CS4220: Pathogen Detection Project

Team snpsnp
A Janhavi A0205165M
Amelia Regina Sutikna A0200711A
Avril Lim A0204887U

1. Introduction

Using metagenomics data to detect pathogens precisely and accurately is important for diagnosis and treatment of severe infection. This paper addresses the multi-class supervised learning classification task of detecting pathogens from a set of 6-mers. Three machine learning algorithms were tested- eXtreme gradient boosting (XGBoost), light gradient boosting (LightGBM) and support vector machines (SVM). We found that tree-based gradient-boosted models LightGBM and XGBoost gave the best results on validation sets, with an average precision of 0.953 and 0.891 respectively.

2. Methodology

2.1 Investigation of Models

eXtreme gradient boosting (XGBoost) is an ensemble learning tree-based method which has been shown to achieve state of the art performance on a multitude of supervised learning classification tasks across domains (1). In particular, recent studies have shown that it achieves superior performance over other techniques in classification tasks involving biomedical and multi-omics data (2,3). Gradient boosted tree-based models build consecutive tree models, minimising the loss function with respect to the prior error each time. XGBoost employs both L1 and L2 regularisation to leaf values as an additional penalty to the loss function - helping to guard against overfitting and increase generalizability of the model. LightGBM is similar to XGBoost in how it utilises gradient boosted decision trees to perform classification tasks. Unlike XGBoost, which uses pre-sort-based algorithms for decision tree learning, LightGBM uses a histogram-based algorithm that buckets continuous features into discrete bins to optimise training speeds (4,5). Additionally, it also carries out a leaf-wise or best-first growth of decision trees, which means that the leaf with the greatest delta loss is grown, unlike XGBoost which grows level-wise. This is done to improve both speed, accuracy, and increase loss reduction (6). If LightGBM is able to perform as well as XGBoost, its speed would be a major advantage for this problem.

To have a comparison against gradient boosting methods, we also delved into Support Vector Machines (SVMs). Similar to the baseline model used to benchmark performance (logistic regression), SVMs also do not natively support classification of more than 2 classes. However, both methods can use meta-strategies such as the One-vs-One strategy or the One-vs-Rest strategy (7) for multi-class problems. We used Scikit-learn's SVM package (8), which uses One-vs-One as the strategy for all multi-class problems.

2.2 Data Exploration

Preliminary analysis of the training data showed a significant class imbalance with a much greater proportion of decoy reads. Among the 10 classes of pathogens, there was also a greater proportion of certain species of pathogens like *B. pseudomallei, P. aeruginosa, M. ulcerans* and *M. tuberculosis* (Fig S1). Due to hardware limitations, our team had to select a subset of the data for training. To address the class imbalance, both stratified and uniform sampling were tested in downstream analysis.

2.3 Feature Selection

The ability of the XGBoost algorithm to rank features by importance in prediction has resulted in its successful application on genomic data to determine predictive features of various bacteria (9,10). Thus, our team investigated its use in reduction of the high dimensional feature space of 2080 for our application. In the training set for feature selection, both stratified and uniform samples were tested. On the stratified sample, an artificially low number of decoy reads was used with the objective of learning the most predictive features of pathogens - since they would be harder to detect at low abundance in real datasets. Features were ranked by importance in terms of information gain in the decision tree.

To identify the important features, we performed empirical tests on training data, which revealed that at least 25% of the feature space was needed to achieve comparable performance to training on the full feature space, in terms of test accuracy on the training set. A set comparison was also performed between the top 25% of features identified from stratified and uniform samples. An intersection of only about half was observed. In comparing the performance of the reduced feature space of both samples, that of the stratified sample achieved slightly better performance in terms of test accuracy on the training data. Thus, this set of 512 features was used for model training.

2.4 Training Data Selection

The reduction in feature space by 75% allowed for a corresponding increase in the number of reads. Even so, we had to determine the ideal ratio of decoy to pathogen reads to include in the training set. Following feature selection, we saw a significant improvement in test accuracy on the training set (Fig S2), with overall more balanced performance in prediction across all classes. From umap plots of species clusters from the reduced feature space (Fig S3), we observed that pathogen species clusters are more concentrated around the centroid as compared to the decoy cluster, which has points that are more widely dispersed, with some relatively small clusters that were close to pathogens. This indicates greater variation among decoy reads and suggested that a greater proportion of training data might need to be apportioned to decoy samples in order for the model to learn to accurately classify decoy reads. Further, in sequencing data from patients it is likely that pathogen

abundance would be much lower, and decoy reads would make up an overwhelming majority. Thus, our team tested the performance of our models on a training set made up of 50%, 75% and 90% decoy reads respectively.

