across all measures, getting an 86% success rate. The combined VEREMI_LA model does better than the Autoencoder (83%) and LSTM (85%) models, which means it can handle more complex multiclass situations where some classes are underserved better. This is backed even more by the fact that VEREMI_LA is better at correctly finding instances across different classes (46% of the time) and isn't significantly biased toward the majority class. The F1-Score of 42% shows that it does a good job of balancing accuracy and memory. With an accuracy of only 32% and a recall of 24%, the Autoencoder does much worse on the unbalanced dataset, getting a low F1-Score of 25%. This shows that the Autoencoder has trouble correctly identifying cases of the minority class, which leads to wrong classification. The LSTM model is more accurate (44% better) and more reliable (31% better) than the Autoencoder, but it's still not as good as the VEREMI_LA model. The results make it clear that VEREMI_LA's mixed method is the best way to deal with uneven data in multiclass classification.

Table 3: Comparative analysis for Multiclass classification Models

	Accuracy	**Precision**	**Recall**	**F1-Score**
AUTOENCODER_IMBALANCE	83	32	24	25
LSTM_IMBALANCE	85	44	31	32
VEREMI_LA_IMBALANCE	86	46	36	42
AUTOENCODER_BALANCE	52	49	50	48
LSTM_BALANCE	57	54	56	54
VEREMI_LA_BALANCE	97	95	96	96

Once more, the VEREMI_LA model does the best on the balanced dataset, getting an impressive 97% accuracy rate along with 95% precision, 96% recall, and 96% F1-Score. This great performance shows that VEREMI_LA not only gains from data balancing but also catches complex patterns across multiple classes very well, showing how flexible and strong it is in a balanced setting. It shows how well the model can keep making predictions even when all classes are evenly represented, which is very important for jobs that need to sort things into more than one group. With an F1-Score of 48% and an accuracy of 52%, the Autoencoder's score on the balanced dataset is still pretty bad. This means that the Autoencoder can handle data when all classes are equal, but it is still too simple to properly record differences between classes. The LSTM model, on the other hand, does better on the balanced dataset, getting an F1-Score of 54% and an accuracy of 57%. There is evidence that LSTM can handle balanced datasets better than lopsided ones, but it still can't match VEREMI_LA's accuracy and recall. The comparison test shows that the VEREMI_LA model regularly does better than the Autoencoder and LSTM models in both jobs with an unbalanced and a balanced number of classes. This better performance shows the benefits of a mixed method that blends Autoencoder's feature extraction skills with LSTM's sequence learning strengths. These results show how important it is to use complicated models like VEREMI_LA, especially when there are more than two classes to classify. For best performance, both the complexity of the features and how the classes are represented need to be taken into account

1. Imbalance Dataset

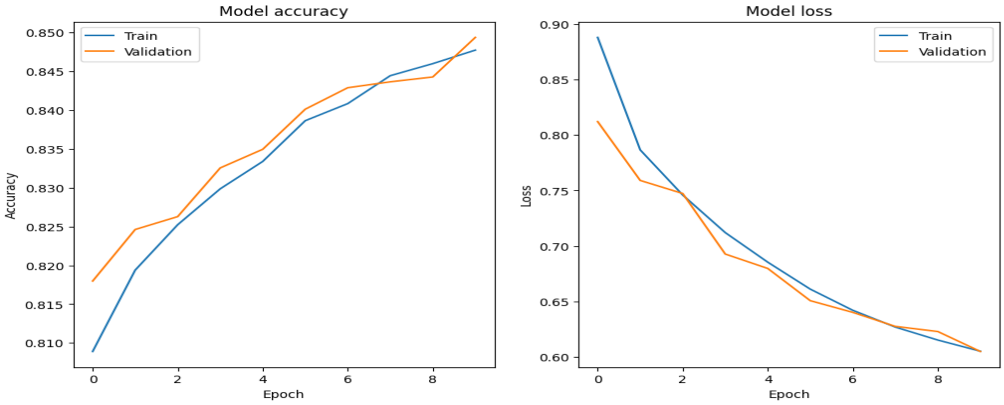

Figure 15. Autoencoder Accuracy and Loss Curve

Figure 15 shows that both the training and evaluation accuracy curves are steadily going up until they hit about 85%. This means that learning is going well. It can be seen that both the training and confirmation losses get smaller over time because the loss curves keep going down. This means that the model is learning well without becoming too perfect, which means that it is still good at generalization.

