Chapter 3: Eye State Classification

3.1 Introduction

Eye state classification analysis is a field of research that focuses on developing techniques to accurately classify the state of the eye, such as open or closed, based on eye movements and other related signals. The state of the eye is an important parameter in a variety of applications, including driver drowsiness detection, human-computer interaction, and medical diagnosis.

The classification of eye state is a challenging task due to the complexity of the eye movement patterns and the variability of individual eye characteristics. Eye state classification analysis aims to develop efficient algorithms and models that can accurately classify the state of the eye in real-time using a variety of input signals, including electrooculography (EOG), video recordings, and infrared imaging.

The development of eye state classification analysis has been driven by the need for accurate and reliable systems that can detect the state of the eye in real-time. In the field of driver drowsiness detection, for example, the accurate detection of eye state can prevent accidents caused by driver fatigue. Similarly, in the field of human-computer interaction, eye state classification analysis can enable the development of more intuitive and efficient interfaces that respond to the user's state of attention and focus.

Recent advancements in machine learning and deep learning techniques have significantly improved the accuracy and speed of eye state classification analysis. These techniques have enabled the development of more sophisticated models that can handle complex eye movement patterns and variations in individual eye characteristics. As a result, eye state classification analysis has become an increasingly important and active area of research in the fields of computer vision, signal processing, and machine learning.

The EEG Eye state Classification dataset can also be used to study the characteristics of EEG signals during different eye states. Researchers have analysed the power spectrum and coherence of EEG signals to identify the frequency bands that are most informative for distinguishing between eye-open and eye-closed states. They have also studied the topographical distribution of EEG signals to identify the brain regions that are most involved in eye state classification.

Electroencephalography (EEG) is a non-invasive technique for measuring electrical activity in the brain. EEG signals can be used to study brain function, diagnose neurological disorders, and even control devices using brain-computer interfaces. One application of EEG signals is the classification of eye states, which can be useful for detecting drowsiness and monitoring vigilance in various settings, such as driving, aviation, and medical procedures.

3.2 Implementation

This analysis of the EEG Eye state Classification dataset can provide insights into the classification of eye states using EEG signals. We have used various classification techniques, such as k-nearest neighbors (KNN) classifier, Logit-Boost classifier, Bidirectional LSTM and Supervised Variational Autoencoder, to classify eye states based on the EEG signals and used the Robust Scaler for feature scaling. The performance of these classification techniques evaluated using different metrics such as accuracy, sensitivity, specificity and etc.

3.2.1 Dataset Description

The EEG Eye state Classification dataset is a widely used dataset for analysing EEG signals and classifying eye states. The dataset consists of EEG recordings from 14 channels, taken from 14 healthy subjects with eyes closed or open. The EEG signals were recorded at a sampling rate of 128 Hz, and each recording lasted for 117 seconds. The dataset includes a total of 14,980 EEG samples, which are divided into two classes: eye-open and eye-closed. The eye state was detected via a camera during the Electroencephalogram (EEG) measurement and added later manually to the file after analysing the video frames. '0' indicates the eye-open and '1' the eye-closed state. All values are in chronological order with the first measured value at the top of the data.

This dataset is available in the UCI machine learning repository [29]. The dataset was donated by Rosler and Suendermann [30]. Using ¨ EMOTIV EEG neuroheadset, the data were derived from one continuous EEG measurement. In EMOTIV EEG neuroheadset, “FC5,” “F8,” “T7,” “P7,” “O1,” “AF3,” and “AF4” correspond to a maximum increase and “F7,” “T8,” “P8,” “O2” “F3,” “F4,” and “FC6” to a minimum decrease in eye states. The 14 attributes are for the signals recorded from different locations on the scalp: “FC5,” “F8,” “T7,” “P7,” “O1,” “AF3,” “AF4,” “F7,” “T8,” “P8,” “O2” “F3,” “F4,” and “FC6.” From 14,980, 8,257 records in the corpus are for the eye open and 6,723 samples for the eye closed.

Data Set Characteristics:	Multivariate, Sequential, Time-Series	**Number of Instances:**	14980
Attribute Characteristics:	Integer, Real	**Total Number of Attributes:**	15
Associated Tasks:	Classification	**Missing Values?**	N/A
No. of samples in class 0	8257	**No. of samples in class 1**	6723

3.2.2 System Configuration

The system configuration plays a vital role in the effective operation of deep learning models, as it directly impacts the amount of computational power available for both training and inference processes. Due to their complexity, deep learning models demand significant

computing resources, including processing power, memory, and storage. The number of resources required is typically proportional to the size of the dataset and the complexity of the model. Therefore, larger datasets and more intricate models necessitate more resources to function efficiently. This Chapter analysis was done by using the following machine configuration:

- **Operating System**: Windows 11
- **Processor**: AMD Ryzen 3 4300U with Radeon Graphics 2.70 GHz
- **Installed RAM**: 8.00 GB (7.36 GB usable)
- **SSD**: 512 GB (476 GB usable)
- **System type**: 64-bit operating system, x64-based processor

3.2.3 Results Analysis and Discussion

This result analysis provides a summary of the findings and presents them in a clear and concise manner. Overall, the result analysis and discussion provide the reader with a comprehensive understanding of the research outcomes and their implications for the field. At the first section we will discuss about encryption and decryption analysis then directly jump to the model diagnosis analysis part. In the model diagnosis part, we will see the dataset overview, feature selection with the correlation matrix, Dataset splitting, cross validation and hyperparameter tuning for model optimization. At last, we choose the best model and discuss about the comparison study.

