technische universität
dortmund

Machine Learning for Physicists

Jet Classification with LHC data

Riana Shaba

Professors:

Dr. Carsten Burgard

Dr. Cornelius Grunwald

July 31, 2025

Contents

1 Introduction

At CERN's most powerful particle accelerator, the Large Hadron Collider (LHC), particle physicists study proton-proton collision events by analyzing enormous amounts of data. The proton-proton bunches collide at a frequency of about 40 MHz, and having such high data rates, the processing and filtering of collision events in real-time or offline needs to be carried out as efficiently as possible, and as so it remains a challenge.
To serve this purpose, machine learning methods and deep learning algorithms are highly motivated and have so far demonstrated to be very successful in various applications in event processing such as in the trigger level which will allow the large experiments at LHC like CMS and ATLAS to preserve new physics signatures such as Higgs physics and dark matter sectors, and other usages like high level physics object classification and physics analyses [1].
In this study we focus on using machine learning techniques to perform the task of jet classification using jet substructure information. A jet is a collimated bunch of particles and physics objects that are produced from the hadronization of quarks and gluons. Jets are a very important tool used to reconstruct hadronic activity, and are one of the most complex objects to deal with in particle physics. Jet substructure tools study the inner structure of jets while identifying certain features, and distinguishing the various radiation profiles of jets from backgrounds [7]. This is interesting because physics processes have overwhelming backgrounds and some trigger strategies rely only on the energy of the jet. So by introducing jet substructure techniques in the hardware trigger, can further greatly reduce backgrounds and preserve significantly more of these types of signals in the future.
Related work [1] that has used the same dataset [10], applies a neural network with three hidden layers and has shown the performance of their classifier, and it dives deeper into the implementation of their neural network in FPGAs for trigger and DAQ. The trained neural network is translated into High-Level Synthesis (HLS) code performed by `hls4ml` package, however we are only interested in their classification task and result, in order to compare how our machine learning algorithm will perform by using the same performance metrics.

Code Link

The code written for making this project is available in Colab:
https://colab.research.google.com/drive/14VqrPImd2r7A-aFBin9RLHWewS9Nf3RO?usp=share_link

2 Case Study

Our focus is to classify whether the jet is a light quark (q), top quark (t), W boson (W), Z boson (Z) or a gluon (g) jet.

2.1 Dataset

The dataset [10] consists of 16 already selected features consisting of mass-like and shape-like observables typically used in LHC analyses, and 5 classes. The mass-like observables are sensitive to the absolute energy scale of the jet, while shape-like observables are sensitive to relative energy scales within the jet. Simulated $W^+W^-, ZZ, t\bar{t}, q\bar{q}$ and gg events, produced first at leading-order, are generated at $\sqrt{s} = 13$ TeV for comparison to LHC center of mass energy run. The jet samples are focused in a relatively narrow p_T kinematic range.
For the selection of features, high-level jet recombination algorithms and substructure tools have developed various observables to identify the origin of a jet based on the structure of its radiation pattern. As a baseline, all the jets are clustered using anti-k_T algorithm, with a distance parameter of $R = 0.8$ [1]. A summary of all the input features used in this analysis with a brief description as presented in [4], is found in the Appendix, Table 1.

2.2 Data Processing

We randomly split the data into training (60%), validation (20%) and testing (20%) datasets just as in the reference paper [1]. After some basic exploration of the features, we check and confirm that the data is balanced where each class amounts to about 20%. This is very important especially in classification tasks, because working on an imbalanced dataset can result into a misleading evaluation and performance of the model. Then we use a `t-SNE` plot in order to have a visual representation of the classes separation as shown in Figure 1. From the plot we see that the top quark, W boson and Z boson jets are relatively more easily separable compared to gluon and quark jets which are more difficult to distinguish. This is useful to draw the conclusions, as we keep in mind the complexity of the jets as physics objects.

2.3 Main Machine Learning Method

2.3.1 `LightGBM` Classifier

The main machine learning algorithm for this study is the `LightGBM` classifier [8], a gradient boosting framework which uses tree-based learning algorithm. Its advantages are relevant for

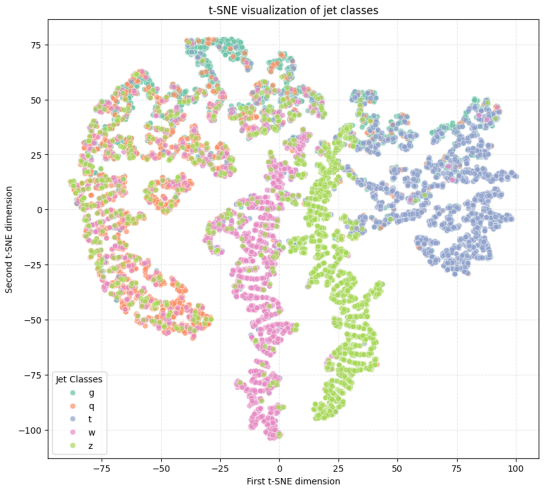
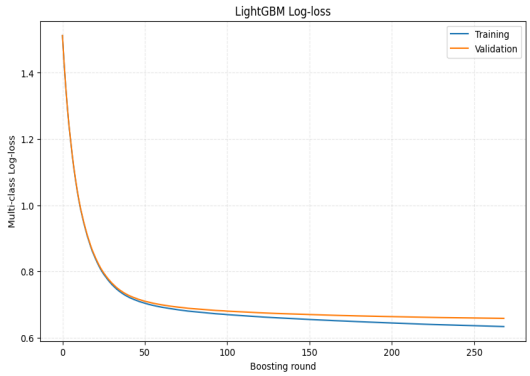


