Identify Glitches in Gravitational Waves

Name:	**Ankur Kumar**
Registration No./Roll No.:	21043
Institute/University Name:	IISER Bhopal
Program/Stream:	DSE
Problem Release date:	August 17, 2023
Date of Submission:	November 19, 2023

1 Introduction

Gravitational wave data analysis is the study of identifying non Gaussian noise transients known as 'Glitches'. A high occurrence rate of glitches in data can mimic or hazard true gravitational wave signals. Hence successful classification of glitches is an important part of gravitational wave analysis. The objective of this project is to classify the types of glitches in gravitational wave data.
The challenge at hand pertains to a classification task involving a data set comprising 6000 instances distributed across 22 classes. Notably, the data set exhibited skewness,1 with a predominant concentration in a specific class labeled 'blip.' To rectify this imbalance, oversampling methods were employed.
Various machine learning algorithms were applied to achieve the desired classification outcome. Among these, the Random Forest Classifier emerged as the most effective, yielding optimal evaluation results.

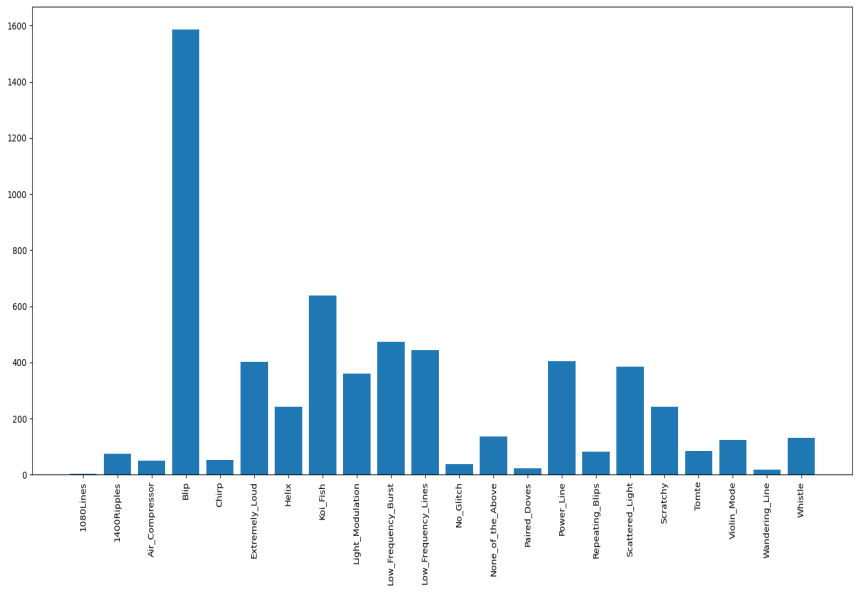


Figure 1: Overview of Data Set

2 Methods

The suggested approach involves leveraging a Random Forest Classifier with optimized hyperparameters through cross-validation. Notably, substantial enhancements are observed when the model un-

dergoes training on an augmented dataset. This augmentation is achieved by employing the Synthetic Minority Oversampling Technique (SMOTE) [1], a method that generates new samples through a K-Nearest Neighbors (KNN) based approach applied to the existing data points.
SMOTE, or Synthetic Minority Oversampling Technique, plays a pivotal role in addressing class imbalance by generating synthetic instances of the minority class. By employing a K-Nearest Neighbors algorithm, SMOTE identifies the minority class samples and creates synthetic counterparts in feature space.
SMOTE works by identifying minority class instances and creating synthetic samples in between them. It does so by selecting a minority class instance, finding its k-nearest neighbors, and then generating synthetic instances along the line segments connecting the chosen instance with its neighbors. This oversampling technique effectively balances class distribution, enhancing the model's ability to learn patterns from the minority class and improving overall classification performance.

The code can be found from the given github link[1]
Various models were explored to address the specific problem at hand. Among them, K-Nearest Neighbors (KNN), Decision Tree, Adaptive Boosting, and Bagging Classifier demonstrated commendable performance. However, SVM, MLP Classifier [2], and Logistic Regression exhibited suboptimal results when applied to the given dataset.

