# Title:
Comparative analysis of Machine Learning Algorithms on **Waveform Generator**.


# Abstract:
The aim of the project is to compare and evaluate the performance of various machine learning models in classifying waveform generator. A multi-feature dataset of waveform signals was considered. The compared models are Logistic Regression, Random Forest, Support Vector Machine, Gradient Boosting (XGBoost), and K-Nearest Neighbors. The data set was split into a training set and a test set with standardized pre-processing when required. Performance was observed based on accuracy and precision measurements, confusion matrices, and classification reports. Tests indicated that ensemble models like Random Forest and XGBoost outperformed other models reliably for classification on waveform generator and were promising candidates for such use. These two algorithms were more robust and showed greater reliability in accurately classifying the waveform generator signals. Logistic Regression and SVM performed moderately well, while KNN demonstrated lower performance due to its sensitivity to feature scaling and high-dimensional data. The tool is demonstrated in the form of an interactive user Streamlit web application for rapid prototyping and visualization.



# Keywords:
Waveform, Classification, Machine Learning, Model Comparison, Streamlit


# 1. Introduction
Waveform Generator plays an important role in different signal processing applications such as telecommunication, seismology, and medical imaging diagnosis. The study explores the performance of different classification models to determine the optimal model for waveform signal classification. The study provides a tangible experience via a Python-based Streamlit web interface for model selection, training, and performance visualization.

# 2. Suggested Methodology:
A modular pipeline was used to compare selected classification models on a standardized dataset of waveforms.

**a. Datasets**
The waveform.csv dataset is taken from Kaggle which includes a few numerical features representing waveform signals along with a target class for supervised classification.

```
df = pd.read_csv('waveform.csv')
print("Dataset shape:", df.shape)
print(df.head())
```

```
Dataset shape: (5000, 22)
   Column1  Column2  Column3  Column4  Column5  Column6  Column7  Column8  \
0    -1.23    -1.56    -1.75    -0.28     0.60     2.22     0.85     0.21
1    -0.69     2.43     0.61     2.08     2.30     3.25     5.52     4.55
2    -0.12    -0.94     1.29     2.59     2.42     3.55     4.94     3.25
3     0.86     0.29     2.19    -0.02     1.13     2.51     2.37     5.45
4     1.16     0.37     0.40    -0.59     2.66     1.00     2.69     4.06

   Column9  Column10  ...  Column13  Column14  Column15  Column16  Column17  \
0    -0.20      0.89  ...      2.89      7.75      4.59      3.15      5.12
1     2.97      2.22  ...      1.24      1.89      1.88     -1.34      0.83
2     1.90      2.07  ...      2.50      0.12      1.41      2.78      0.64
3     5.45      4.84  ...      2.58      1.40      1.24      1.41      1.07
4     5.34      3.53  ...      4.30      1.84      1.73      0.21     -0.18

   Column18  Column19  Column20  Column21  Column22
0      3.32      1.20      0.24     -0.56         2
1      1.41      1.78      0.60      2.42         1
2      0.62     -0.01     -0.79     -0.12         0
3     -1.43      2.84     -1.18      1.12         1
4      0.13     -0.21     -0.80     -0.68         1

[5 rows x 22 columns]
```

**b. Data Preprocessing**

1. Feature-target split using pandas.
2. Removing null values / redundant values.
3. Fixing missing values.
4. Train-test split using sklearn.model_selection.train_test_split.
5. Standardization of features using StandardScaler.
6. Stratification in order to maintain class balance in splits.

**c. Model Selection**

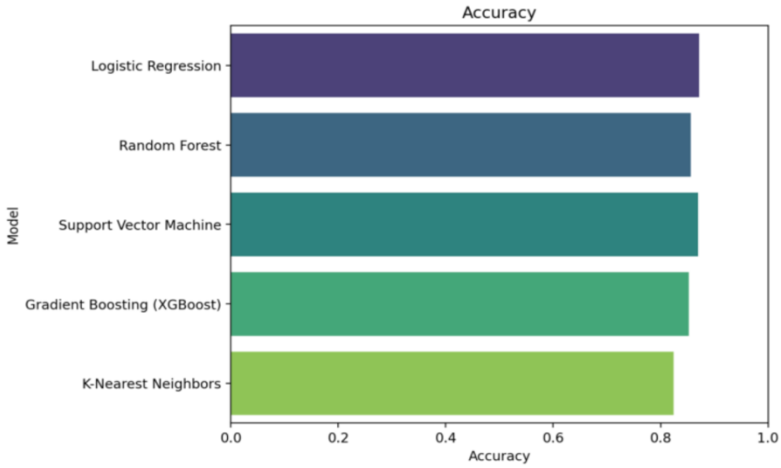
Five Classification models were considered :