3.2.3.1 Encryption and Decryption time analysis

In this section we will discuss about the Encryption and Decryption time analysis and comparison with an existing model (HBESDM-DLD) [1] which was use the SIMON cipher as encryption technique with optimal key generation process using Group teaching Optimization algorithm (GTOA). Table 3.1 describe the comparison between our SPECK-NOA model and the existing Model.

Table 3.1: Encryption and Decryption time analysis and Comparison

Securing Model	*Data size (%)*	*Encryption Time (sec)*	*Decryption Time (sec)*
SPECK-NOA	**20**	**2.47**	**2.43**
	40	**3.08**	**3.03**
	60	**4.19**	**4.06**
	80	**6.38**	**6.18**
	100	**12.50**	**12.56**
SIMON-GTOA (HBESDM-DLD) [12]	20	3.03	3.05
	40	3.83	3.82
	60	5.10	5.06
	80	7.71	7.62
	100	15.61	15.21

We can see in the Table 3.1 that time required for existing technique is much higher than our SPECK-NOA model for both encryption and decryption. The Table 3.1 is clearly depicted in Figure 3.1 where each model's encryption time and decryption time are showing in different plot basis on percentage of the data size.

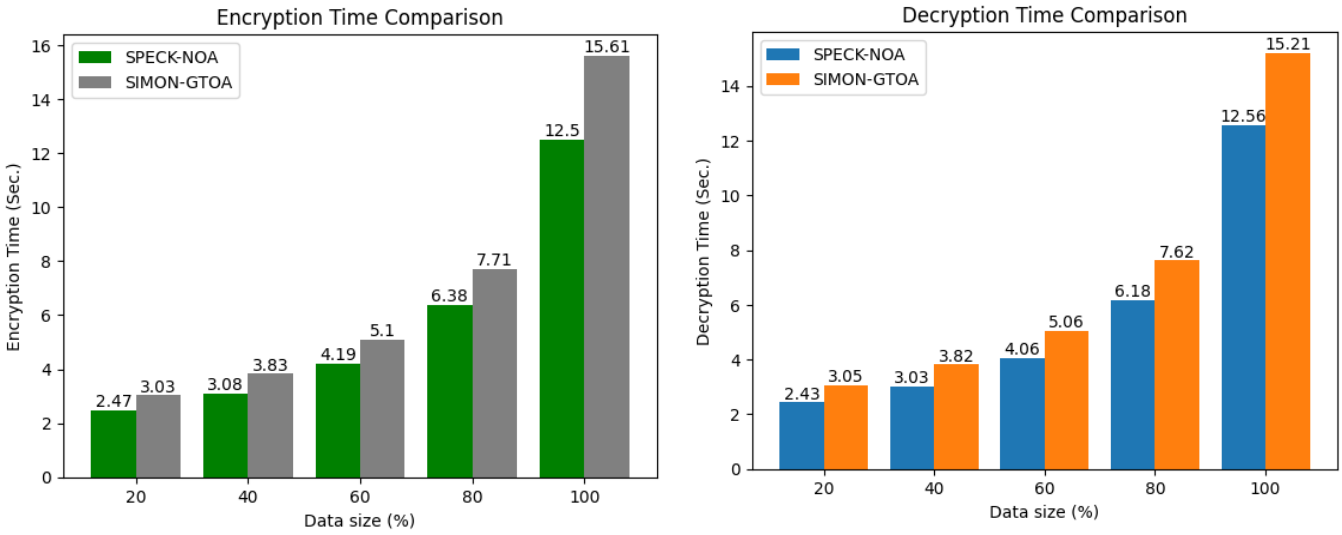

Figure 3.1: Comparison Analysis of Encryption and Decryption time

After decrypt the dataset, we need to diagnosis the dataset. So now we come the diagnosis part, where we will see the analysis of the diagnosis model. The below all sections are for the diagnosis analysis.

3.2.3.2 Dataset Overview

This section 1st plots the ratio of the class in dependent feature then describe the statistics of the dataset with the help of counting, Mean, Standard deviation (std), minimum value, 25% quartile, 50% quartile, 75% quartile and maximum value of each feature. Figure 3.2 describe the ratio of the class level in dependent feature and Table 3.2 clearly describe the dataset. We can see in the figure 3.2, the class level 0 or eye open state have 55.12% of records and the class level 1 or eye closed state have 44.88% of the records.

The goal of the dataset overview is to provide readers with a clear understanding of the dataset and how it was prepared for analysis. It is crucial to include this information in research papers as it allows other researchers to replicate the study and verify the findings. A well-described dataset overview section can also help to establish the credibility and reliability of the study.

