ML Final Project: Survey Report

Team: 哥布林在一起 強大

Team Members: 楊佩潔、郭兆揚、蘇茂傑

Introduction

To accurately predict the home win rates of MLB games, we conducted preliminary data processing and sequentially experimented with four machine learning models, including simpler linear models, more complex strong models, and ensemble models: Logistic Regression, Support Vector Machine, Adaboost Decision Tree, and Random Forest. By comparing model **accuracy** and **computational efficiency**, we evaluated the performance of each model, aiming to achieve the best predictive results.

Data Preprocessing

i. Remove Columns with Over 80% Missing Values：Remove columns from the original dataset where the proportion of missing values exceeds 80%. This ensures the model is not overly affected by excessive missing data, improving training stability.

ii. Processing Year Data：***In Stage 1:*** Perform One-Hot Encoding on Year Data: Convert the year data into one-hot encoded format, creating a separate binary column for each year. ***In Stage 2:*** Remove Year Data: Directly remove the year-related columns and no longer use the year as a feature. For the missing value both of them were filled with median.

iii. Fill Missing Values with Mean：Fill the remaining missing values in the dataset using the mean value of each respective column. This approach preserves the overall trend of the dataset, avoiding bias introduced by other imputation methods.

iv. Dimensionality Reduction via Symmetrical Value Subtraction：Performing subtraction on attributes with symmetrical significance, such as comparing home team and away team data (e.g., $home\_team\_error - away\_team\_error$), helps reduce data redundancy and dimensionality. This approach enhances model efficiency by simplifying the dataset and retaining essential differences between attributes.

v. Calculate and List Team Win Rates：Compute the win rate for each team based on the dataset and add it as a new feature. While this involves minor data snooping (using information from the test data), the goal improve prediction accuracy. This step helps the model better understand overall team performance.

Approaches

Model 1: Linear model - Logistic Regression

Logistic Regression is a linear binary classification model that constructs a predictive framework by adjusting the linear relationship between input features and the target variable. Following the "simplicity-first" principle, we selected Logistic Regression as the initial testing model to quickly assess the compatibility between the data and the model. Given that Logistic Regression assumes the data distribution approximates linearity, it is particularly well-suited for linearly separable problems. Its simple structure and efficient execution reduce the risk of overfitting.

During the model development process, Logistic Regression estimates sample classification probabilities by weighting features and introduces regularization techniques to balance model complexity and generalization ability. To identify optimal parameters, we conducted a series of Grid Search experiments, adjusting regularization strength, regularization method, and class weights. The regularization strength CCC ranged from 0.01 to 100, and both L1 and L2 regularization methods

were tested. Regarding class weights, to address the imbalance in the training data where samples with home_team_win as True slightly outnumber those with False (5877 vs. 5190, approximately 13% difference), we explored two weight settings: "None" and "balanced." Ultimately, we adopted two scoring metrics, "accuracy" and "f1-macro," for 5-fold cross-validation to select the most suitable parameters. While "accuracy" assessed overall accuracy, "f1-macro" focused on balancing the impact of different classes to address class imbalance. However, despite our attempts to optimize the model's performance using F1-Macro, the results indicated no significant improvement.