2.5 Hyperparameter tuning

For the XGBoost model, hyperparameter tuning was carried out on a predefined set of parameters using GridSearchCV. Notably, a learning rate of 0.10 and maximum tree depth of 4 were found to be ideal. To prevent overfitting, a plot of multiclass log loss scores on train and test data with each iteration was also monitored. From these plots, it was determined that 200 estimators were sufficient to achieve minimization of loss without widening the gap between train and test accuracy. Plots before and after hyperparameter tuning (Fig S4) also illustrate a decrease in the difference between mlogloss scores of train and test data- indicating reduced tendency of the model to overfit.

For the LightGBM model, we first trained using a set of parameters that were commonly used. This preliminary model already had sufficiently decent training accuracies and precision scores on the test data but hyperparameter tuning was still conducted to see if further improvements were possible. It was run using sklearn's GridSearchCV with 3-fold cross-validation. Some noteworthy parameters decided by this optimization include the maximum depth, in which 5 was found to be the ideal value, as well as lambda l1 and lambda l2, which were set to 1.0 and 0.25, respectively.

For the SVM model, we first ran a preliminary test and trained 4 models using the linear, polynomial, RBF and Sigmoid kernel functions. The results on the training data showed that the polynomial kernel had the best performance in terms of test accuracy on the training data. Then, we selected the degree of the polynomial and the regularisation parameter, C using GridSearchCV. Notably, values 10 and 2 for C and degree were found to be ideal. The final tuned parameters for all three models are described in Table S1.

3. Results

For the validation set, we tried several possible probability thresholds for all 3 models. The optimum threshold values were chosen such that the performance on the validation set is maximised. Then, we used the same threshold values for prediction in the final validation set. All models outperformed a baseline logistic regression model. The breakdown precision per patient is provided in Figure 1.

