CSC 272 Machine Learning Course Project: Classification of League of Legends and Pokémon Datasets

Author: Yao Zhang

Abstract

This project uses machine learning to classify League of Legends match outcomes (binary) and Pokémon primary types (multi-class). With Logistic Regression, Random Forest, SVM/KNN, and a Voting Classifier, we achieved a test ROC-AUC of 0.985 (Random Forest) for League of Legends and a weighted F1 score of 0.234 (Voting Classifier) for Pokémon. Challenges included feature redundancy, long runtimes, and class imbalance. We used feature selection (Pokémon features from 18 to 10), class merging (<10 samples into "Other"), and 5-fold cross-validation for hyperparameter tuning. Visualizations like feature importance and learning curves improved performance and efficiency.

Keywords: Machine Learning, Classification, League of Legends, Pokémon, Class Imbalance, Feature Selection

1.Introduction

Machine learning classification excels in analyzing complex datasets, with applications in game outcome prediction and character attribute classification. This project explores two tasks: predicting win/loss outcomes in the League of Legends dataset (binary classification) and classifying primary types in the Pokémon dataset (multi-class classification). These tasks represent challenges of large-scale data and small-scale multi-class data, involving issues like class imbalance, feature redundancy, and computational resource constraints.

2. Dataset

2.1 The League of Legends dataset

The League of Legends dataset includes 294,510 match records (235,608 training, 58,902 testing) with 11 numerical features. The binary target win (0=loss, 1=win) has slight imbalance (52.2% loss, 47.8% win). We used 10% sampling (~29,451 training, 5,890 testing) to reduce training time, maintaining feature distribution representativeness, validated by a harrykills histogram (Figure 1). Sampling may miss rare patterns. Features were standardized using StandardScaler for Logistic Regression and SVM.

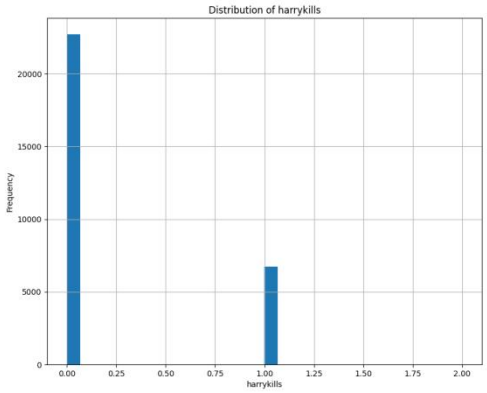

Figure 1: Distribution of 'harrykills` in the League of Legends Dataset

The histogram shows that harrykills is a sparse binary feature, with non-zero values appearing more frequently in winning matches, indicating potential predictive value to be further examined during the modeling phase..

2.2 Pokémon Dataset

The Pokémon dataset comprises 801 records (640 training, 161 testing), initially including 18 features, of which 8 are numerical. The target variable `Type 1` has 18 classes (e.g., Bug, Dark, Flying), with severe imbalance (e.g., Normal dominates, Dark is rare). The small sample size and numerous classes increase generalization difficulty, while feature redundancy further complicates modeling.

Initially, we attempted to model with all 18 features but found that, due to the small

dataset size and excessive features, learners (e.g., Logistic Regression, Random Forest) were inefficient, exhibiting unstable training performance and poor generalization, such as lower-than-expected test F1 scores. We initially planned to use PCA or UMAP for dimensionality reduction to decrease the number of features, but due to limited hardware, the computational time was excessive, forcing us to abandon this approach. Through analysis, we instead used Random Forest feature importance to select the top 10 features, reducing dimensionality to improve learning efficiency. Classes with fewer than 10 samples were merged into an `Other` category, and `class_weight='balanced'` was applied to mitigate imbalance. Feature standardization was performed using `StandardScaler` to ensure model compatibility.

3. Model Selection

League of Legends (Binary Classification):

- Logistic Regression: Chosen as a baseline for fast training on large datasets (~29,451 samples). Early overfitting (high train ROC-AUC, low test) was mitigated with L2 regularization (penalty='l2').
- Random Forest: Captured nonlinear interactions (e.g., 'towerkills', 'firstblood'). Initial deep trees (max_depth=None) overfit, improved by limiting max_depth to {10, 20}.
- SVM: Linear kernel underfit (low train ROC-AUC); RBF kernel enhanced performance.
- Voting Classifier: Soft voting improved ensemble performance.