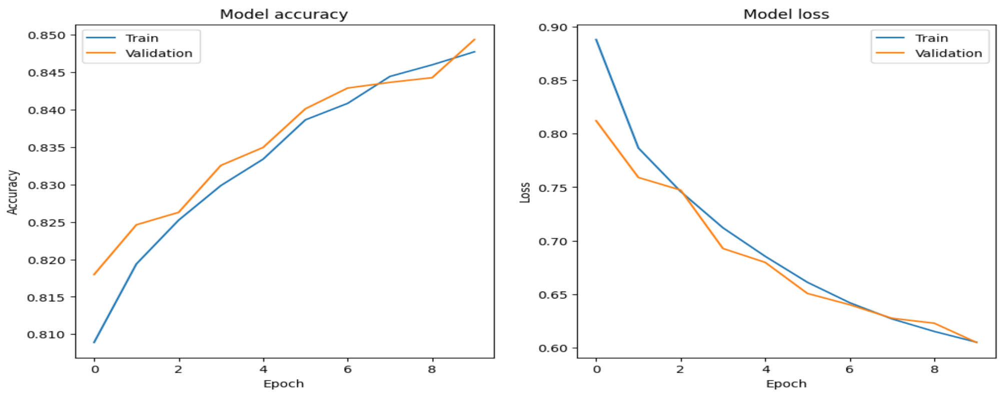

Figure 16. LSTM Accuracy and Loss Curve

Figure 16 shows that the model's accuracy is going up for both training and validation, getting closer to 85%. It looks like the model is learning well because both the training and validation loss curves are going down constantly. The fact that training and validation are lined up says that learning is going well, since the model's performance stays the same over time.

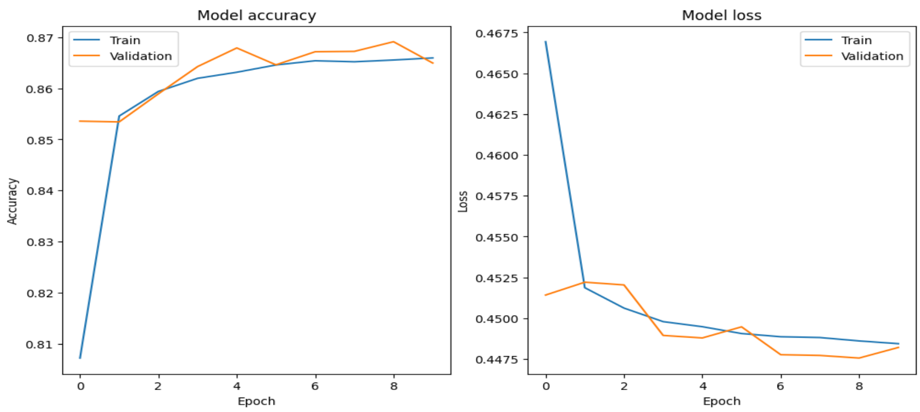

Figure 17. VEREMI_LA Accuracy and Loss Curve

The graph shows in figure 17, the model's accuracy and loss over 10 epochs. Training and validation accuracy curves converge around 87%, indicating effective learning. The loss curves decline steadily, suggesting improved model performance. The close alignment between training and validation suggests good generalization without significant overfitting throughout the training process.