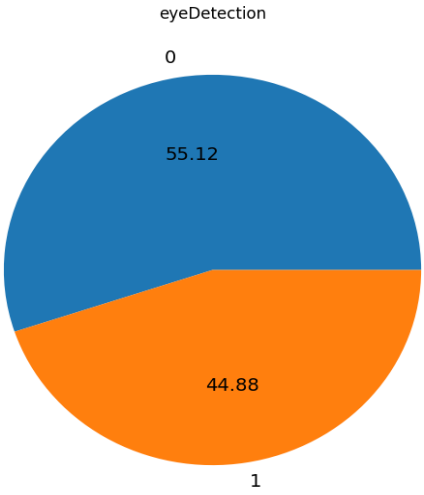

Figure 3.2: Pie plot of the Dependent variable

Table 3.2: Dataset Descriptions

Attributes	Count	Mean	Std	min	25% quartile	50% quartile	75% quartile	max
AF3	14980	4321.92	2492.07	1030.77	4280.51	4294.36	4311.79	309231.00
F7	14980	4009.77	45.94	2830.77	3990.77	4005.64	4023.08	7804.62
F3	14980	4264.02	44.43	1040.00	4250.26	4262.56	4270.77	6880.51
FC5	14980	4164.95	5216.40	2453.33	4108.21	4120.51	4132.31	642564.00
T7	14980	4341.74	34.74	2089.74	4331.79	4338.97	4347.18	6474.36
P7	14980	4644.02	2924.79	2768.21	4611.79	4617.95	4626.67	362564.00
O1	14980	4110.40	4600.93	2086.15	4057.95	4070.26	4083.59	567179.00
O2	14980	4616.06	29.29	4567.18	4604.62	4613.33	4624.10	7264.10
P8	14980	4218.83	2136.41	1357.95	4190.77	4199.49	4209.23	265641.00
T8	14980	4231.32	38.05	1816.41	4220.51	4229.23	4239.49	6674.36
FC6	14980	4202.46	37.79	3273.33	4190.26	4200.51	4211.28	6823.08
F4	14980	4279.23	41.54	2257.95	4267.69	4276.92	4287.18	7002.56
F8	14980	4615.21	1208.37	86.67	4590.77	4603.08	4617.44	152308.00
AF4	14980	4416.44	5891.29	1366.15	4342.05	4354.87	4372.82	715897.00
eyeDetection	14980	0.45	0.50	0.00	0.00	0.00	1.00	1.00

Now on the next section, we'll see the correlation matrix of the dataset and select the important features based on the correlation if needed.

3.2.3.3 Correlation Matrix and Feature Selection

Feature selection is a critical step in machine learning, particularly when working with high-dimensional datasets. One commonly used technique for feature selection is Pearson Correlation, which measures the linear correlation between features and the target variable. The

use of Pearson Correlation for feature selection has been explored in many research papers, and it has been shown to be effective in identifying relevant features and improving model performance. By selecting only, the most relevant features, the model can be trained more efficiently, reducing computational time and improving accuracy. Here we select the independent features based on the relation with the dependent feature. In our study we follow a rule as number of features in our dataset is very low. The rule is if more than 70% of the independent features are correlated at least 5% with the class variable then we discard the features which has less than 5% correlation with the dependent variable. In Figure 3.3, we show the correlation matrix of the whole dataset.

