1 Introduction

This experiment comprises two stages: first, we tackled a multi-class classification task on the original dataset; then, on a partially labeled dataset with suspected distribution shift, we analyzed the shift type and applied appropriate mitigation strategies.

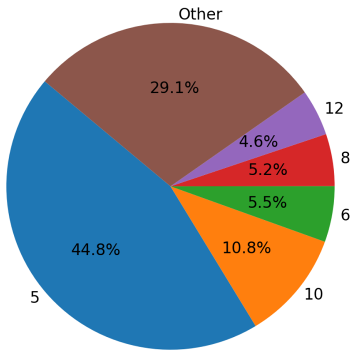
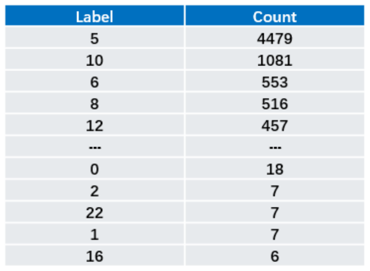


Figure 1: Class Distribution



Label	Count
5	4479
10	1081
6	553
8	516
12	457
---	---
0	18
2	7
22	7
1	7
16	6

Figure 2: Distribution Table

For the training dataset, we first confirmed the absence of missing values and verified that all features are of float type, eliminating the need for nan-filling or one-hot encoding. At the label level, the class distribution—illustrated in the pie chart and table in Figure 1 and 2—shows that Class 5 accounts for 44.8% of the samples, and the top five classes together represent 70.9%. In contrast, the least frequent class (Class 16) contains only six samples, which indicating a severe class imbalance. At the feature level, we examined potential linear dependencies and redundancy using Pearson's r and Principal Component Analysis (PCA). As shown in Figure3 and Figure 4, no strong linear correlations were observed. However, a substantial proportion of variance is explained by the first half of the principal components, suggesting notable feature redundancy within the dataset.

Distribution shift can be categorized into covariate shift, label shift, and concept drift. At the data level, Kolmogorov–Smirnov (K-S) tests on each feature between the training set and test set 2 revealed significant distribution changes in 268 out of 300 features, confirming the change in $P(X)$. Although $P(y|X)$ hasn't been verified yet, it is suggesting that covariate shift has likely occurred. After comparing the label distributions, and conducting K-S test on the conditional distributions of X given each label, we observed that $P(y)$ had shifted while $P(X|y)$ remained largely unchanged, suggesting a high likelihood of label shift (All experiments are illustrated in Appendix Section A).

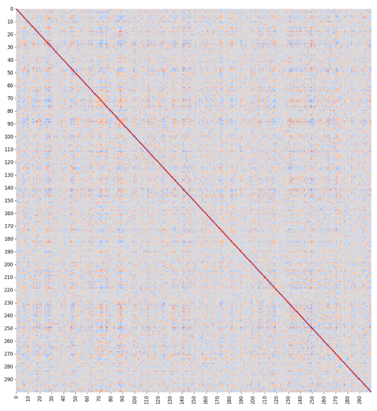


Figure 3: Pearson's r

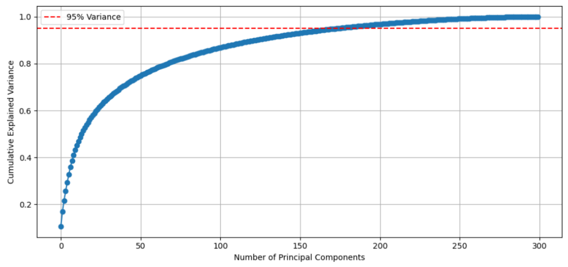


Figure 4: PCA Cumulative Explained Variance

Learning on extremely imbalanced data has been extensively studied, with He and Garcia [1] proposing effective data processing strategies, while Zhou [2] systematically explored ensemble learning techniques for robust model construction. Building upon these foundations, we integrated both approaches in our framework. For distribution shift, we conducted identification and mitigation based on the formal definitions and solutions introduced by Huyen Chip in [3].