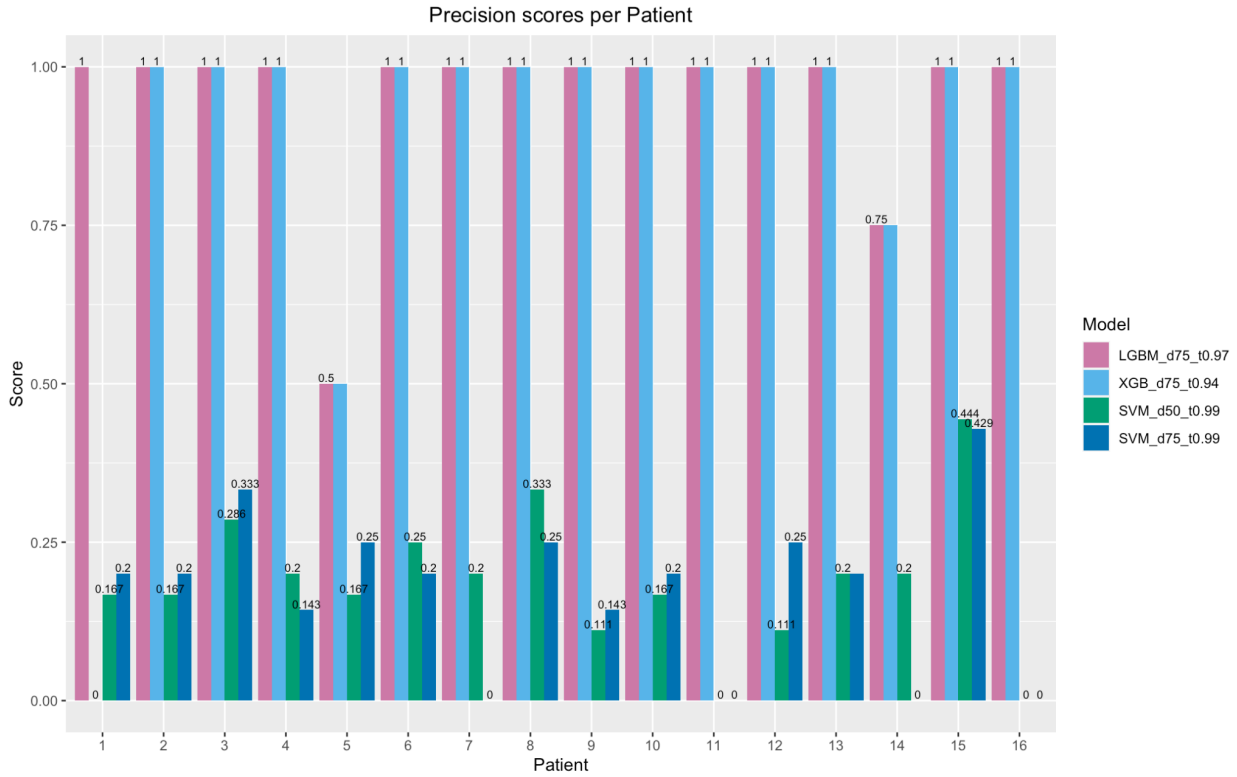

Figure 1: Patient breakdowns of precision scores from each model for initial validation set.
Plot generated using ggplot2 (11) and reshape2 (12) packages on R version 4.0.3.
Legend (from left to right): Pink - LightGBM model with 75% decoy and probability threshold of 0.97,
Light Blue - XGBoost model with 75% decoy and probability threshold of 0.94,
Green - SVM model with 50% decoy and probability threshold of 0.99,
Dark Blue - SVM model with 70% decoy and probability threshold of 0.99

3.1 Significance of training set

When trained on a dataset of 50% decoy and 50% pathogen reads, we noticed that all 3 models were tuned to predict pathogens even in the absence of any pathogens (i.e. patient 11 & 16) on the validation sets. This was unsurprising, as decoy reads are expected to make up a large majority in real patient datasets, compared to the training data. This issue was resolved when models were trained on datasets containing 75% and 90% decoy reads. Additionally, we found that the training set of 75% decoy reads gave optimal performance on the held-out validation set in terms of precision. From Table 1, XGBoost and Light GBM showed significant improvements in precision of predictions on the validation set after increasing the proportion of decoy reads in the training set from 50% to 75%.

Table 1: Average precision of models on validation set trained with different proportions of decoy reads

Training Data	Average Precision on Validation Set			
	XGBoost	LightGBM	SVM	Baseline
50% decoy	0.547	0.572	0.188	0.151
75% decoy	0.891	0.953	0.175	

3.2 Poor performance of SVM

Surprisingly, SVM showed poor performance on the validation set, even when a higher percentage of decoy reads were introduced into the training set. This was despite good performance of SVM on the test set during the train-test split, showing an accuracy of 0.945 on the 50% decoy training set and 0.935 on the 75% decoy training set. The difference in performance could be due to 2 reasons. Firstly, while the initial testing relies on row-wise prediction, the validation test does set-wise prediction. In the latter, the effect of misclassifications is much more inflated. Secondly, SVM's performance on imbalance datasets is known to be poor (13). Thus, it is possible that training on an even more imbalanced dataset (75% decoy) led to even poorer performance instead of improving performance on the validation set.

3.3 LightGBM vs. XGBoost

On the initial validation set, LightGBM performed better than XGBoost (Table 1). From the patient breakdown in Figure 1, we see that the XGBoost model was unable to predict for patient 1, which LightGBM could. This suggests that LightGBM could be more sensitive in predicting pathogens. Moreover, the major advantage of LightGBM lies in its training speed, which is significantly faster than XGBoost by up to ten times for our training dataset, while also providing the highest precision. However, despite this difference, both models predicted the same pathogens for every patient in the second validation set.

4. Conclusion: Lessons learnt & Future work

A major takeaway our team gained from this project was the importance of the composition of training data. Hardware restrictions like RAM made it necessary for us to find a way to utilise the large training dataset as efficiently as possible. Though we had initially assumed that a higher proportion of pathogen reads was most important in developing a model sensitive enough to pick up pathogens at very low abundance in validation datasets, the significant improvement observed after training on a higher proportion of decoy data proved the importance of training data being close to real datasets in its class distribution. The high precision achieved by both tree-based methods also suggest that they are suitable for metagenomics classification tasks, and effective in reducing the high-dimensional feature space from sequencing data to the most important subset.