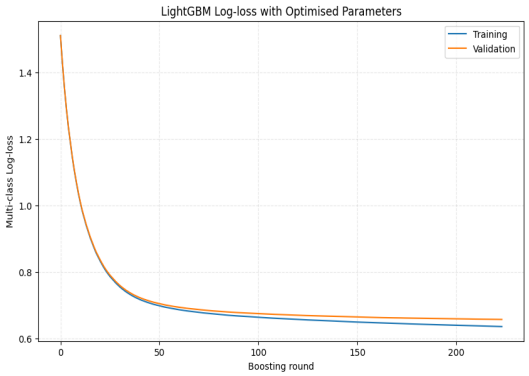
Figure 1: Class separation visualization with t-SNE plot.

this problem because it can handle large and complex datasets efficiently, it can capture non-linearities well, and has a fast training speed and low memory usage to run. LightGBM builds trees sequentially, where each new tree focuses on correcting the errors made by the previous trees. It uses the gradient of the loss function, and for our problem we choose the multi-class loss function `multi_logloss` as our metric, which measures how well the model predicts the probabilites for each class. Instead of evaluating every split, it bins continuous features into discrete histograms, which speeds up the training. It grows leaf-wise, so best leaf first, instead of level-wise. This leads to deeper trees and better accuracy but needs careful regularization to avoid overfitting [2, 9].

First we define a baseline LightGBM model with manually-set values for the parameters. We train the model on the training dataset and evaluate its performance on both the training and validation datasets. In order to monitor the training process and learning behavior of the model, we plot the log-loss curves for both datasets in order to check for any overfitting or underfitting. We also log `multi_error`, which is the multi-class classification error that tells us how often does the model gets the class label wrong, to track accuracy-related trends. To reduce overfitting which was an initial problem with our model, and to save time by preventing unnecessary training, we use early stopping with a patience of 5, which stops the training if the validation loss does not improve after 5 rounds. This is a way to asses whether the model is learning meaningfully and helps us for the next step which is the tuning of the hyperparameters. Using this methodology, we kept the results where the model doesn't show overfitting as could be seen from the performance curves for both the training and validation in Figure 2a. After the training of the initial model without the optimized hyperparameters, we evaluate the model on the validation dataset and get an accuracy of 76.42%, where the LightGBM output,



(a) LightGBM Multi-Class Log-loss before optimization of the hyperparameters.



(b) LightGBM Multi-Class Log-loss after optimization of the hyperparameters.

Figure 2: LightGBM Multi-Class Log-loss for each boosting round for both the training and validation dataset before and after optimization of the hyperparameters. We visually monitor the training process and the learning behavior of the model.

the ROC curves and the normalized Confusion matrix are found in the Appendix, Figure 6.

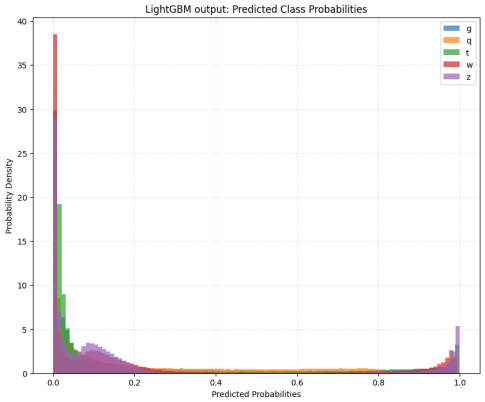
2.3.2 Hyperparameter Optimization using `Optuna`

To optimize the hyperparameters we use `Optuna`, which is a hyperparameter tuner for LightGBM that efficiently explores the parameter space using techniques such as the Tree-structured Parzen Estimator (TPE), while optimizing the hyperparameters in a stepwise manner [5]. It automates the search for finding the optimal values to improve performance, and in our case, to minimize the multiclass log-loss function of the LightGBM model [6]. To ensure robust and reliable evaluation of each hyperparameter, we integrate a 5-fold Stratified K-Fold cross-validation within Optuna's objective function. Despite the dataset being already balanced across the five jet classes, random data splits can introduce imbalances at the fold level. Since Stratified K-Fold preserves the original class proportions within each training and validation fold, it guarantees that each fold maintains the class proportions similar to the dataset. This way it avoids any bias in evaluation metrics, reduces variance and we have a more stable and fair model evaluation during the tuning process. In Figure 2 we show the plots of the performance of both the training and validation datasets during the training, where the model with the best parameters after optimization is shown in Figure 2b. After training the model using the optimal parameters obtained from the tuning process, we evaluate its performance on the validation dataset, achieving an accuracy of 76.45%. The LightGBM output, along with the ROC curves and the normalized confusion matrix, are presented in Appendix Figure 7. Finally, we evaluate the model on the test dataset to assess its generalization performance, where we achieve an accuracy of 76.49%. The results of the evaluation are given in Figure 3.