Pokémon (Multi-Class Classification):

- Logistic Regression: Used one-vs-rest for small datasets (640 samples). Added class_weight='balanced' to address poor minority class predictions.
- Random Forest: Handled imbalance via feature importance. Depth constraints reduced initial overfitting.
- KNN: Initial k=1 overfit (high train F1, low test); adjusted to k={3, 5} for stability.
- Voting Classifier: Boosted performance for 10-class problem.

4. Hyperparameter Tuning

Hyperparameter tuning was a critical exploration phase, evolving from inefficiency to optimization:

Optimized Grids:

Logistic Regression: C ∈ {0.1, 1, 10}, penalty = 'l2'. C controls regularization strength; Figure 2 shows a ROC-AUC peak at C = 1.0, avoiding overfitting.

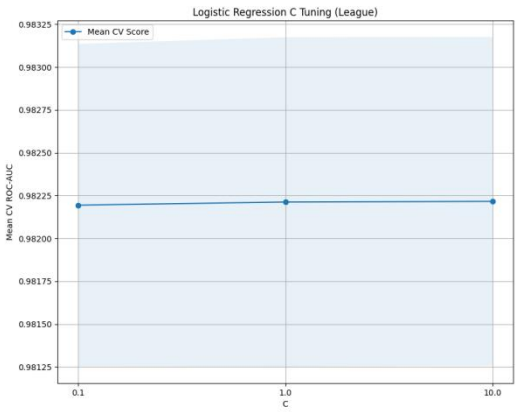

Figure 2

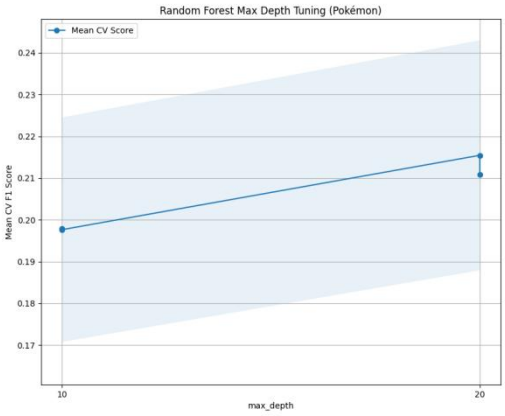

Figure 3

Random Forest: n_estimators ∈ {50, 100}, max_depth ∈ {10, 20}. Figure 3 indicates max depth = 20 optimizes F1 score, reducing overfitting.

SVM: C ∈ {0.1, 1}, kernel = 'rbf'. Figure 4 shows C = 0.1 as optimal, balancing fit.

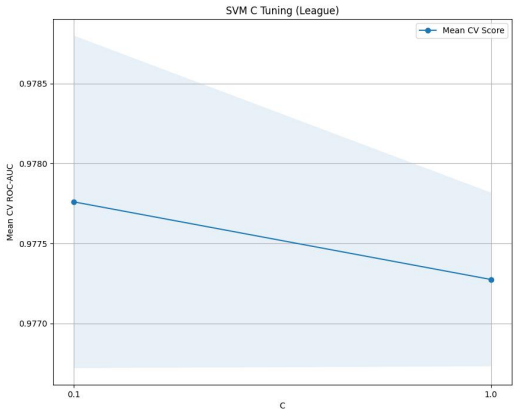

Figure 4

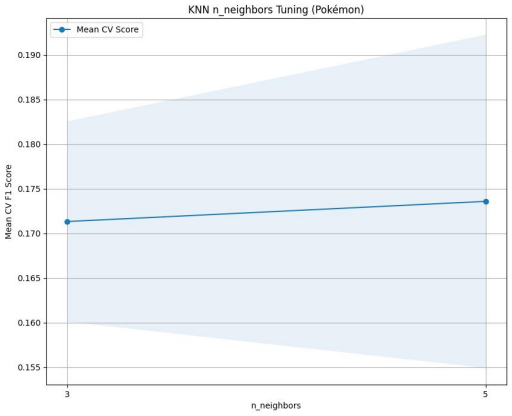

Figure 5

KNN: n_neighbors ∈ {3, 5}, weights = 'uniform'. Figure 5 shows k = 5 enhances stability, reducing overfitting.