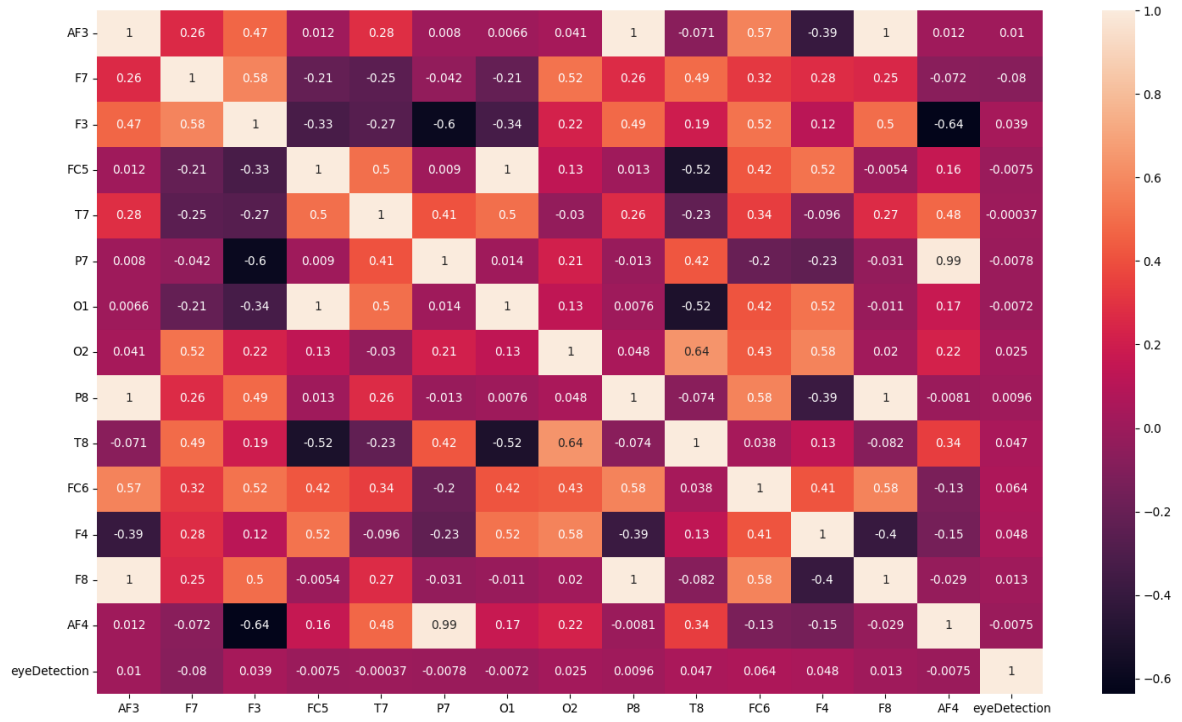

Figure 3.3: Pearson Correlation of features w.r.t each other

Pearson correlation is a statistical method used to measure the linear relationship between two variables. It calculates the correlation coefficient, which ranges from -1 to 1, with values closer to -1 indicating a negative correlation, values closer to 1 indicating a positive correlation, and values closer to 0 indicating no correlation. The Pearson correlation coefficient is a valuable tool for identifying patterns and trends in data, and it can provide insight into the strength and direction of relationships between variables. In Figure 3.3 we can observe that there is very less correlation between each independent variable.

For the feature selection purpose, we mostly focus on the target variable. So, we will select the features by totally depend on the Figure 3.3.1 where visible the Pearson's Correlation of all features w.r.t target. We can observe the figure 3.3.1 and conclude that there are only 2 features which has more than 5% correlation. One is ‘FC6’ which is 6% positively correlated and another one is ‘F7’ which is 8% negatively correlated. So, from the Figure 3.3, we can conclude that not more than 70% of the independent features are at least 5% correlated with the target variable. So, there is no necessity to select the most relevant features.

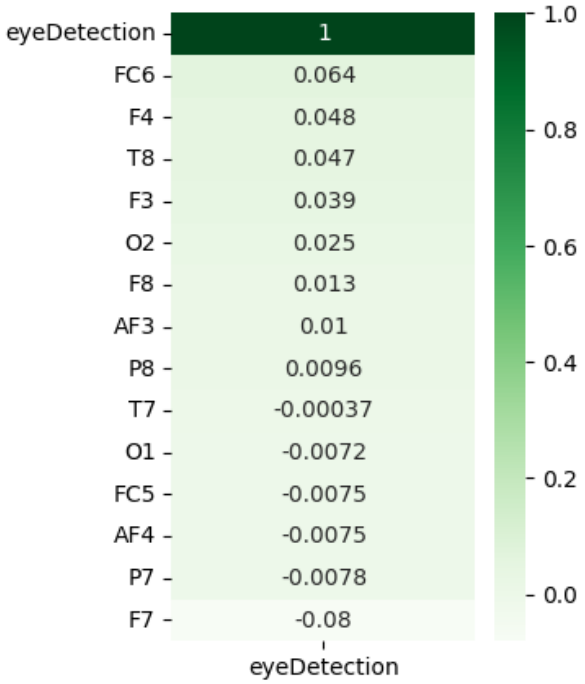

Figure 3.3.1: Pearson's Correlation of features w.r.t target

As no features are relevant so we need to train the models with all the features. Now, we directly move to the model building part that start with splitting the dataset.

3.2.3.4 Dataset Splitting

Dataset splitting is an essential step in research as it helps evaluate the performance of a model on unseen data. By splitting the dataset into training, validation, and test sets, researchers can train the model on one subset, tune the model's hyperparameters on another subset, and evaluate its performance on a final subset that has not been seen during training. Proper dataset splitting ensures that the model's performance is not biased towards the training data, and it can generalize well to new, unseen data. Therefore, careful consideration and reporting of the dataset splitting process are crucial for the credibility and reproducibility of research.

For getting the better performance, 1st we randomly separate the 20% of the data as test data from the dataset. Now from the rest data we split the data into training data and validation data using 3 different validation split ratio (10%, 20%, 25%). For each splitting run the normal model with the scaled training data and validate with the scaled validation data (Robust Scaler is used to scale the training data then transform the validation data based on that). Performance Validation was done using three different classification evaluation metrics (Precision, Recall and F1 Score) to find the best validation split ratio for cross validation. Table 3.3 describe each model’s performance metrics for the validation data.