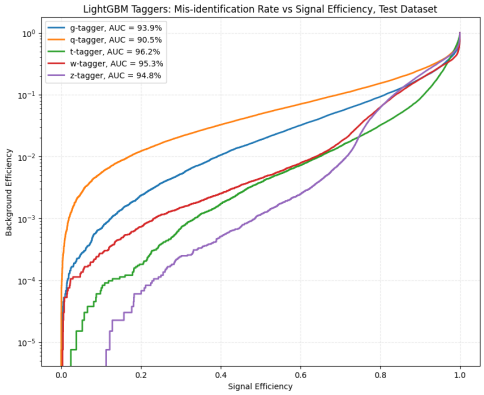
The output histograms show the predicted probabilities for each jet class, and how confident the model is at predicting them. What is noticeable is that the model is very confident at not predicting the W-jet class , as it peaks the highest with a probability of zero. This indicates that W-jets have similar features to other classes making them harder to distinguish. In the same region we have the top-jets, and gluon-jets. A general problem when considering the jet substructure, is that quark/gluon-initiated jets look very similar to W-initiated jets which are reconstructed as large-radius jets. This can also be seen in the Confusion Matrix where the model is mainly confusing the W-jets with q-jets.
On the other hand, the Z-jet class has a relatively higher peak compared to the other classes at a probability of 1, meaning that the model can classify this class with confidence, so the Z-jets must have a feature that makes them separable compared to the other classes. In the middle we see dominantly the quark-jets that are predicted with a moderate confidence.
All this hints at possibly high correlations between the features and redundant information for the model to make any significant distinction. The ROC curves are used in the high-energy physics convention with flipped axes with a logarithmic scale on the y-axis to emphasize the background rejection. Ideally we would have the curves in the bottom right corner. These curves show how well the model can distinguish a certain jet from all the other jet types. In contrast to what the predicted probabilities and confusion matrix show, the model is able to distinguish the W jets very well from the rest, but it struggles when it has to choose between all of the classes. This means it has learned to classify W-jets from background very well. For gluons we have an overall good performance, meanwhile for quarks we have the lowest performance because distinguishing quarks and gluon jets is a known problem due to QCD. For top quarks the model performs the best which is expected because of their high mass and high multiplicity that makes them easier to identify. Z boson gets confused with quarks like the W boson, but the model can separate them from each other well.
The best performance is for the top quark t-jet with 83% as can also be seen from the curve it represents. We also achieve about 1% higher results considering these metrics compared to the results from the reference paper shown in the Appendix, Figure 5.

As a side task, we explored additional Boosted Decision Trees algorithms to check how they would perform with our dataset, such as `XGBoost` and `CatBoost`[1] classifiers. XGBoost [11] grows decision trees in a level-wise manner, in comparison to LightGBM which uses leaf-wise growth. It offers more control on complex datasets and it tends to generalize better. The objective function used for this model is `multi:softprob` which predicts the class probabilities and the multiclass log-loss `mlogloss` as the loss function. We use early stopping with a pa-

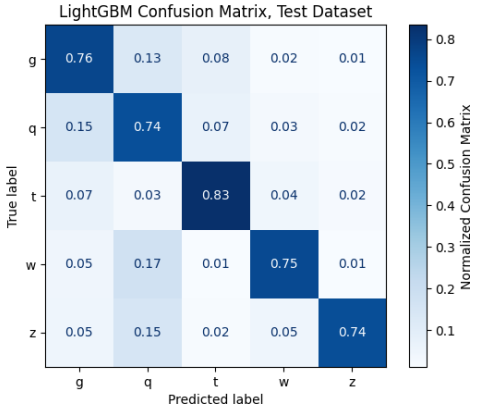
[1]CatBoost Classifier part of this work was implemented by my project teammate.



LightGBM output histograms



Normalized Confusion Matrix



ROC curves

Figure 3: LightGBM output and metrics with the optimized hyperparamters evaluated on the test dataset, with an accuracy of 76.49%.

tience of 5, and after evaluating the initial model, without hyperparameter optimization, on the validation dataset we get an accuracy of 76.64%. Optuna is used for the tuning, with a 5-fold Stratified K-fold cross validation. The accuracy after evaluating on the validation dataset is 76.66%, and the results of the evaluation on the testing dataset are found in Appendix, Figure 8 where the accuracy reaches 76.79%. CatBoost [3], which is another gradient boosting algorithm similar to XGBoost and LightGBM, that grows symmetric trees sequentially to minimize the loss function, and with an ordered boosting to prevent overfitting. The results on the test dataset, with an accuracy of 76.53% are given in Appendix, Figure 9.