A suggestion for future work would be to consider an ensemble method using the predictions of both the XGBoost model and the LightGBM model. This would require tests on several other independent validation sets to get an idea of whether this would actually improve prediction compared to using either model alone. Instead of using the threshold of training accuracy with the full feature space as a cut-off to determine the proportion of predictive features, different proportions of the reduced feature space could be tested and evaluated to determine the most suitable one. This could allow us to further reduce the dimensionality of the input data and train on more samples.

Additionally, the hardware and time constraints meant that we were only able to perform minimal hyperparameter tuning as we were unable to tune many combinations of parameters at once. Thus, a suggestion for future work would be to try a wider range of values for hyperparameter tuning given access to better hardware, or use RandomizedSearchCV to test a wider range of values for our parameters.

References

1. Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining [Internet]. New York, NY, USA: Association for Computing Machinery; 2016 [cited 2022 Apr 15]. p. 785–94. (KDD '16). Available from: https://doi.org/10.1145/2939672.2939785
2. Ma B, Meng F, Yan G, Yan H, Chai B, Song F. Diagnostic classification of cancers using extreme gradient boosting algorithm and multi-omics data. Comput Biol Med. 2020 Jun 1;121:103761.
3. Dimitrakopoulos GN, Vrahatis AG, Plagianakos V, Sgarbas K. Pathway analysis using XGBoost classification in Biomedical Data. In: Proceedings of the 10th Hellenic Conference on Artificial Intelligence [Internet]. New York, NY, USA: Association for Computing Machinery; 2018 [cited 2022 Apr 15]. p. 1–6. (SETN '18). Available from: https://doi.org/10.1145/3200947.3201029
4. Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, et al. LightGBM: a highly efficient gradient boosting decision tree. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. Red Hook, NY, USA: Curran Associates Inc.; 2017. p. 3149–57. (NIPS'17).
5. Light Gradient Boosting Machine [Internet]. Microsoft; 2022 [cited 2022 Apr 18]. Available from: https://github.com/microsoft/LightGBM
6. Shi H. Best-first Decision Tree Learning [Internet] [Thesis]. The University of Waikato; 2007 [cited 2022 Apr 18]. Available from: https://researchcommons.waikato.ac.nz/handle/10289/2317
7. Brownlee J. 4 Types of Classification Tasks in Machine Learning [Internet]. Machine Learning Mastery. 2020 [cited 2022 Apr 18]. Available from: https://machinelearningmastery.com/types-of-classification-in-machine-learning/
8. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. J Mach Learn Res. 2011;12(85):2825–30.
9. Nguyen M, Long SW, McDermott PF, Olsen RJ, Olson R, Stevens RL, et al. Using Machine Learning To Predict Antimicrobial MICs and Associated Genomic Features for Nontyphoidal Salmonella. J Clin Microbiol. 57(2):e01260-18.
10. Ding C, Han H, Li Q, Yang X, Liu T. iT3SE-PX: Identification of Bacterial Type III Secreted Effectors Using PSSM Profiles and XGBoost Feature Selection. Comput Math Methods Med. 2021 Jan 6;2021:e6690299.
11. Wickham H, Chang W, Henry L, Pedersen TL, Takahashi K, Wilke C, et al. ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics [Internet]. 2021 [cited 2022 Apr 18]. Available from: https://CRAN.R-project.org/package=ggplot2
12. Wickham H. reshape2: Flexibly Reshape Data: A Reboot of the Reshape Package [Internet]. 2020 [cited 2022 Apr 18]. Available from: https://CRAN.R-project.org/package=reshape2
13. Khoong WH. When Do Support Vector Machines Fail? [Internet]. Medium. 2021 [cited 2022 Apr 18]. Available from: https://towardsdatascience.com/when-do-support-vector-machines-fail-3f23295ebef2