2 Methodology

Gradient Boosting Decision Trees (GBDT) is an efficient ensemble learning technique well-suited for classification tasks. It builds a sequence of shallow trees, each trained to correct the errors of its predecessor, thereby improving overall performance iteratively. GBDT models are known for their strong generalization ability and inherent feature selection during training. In this study, we adopted three widely used GBDT implementations: XGBoost [4], LightGBM [5], and CatBoost [6], with brief introductions provided in Appendix Section C. Also due to their decision tree-based nature, no standardization, normalization, or regularization was applied, as such models typically operate on feature thresholds and do not require these preprocessing steps [7].

While GBDT models help reduce bias, their relatively complex structure may increase variance. To mitigate this, in addition to hyperparameter tuning, we adopted a k-fold inference strategy. Specifically, the training set was partitioned into k folds, and three types of GBDT models were each trained across all folds, resulting in $3k$ base models. Each base model generated predictions on the test set, and final predictions were obtained via ensembling. Among several ensemble strategies considered, we selected stacking with L2-regularized logistic regression as the meta learner based on experimental performance. The meta learner was trained using base learner predictions on their corresponding validation folds (Figure 5), and data for predictions were constructed by averaging probabilities within each model type before concatenation (Figure 6).

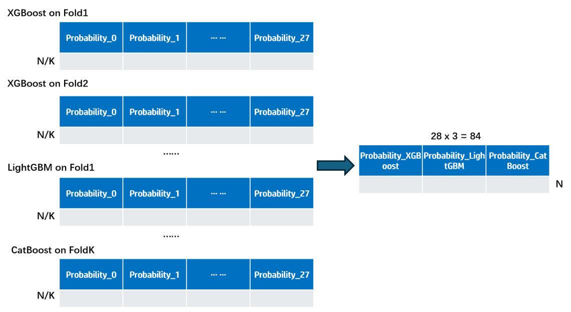


Figure 5: Meta Learner Training Set Structure

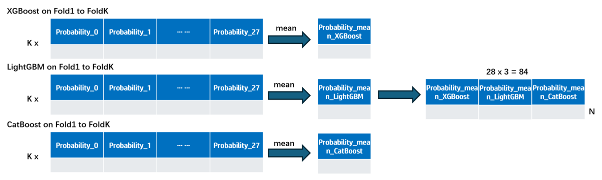

XGBoost on Fold1 to FoldK
LightGBM on Fold1 to FoldK
CatBoost on Fold1 to FoldK
K x
Probability_0
Probability_1
Probability_27
mean
Probability_mean_XGBoost
Probability_mean_LightGBM
Probability_mean_CatBoost
28 x 3 = 84
N

Figure 6: Meta Learner Test Set Structure

To ensure comprehensive and unbiased evaluation, we fixed the random seed to 42 and employed 5-fold cross validation throughout the experiment. The choice of 5 folds aligns with the nested structure of our k-fold inference framework, involving both outer and inner validation loops. Given that the rarest class contains only 6 samples, using more than 5 folds would risk excluding certain classes from individual splits. Hence, 5 was the maximum feasible fold number to preserve class representation in all folds.

As specified in the task description, naturally weighted cross-entropy loss was adopted as the primary evaluation criterion throughout the experiment. In addition, we considered other metrics commonly used for imbalanced multi-class classification, including balanced accuracy and macro F1 [8], which assign equal importance to all classes regardless of their frequency. However, these were not used directly for model selection.

To assess whether differences in generalization performance were statistically significant, we employed formal statistical testing rather than relying solely on mean comparisons. We first tested whether the performance scores followed a normal distribution, then depending on the distribution and the number of groups (two or more), appropriate methods such as paired t-tests or ANOVA were applied accordingly [9]. The complete procedure and choice of tests are illustrated in the flowchart in Appendix Section B.

To address the severe class imbalance in the dataset, three commonly used row sampling strategies are tested (random over sampling + random under sampling, SMOTE + random under sampling + ENN and ADASYN + Tomek Links). For each method, we conducted grid search to determine the optimal sampling ratios and ultimately selected the best-performing strategy among the three.

To mitigate information redundancy, we performed feature selection based on either importance scores or SHAP (SHapley Additive exPlanations) values [10]. Using a step size of 2%, we determined the optimal proportion of top-ranked features to retain for each method, and subsequently selected the better-performing one.