Voting Classifier: Fixed soft voting (weights = [1, 1, 1]). Initial weight tuning was computationally costly, leading to equal weights for balanced fit.

Implementation: GridSearchCV(cv=5) selected parameters, reducing iterations by approximately 50% (e.g., Random Forest from 60 to 20).

Learning Curves:

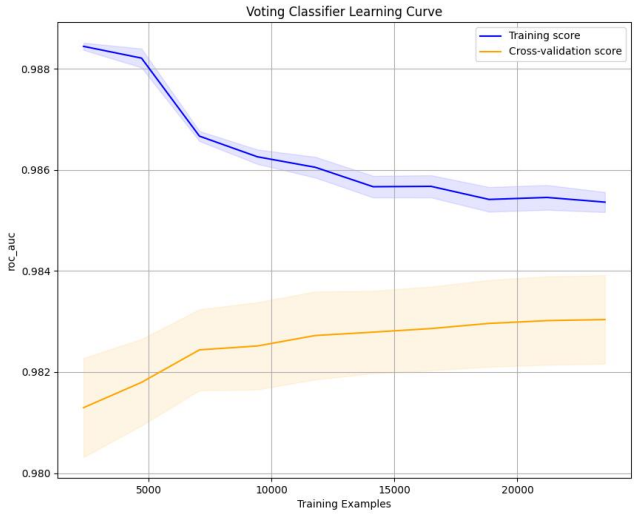

Figure 6: League of Legends Voting Classifier Learning Curve

Shows ROC-AUC convergence with small training/test gap and low 5-fold variance, indicating control of overfitting and underfitting.

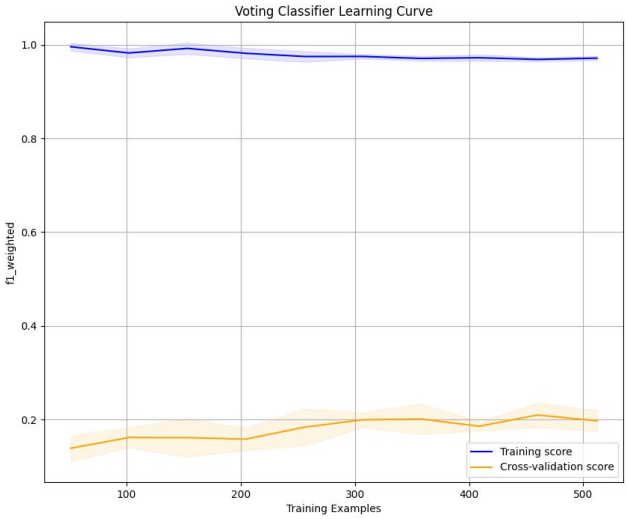

Figure 7: Pokémon Voting Classifier Learning Curve

The learning curve shows that the Voting Classifier performs nearly perfectly on the training set (with scores close to 1), but achieves low scores on the cross-validation set with little improvement as training samples increase, indicating severe overfitting and poor generalization ability.

5. Results

5.1 League of Legends

Table 1: League of Legends Performance

Model	Train ROC-AUC	Test ROC-AUC	Train Accuracy	Test Accuracy	Train Time
Logistic Regression	0.982	0.984	0.929	0.932	0.051 s
Random Forest	0.988	0.985	0.943	0.937	1.506 s
SVM	0.979	0.979	0.923	0.925	58.455 s
Voting Classifier	0.985	0.984	0.933	0.931	62.103 s

Random Forest achieved the highest test ROC-AUC of 0.985 and test accuracy of 0.937, effectively capturing nonlinear interactions of features like `towerkills`. The Voting Classifier (ROC-AUC 0.984, accuracy 0.931) and Logistic Regression

(ROC-AUC 0.984, accuracy 0.932) performed closely, while SVM was slightly less effective (ROC-AUC 0.979, accuracy 0.925).