Table 3.3: Splitting Evaluation Metrics on validation data

Model	Train-Validation Split Ratio	Precision	Recall	F1 Score
KNN	90%-10%	0.9605	0.9586	0.9595
	80%-20%	0.9584	0.9563	0.9572
	75%-25%	**0.9606**	**0.9593**	**0.9599**
Logit-Boost	90%-10%	0.7326	0.7285	0.7297
	80%-20%	0.7412	0.7351	0.7364
	75%-25%	**0.7464**	**0.7434**	**0.7443**
BI-LSTM	90%-10%	0.7607	0.7568	0.7595
	80%-20%	0.7548	0.7563	0.7527
	75%-25%	**0.7686**	**0.7593**	**0.7598**
SVAE	90%-10%	0.8570	0.8486	0.8075
	80%-20%	**0.8568**	**0.8775**	**0.8527**
	75%-25%	0.8076	0.8175	0.8298

If we minutely observe the Table 3.3, we can conclude that KNN classifier, Logit-Boost classifier, Bi-LSTM has highest precision, recall and f1 score on **75%-25%** split ratio than all other split. So, the best split ratio is set for these 3 models as 75%-25%. And for SVAE, 80%-20% split has the highest recall and f1 score than all other split. So, the best split ratio is set for SVAE is 80%-20% as maximum (2) metrics are better.

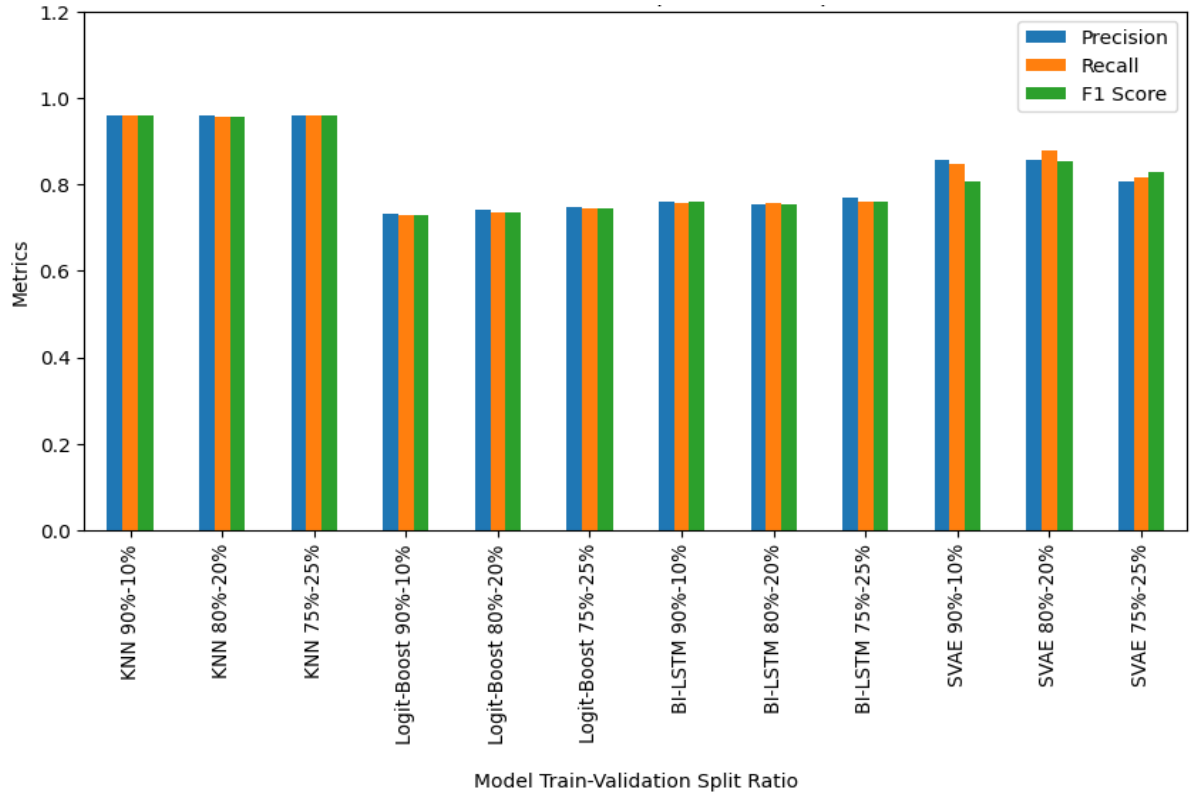

Figure 3.4: Comparative analysis of all model basis on dataset splitting

To finding the results of the Table 3.3, we used

- Default parameter for **KNN and Logit-Boost classifier**.
- **For Bi-LSTM**, number of LSTM layer 1 with 128 nodes, number of dense layers 1 with 32 nodes, epochs 10, batch size 128 and the loss function 'binary crossentropy'
- **For SVAE**, number of encoder and decoder layers 1 with 128 nodes, Number of dense layers 1 with 32 nodes, epochs 10, batch size 128, supervised loss 'binary crossentropy'

The figure 3.4 depicted for comparative analysis between each model. From that figure we can say that the KNN classifier with default parameter works better than others models. Now with the best validation split ratio for each model, we will see the cross-validation performance in the next section.

3.2.3.5 Cross Validation

Cross validation involves partitioning a dataset into multiple subsets and iteratively training and testing the model on different subsets to obtain an estimate of its performance. Cross-validation helps to prevent overfitting and provides a more accurate estimate of the model's generalization performance. It also allows for the comparison of different models and their hyperparameters to choose the best-performing one. This study used the types of cross-validation that are k-fold cross-validation, and stratified k-fold cross-validation and shuffle split cross-validation. In each Cross-validation technique for each model the training and validation data are also scaled by Robust Scaler. Table 3.4 describe the performance metrics basis on the validation data. Here the mean of each evaluation metrics is calculated. Because of multiple combinations of the subsets of 80% train data with the best split ratio, there is generate multiple evaluation metrics for each model, so we take the average of each evaluation metrics. To select the best Cross validation technique for hyperparameter tuning the mean balanced accuracy, mean accuracy and mean AUC score are used for each model.