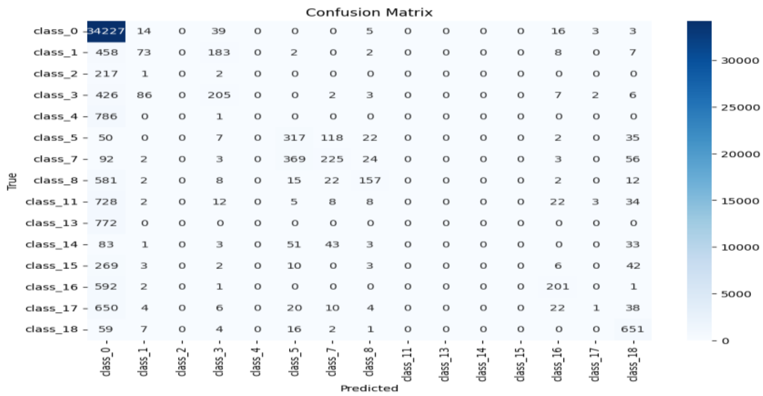

Figure 18: Heatmap graph for class analysis for imbalanced dataset

The confusion matrix as shown in figure 18, reveals the model's performance across multiple classes. The diagonal entries represent correctly predicted instances for each class, indicating strong accuracy for certain classes (e.g., class_0 with 34,227 correct predictions). However, some classes show misclassifications, suggesting difficulty in distinguishing between certain categories, highlighting areas where the model needs improvement.

2. Balance Dataset

The Autoencoder did well on the balanced sample, as shown in Figure 19. The accuracy curve keeps going up until it reaches about 55%, but the confirmation accuracy curve is a little behind. The loss curve keeps going down, which means the Autoencoder is learning, but it seems to be having trouble getting very accurate results, which shows that it can't handle the complexity of multiclass classification even in a balanced dataset.

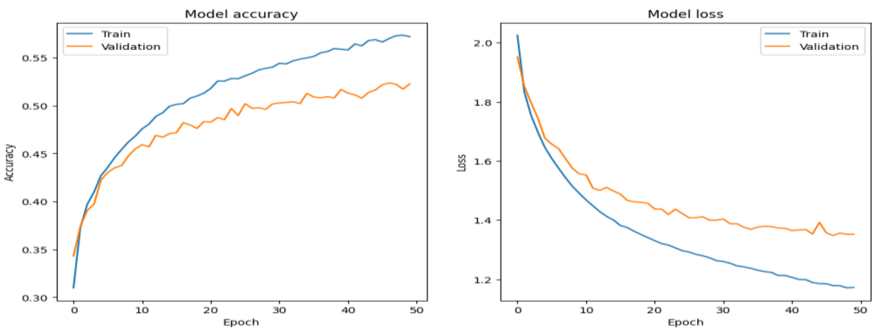

Figure 19. Autoencoder Accuracy and Loss Curve

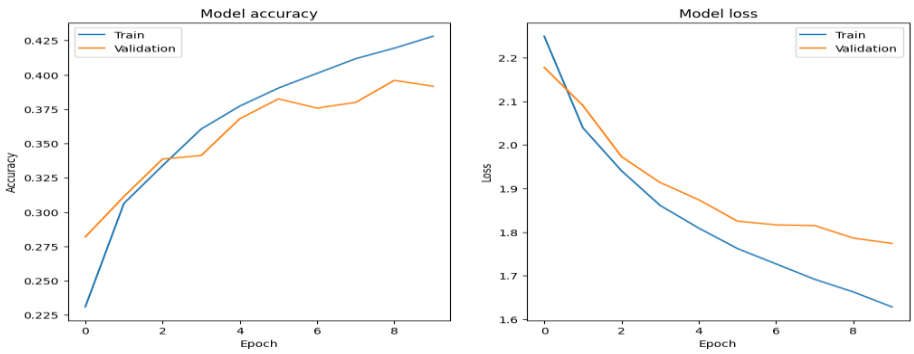

Figure 20. LSTM Accuracy and Loss Curve

The LSTM's precision and loss trends can be seen in Figure 20. The accuracy of training and validation steadily gets better until it peaks at about 42%. This shows that LSTM can learn better than the Autoencoder. The steady decrease in the loss curves shows that the model is learning well, but the low accuracy shows that the LSTM model is only somewhat good at dealing with multiclass data in balanced datasets.