Bayesian hyperparameter optimization [11] was applied to all three GBDT models and the logistic regression meta learner to enhance performance. The tuning targeted key parameters, including the regularization strength C for logistic regression and learning-related hyperparameters such as learning_rate and max_depth for GBDT models.

By applying the trained model to labeled samples from the new test set and observing almost consistent performance across classes (i.e. $P(y|X)$ might not changed), we infer that concept shift is unlikely to be present. To address potential covariate shift, we trained a logistic regression classifier to distinguish between training and test samples, and used the predicted ratio $P(test)/P(train)$ as sample weights. The models was then fine-tuned using 202 labeled samples from the new dataset [12]. For label shift, we performed Bayes posterior correction to correct the prediction from our original model [13].

3 Results

After all the experiments, we've determined that SMOTE + random under sampling + ENN combination and SHAP-based feature selection will give us the best performance. The optimal results and corresponding statistical comparisons before and after tuning on LightGBM as an example are summarized in Appendix Section D for all three experiments. Then, by applying the ensemble strategy, we achieved a final 5-fold cross validation mean of approximately 0.0046 on the training set, which is a 50% improvement comparing to the naive model, which assigns uniform probabilities to all classes (0.0097). To evaluate this result, we used two baseline models for comparison: a naive model, a XGBoost model, and the results are shown in Figure 7. Other metrics for the final model are shown in Figure 8.

A comparison of SHAP-based feature selection across the three models reveals 98 shared features, with top-ranked ones including features such as 17, 99, and 270. Meanwhile, differences in selected features among models contribute to the diversity and complementarity of the ensemble.

Although this may not fully represent the effect of our methodologies on distribution shift, after addressing covariate and label shift, the final weighted cross-entropy loss on the labeled samples in test set 2 represents a decrease of compared to the original model.

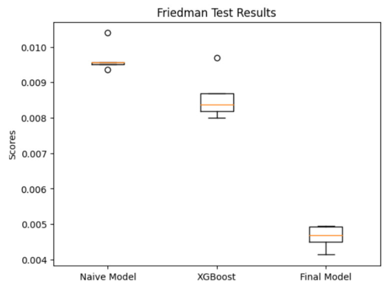
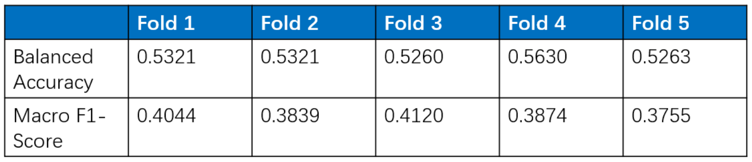


Figure 7: Naive Model vs. XGBoost vs. Final Model



	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Balanced Accuracy	0.5321	0.5321	0.5260	0.5630	0.5263
Macro F1-Score	0.4044	0.3839	0.4120	0.3874	0.3755

Figure 8: Other Metrics

4 Discussion

Guided by the bias-variance decomposition of error, we first conducted 5-fold cross validation to compare XGBoost with logistic regression. As shown in Figure 9, the paired t-test yielded a p-value of 0.0086, indicating that the limited capacity of logistic regression resulted in higher bias, while increasing model complexity led to improved performance. The benefit of ensemble learning was further validated by comparing a single XGBoost model with its five-fold inference counterpart using simple averaging (Figure 10), where a p-value of 0.0007 confirmed a significant performance gain and enhanced robustness against overfitting. These results provided a clear empirical basis for the modeling strategy.

