

Classification of Glitches in the Gravitational Wave DataSet

Name:	Arijit Ghosh
Roll No.:	21051
Institute Name:	IISER Bhopal
Program:	BS-MS Natural Science
Stream:	Physics
Problem Release date:	August 17, 2023
Date of Submission:	November 19, 2023

1 Introduction

The era of Gravitational-Wave (GW) Astronomy started in 2015 with the detection of the GWs emitted in the coalescence of two black holes by the Advanced LIGO detectors. To date, the LIGO-Virgo-KAGRA (LVK) collaboration has reported results from three observing runs comprising 90 confident detections.[1]

GW detectors operate in extremely low noise conditions, which may be disrupted by the ground motion surrounding the detectors or earthquakes, storms or even anthropogenic sources of noise. With a higher detection rate, non-Gaussian noise transients of instrumental or environmental origin, commonly dubbed “glitches”, will become increasingly more of a concern for the LVK detectors.



Figure 1: Overview of Data Set using pairplot()

In this project, we would try to classify the glitches using ML by their features like Central frequency, peak frequency, snr(signal and noise ratio), bandwidth and duration. There are eight features, two of which are categorical. I have ignored ‘GPstime’ because that does not directly

contribute to glitch classification. There are 22 target classes. The highest target class ‘blip’ has 1587 instances, and the lowest class ‘1080Lines’ has only four instances. There are a total of 6000 instances. There is no missing value in the dataset.

2 Methods

The models used in this project are Logistic Regression, Random Forest, Support Vector Machine, K Nearest Neighbour Classifier, Decision Tree Classifier and Adaptive boosting.[4] I have also done feature selection.

Though there is no missing value, some data preprocessing is still needed. I have used One-Hot Encoding to convert the categorical feature to a numerical feature. I have only converted the ‘ifo’ feature to a numerical feature. I have removed ‘id’ and ‘GPStime’ from the dataset. After all of that, I have seven attributes in the dataset. I have also used Pipeline, so I don’t have to scale the data every time.

Feature Selection: I have done the feature selection for the best estimator: Random forest. And the performance(f-macro) has increased for that model.

Hyperparameters: These are the higher-level parameters that we set manually before starting the training, which is based on properties such as the characteristics of the data and the capacity of the algorithm to learn.[1] In sklearn, we can use exhaustive ‘GridsearchCV’, which explores all the combinations of given hyperparameters. It is efficient but time-consuming. Or we can use ‘RandomizedsearchCV’, which explore the specific number of iteration (less than the total combinations) and give the hyperparameters based on that iteration combinations only. But there is a trade-off of efficiency occurs. I have only used ‘GridSearchCV’ in this project for all the models.[3]

- For Logistic Regression, I have tuned the ‘C’ value, ‘penalty type’, ‘maximum iteration number’ and ‘class weight’.
- For Random Forest, I have tuned ‘class weight’, ‘number of decision trees’, ‘maximum depth of the decision tree’, ‘splitting criterion’ and ‘Number of sample splitting’.
- For SVM, I have tuned ‘kernel’, ‘class weight’, ‘gamma value’, and ‘probability’.
- For KNN, I have tuned the ‘number of neighbours’, ‘distance type’ and ‘weight of the neighbours’.
- For the Decision Tree, I have tuned ‘splitting criterion’, ‘max depth’, ‘minimum number of sample leaf’ and ‘class weight’.
- For AdaBoost, I have tuned ‘number of estimators’ and ‘learning rate’.
- For Feature Selection, I have tuned the ‘number of features to select’.

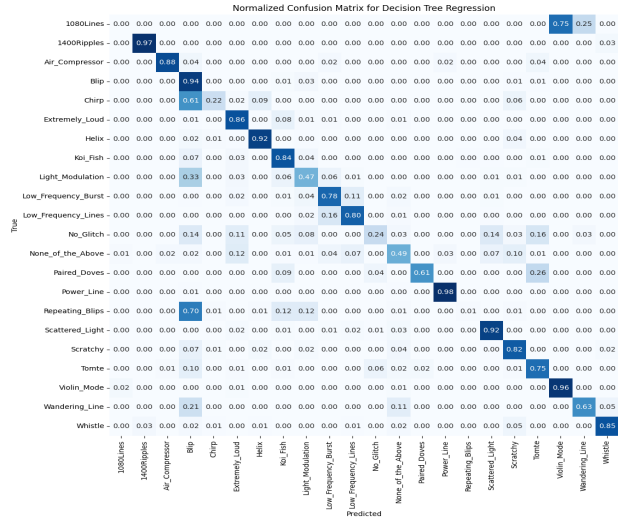
3 Experimental Setup

To evaluate the model, I have used macro-averaged precision, recall, f-measure. I have used Stratified K-fold (k=5) to split the data into training and evaluation sets and then evaluate the test set based on the test set. For this purpose, I have taken the mean value of all the evaluation scores in k folds.