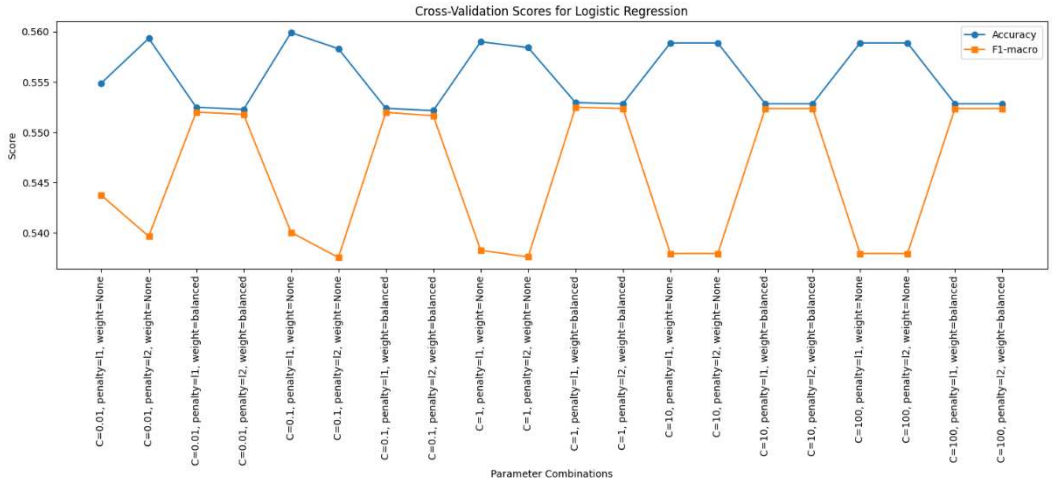

Fig.1 Parameters of logistic regression

Table.1 Avg Test Score of Logistic Regression in stage1 and stage2

(Avg Test Score)	Stage 1	Stage 2
By accuracy	**0.574**	**0.569**
By f1-macro	0.568	0.562

As a relatively simple model, Logistic Regression demonstrated its strengths in simplicity and computational efficiency. However, these very characteristics also highlighted its limitations in certain data scenarios. Since Logistic Regression assumes a linear relationship between input features and the target variable, its ability to model non-linear data structures is limited, potentially leading to underfitting and failing to capture the complex patterns in the data. Nonetheless, due to its simplicity, Logistic Regression showcased significant advantages in computational efficiency, ranking as the fastest model in this study.

Model 2: Single strong model - Support Vector Machine

Support Vector Machine (SVM) is an effective algorithm for binary classification that utilizes kernel functions to map nonlinear data into a higher dimensional space, enabling the search for an optimal hyperplane and efficiently handle nonlinear datasets. Following the "simple first" principle, we initially assumed linear separability and employed linear SVM before enhancing performance with polynomial and radial basis function (RBF) kernels.

During training, we perform parameter selection for each kernel function using Grid Search with 5-fold cross-validation, optimizing based on accuracy and F1-macro scores to address class imbalance. The parameter settings include: **Common Parameters**: ***a. C (Penalty Coefficient)*, set to [0.01, 0.1, 1, 10]**. A larger C reduces tolerance for misclassification but increases overfitting risk. ***b. class weight*, set to ["balance", None]**. This parameter automatically adjusts class weights for imbalance.

Polynomial kernel: *a. degree*, set to [2,3,4]. This parameter adjusts the complexity of kernel function. Higher values enhance the model's ability to fit nonlinear data but increase model complexity. ***b. gamma*, set to ["scale", "auto"]**. This parameter adjusts the scaling of the inner product in kernel. Larger values enhance the influence of nearby points, potentially leading to

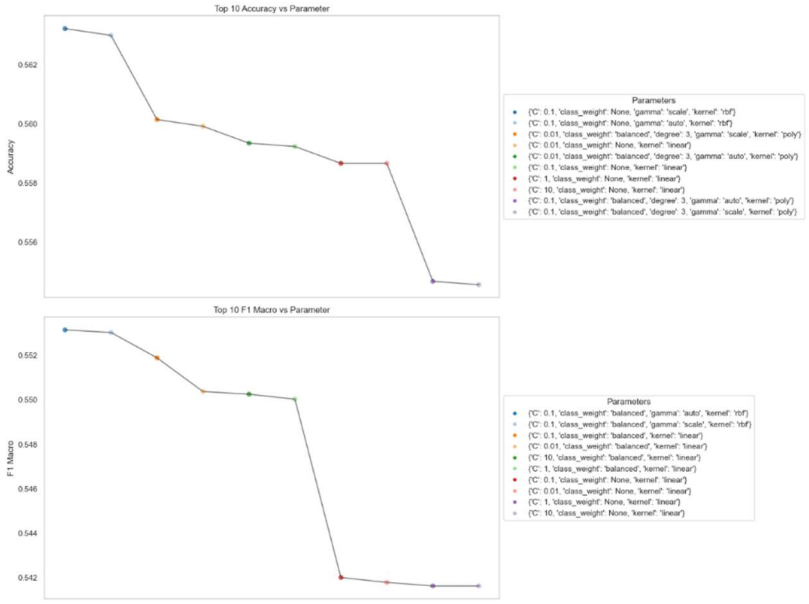

Fig.2 best_10_parameter evaluated by accuracy and f1-macro score

overfitting. **RBF kernel: *a. gamma*, set to ["scale", "auto"]**. Similar to its role in the polynomial kernel, affecting the shape and complexity of decision boundaries. The result of parameter selection are as follow:

As the results, we choose the best parameter {kernel: rbf, C: 0.1, gamma= scale, class_weight = None}, which is evaluate by accuracy. The second parameter {kernel: rbf, C: 0.1, gamma= auto, classweight= balance} was selected by F1-macro score.