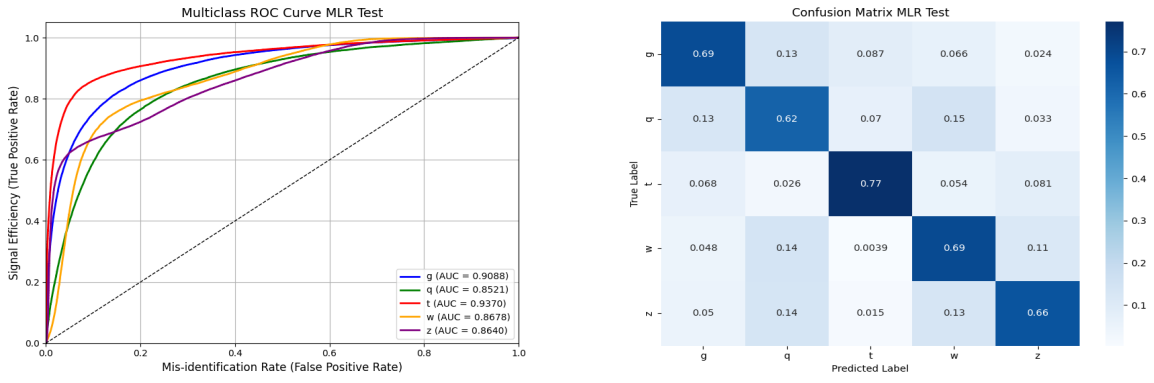
Following our results from the LightGBM classifier, especially the output probabilities and the confusion matrix which showed where the model was struggling most, we think that even though the dataset has already selected features, there are some features that very similar to

each other giving redundant information which does not help the model to learn and distinguish well between certain classes. For this purpose we plotted the Correlation Matrix and the Feature Importance for the final optimized model which is shown in Appendix, Figure 10. Another feature selection and training the model, and evaluating it on validation dataset gave more or less the same results for our metrics, and a much lower accuracy of 76.10%. We did not dive much deeper into this because we think it is part of the complexity and the capacity of what we can achieve with this dataset.

2.4 Alternative Machine Learning Method

As an alternative method that is much simpler compared to boosted decision trees, we use Multinomial Logistic Regression. It is a generalization of logistic regression to multiclass problems. The model estimates the probability that a given input belongs to each class by applying the softmax function to a linear combination of the input features. It is a simple method, fast and interpretable, but it assumes linear decision boundaries between the classes, so it is unable to capture the nonlinearities and complex features that are present in this dataset and jet classification problem.

Its performance was much lower compared to the previous, more advanced and sophisticated methods. The results from the evaluation on the validation dataset give an accuracy of 68.90% and are given in Appendix, Figure 11. The evaluation on the testing dataset gives an accuracy of 68.83% and is shown in Figure 4. These results show how far we can go using a linear



(a) ROC curves

(b) Normalized Confusion Matrix

Figure 4: Multinomial Logistic Regression metrics for the model evaluated evaluated on testing dataset, with an accuracy of 68.83%.

classifier for approaching complex problems, and that more sophisticated models will give a much better performance.

3 Conclusions

In this project we studied a multiclass classification problem of jets based on the jets substructure. The motivation was to see if a boosted decision tree algorithm such as LightGBM Classifier would perform better than a Neural Network with three hidden layers that we looked as a reference, and compare their metrics results with ours.

Our main machine learning method LightGBM gave us an accuracy of 76.49% on the test dataset with AUC percentages of about 1% better than the reference material. Because the difference was not very significant, we explored additional powerful models such as XGBoost and CatBoost, that seemed to perform comparatively better, like XGBoost with a final accuracy of 76.79% on the test dataset.

Then our alternative method, the Multinomial Logistic Regression showed exactly how far we can go with a linear model when it comes to multi-classification tasks in such complex cases as with particle physics objects like jets. They performed much lower compared to the boosted decision tree models, giving an accuracy on the test dataset of 68.83%.

In the end, we have slightly better results compared to our target, but they are not so significant. Considering the general problems and difficulties when dealing with jets, their class separation, and the complexity of the dataset itself, we conclude that we have reached the capacity of the results we can get from this dataset, with all its features, and that there are possibly better models, like XGBoost in our study, that could perform better than LightGBM.