The ensemble strategies evaluated, along with their mean weighted cross-entropy loss from 5-fold cross validation, are summarized in Figure 11. Among them, stacking with L2-regularized logistic regression as the meta learner (before Bayes Tuning) achieved the best performance (0.005098). In contrast, using more complex models such as Random Forest or XGBoost as meta learners resulted in degraded performance, even worse than simple

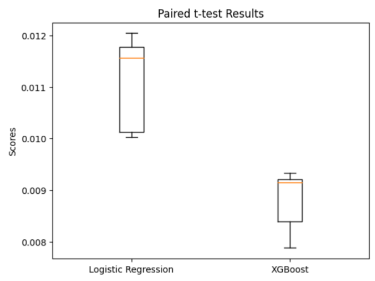


Figure 9: LR vs. XGBoost

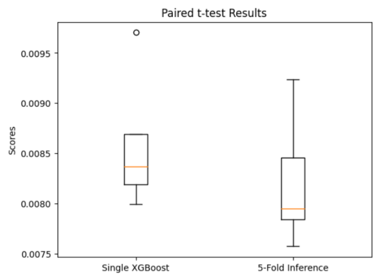
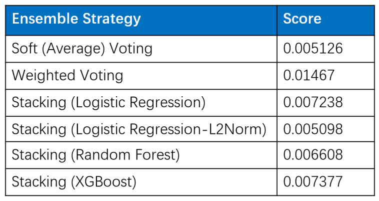


Figure 10: Single XGBoost vs. Ensemble



Ensemble Strategy	Score
Soft (Average) Voting	0.005126
Weighted Voting	0.01467
Stacking (Logistic Regression)	0.007238
Stacking (Logistic Regression-L2Norm)	0.005098
Stacking (Random Forest)	0.006608
Stacking (XGBoost)	0.007377

Figure 11: Ensemble Strategies

averaging. This is likely due to overfitting, as the meta learner operates on features already abstracted by the base learners. This interpretation is further supported by the performance drop observed when removing L2 regularization from logistic regression.

Accuracy-based metrics are unsuitable for this highly imbalanced classification task. For example, a model that only predicts the top five majority classes could still achieve an accuracy of around 70%, while entirely ignoring minority classes. This is clearly misaligned with our objective of fair treatment across all categories. However, it is also worth noting that minimizing weighted cross-entropy loss does not always lead to better classification performance. This is because the loss is computed based on the predicted probabilities assigned to the correct labels, rather than discrete prediction accuracy. A counterexample illustrating this limitation is provided in Appendix Section E.

Future work may explore alternative ensemble strategies such as one-vs-rest modeling to better capture minority classes, apply statistical testing during hyperparameter tuning for more robust evaluation rather than mean value comparison, and conduct further research on distribution shift, particularly on concept drift.

5 Conclusion

This study addressed class imbalance and distribution shift in a multi-class classification task. By using GBDT models and ensemble strategies, SMOTE based row sampling, SHAP based feature selection and Bayesian tuning, our approach achieved a performance improvement of over 50% in comparison to the naive model. For distribution shift, we accurately identified its type based on the formal definition and implemented weighted training and Bayes posterior correction to enhance model performance.

References

[1] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[2] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms.* CRC Press, Boca Raton, FL, 2012.

[3] Huyen Chip. Data distribution shifts and monitoring, 2022. Accessed: 2025-04-22.

[4] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 785–794. ACM, 2016.

[5] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 3146–3154, 2017.

[6] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: Unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 6638–6648, 2018.

[7] Max Kuhn and Kjell Johnson. *Feature Engineering and Selection: A Practical Approach for Predictive Models.* CRC Press, 2019.

[8] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[9] Francois Deryckel. Machine learning with r - chapter 2: Tests and inferences. `https://fderyckel.github.io/machinelearningwithr/testinference.html`, 2020. Accessed: 2025-04-19.

[10] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

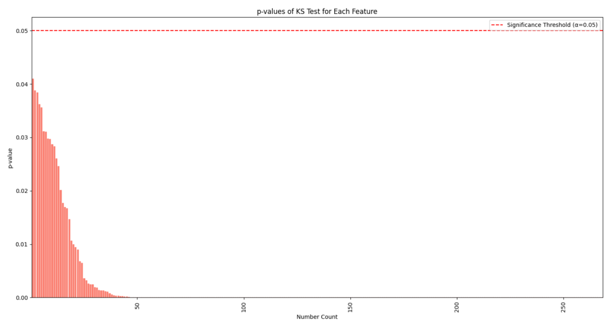
[11] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

[12] Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning under covariate shift. *Journal of Machine Learning Research*, 10(Sep):2137–2155, 2009.