Supplementary Figures

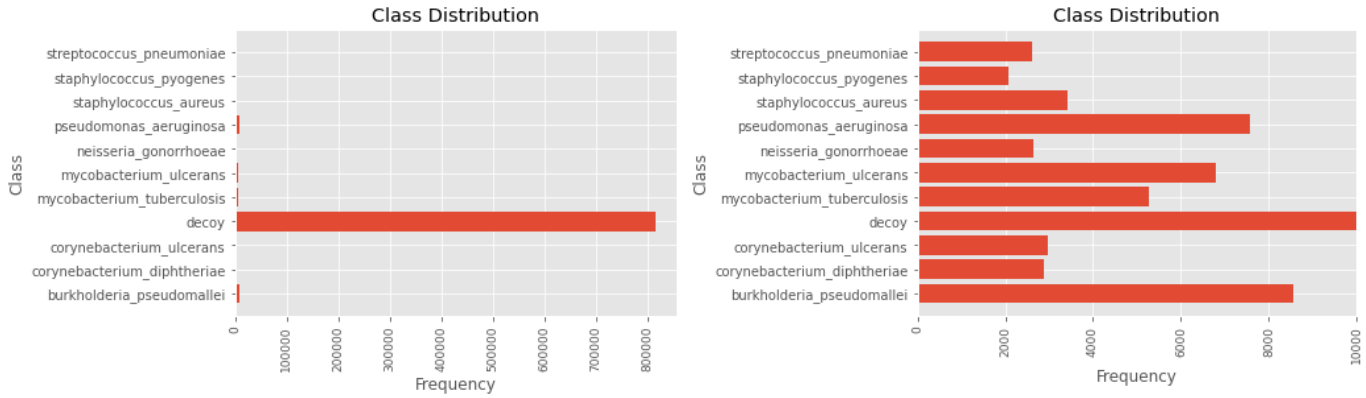

Figure S1: Histogram plots of class distribution in training data by frequency count for n in range (0, 900,000) (left) and n in range (0, 10,000) (right)

Train Accuracy: 0.9783440883736192
Test Accuracy: 0.8832020997375328
Classification Report:

	precision	recall	f1-score	support
0	0.98	0.97	0.97	415
1	0.86	0.86	0.86	416
2	0.83	0.85	0.84	416
3	0.91	0.93	0.92	416
4	0.76	0.76	0.76	415
5	0.75	0.74	0.74	415
6	0.99	0.97	0.98	416
7	0.94	0.93	0.94	415
8	0.92	0.94	0.93	416
9	0.92	0.89	0.90	416
10	0.87	0.88	0.87	416
accuracy			0.88	4572
macro avg	0.88	0.88	0.88	4572
weighted avg	0.88	0.88	0.88	4572

Train Accuracy: 0.9779472914456139
Test Accuracy: 0.928596008258775
Classification Report:

	precision	recall	f1-score	support
0	0.98	0.98	0.98	1712
1	0.86	0.82	0.84	577
2	0.88	0.82	0.85	592
3	0.96	0.98	0.97	8486
4	0.79	0.75	0.77	1053
5	0.80	0.84	0.82	1357
6	0.96	0.97	0.96	527
7	0.96	0.96	0.96	1514
8	0.88	0.89	0.89	682
9	0.88	0.75	0.81	416
10	0.89	0.80	0.84	520
accuracy			0.93	17436
macro avg	0.89	0.87	0.88	17436
weighted avg	0.93	0.93	0.93	17436

Figure S2: Comparison of XGBoost classification performance on training set before (left) and after (right) feature selection

