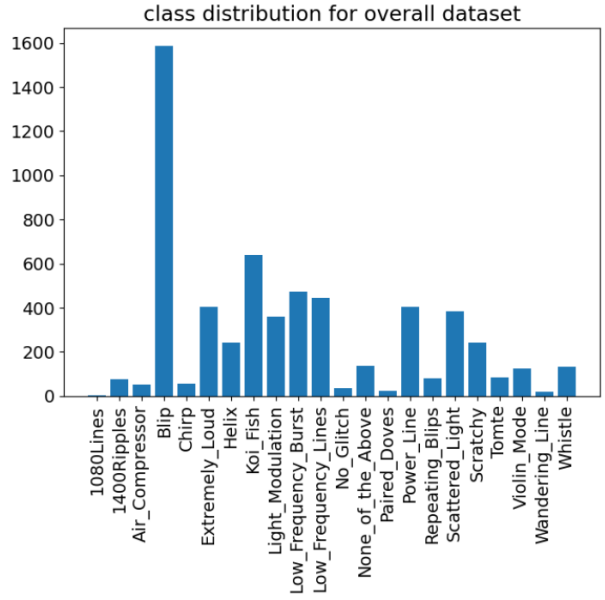
Identify Glitches in Gravitational Waves

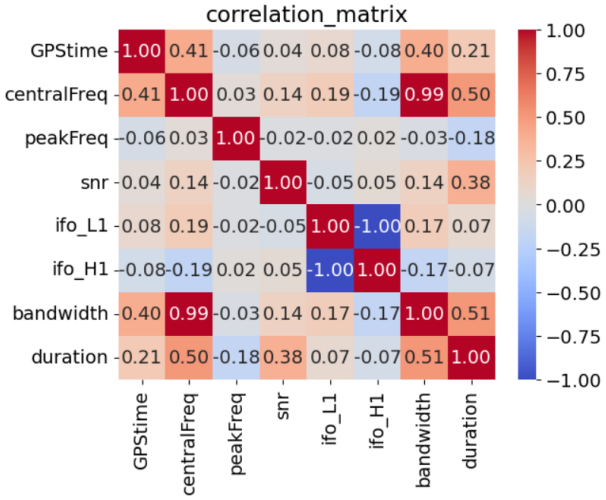
Name:	**Ujjwal Kishor Sahoo**
Registration No./Roll No.:	21293
Institute/University Name:	IISER Bhopal
Program/Stream:	Physics
Problem Release date:	August 17, 2023
Date of Submission:	November 19, 2023

1 Introduction

Gravitational waves are distortions in spacetime caused by accelerated masses propagating as waves from their source at the speed of light. Detecting gravitational waves requires understanding instrument responses amid environmental noise. Anomalous non-Gaussian noise events, termed 'Glitches,' are crucial to study due to their frequent occurrence in LIGO data, posing challenges in distinguishing them from genuine gravitational wave signals. The training data provides glitch characteristics, including bandwidth. There are no missing values in the data set, eliminating the need for imputation. This data set poses a compelling classification challenge due to its multi-class nature, with each class featuring varying instances. Machine learning models such as K-Nearest Neighbours, Support Vector Machines, Logistic Regression, Random Forest, and Decision Trees from the sci-kit-learn library are employed to classify glitches. The model is then applied to the test data, including glitches and unique identification labels. In a related aspect of the project, One Hot Encoding handles the categorical variable $'ifo'$.



((a)) The class overall distribution



((b)) The correlation matrix for the dataset

2 Methods

This project uses machine learning models such as KNNs, SVMs, Logistic Regression, AdaBoost, Decision Trees, and Random Forest. Handling categorical values in the $'ifo'$ feature involves using

Table 1: Performance Of Different Classifiers Using All Features

Classifier	Precision	Recall	F-measure
Decision Tree	0.70	0.71	0.70
K-Nearest Neighbor	0.64	0.55	0.58
Logistic Regression	0.50	0.44	0.44
Random Forest	0.83	0.75	0.77
Support Vector Machine	0.01	0.105	0.014
AdaBoost Technique RF	0.69	0.70	0.69

One-Hot Encoding, which converts them into numerical values for model compatibility. The code runs twice on the training data, initially without $'ifo'$, resulting in low scores for all models. The second run involves One-Hot Encoding, leading to improved results. Normalisation is applied, and a correlation matrix reveals a strong link between $'bandwidth'$ and $'centralFreq.'$

The features $'id'$ and $'GPStime'$ are removed as they weren't providing any vital information about the dataset, and by individual knowledge, they can be neglected. And while neglecting the two features, we are certainly getting better results than expected.