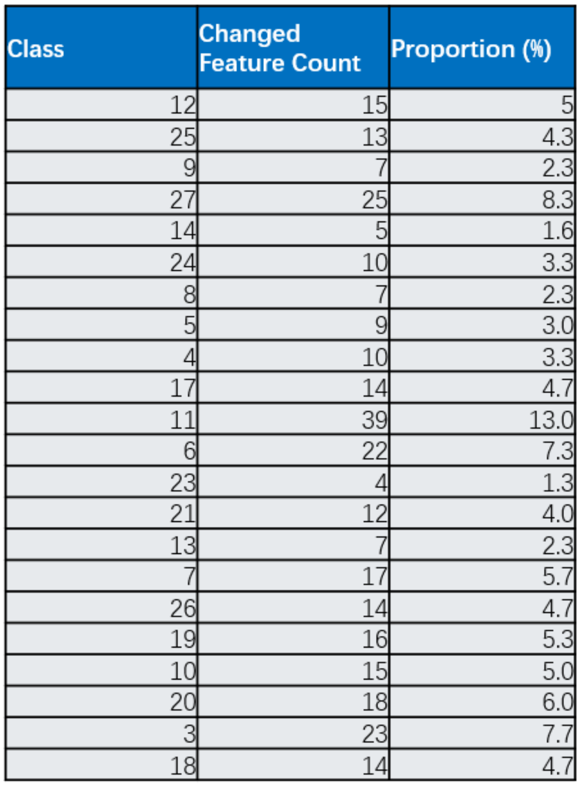
[13] Marco Saerens, Patrice Latinne, and Christine Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. *Neural computation*, 14(1):21–41, 2002.

Appendix

A Tests on Dataset for Distribution Shifts



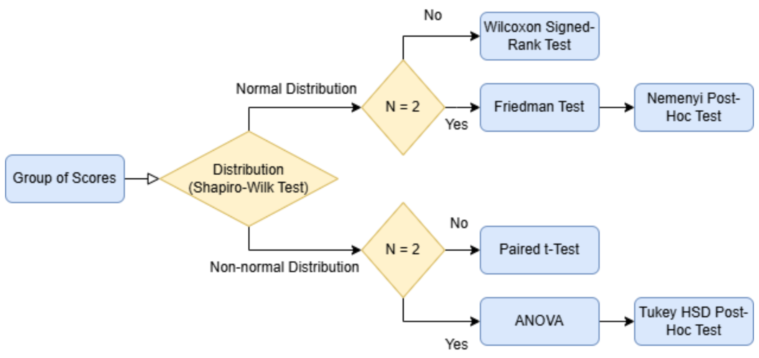
Histogram of K-S Test for Each Features



Class	Changed Feature Count	Proportion (%)
12	15	5
25	13	4.3
9	7	2.3
27	25	8.3
14	5	1.6
24	10	3.3
8	7	2.3
5	9	3.0
4	10	3.3
17	14	4.7
11	39	13.0
6	22	7.3
23	4	1.3
21	12	4.0
13	7	2.3
7	17	5.7
26	14	4.7
19	16	5.3
10	15	5.0
20	18	6.0
3	23	7.7
18	14	4.7

Changed Features per Class

B Flowchart for Statistical Test



Flowchart for Statistical Test

C Brief Introduction about GDBT Models

- **XGBoost**: It integrates second-order gradient optimization and regularization techniques, enhancing both the accuracy and stability of the model. Its parallel tree construction algorithm speeds up the training process and improves the model's ability to generalize to new cases.
- **LightGBM**: It utilizes a histogram-based learning approach and a leaf-wise growth strategy, and allows for rapid training while preventing overfitting through discretization and built-in regularization strategies.
- **CatBoost**: It stands out with its ordered boosting mechanism, which effectively manages categorical features, significantly reducing the risk of overfitting and simplifying the data preparation process.

D Results for Tuning, with Comparison on LightGBM as an Example

It is worth noting that, aside from feature selection, the other two preprocessing strategies led to significant improvements in model performance. This is due to the nature of GBDT models which already possess a certain degree of built-in feature selection capability, thereby diminishing the impact of explicit feature selection.