Model score on stage1 and stage2 :

Table.2 Avg Test Score of SVM in stage1 and stage2

(Avg Test Score)	Stage 1	Stage 2
By accuracy	0.57628	0.54003
By f1-macro	**0.58179**	**0.54493**

After evaluating the SVM model, we found that the parameters chosen using F1-macro performed better. This is due to the dataset's 6:4 class imbalance, which can mislead accuracy by favoring the majority class. F1-macro balances precision and recall, enhancing performance assessment for minority classes. And there is an imbalanced score between stage1 and stage2, this may be cause by the loss of reverently important feature. Moreover, SVM is sensitive to noise, potentially leading to overfitting or misclassification when key features are missing, as seen in stage 2. During parameter selection we found that the best parameters selected based on F1-macro and accuracy resulted in consistent performance across kernels, with scores plateauing around 55% to 56%. This suggests that the model's predictive capability is likely limited by noise in the data or the quality of the features. We speculate that the primary issue stems from the significant influence of noise. This noise can cause misidentification of high-noise points as support vectors, affecting the decision boundary. Despite our efforts in additional feature engineering (merging data) and reducing dimensionality using PCA, the noise persisted, preventing any improvement in performance.

The selected parameters prioritize flexibility, resulting in a high number of support vectors and

complex decision boundaries primarily driven by the noise in the data. This increases training time due to quadratic scaling with data size and slows down predictions since all support vectors are used during inference. Furthermore, high-dimensional kernels exacerbate computational inefficiency.

Model 3: Aggregation model - Adaboost Decision Tree

Adaboost decision tree has the following characteristics: it combines weak classifiers with weights (α) to form a strong classifier, offering certain resistance to overfitting. It is suitable for data with linear or non-linear distributions. In scenarios without too much noise, it can adjust sample weights (u) to focus more on difficult-to-classify data points, making the model more robust to outliers. Based on these characteristics, we decided to use the Adaboost decision tree for prediction.

For parameter selection, we conducted grid experiments to determine optimal values for the following parameters: *a.* the maximum depth of decision trees (to control the power of individual classifiers and ensure moderate fitting) with `estimator__max_depth` set to [1, 2, 3, 4]; *b.* the number of decision trees (to control the aggregation power and ensure moderate fitting) with `n_estimators` set to [50, 100, 200]; *c.* learning rate (a trade-off between efficiency and stability) with `learning_rate` set to [0.01, 0.1, 1, 10].

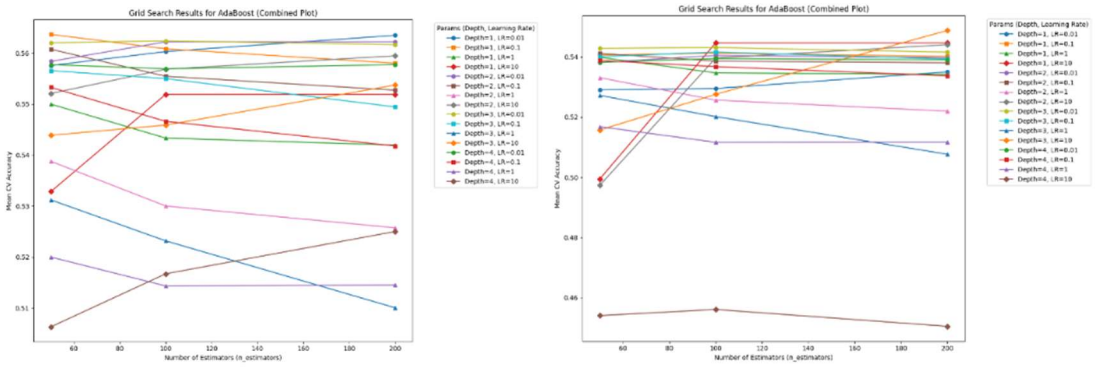

Fig.3 (5-fold cv) accuracy/f1-macro scores of adaboost decision tree

Grid experiment results of adaboost decision tree:

Scoring by "accuracy": *estimator__max_depth = 1, n_estimators = 50, learning_rate = 0.1*

Scoring by "f1-macro": *estimator__max_depth = 3, n_estimators = 200, learning_rate = 10*

Table.3 Avg Test Score of adaboost decision tree

(Avg Test Score)	Stage 1	Stage 2
By accuracy	**0.582**	**0.583**
By f1-macro	0.573	0.564

After submitting the results, we found that the best parameters selected using accuracy scoring allowed the AdaBoost Decision Tree to achieve approximately 58% accuracy in both stages, it is a relatively good result. We speculate that this is because AdaBoost Decision Tree inherently adjusts sample weights, so the sample balancing provided by F1-macro could not further benefit this model. In terms of accuracy, it may be limited by the excessive noise within the samples. AdaBoost's

characteristic of assigning more weight to hard-to-classify data points (often noise) further prevents accuracy from improving. Regarding computational efficiency, since AdaBoost requires retraining a base classifier in each iteration, the accumulated computational cost significantly increases. Additionally, the subsequent weak classifiers focus on hard-to-classify samples (often noise), further leading to a waste of computational resources.

Model 4: Aggregation model – Random Forest

Random forest, in addition to handling linear/non-linear data, has the following characteristics: it employs Bagging (random sampling with replacement + multiple models aggregation) to generate multiple subsets of data and builds independent decision trees based on them. The results are then aggregated, ensuring the model's accuracy and stability. Each tree is trained on different subsets obtained via random sampling, so noise does not overly impact the entire model. Each tree randomly uses a subset of features as candidate features, ensuring diversity among the trees, reducing computational costs, and making the model more robust and less prone to overfitting. Furthermore, during the splitting process, each decision tree selects features that maximize impurity reduction. Noise features generally contribute little to impurity reduction, making them less likely to be selected. Even if selected, they only affect a small number of nodes or trees. Therefore, random forest maintains accuracy even in high-dimensional datasets with many noise features.

Before conducting the grid experiment, we first needed to determine a reasonable number of decision trees. We aimed for a number that could provide stable results without consuming excessive computational resources. After experimental selection, we adopted `n_estimators=200` as the number of decision trees, as this number provides relatively stable results while not consuming too many computational resources.

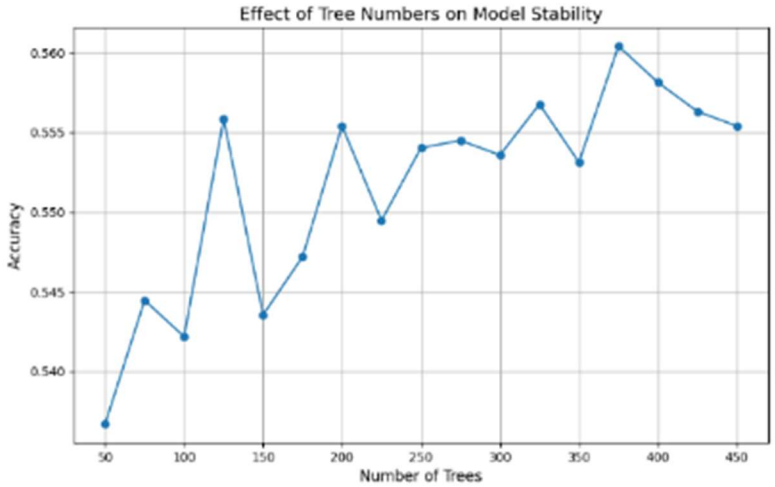

Fig.4 Accuracy to each n_estimators (to determine the proper number of decision trees)

For parameter selection, we conducted grid experiments to determine the optimal values for the following parameters: *a.* the maximum depth of decision trees to ensure the complexity of individual trees is appropriate, avoiding overfitting or underfitting, with `max_depth` set to [5, 7, 9]; *b.* the

minimum number of samples required to split a node to control the extent of tree splitting, for the same reasons, with `min_samples_split` set to [5, 10, 20]; *c.* class weights to address data imbalance, specifically including the options `class_weight` set to ["balanced", None].