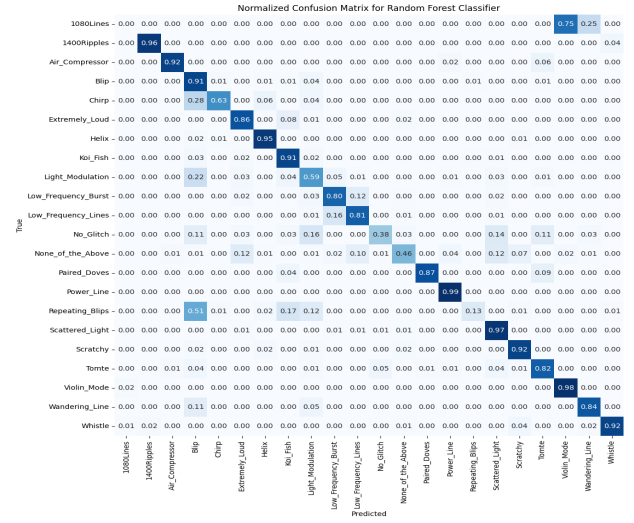
The best hyperparameters for this classification problem are,

- For Logistic Regression C=0.001, class weight=‘balanced’,
- For Random Forest class weight=‘balanced’, max depth=20, max features=‘log2’, number of estimators=200, min samples split=8
- For SVM C=10, class weight=‘balanced’, kernel=‘linear’, probability=True
- For Decision Tree criterion=‘entropy’, max depth=9, min samples split=10
- For KNN number of neighbors=4, distance type=‘Manhattan’, weights=‘distance’
- For AdaBoost base estimator= ‘Decision Tree’, criterion= ‘entropy’, number of estimator=200, learning rate=1
- For the Random Forest of Selected Features, the number of selected features is 6 out of 7, Random Forest with the best hyperparameters mentioned above.

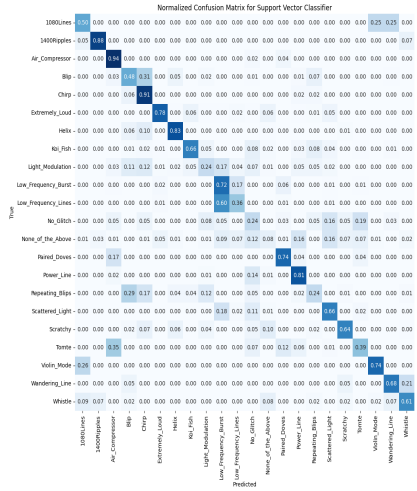
4 Results and Discussion



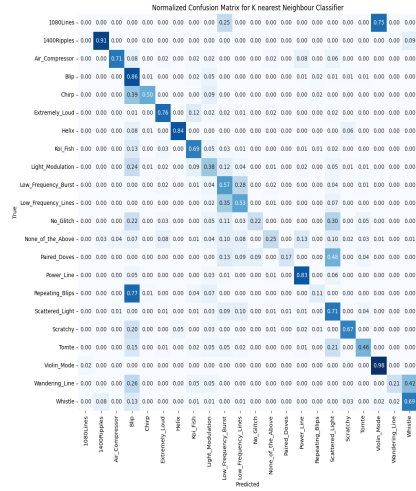
(a) Normalized Confusion Matrix for Decision Tree



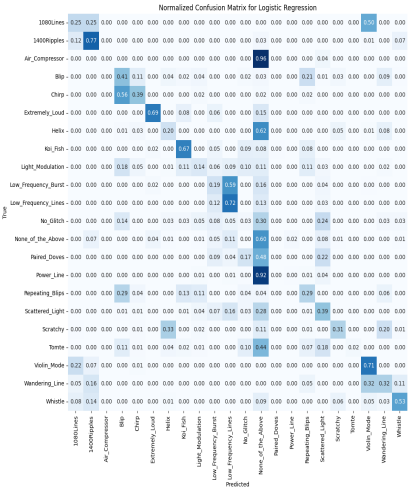
(b) Normalized Confusion Matrix for Random Forest