Experiments are run, removing bandwidth and centralFreq individually. Removing bandwidth results in an F-measure of 0.808, and centralFreq yields 0.804. During normalisation, removing these features individually leads to lower F-measures, highlighting the impact of correlated features on model performance.

Through GridSearchCV, I tuned the hyper-parameters for various classifiers like SVM, RFC, LR, etc., including hyper-parameters such as C, $kernel$, $random\_state$, etc. The parameters are tuned perfectly, and then we get the desired result, as in Table 1 and Table 2.

This project uses effective preprocessing, utilising One-Hot Encoding and feature removal experiments. The correlation matrix guides decisions, uncovering the critical balance between features. The iterative nature of model development is necessary, which leads to effective machine learning models. The focus is on practical strategies, emphasising the essential steps to enhance model performance and the iterative process of achieving effective outcomes.

Click GitHub to visit the GitHub repository.

3 Experimental Setup

Evaluation criteria for machine learning models include precision, measuring positive prediction accuracy; recall, assessing the model's ability to identify relevant instances; F1 score, striking balance between precision and recall. Implementing advanced models like K-nearest Neighbors (KNN), Support Vector Machines (SVM), Random Forest, Decision Trees, and Logistic Regression demands fine-tuning hyperparameters. For the case of SVM, we need to tune the hyperparameters such as C and $kernel$. While doing the same for all the classifiers, we found out that the evaluation criteria of the Random Forest classifier was the best fit for this model. I tuned the hyperparameters $n\_estimators$ (number of trees in the forest), $criterion$ (function used to measure the quality of a split), max_depth (maximum depth of each tree in the forest) and min_sample_split (minimum number of samples required to split an internal node) to 130, gini, 20 and 3, respectively and got the score of 0.77.

The library involving all the necessary classifiers is SK-Learn, and I ran the code multiple times to find the best result. Matplotlib and Seaborn libraries were used to plot the confusion matrices between the classes.

Table 2: Performance Of Different Classifiers Using All Features Except Bandwidth

Classifier	Precision	Recall	F-measure
Decision Tree	0.72	0.71	0.71
K-Nearest Neighbor	0.50	0.49	0.50
Logistic Regression	0.29	0.29	0.27
Random Forest	0.82	0.75	0.77

4 Results and Discussion

Using the Random Forest classifier and excluding "bandwidth" produced the graph below. The graph in (Figure 2) shows how many data points are classified into which classes. The second plot in (Figure 3) tells us about the confusion matrix between classes while removing the bandwidth feature in the test dataset. From the training dataset, the distribution of the glitches is plotted above, and following that, we can see that the result I got is proportional to it. I have made a table for the classifier I used by indicating its Precision, Recall and F-measure.