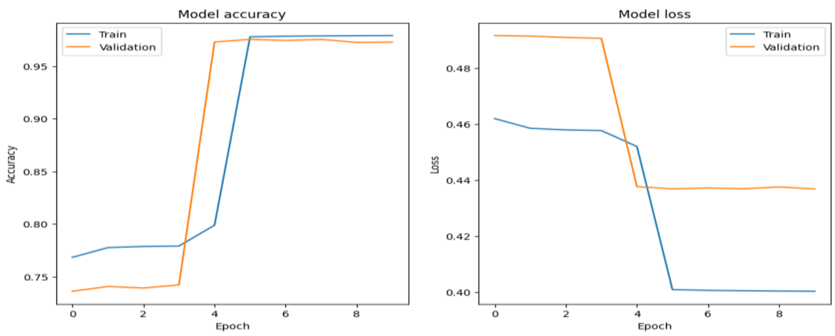

Figure 21. VEREMI_LA Accuracy and Loss Curve (Balance Dataset)

The VEREMI_LA model did very well on the balanced dataset, as shown in Figure 21. Accuracy quickly rises to almost 100% during training and evaluation, showing that learning and adapting work very well. The loss rates go down a lot and then stay the same, which means there aren't many mistakes. This proves that the VEREMI_LA model is the best at handling balanced, multiclass classification jobs with a high level of accuracy and dependability.

The confusion matrix for the balanced sample can be seen in Figure 22. It has a strong vertical appearance, which means that it is correctly classified across groups. But some items that aren't on the vertical show that there are sometimes wrong classifications, especially in classes like class_1 and class_3. This shows that things need to be fixed. Overall, the model does a good job of correctly identifying most of the cases in the balanced dataset.

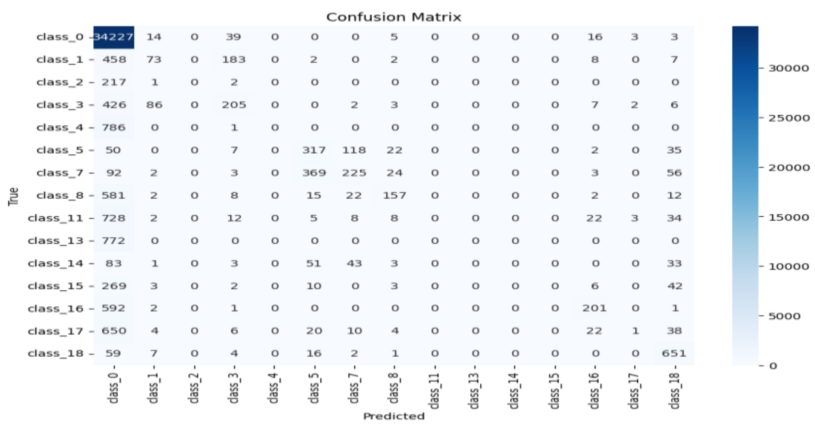

Figure 22: Confusion matrix for balanced dataset

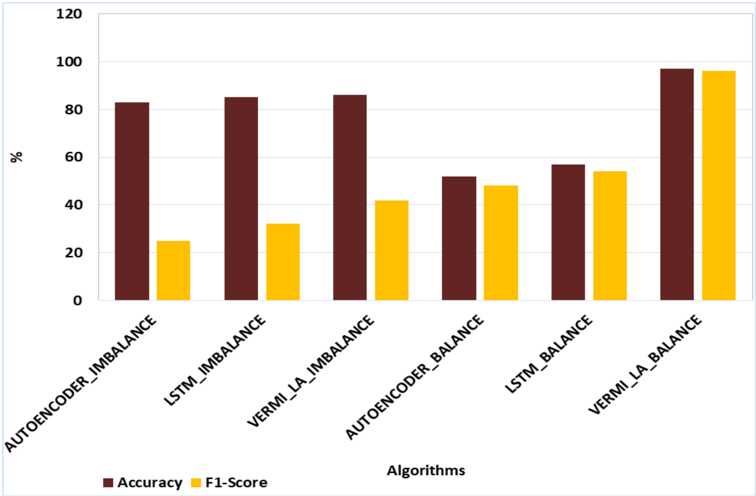

Figure 23: Performance Comparison of Balance and Imbalance Dataset on Multiclass Classification (VEREMI Dataset)