1. Logistic Regression
   Logistic regression predicts probabilities and assigns data points to binary classes (e.g., spam or not spam).
2. Random Forest
   Random forest is an ensemble method that combines multiple decision trees.
3. Support Vector Machine (SVM)
   SVMs find the best boundary (called a hyperplane) that separates data points into different classes.
4. XGBoost
   Advanced version of Gradient Boosting that includes regularization to prevent overfitting. Faster than traditional Gradient Boosting, for large datasets.
5. K-Nearest Neighbors
   KNN is a simple algorithm that predicts the output for a new data point based on the similarity (distance) to its nearest neighbors in the training dataset, used for both classification and regression tasks.

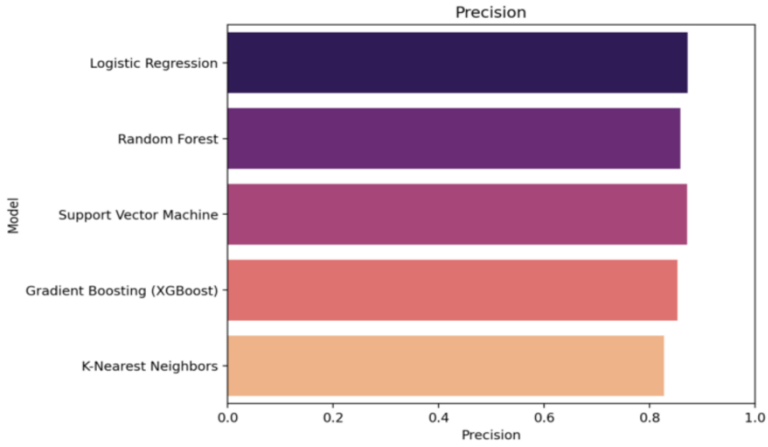
**d. Evaluation Metrics**

Models were assessed based on:

1. Accuracy




2. Weighted Precision

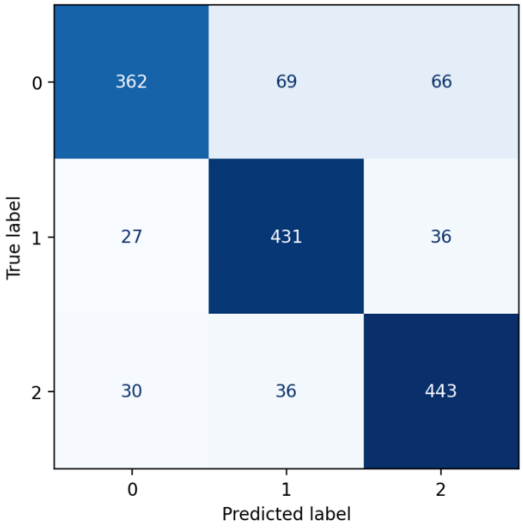



3. Classification Report

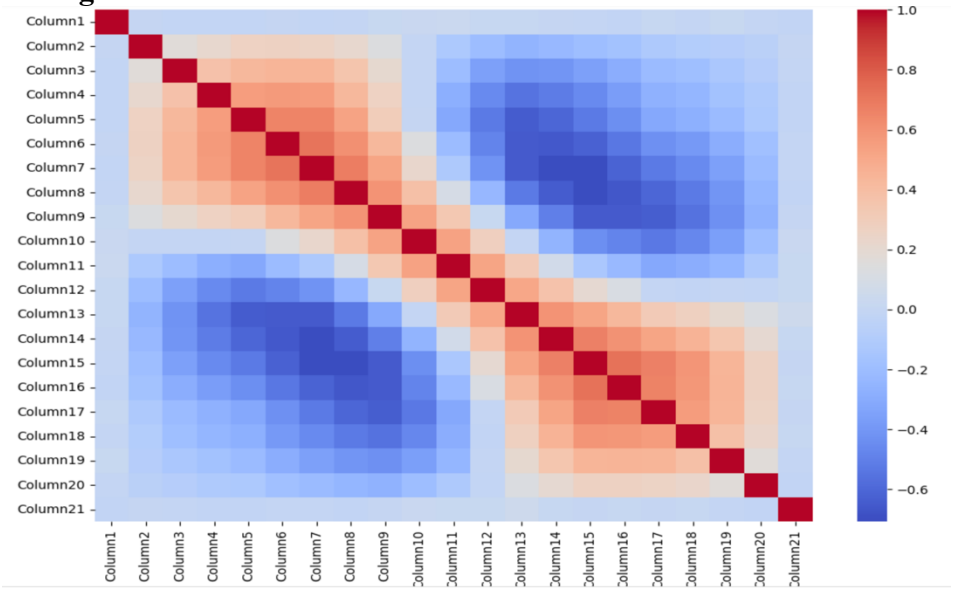
## Classification Report: K-Nearest Neighbors