A Appendix

A.1 Reference Paper Results for Comparison

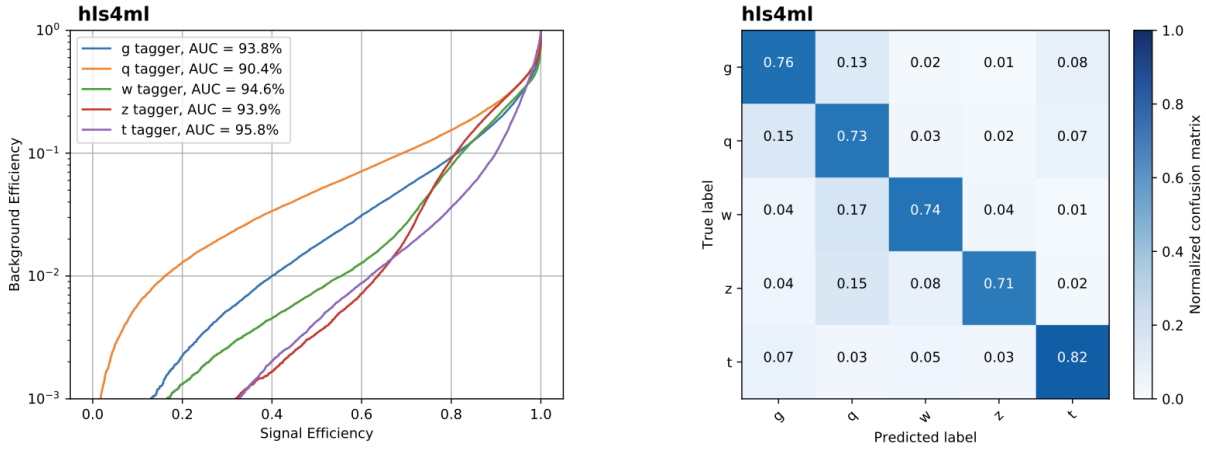


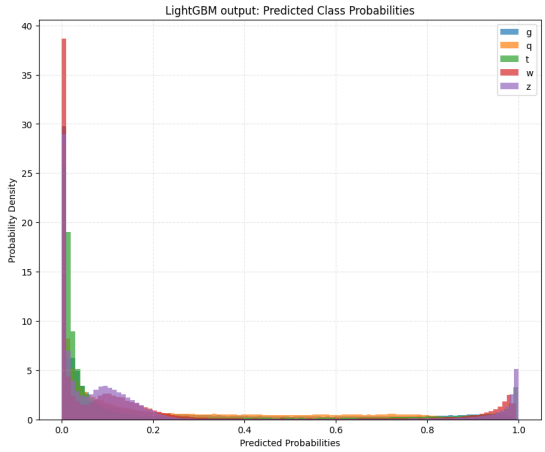
Figure 5: ROC curves and the Confusion Matrix using a Neural Network with three hidden layers. Taken from Ref. [1].

A.2 Dataset

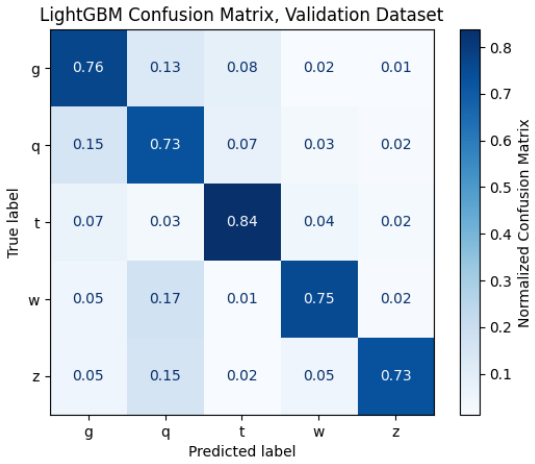
Features	Description
m_{mMDT}	Groomed jet mass using modified mass drop, a jet grooming method to remove soft and wide-angle radiation from the jet cone.
$n_2^{\ \beta=1,2}$	Energy correlation ratios that identify 2-prong jet substructure from W/Z decays using only information on the energies and angles of particles within a jet. β is the angular exponent that determines the sensitivity to collinear vs wide-angle radiation.
$m_2^{\ \beta=1,2}$	Groomed jet mass with an angular exponent β. Sensitive to how jet mass behaves under soft/collinear radiation suppression.
$c_1^{\ \beta=0,1,2}$	2-point energy correlation functions normalized to emphasize differences in jet radiation patterns. β is the angular exponent, where for $\beta = 0$ it is proportional to jet p_T. $\beta = 1, 2$ increase sensitivity to angular structure, useful for *q/g* discrimination.
$c_2^{\ \beta=1,2}$	Ratio of 3-point to 2-point energy correlation functions, useful for probing multi-prong substructure and identifying *W/Z* bosons. β is the angular exponent.
$d_2^{\ \beta=0,1,2}$	3-point energy correlation ratio that probes 2-prong substructure using higher-order energy correlations. β is the angular exponent. It is used to distinguish boosted decays from QCD jets.
$d_2^{\ (\alpha,\beta)=(1,1),(1,2)}$	Generalized d_2 that uses separate angular exponents: α for energy and β for angles, allowing finer control over jet shape sensitivity.
$\sum z \log z$	Sum over the jet constituents. Lower for narrow, quark-like jets; higher for broader, gluon-like jets.
multiplicity	Number of constituents in the jet, powerful for discriminating q/g and W/Z from QCD jets.