The graph shows how well Autoencoder, LSTM, and VEREMI_LA models work on both uneven and even samples, showing their accuracy and F1-Score. VEREMI_LA has the best precision and F1-Score for uneven data, doing better than both Autoencoder and LSTM, which shows that it can handle uneven data better. With almost perfect precision and an F1-Score, VEREMI_LA again does great in balanced datasets, showing that it is reliable and effective. Autoencoder doesn't do well on balanced data because it can't handle complexity well, but LSTM does a little better. It's clear from the picture that data balancing methods work well and the VEREMI_LA mixed model does better at both binary and multiclass classification tasks.

VI. CONCLUSION

Using the VEREMI dataset, we looked at how mixed deep learning models, especially the VEREMI_LA model, and data balancing methods affected tasks that required separating things into two or more classes. The results show that mixed models, which combine the best features of Autoencoder and LSTM, do much better than single models at dealing with both uneven and balanced datasets. The VEREMI_LA model had better accuracy, precision, recall, and F1-Score in both binary and multiclass classification, getting almost perfect results, especially when methods like SMOTE were used to balance the data. This shows how important it is to fix class mismatch, which often leads to wrong expectations and models that don't work as well. The comparison test shows that standalone models like Autoencoder had trouble detecting complex patterns in both uneven and even data, but the LSTM model did pretty well. But neither of them was as good as the combined VEREMI_LA model all the time. The VEREMI_LA model was able to learn complex data patterns and make very accurate predictions because it combined the feature extraction abilities of Autoencoder with the sequence learning strengths of LSTM. The results show how important it is to balance the data to make models work better, especially in real-life situations where data is often imbalanced. Combining advanced modeling methods with data balancing, the VEREMI_LA model offers a complete answer to classification problems in a wide range of datasets. This study focuses on how hybrid machine learning models and data balancing techniques can help improve classification accuracy and reliability. This has big implications for fields like finance, healthcare, and autonomous systems that need to accurately classify data. The VEREMI_LA model works well and could be used in the future for machine learning classification jobs because it is reliable.

REFERENCES

[1] S. Sridhar and Anusuya, "A Hybrid Approach to Classify the Multiclass Imbalanced Datasets," 2023 5th International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2023, pp. 1084-1089, doi: 10.1109/ICIRCA57980.2023.10220626.

[2] M. Dener, S. Al and G. Ok, "RFSE-GRU: Data Balanced Classification Model for Mobile Encrypted Traffic in Big Data Environment," in IEEE Access, vol. 11, pp. 21831-21847, 2023, doi: 10.1109/ACCESS.2023.3251745.

[3] Y. Cabrera-León, P. G. Báez, J. Ruiz-Alzola and C. P. Suárez-Araujo, "Classification of Mild Cognitive Impairment Stages Using Machine Learning Methods," 2018 IEEE 22nd International Conference on Intelligent Engineering Systems (INES), Las Palmas de Gran Canaria, Spain, 2018, pp. 000067-000072, doi: 10.1109/INES.2018.8523858.

[4] A. Mellor, S. Boukir, A. Haywood and S. Jones, "Exploring issues of training data imbalance and mislabeling on random forest performance for large area land cover classification using the ensemble margin", J. Photogramm. Remote Sens., vol. 105, pp. 155-168, 2015.

[5] K. Napierala and J. Stefanowski, "Types of minority class examples and their influence on learning classifiers from imbalanced data", J. Intell. Inf. Syst., vol. 46, pp. 563-597, 2016.

[6] P. Yildirim, "Chronic Kidney Disease Prediction on Imbalanced Data by Multilayer Perceptron: Chronic Kidney Disease Prediction", 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), pp. 193-198, 2017.

