ML Final Report

Mr. O'clock insanely practices

December 21, 2024

1 Introduction

This final project is aim to use machine learning to predict HTMLB outcome. We have explore through many algorithms and data preprocess, and result in 0.58989 accuracy in predicting same season outcome(stage1) and 0.55473 in predicting a whole new season(stage2). We are going to break it down what have we done with every model and preprocess and what are the results.

2 Data Preprocess

Dealing with NAN

There are many NAN in all kinds of data. To deal with them, we can either fill in 0, fill in mean of other non-NAN rows, or fill in mean of the data of same team.

Team Attribute and Pitcher Attribute

We try to add 60 boolean columns: is_home_team_{team name} and is_away_team_{team name} which indicate that if home team or away team equal to team name. Same method can be used on pitcher name.

However we try it on random forest an observed that these attribute may have low importance. We guess that perhaps this kind of data is too scatter to make it important

Alternatively, we can labelize string columns, this may work in algorithm based on decision tree.

Drop std and skew

Since we think std and skew has little relationship with winning or lose. We try to drop all of those column.

The result is that there are no difference in accuracy. However, training speed is faster using this data.

3 Models

KNN

PLA

Linear Regression

Logistic Regression

SVM

Random Forest

CATboost

Catboost is an algorithm based on Gradient Boosting Tree, which is similar to XGboost and Light-

GBM. However, Catboost can deal with string data type directly, which is a very good characteristic for this problem since we think string attribute like team name or pitcher name are important for training and cannot be dropped.

In all of the following experiment, i use mean of column to replace numerical column's NAN, and use 'NAN' to replace all NANs in string column.

In the initial tries, we use parameters as following

```
iterations = 500
depth = 5
learning_rate = 0.05
loss_function = 'Logloss'
```

This is our first algorithm that reaches 0.58 in public test. However, it is still not good enough to pass benchmark.

To find a better set of parameters, we use optuna to help us find it. And to mimic to stage1 / stage2 situation we will met, we use following different strategy to cut training and validation test for optuna.

stage1 Since we are predicting the last half season given each year's first half season, we use the last 0.2 of each season's data as validation data, and other as training data, and feed these to optuna to find the best parameters.

stage2 We are predicting a whole new year. Hence we take 2023's data as validation data, and other as training data, and feed these to optuna to find best parameters.

When using optuna, we search in following range

```
params = {
    "iterations": trial.suggest_int("iterations", 100, 1000),
    "depth": trial.suggest_int("depth", 4, 10),
    "learning_rate": trial.suggest_float("learning_rate", 1e-3, 0.3, log=True),
    "l2_leaf_reg": trial.suggest_float("l2_leaf_reg", 1, 10),
    "bagging_temperature": trial.suggest_float("bagging_temperature", 0, 1),
    "random_strength": trial.suggest_float("random_strength", 0.1, 10),
    "loss_function": "Logloss",
    "cat_features": categorical_features,
    "verbose": 0,
}
```

The best parameters we found is

```
stage1_best_params = {
    'iterations': 671,
    'depth': 6,
    'learning_rate': 0.011125753493784871,
    'l2_leaf_reg': 8.31605718532868,
    'bagging_temperature': 0.16596075573738783,
    'random_strength': 7.585473310850967
}
stage2_best_params = {
    'iterations': 700,
    'depth': 6,
    'learning_rate': 0.021691769231380345,
    'l2_leaf_reg': 1.1238482611810492,
    'bagging_temperature': 0.207863787648766,
    'random_strength': 5.938894924700072
}
```

Which lead to a 0.57985 accuracy in stage1. And 0.54493 in stage2.

Blending and Bagging

4 Conclusion