Table 1: Features in the dataset, where the only the groomed jet mass is a mass-like observable, while the rest are shape-like observables. From Ref. [1, 4].

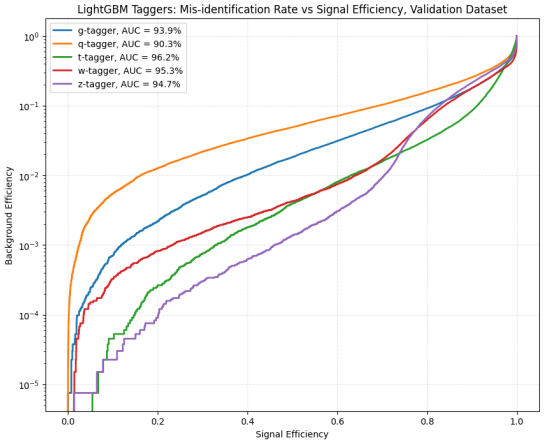
A.3 `LightGBM` Classifier Results



LightGBM output histograms

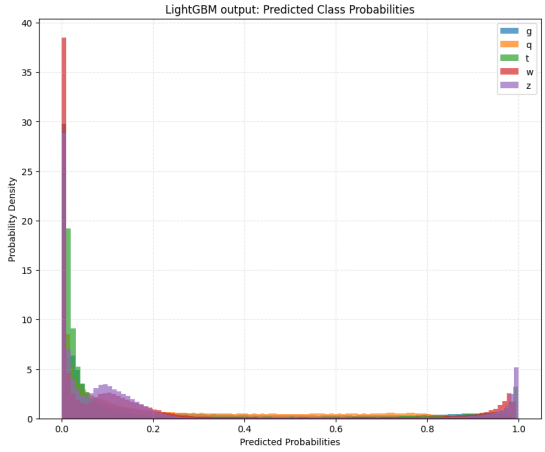


Nomalized Confusion Matrix

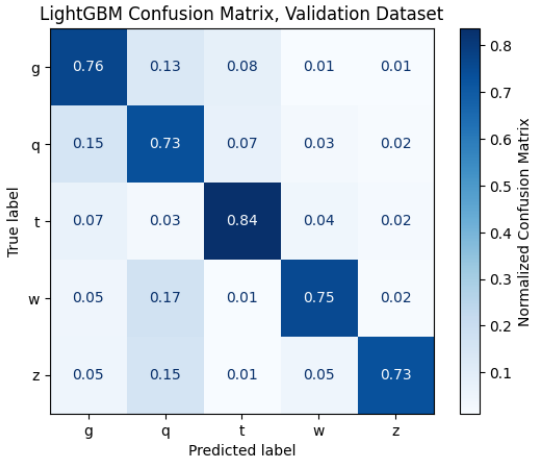


ROC curves

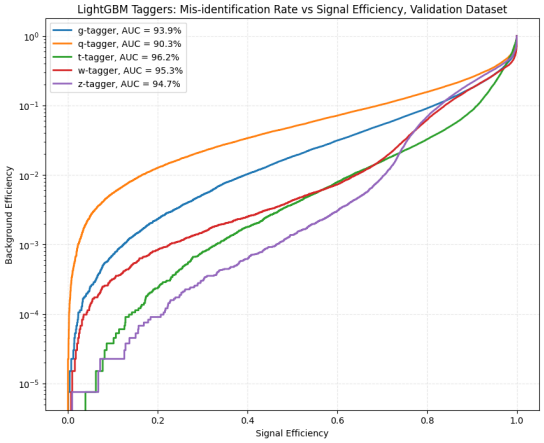
Figure 6: LightGBM output and metrics for the initial model evaluated on validation dataset, with an accuracy of 76.42%.



LightGBM output histograms



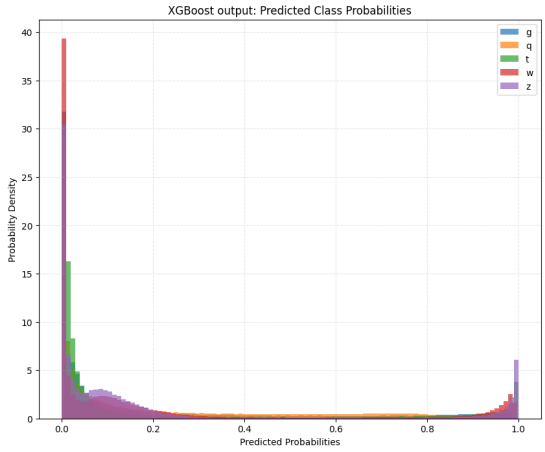
Normalized Confusion Matrix



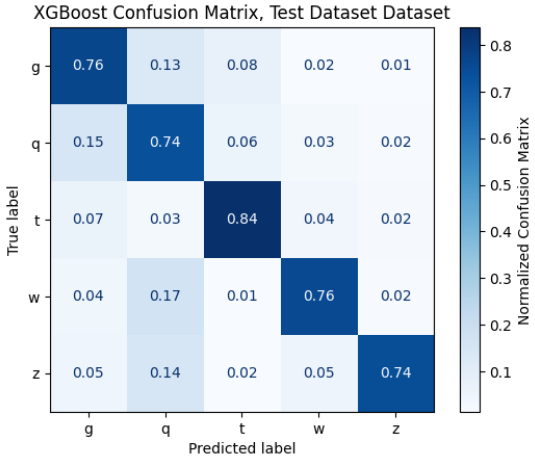
ROC curves

Figure 7: LightGBM output and metrics with the optimized hyperparamters model evaluated on the validation dataset, with an accuracy of 76.45%.