MinMax scaler was used to scale the data. It ensured that all features are on a consistent scale, preventing larger-magnitude features from dominating certain algorithms. This standardization facilitated faster convergence in gradient-based optimization methods and enhances the performance of distance-sensitive algorithms like K-Nearest Neighbors. Additionally, Min-Max scaling helped mitigating the influence of outliers, making models more robust.

One-Hot Encoding was applied to the 'ifo' column since it held categorical data with values 'L1' and 'H1.' Since there is no inherent rank associated with these values, ordinal encoding isn't suitable. Therefore, One-Hot Encoding, chosen over ordinal encoding, proves to be a better fit for this scenario. This method has been employed in the preprocessing step to appropriately represent the categorical information in the data.

In our pursuit of optimizing the model's performance, Principal Component Analysis (PCA) was implemented along with various feature selection methods. While these techniques successfully reduced computing time by dimensionality reduction, the trade-off came in the form of a noticeable decline in model efficacy. Despite the efficiency gains, the model experienced a reduction in accuracy, F1 score, recall, and precision.

The code for this project contains pipeline for classifying glitch data. It uses different classifiers like AdaBoost, Bagging, Neural Networks, K-Nearest Neighbors, Decision Trees, Logistic Regression, Naive Bayes, Random Forest, and Support Vector Machines. The classification class is created with options for the data path, classifier choice, and the number of selected features. The code loads and preprocesses data, selects a classifier, and performs cross-validation with hyperparameter tuning. It evaluates the model using metrics like precision, recall, and F1-score, displaying the results along with a confusion matrix. The code handles imbalanced datasets using SMOTE for oversampling. It's designed for glitch data stored in CSV files, and you can choose the classifier and its parameters when you create an instance of the class.

3 Experimental Setup

The evaluation criteria employed in assessing the model's performance encompassed four key metrics: F1 measure, accuracy, precision, and recall [3]. F1 measure, the harmonic mean of precision and recall, offers a balanced assessment of a model's ability to simultaneously achieve high precision and recall. Accuracy represents the ratio of correctly predicted instances to the total instances, providing an overall measure of correctness. Precision measures the accuracy of positive predictions, indicating the model's ability to correctly identify true positive instances. Recall, on the other hand, gauges the model's capacity to capture all relevant instances by measuring the ratio of true positive predictions to the total actual positives. Together, these metrics offer a comprehensive evaluation of the model's

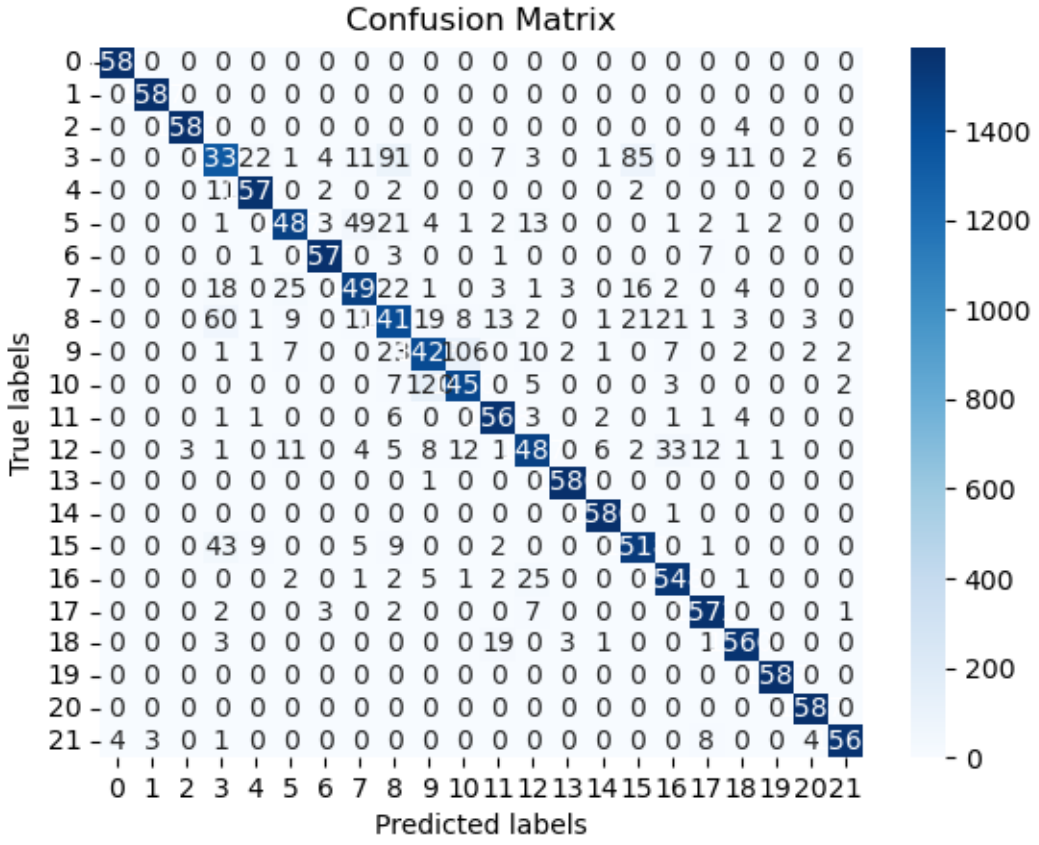
[1] https://github.com/Aksh3141/ML-project.git