Model	Method	Weighted Cross-Entropy Loss (Average)	(under_sampling, over_sampling)
XGBoost	random over and under sampling	0.008578066	(1000, 100)
XGBoost	SMOTE+random under sampling+Edited Nearest Neighbours	0.007278281	(1000, 400)
XGBoost	ADASYN+Tomek Links	0.008564232	(1100, 800)
LightGBM	random over and under sampling	0.013211832	(1200, 500)
LightGBM	SMOTE+random under sampling+Edited Nearest Neighbours	0.009619888	(1400, 500)
LightGBM	ADASYN+Tomek Links	0.012612577	(1600, 600)
CatBoost	random over and under sampling	0.007647356	(1000, 200)
CatBoost	SMOTE+random under sampling+Edited Nearest Neighbours	0.005996197	(1000, 400)
CatBoost	ADASYN+Tomek Links	0.007290376	(1400, 500)

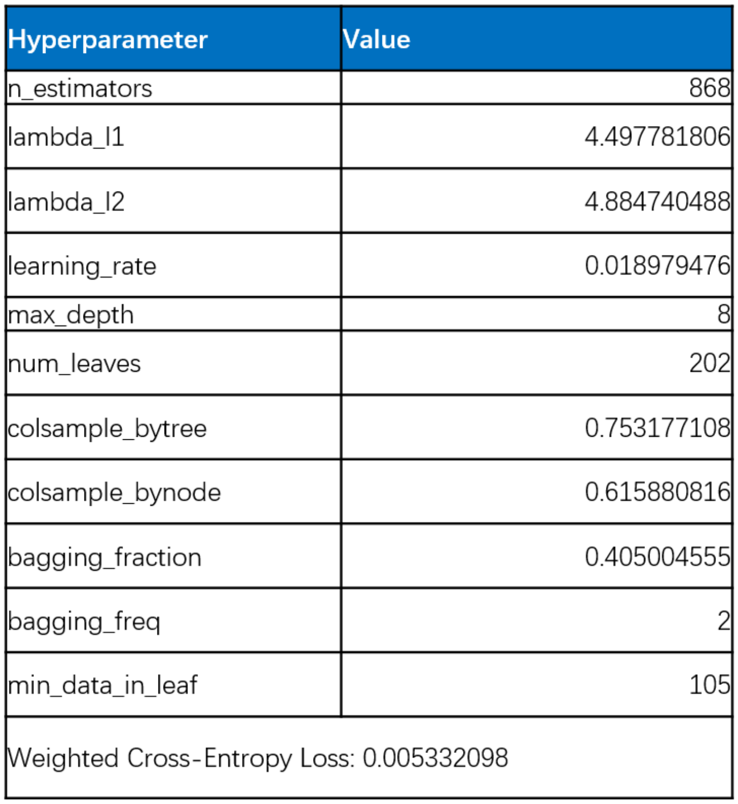
Experiment Result for Row Sampling

Model	Method	Weighted Log Loss	Ratio
XGBoost	Importance	0.007227115	0.94
XGBoost	SHAP	0.006894432	0.44
LGBM	Importance	0.009517474	0.86
LGBM	SHAP	0.009401108	0.8
CatBoost	Importance	0.005906308	0.96
CatBoost	SHAP	0.005899048	0.6

Experiment Result for Feature Selection

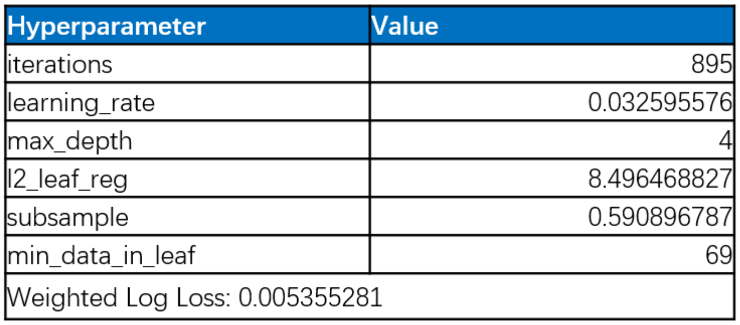
Hyperparameter	Value
n_estimators	519
learning_rate	0.019464995
max_depth	5
lambda	0.266294078
alpha	0.093557212
subsample	0.448423575
colsample_bytree	0.564700608
colsample_bynode	0.544157088
colsample_bylevel	0.711328175
min_child_weight	7.736242052
gamma	0.394395367
Weighted Cross-Entropy Loss: 0.005388398	