[7] Pasapitch Chujai, Kittipong Chomboon, Kedkarn Chaiyakhan, Kittisak Kerdprasop and Nittaya Kerdprasop, "A Cluster Based Classification of Imbalanced Data with Overlapping Regions Between Classes", Proceedings of the International Multiconference of Engineers and Computer Scientists, vol. I, 2017.

[8] Hongzhi Wang, Mohamed Jaward Bah and Mohamed Hammad, "Progress in Outlier Detection Techniques: A Survey", IEEE Access.

[9] Y. Qian, Y. Liang, M. Li, G. Feng and X. Shi, "A resampling ensemble algorithm for classification of imbalance problems", Neurocomputing, vol. 143, pp. 57-67, 2014.

[10] Radial-Based Oversampling for Multiclass Imbalanced Data Classification, Bartosz Krawczyk, Member, IEEE, Michal Koziarski, and Michal Wozniak, Senior Member, IEEE.

[11] S. Sridhar and A. Kalaivani, "A Two Tier Iterative Ensemble Method To Tackle Imbalance In Multiclass Classification", IEEE 2020 International Conference on Decision Aid Sciences and Application (DASA).

[12] S. Wang and X. Yao, "Multiclass imbalance problems: Analysis and potential solutions", IEEE Trans. Syst. Man Cybern. B Cybern., vol. 42, no. 4, pp. 1119-1130, Aug. 2012.

[13] J. Jeong, H. Jeong and H. -J. Kim, "BAMTGAN: A Balanced Augmentation Technique for Tabular Data," 2023 9th International Conference on Applied System Innovation (ICASI), Chiba, Japan, 2023, pp. 205-207, doi: 10.1109/ICASI57738.2023.10179533.

[14] A. I. Al-Alawi and M. S. Albuainain, "Machine Learning in Human Resource Analytics: Promotion Classification using Data Balancing Techniques," 2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS), Manama, Bahrain, 2024, pp. 1-5, doi: 10.1109/ICETSIS61505.2024.10459566.

[15] M. M. Hasan Bhuiyan, S. A. Poly and A. Kumar Acharyan, "Comparative Study on Brainstroke Prediction Using Data Balancing Techniques," 2024 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC), Bhubaneswar, India, 2024, pp. 1-5, doi: 10.1109/ASSIC60049.2024.10507962.

[16] G. Parmar, R. Gupta, T. Bhatt, G. J. Sahani, B. Y. Panchal and H. Patel, "Data Re-Balancing using Fuzzy Clustering and SMOT Mechani," 2023 4th International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2023, pp. 1714-1718, doi: 10.1109/ICESC57686.2023.10192964.

[17] S Bej, N Davtyan, M Wolfien, M Nassar and O. Wolkenhauer, "LoRAS: an oversampling approach for imbalanced datasets", Mach Learn [Internet], vol. 110, no. 2, pp. 279-301, 2021,

[18] Desuky AS and S. Hussain, "An Improved Hybrid Approach for Handling Class Imbalance Problem", Arab J Sci Eng [Internet], vol. 46, no. 4, pp. 3853-64, 2021

[19] D Brzezinski, Minku LL, T Pewinski, J Stefanowski and A. Szumaczuk, "The impact of data difficulty factors on classification of imbalanced and concept drifting data streams", Knowl Inf Syst [Internet], vol. 63, no. 6, pp. 1429-69, 2021

[20] H Ahmad, B Kasasbeh, B Aldabaybah and E. Rawashdeh, "Class balancing framework for credit card fraud detect ion based on clustering and similarity-based select ion (SBS)", Int J Inf Technol [Internet], 2022

[21] M. Y. Arafat, S. Hoquef, S. Xuf and D. M. Farid, "Advanced Data Balancing Method with SVM Decision Boundary and Bagging", 2019 IEEE Asia-Pacific Conf. Comput. Sci. Data Eng. CSDE 2019, 2019.

[22] Dataset Used: https://www.kaggle.com/datasets/cyberdeeplearning/veremiap