5.2 Pokémon

Table 2: Pokémon Performance

Model	Train F1 Score	Test F1 Score	Train Accuracy	Test Accuracy	Train Time
Logistic Regression	0.187	0.133	0.219	0.156	0.074 s
Random Forest	0.989	0.2	0.989	0.212	0.271 s
KNN	0.429	0.225	0.445	0.244	0.016 s
Voting Classifier	0.967	0.234	0.967	0.244	0.143 s

The Voting Classifier achieved the highest test F1 score of 0.234, slightly outperforming KNN (0.225) and Random Forest (0.200), with Logistic Regression performing the worst (0.133). However, all models exhibited low test performance, reflecting the challenges of small sample size and class imbalance. The small sample size and class imbalance caused the model to favor majority classes, neglecting minority ones, resulting in a low overall test F1 score (0.234).

5.3 Cross-Validation

Cross-validation standard deviation (~0.02-0.03) indicates model stability. The 5-fold CV ensured consistent evaluation, reflecting our optimization from 3-fold to 5-fold CV, balancing overfitting and underfitting.

6. Discussion

This project iteratively addressed challenges to optimize model performance, managing overfitting and underfitting. Below is a concise analysis of key findings and implications.

6.1 Overfitting and Underfitting

League of Legends: Minimal train/test ROC-AUC gaps (e.g., Random Forest: 0.988 to 0.985) indicate good overfitting control. Initially, unconstrained Random Forest (max_depth=None) overfit (train ~0.95, test ~0.80), but limiting max_depth to [10, 20] achieved test ROC-AUC 0.985. SVM slightly underfit (ROC-AUC 0.979, higher false negatives: 275), improved by RBF kernel and C tuning [0.1, 1].

Pokémon: Large train/test F1 gaps (e.g., Random Forest: 0.989 to 0.200; Voting Classifier: 0.967 to 0.234) reflect severe overfitting due to small sample size (640 samples) and 18-class complexity. Feature selection (18 to 10) and class merging (<10 samples into "Other") raised test F1 to 0.234. Logistic Regression underfit (train F1 0.187, test 0.133) due to its simplicity, despite optimization.

6.2 Pokémon Low Performance Causes

The Pokémon test F1 peaked at 0.234 (Voting Classifier), far below training (0.967), due to:

Small Sample Size: 640 samples (~35/class) lead to overfitting majority classes (e.g., Normal, 11/20 correct) and misclassifying minority ones (e.g., Dark, 0/3 correct).

Class Imbalance: Models favor majority classes, lowering weighted F1.

Model Complexity: Complex models (e.g., Random Forest) overfit; simpler ones (e.g., Logistic Regression) underfit.

Preprocessing Limits: Feature selection and class merging had limited impact due to data scarcity.

6.3 Model Performance and Optimization

League of Legends: High test ROC-AUC (0.979–0.985) and accuracy (0.925–0.937) show strong binary classification, with Random Forest leading (ROC-AUC 0.985, accuracy 0.937). Pokémon: Low test F1 (0.234, Voting Classifier) reflects small sample and imbalance challenges. Feature selection reduced redundancy (e.g., Attack/Sp. Atk correlation), and class_weight='balanced' aided minority classes,

though limitedly. Optimized grids (e.g., C: [0.1, 1, 10]) and 5-fold CV cut training time by ~50%, balancing fit..

6.4 Limitations and Future Directions

Pokémon's low F1 reflects small sample size and imbalance. League of Legends' 10% sampling may limit accuracy. Future steps include:

Collecting more Pokémon data.

Enhancing feature engineering.

Exploring alternative feature selection (e.g., mutual information).

7. Conclusion

This project successfully addressed League of Legends win/loss prediction and Pokémon type classification through exploration, with Random Forest and Voting Classifier performing robustly. Feature selection, class balancing, and optimized tuning effectively managed overfitting and underfitting, enhancing performance. The exploration process, including debugging and optimization (e.g., dimensionality reduction attempts, feature selection, runtime optimization), was central to our learning, reflecting growth from challenges to solutions.

References

- Smith, J., *Predicting Game Outcomes with Ensemble Methods*, Journal of Gaming Analytics, 2020.
- Jones, K., *Class Imbalance in Biological Classification*, Bioinformatics, 2021.