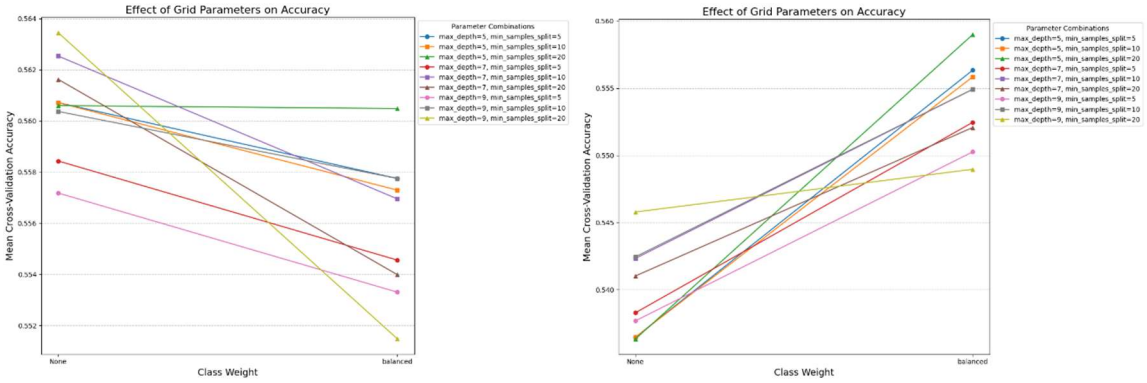

Fig.5 (5-fold cv) accuracy/f1-macro scores of random forest

Grid experiment results of random forest:

Scoring by “accuracy”: max_depth = 9, min_samples_split =20, class_weight = None
Scoring by “f1-macro”: max_depth = 5, min_samples_split =20, class_weight = ”balance”

Table.4 Avg Test Score of Random Forest

(Avg Test Score)	Stage 1	Stage 2
By accuracy	0.588	0.574
By f1-macro	**0.597**	**0.592**

After submitting the results, we found that the parameters selected using F1-macro scoring allowed the Random Forest to achieve approximately 59% accuracy in both stages, which is the highest prediction accuracy among all our models. We speculate that F1-macro scoring helped improve the accuracy of Random Forest on the imbalanced dataset. In terms of accuracy, although the dataset contains a significant amount of noise, the characteristics of Random Forest, such as random sampling and random feature selection, make it better at handling noise, resulting in the highest accuracy among all models. Regarding computational efficiency, although Random Forest is an ensemble model that requires training multiple weak classifiers, its random feature selection reduces the computational cost to some extent, which means that its computational efficiency can be ensured.

Conclusion

Based on the experimental results, we found that Logistic Regression, despite having the highest computational efficiency, struggled to handle nonlinear data due to its simple model structure, resulting in an accuracy of only around 55%. SVM achieved an accuracy of approximately 55%-56% and showed some improvement in handling imbalanced data, but its performance was significantly limited by noise, and its computational efficiency was relatively low. AdaBoost Decision Tree achieved an accuracy of about 58%, which was relatively better, but its inherent weight allocation mechanism made it overly sensitive to noise, and it had the lowest computational efficiency. In contrast, Random Forest achieved the highest accuracy at around 59%, demonstrated strong noise resistance, and had moderate computational efficiency.

Ultimately, based on the balance between accuracy and computational efficiency, we chose

Random Forest as our final recommended model. Its characteristics of random sampling and feature selection allowed it to excel in both accuracy and noise resistance. Additionally, its computational efficiency was moderate, with resource utilization being well-balanced among ensemble models. However, as an ensemble model, Random Forest still requires training multiple weak classifiers, resulting in slightly higher computational costs, though still within acceptable limits for this study.

Reference

1. Lecture slides
2. ChatGPT
3. https://hackmd.io/@johnnyasd12/BJ9bqqevD/
4. https://blog.csdn.net/qq_43190189/article/details/105778058
5. https://www.ibm.com/think/topics/random-forest
6. https://medium.com/jameslearningnote/%E8%B3%87%E6%96%99%E5%88%86%E6%9E%90-%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-%E7%AC%AC2-4%E8%AC%9B-%E8%B3%87%E6%96%99%E5%89%8D%E8%99%95%E7%90%86-missing-data-one-hot-encoding-feature-scaling-3b70a7839b4a
7. https://rpubs.com/skydome20/R-Note14-SVM-SVR
8. Package: Python-sklearn, https://scikit-learn.org/1.5/supervised_learning.html

Workload

楊佩潔：SVM

郭兆揚：Adaboost Decision Tree, Random Forest

蘇茂傑：Data Preprocessing, Logistic Regression