Table 3.4: Cross Validation evaluation metrics

Model	Cross Validation	Mean Balanced Accuracy	Mean Accuracy	Mean AUC score
KNN	Kfold	0.9503	0.9286	0.9350
	Stratified Kfold	0.9484	0.9536	0.9570
	Shuffle Split	**0.9660**	**0.9639**	**0.9689**
Logit-Boost	Kfold	0.8327	0.8250	0.8270
	Stratified Kfold	0.8412	0.8351	0.8364
	Shuffle Split	**0.8446**	**0.8434**	**0.8443**
BI-LSTM	Kfold	0.7877	0.7968	0.7893
	Stratified Kfold	0.7746	0.7362	0.7227
	Shuffle Split	**0.7986**	**0.7893**	**0.7998**
SVAE	Kfold	**0.8670**	**0.8886**	**0.8675**
	Stratified Kfold	0.8668	0.8657	0.8572
	Shuffle Split	0.8367	0.8275	0.8398

If we carefully observe the Table 3.4, we can conclude that KNN classifier and Logit-Boost classifier has the highest mean balanced accuracy, mean accuracy and mean AUC score on **shuffle split** than all other CV technique. For Bi-LSTM, Shuffle split CV technique has highest balanced accuracy and AUC score than all other CV techniques (maximum 2 metrics are better so chosen). So, the best CV technique is set for these 3 models as **shuffle split**. And for SVAE, **kfold** cross validation has chosen as the best CV technique because **kfold** has highest Balanced Accuracy, Accuracy and AUC score than all other CV techniques.

To finding the results of the Table 3.4, used the same parameter mentioned in the preceding section.

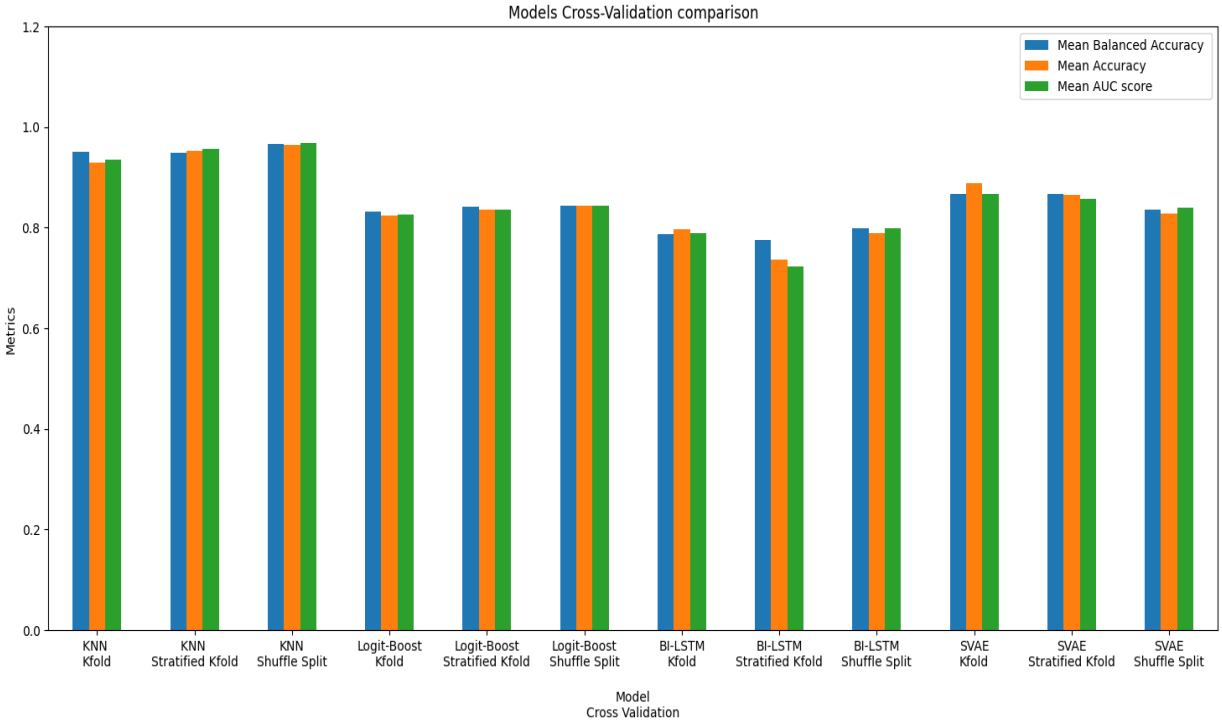

Figure 3.5: Comparative analysis of all model basis on cross-validation

The figure 3.5 can use for comparative analysis between each model. From that figure we can say that the KNN classifier with default parameter works better than others models. Now with the best cross-validation technique for each model, we will see the Hyperparameter tuning in the next section.