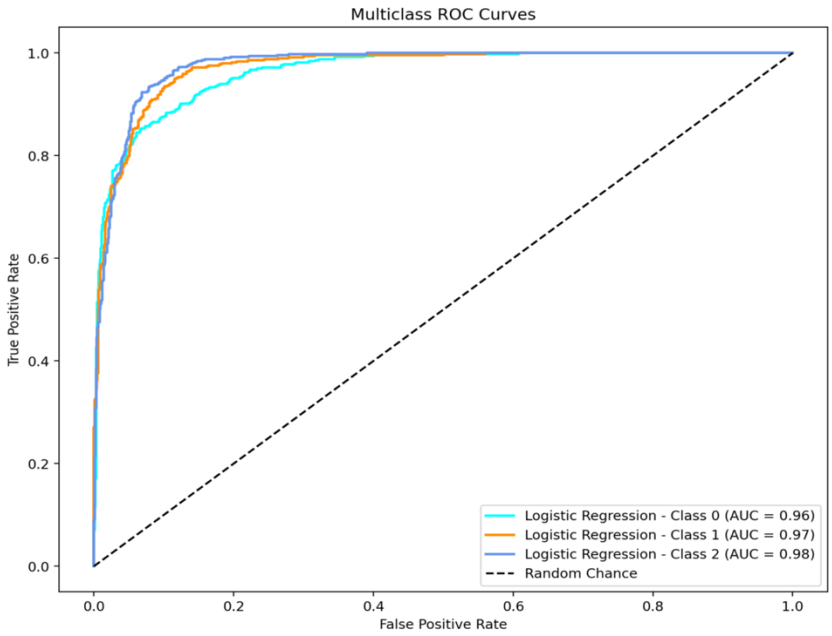
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.86 | 0.73 | 0.79 | 497 |
| 1 | 0.80 | 0.87 | 0.84 | 494 |
| 2 | 0.81 | 0.87 | 0.84 | 509 |
| accuracy | | | 0.82 | 1500 |
| macro avg | 0.83 | 0.82 | 0.82 | 1500 |
| weighted avg | 0.83 | 0.82 | 0.82 | 1500 |

4. Confusion Matrix (plotted with matplotlib and sklearn)




## e. Pictorial Diagram



# 3. Result & Discussion:

All the models were tested on the above parameters. The results are as follows:

1. XGBoost and Random Forest performed the best in terms of accuracy and precision.
2. SVM and KNN were assisted by feature scaling but performed less well.
3. Logistic Regression performed reasonably well, implying that linear separability is limited in this data.
4. Confusion matrices revealed class-level strengths and weaknesses.
5. Accuracy and precision bar plots readily demonstrated model comparison.

# 4. Conclusion & Future Work:

Ensemble methods, particularly Random Forest and XGBoost, provided superior performance for waveform classification as they were more robust and showed greater reliability in accurately classifying the waveform generator signals.
Future work could include:

1. Hyperparameter tuning for further performance improvement
2. Exploring deep learning techniques like CNNs
3. Applying dimensionality reduction techniques
4. Verifying real-time waveform classification accuracy

# References:


1. Waveform Generator Dataset from Kaggle : https://www.kaggle.com/datasets/annsanababy/waveform-dataset?select=waveform.data
2. Colab Notebook for ML code : https://colab.research.google.com/drive/14oGwtsebm6QyyWYwGpfsrUrTa81sH7jL#scrollTo=pT05PoJGyVqd
3. Logistic Regression : https://www.geeksforgeeks.org/understanding-logistic-regression/
4. Random Forest : https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/
5. KNN : https://www.ibm.com/think/topics/knn#:~:text=The%20k%2Dnearest%20neighbors%20(KNN)%20algorithm%20is%20a%20non,used%20in%20machine%20learning%20today.
6. XGBoost : https://www.nvidia.com/en-in/glossary/xgboost/#:~:text=What%20is%20XGBoost%3F,%2C%20classification%2C%20and%20ranking%20problems.
7. SVM : https://scikit-learn.org/stable/modules/svm.html
8. Matplotlib https://www.w3schools.com/python/matplotlib_intro.asp and Seaborn https://www.w3schools.com/python/numpy/numpy_random_seaborn.asp for graph plotting
9. Streamlit for project Deployment : https://share.streamlit.io/?utm_source=streamlit&utm_medium=referral&utm_campaign=main&utm_content=-ss-streamlit-io-cloudpagehero
10. Scikit-learn for more information of machine learning algorithms and its working https://scikit-learn.org/stable/