Table 3: Performance Of Different Classifiers Using All Features

Classifier	Precision	Recall	F-measure
Random Forest	0.86	0.78	0.80

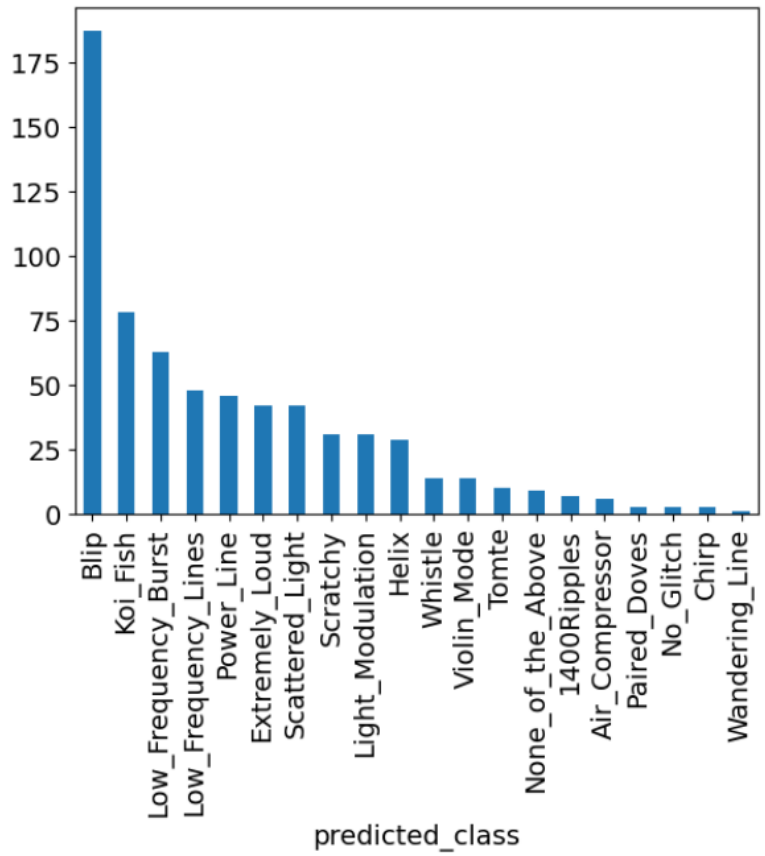


Figure 2: Class Distribution of Test Dataset

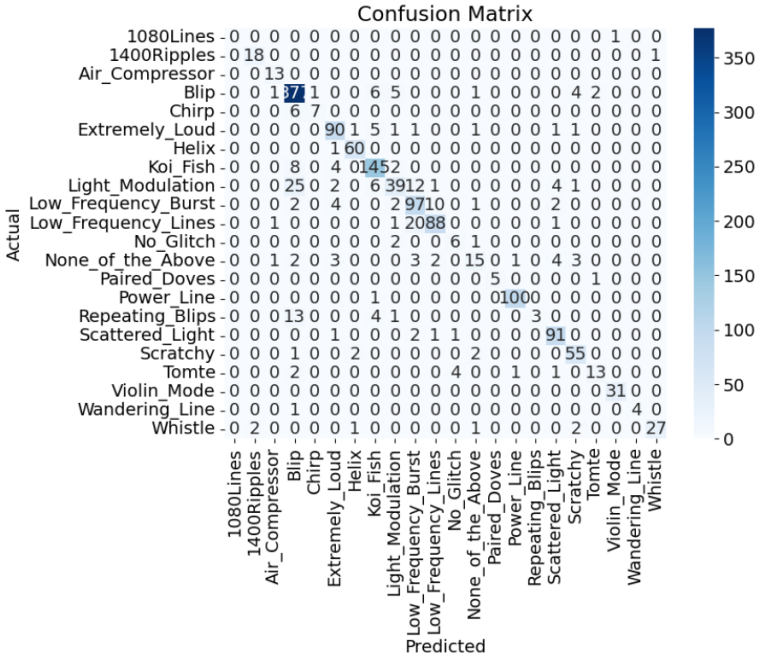


Figure 3: The confusion matrix for the dataset after removing bandwidth

5 Conclusion

From the correlation matrix, we found that bandwidth and centralFreq are highly compatible and removing one of them will positively affect our model. The glitch distribution plot shows fewer non-glitch events and more Blip glitches, challenging a reliable classification model. Detecting gravitational waves from black hole mergers is challenging due to physics laws, and 'No glitch' events are rare.

Building a solid glitch classification model reveals the complexity of spotting gravitational waves. Despite challenges, finding a solution is crucial for the Gravitational Wave community. Combining sturdy hardware with adaptable machine learning, especially Quantum Computation, is practical.

In summary, detecting gravitational waves needs a mix of tech and machine learning. Using robust hardware and flexible machine learning, this collaboration could uncover new aspects of gravitational waves and enhance our understanding of astrophysical phenomena.

References

[1] Robert E. Colgan, K. Rainer Corley, Yenson Lau, Imre Bartos, John N. Wright, Zsuzsa Márka, and Szabolcs Márka (2020), Efficient gravitational-wave glitch identification from environmental data through machine learning, **Physical Review**.

[2] R. Duda, P. Hart (2001), Pattern Classification, 2nd ed.