effectiveness in terms of precision, recall, overall accuracy, and the harmonic balance between precision and recall captured by the F1 measure. In a Random Forest model, several hyperparameters influence the behavior and performance of the ensemble. Some important parameters which were tuned to get the best results are as follow:-

- Criterion: criterion is a parameter that defines the function used to measure the quality of a split in each decision tree within the forest. The two commonly used criteria are "gini" and "entropy." "Gini" measures impurity in terms of misclassification, while "entropy" considers information gain.
- Max Depth: max depth determines the maximum depth of each decision tree in the Random Forest. A deeper tree can capture more complex relationships in the data, but it also increases the risk of overfitting, especially if the dataset is small.
- Number of estimators: n estimators defines the number of decision trees in the Random Forest ensemble. A higher number of estimators can enhance the model's predictive performance, but it comes at the cost of increased computational resources.

4 Results and Discussion

The Evaluation score of the various modes in given in the following Table 1 to show the experimental results. The given image shows the confusion matrix for random forest model.



Random Forest model emerged as the top-performing algorithm, showcasing the highest scores across multiple metrics. It outperformed other models, including Decision Tree, AdaBoost, and K-Nearest Neighbors (KNN), which also demonstrated good results. Decision Tree and AdaBoost benefitted from their inherent ability to capture complex relationships in the data, while KNN leveraged its effectiveness in handling patterns within proximity. Another reason for KNN getting this good results

is the fact the the oversampling performed in the dataset is done by SMOTE which uses a KNN based method to find new points.

Contrasting results were observed with MLP (Multi-Layer Perceptron) Neural Network, Logistic Regression, and Support Vector Machine (SVM). These models exhibited suboptimal performance, and several factors may contribute to their lower evaluation scores. MLP and Logistic Regression might struggle with capturing intricate non-linear relationships present in the dataset, potentially leading to reduced accuracy. SVM, on the other hand, might be sensitive to the data distribution and kernel choice, impacting its ability to discern the underlying patterns effectively.

Table 1: Performance Of Different Classifiers Using All Features

Classifier	Precision	Recall	F-measure	Accuracy
Random Forest	0.96	0.96	0.96	0.96
Adaptive Boosting	0.86	0.87	0.86	0.89
Decision Tree	0.78	0.79	0.78	0.85
K-Nearest Neighbor	0.91	0.91	0.91	0.91
Logistic Regression	0.32	0.40	0.33	0.40
Multinomial Naive Bayes	0.25	0.33	0.25	0.34
Multi-Layer Perceptron	0.45	0.46	0.46	0.50
Support Vector Machine	0.33	0.32	0.33	0.31

5 Conclusion

The Random Forest model emerged as the most effective choice for our dataset, particularly following the implementation of oversampling techniques to address data imbalance. This outcome underscores the significant impact that unbalanced data can have on model performance. Beyond this, it becomes apparent that for optimal results, the application of scaling methods is crucial for specific models. Normalizing the data proves to be especially beneficial, enhancing the performance of these models and contributing to more accurate predictions.

References

[1] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[2] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA, 2015.

[3] Tzu-Tsung Wong and Po-Yang Yeh. Reliable accuracy estimates from k-fold cross validation. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1586–1594, 2019.