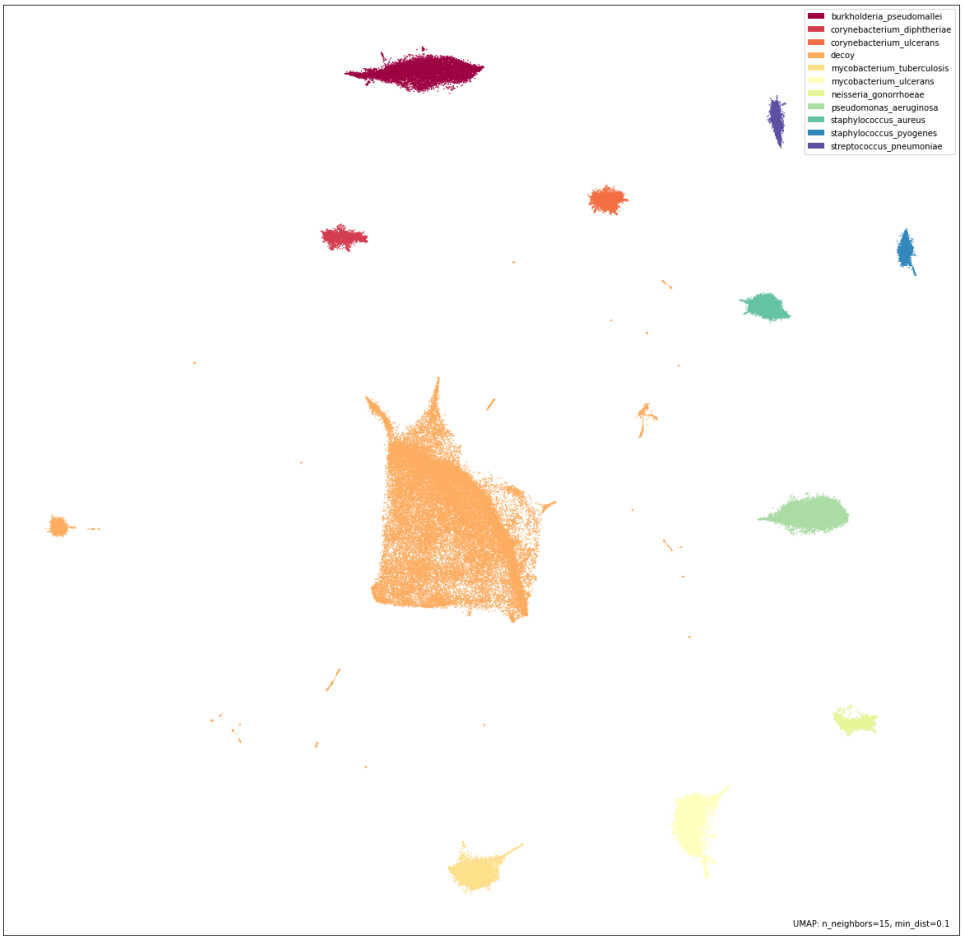

Figure S3: UMAP plot of species clusters after feature selection (classes listed in alphabetical order)

Table S1: Table of tuned hyperparameters for each model, and their names in the submission folder

Model	XGBoost	LightGBM	SVM
Tuned hyperparameters	max_depth=4 learning_rate=0.1 gamma=0, reg_lambda=0 scale_pos_weight=1 base_score=0 reg_alpha=1	boosting=gbdt num_leaves=31 feature_fraction=0.5 bagging_fraction=0.5 bagging_freq=20 learning_rate=0.05 n_estimators=500 max_depth=5 lambda_l1=1.0 lambda_l2=0.25	kernel = poly C = 10 degree = 2
Model name	xgb-d75-tuned.joblib	lgbm_model-final2.joblib	poly2_model.sav

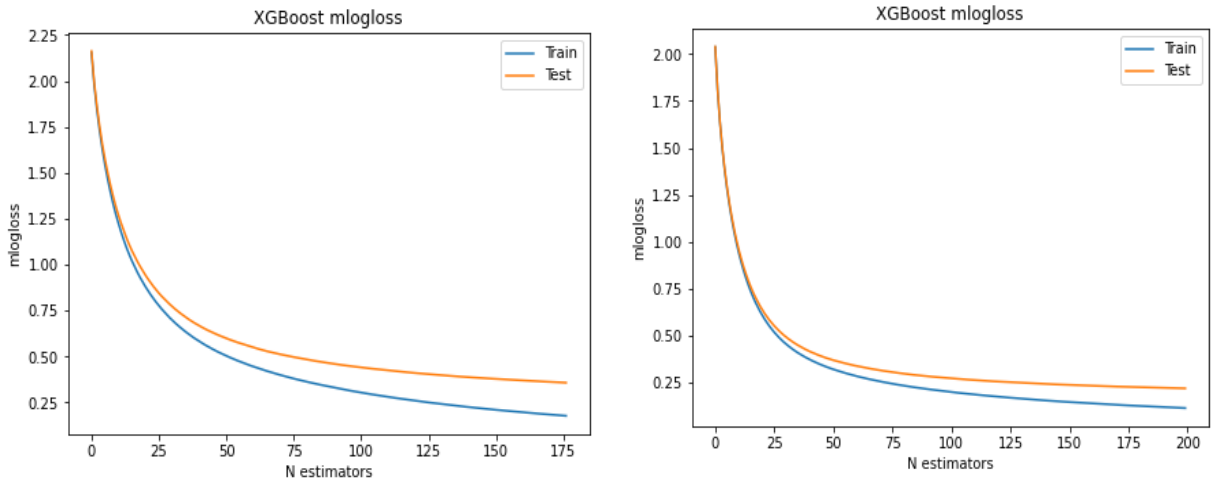

Figure S4: Multi-class log loss plots on training data before (left) and after (right) hyperparameter tuning