3.2.3.6 Hyperparameter Tuning

In this part, first we found the best parameter based on the cross validation with the training and validation data using a parameter list for each model. Then Fit the models with the 80% scaled train data and predict the evaluation metrics based on the 20% scaled test data. This test results helps to choose the best model. In Table 3.5, all test results are present. The updated version of all model is following:

Updated KNN:

Best Split: 75%-25%

Best CV technique: Shuffle Split

After the performance evaluation--

Best Model optimization technique: Optuna

Reason: The mention optimization technique has highest F1 Measure, Specificity and Cohen Kappa score than all other optimization techniques.

Updated Logit-Boost:

Best Split: 75%-25%

Best CV: Shuffle Split

After the performance evaluation--

Best Model optimization: Hyperopt

Reason: The mention optimization technique has highest F1 Measure, Specificity and Cohen Kappa score than all other optimization techniques.

Updated BI-LSTM:

Best Split: 75%-25%

Best CV: Shuffle Split

After the performance evaluation--

Best Model optimization: Hyperopt

Reason: The mention optimization technique has highest F1 Measure, Specificity and Cohen Kappa score than all other optimization techniques.

Updated SVAE:

Best Split: 80%-20%

Best CV: Kfold

After the performance evaluation--

Best Model optimization: Optuna

Reason: The mention optimization technique has highest F1 Measure, Specificity and Cohen Kappa score than all other optimization techniques.

If we look at the Table 3.5, KNN is the best model among all other model. The SVAE model also get better result compare to another model. The KNN get best result may be of the non-linearity in the dataset. The Figure 3.6 also use for comparative analysis between each model.

Table 3.5: Model Optimization using Hyperparameter Tuning

Model	Model Optimization	F1 Measure	Specificity	Kappa
KNN	Hyperopt	0.9484	0.9536	0.9461
	Optuna	**0.9767**	**0.9639**	**0.9528**
	Cuckoo search	0.9627	0.9450	0.9461
Logit-Boost	Hyperopt	**0.8464**	**0.8434**	**0.8531**
	Optuna	0.8377	0.8428	0.8428
	Cuckoo search	0.8246	0.8362	0.8461
BI-LSTM	Hyperopt	**0.9270**	**0.9386**	**0.9261**
	Optuna	0.9268	0.9357	0.9128
	Cuckoo search	0.9167	0.9275	0.9061
SVAE	Hyperopt	0.9384	0.9436	0.9316
	Optuna	**0.9460**	**0.9593**	**0.9428**
	Cuckoo search	0.9446	0.9571	0.9411