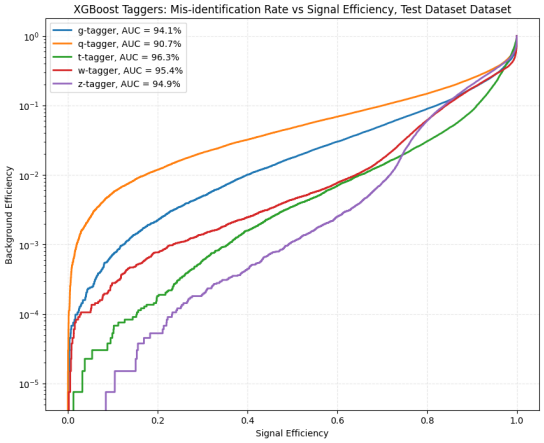
A.4 `XGBoost` Classifier Results



LightGBM output histograms



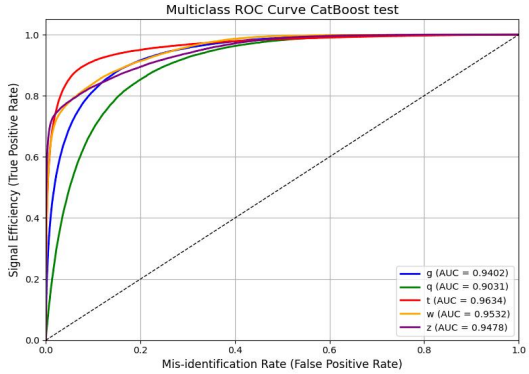
Normalized Confusion Matrix



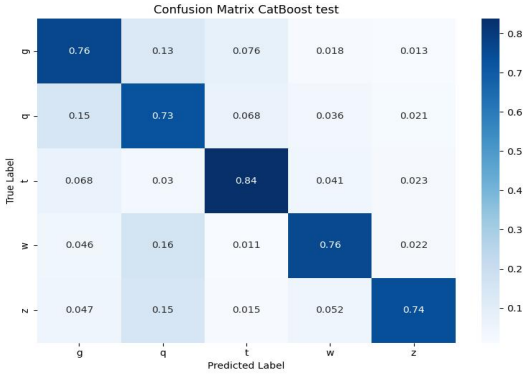
ROC curves

Figure 8: XGBoost output and metrics with the optimized hyperparamters model evaluated on the test dataset, with an accuracy of 76.79%.