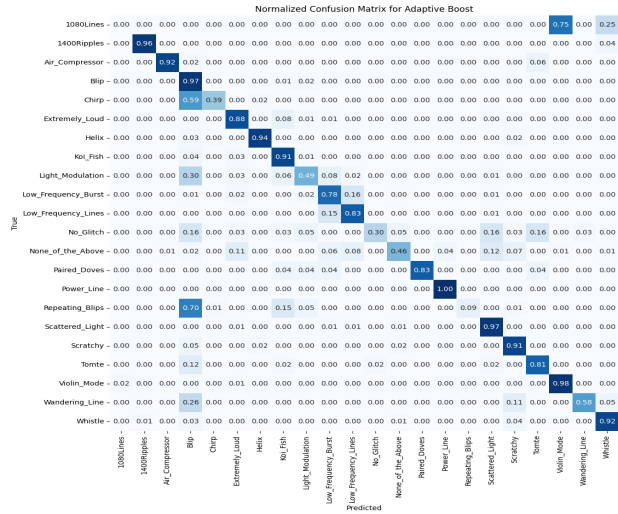
(c) Confusion Matrix for Support Vector Classification



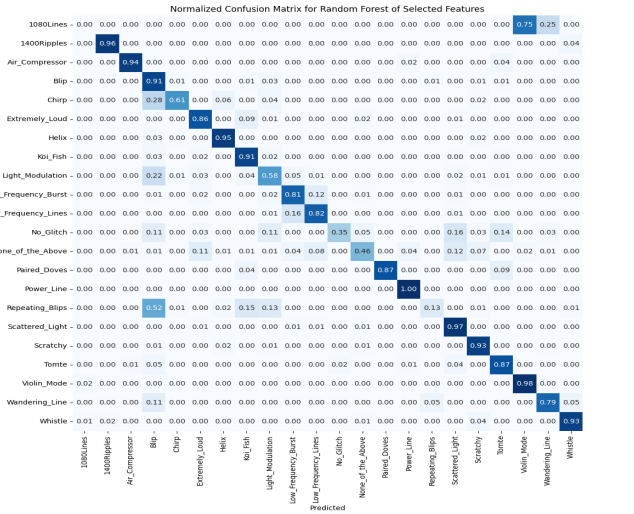
(d) Normalized Confusion Matrix for KNN



(e) Normalized Confusion Matrix for logistic Regression



(f) Normalized Confusion Matrix for Adaptive Boost



(g) Normalized Confusion Matrix for Random Forest after Feature Selection

Figure 2: Confusion Matrix for all the models

Table 1: Performance Of Different Classifiers

Classifier	Macro-avg Precision	Macro-avg Recall	Macro-avg F-measure
Logistic Regression	0.5251	0.6203	0.5247
Random Forest Classifier	0.7738	0.7536	0.7523
Support Vector Machine	0.5324	0.5935	0.5053
K-Nearest Neighbor	0.6049	0.5591	0.5707
Decision Tree Classifier	0.7182	0.6878	0.6932
Adaptive Boosting	0.8242	0.7148	0.7388
After Feature selection Random Forest	0.7963	0.7644	0.7653

From Table 1 and Figure 2, we can see that Adaptive Boost has a good precision score but a worse recall score than Random Forest. If this was a binary classification problem (like Non-Glitch or Glitch), we should have considered Adaptive Boost because without that we might have lost some information about GM wave. But this is a multi-class classification problem of glitches. So, we should consider the F-measure as the evaluation criterion. F-measure tells us how much the prediction has included false instances and excluded actual instances.

Logistic Regression and SVM miserably fail in this classification problem.

So, I have used Random Forest after feature selection with the hyperparameters specified above for the test set prediction.

5 Conclusion

From the outcomes, we can observe that:

1. Random Forest has the best macro averaged F-measure. And I have considered that for predicting the test data.
2. The low precision and recall score of all the models may be due to minority classes. During cross validation, the data set may be split in a biased way.
3. Identification of the glitches is a critical task for GM wave detection, and classification or analysis of them can tell us which component of the instrument or which natural effects can cause noise.

References

- [1] B. P. Abbott et al. (LIGO Scientific, Virgo), Phys. Rev. Lett. 116, 061102 (2016), 1602.03837.
- [2] Tanay Agarwal, Hyperparameter Optimization in Machine Learning.
- [3] <https://scikit-learn.org/stable/modules/gridsearch.html>
- [4] Tom M. Mitchell, Machine Learning