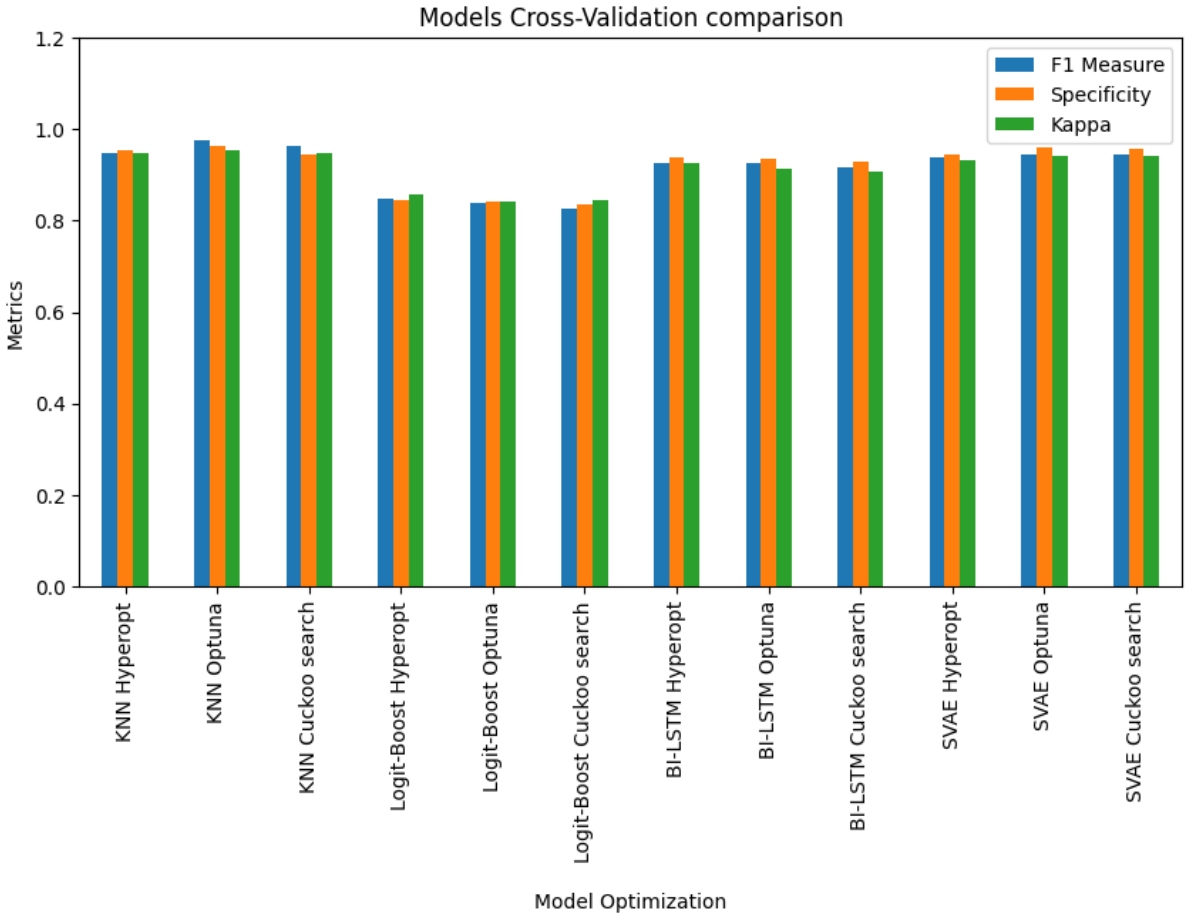

Figure 3.6: Comparative analysis of all model basis on Hyperparameter tuning

Now we compared our model with all other model using some special evaluation metrics in the next section.

3.2.3.7 Comparative Study

The Table 3.6 describe the comparative analysis that involves analysing and comparing two or more entities or phenomena to identify similarities, differences, and patterns. The goal of a comparative study is to gain a deeper understanding of the entities being compared, to identify the strengths and weaknesses of each, and to draw conclusions about their relative performance.

Table 3.6: Comparative analysis with existing model

Models	Precision	Recall	Accuracy	F1 Score	Kappa
KNN	0.97	0.96	0.97	0.97	0.95
SVAE	0.94	0.96	0.95	0.96	0.94
HBESDM-DLD (VAE) [12]	0.93	0.95	0.93	0.93	0.92
Bi-LSTM	0.92	0.94	0.93	0.92	0.93
Logit-Boost	0.83	0.84	0.84	0.83	0.85

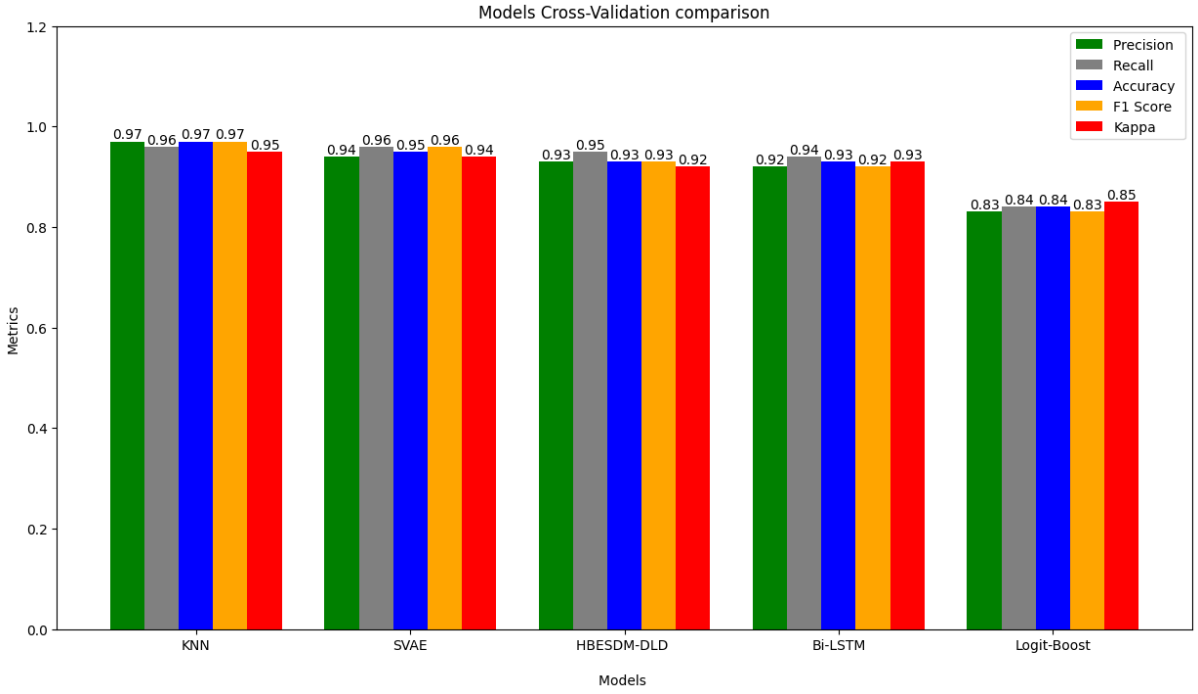

Figure 3.7: Comparative Analysis with existing model

In the figure 3.7 we clearly observe that KNN Classifier give the highest 4 evaluation metrics among five. Just the recall is same as SVAE. So, we can conclude that KNN classifier using Optuna optimization technique give better performance than other model based on five classification evaluation metrics (Precision, Recall, Accuracy, F1 Score, Cohen Kappa). These five-evaluation metrics are used to compare because of the existing model's [12] performance is measured by the same.