A.5 `CatBoost` Classifier Results



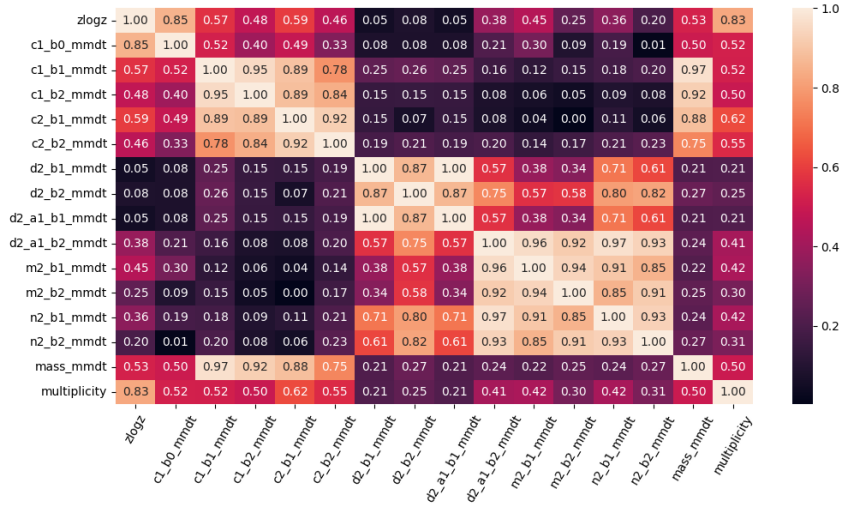
(a) ROC curves



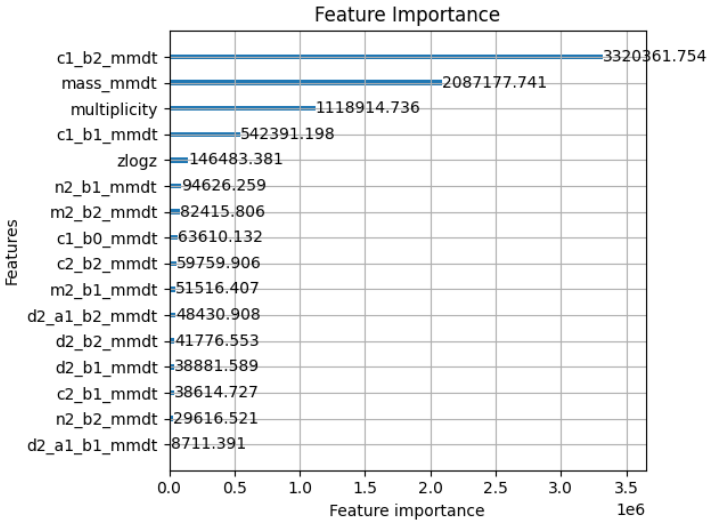
(b) Normalized Confusion Matrix

Figure 9: CatBoost metrics for the model evaluated on testing dataset, with an accuracy of 76.53%.

A.6 Correlation Matrix and Feature importance for LightGBM



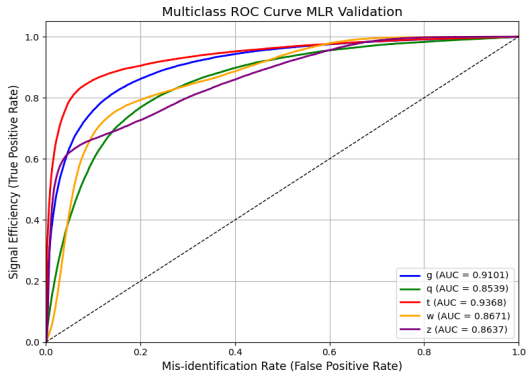
Correlation Matrix



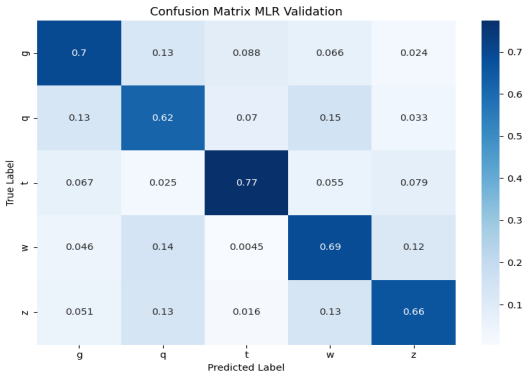
ROC curves

Figure 10: Feature importance plot for LightGBM classifier.

A.7 Multinomial Logistic Regression



(a) ROC curves



(b) Normalized Confusion Matrix

Figure 11: Multinomial Logistic Regression metrics for the model evaluated evaluated on validation dataset, with an accuracy of 68.90%.

References

[1] J. Duarte et al. "Fast inference of deep neural networks in FPGAs for particle physics". In: *Journal of Instrumentation* 13.07 (2018), P07027. DOI: 10.1088/1748-0221/13/07/P07027.

[2] Prashant Banerjee. *LightGBM Classifier in Python*. n.d. URL: https://www.kaggle.com/code/prashant111/lightgbm-classifier-in-python.

[3] CatBoost Developers. *CatBoost Documentation*. 2025. URL: https://catboost.ai/docs/en/.

[4] E. Coleman et al. "The importance of calorimetry for highly-boosted jet substructure". In: *Journal of Instrumentation* 13.01 (Jan. 2018), T01003. DOI: 10.1088/1748-0221/13/01/T01003. URL: https://dx.doi.org/10.1088/1748-0221/13/01/T01003.

[5] Optuna Developers. *optuna.integration.lightgbm.LightGBMTuner — Optuna 2.0.0 documentation*. https://optuna.readthedocs.io/en/v2.0.0/reference/generated/optuna.integration.lightgbm.LightGBMTuner.html. 2020.

[6] Optuna Developers. *optuna.study.Study — Optuna Stable Documentation*. 2025. URL: https://optuna.readthedocs.io/en/stable/reference/generated/optuna.study.Study.html.

[7] Roman Kogler et al. "Jet substructure at the Large Hadron Collider". In: *Reviews of Modern Physics* 91.4 (Dec. 2019). ISSN: 1539-0756. DOI: 10.1103/revmodphys.91.045003. URL: http://dx.doi.org/10.1103/RevModPhys.91.045003.

[8] LightGBM Developers. *LGBMClassifier — LightGBM Documentation*. 2024. URL: https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html.

[9] Microsoft. *LightGBM Parameters*. https://lightgbm.readthedocs.io/en/latest/Parameters.html.

[10] Maurizio Pierini et al. *HLS4ML LHC Jet dataset (150 particles)*. Zenodo. Dataset. 2020. DOI: 10.5281/zenodo.3602260. URL: https://doi.org/10.5281/zenodo.3602260.

[11] XGBoost Developers. *Introduction to Boosted Trees*. Accessed: 2025-07-28. 2024. URL: https://xgboost.readthedocs.io/en/stable/tutorials/model.html.