Experiment Result for Bayes Tunning on XGBoost



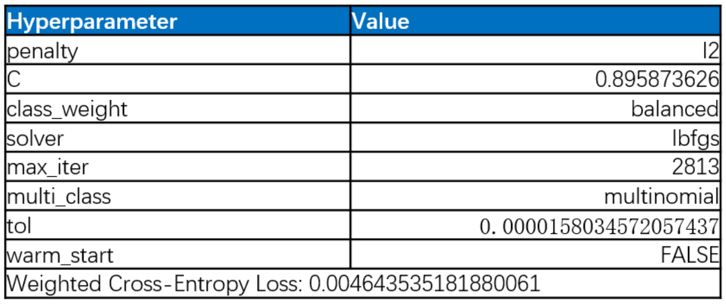
Hyperparameter	Value
n_estimators	868
lambda_l1	4.497781806
lambda_l2	4.884740488
learning_rate	0.018979476
max_depth	8
num_leaves	202
colsample_bytree	0.753177108
colsample_bynode	0.615880816
bagging_fraction	0.405004555
bagging_freq	2
min_data_in_leaf	105
Weighted Cross-Entropy Loss: 0.005332098	

Experiment Result for Bayes Tunning on LightGBM



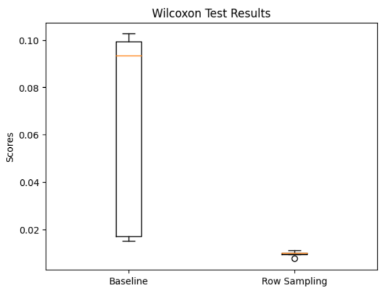
Hyperparameter	Value
iterations	895
learning_rate	0.032595576
max_depth	4
l2_leaf_reg	8.496468827
subsample	0.590896787
min_data_in_leaf	69
Weighted Log Loss: 0.005355281	

Experiment Result for Bayes Tunning on CatBoost

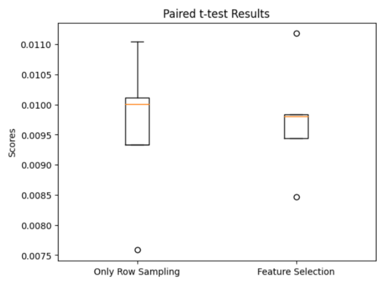


Hyperparameter	Value
penalty	l2
C	0.895873626
class_weight	balanced
solver	lbfgs
max_iter	2813
multi_class	multinomial
tol	0. 0000158034572057437
warm_start	FALSE
Weighted Cross-Entropy Loss: 0.004643535181880061	

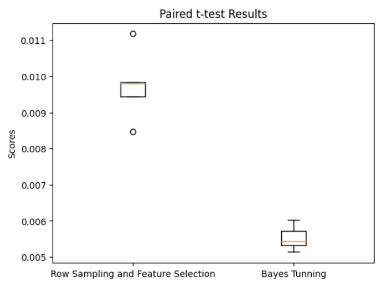
Experiment Result for Bayes Tunning on Meta Logistic Regression



Row Sampling



Feature Selection



Bayes Tuning

E Why Weighted Cross-Entropy Loss May Not Lead to a Good Performance

Consider this simple example: in a binary classification task, the ground truth labels for the test set are [1, 0], and we have model A outputs the predicted probabilities [[0.4, 0.6], [0.1, 0.9]], while Model B just gives equal probability to all the classes for all the samples (i.e. [[0.5, 0.5], [0.5, 0.5]]). In this scenario, the weighted cross-entropy loss (calculated using the given code) for Model A is approximately 0.70, whereas for Model B it is around 0.34. We may reasonably infer that Model A has somehow learned meaningful patterns from the data, whereas Model B is merely making random guesses. However, despite this, Model B achieves a weighted cross-entropy loss that is more than 50% lower than that of Model A.