Clash Royale Match Outcome Prediction

Faraa Awoyemi, Lilia Benabdallah, Ilyes Ben Younes, Lea Hadj-Said

1 Introduction

The goal of this project is to predict the result of a 1v1 Clash Royale match using only the decks of both players. We work on a supervised classification problem where the model must predict if Player 1 wins or loses the match.

We found our datasets on Kaggle. The first dataset contains real competitive matches played on the top ladder. It is not a tournament dataset. Each row corresponds to one independent match between two players. The second dataset contains information about the cards such as elixir cost, rarity, win rate and usage rate.

At first, we only use deck composition. Then, in Phase 2, we add more strategic information to improve the predictions.

2 Dataset Description

The main dataset contains 2311 matches. For each match, we have the 8 cards of Player 1, the 8 cards of Player 2, and the number of crowns scored by both players. We also add a new column called `winner` or `win_p1` which is equal to 1 if Player 1 has more crowns than Player 2, and 0 otherwise.

This dataset is a collection of matches played by different players at high level. The dataset is naturally imbalanced. Player 1 wins around 72% of the matches. This happens because Player 1 is often the player who is higher on the ladder or in a winning streak. This imbalance reflects real game conditions and should not be artificially corrected, but it must be taken into account when evaluating the models.

3 Exploratory Data Analysis

We start by checking the quality of the dataset. There are no missing values, no duplicated rows, and all crown values are valid between 0 and 3. This means that no real data cleaning is required.

We then analyze the most frequent cards for Player 1 and Player 2. Some cards are clearly over-represented. This shows that the dataset follows the current meta of the game. This also creates a natural bias, because the model will automatically learn better the cards that appear very often.

We also study the correlation between the presence of each card in Player 1's deck and the number of crowns scored by Player 1. We observe that some cards like Golem, Mini P.E.K.K.A, Zap or Bandit are positively correlated with Player 1's victory. This does not mean that these cards guarantee a win, but they statistically increase the chances of winning in this dataset.

4 Data Preprocessing

Since the card names are categorical data, we cannot use them directly in machine learning models. We apply One-Hot Encoding on the 16 card slots (8 for Player 1 and 8 for Player 2).

After encoding, each match is represented by a large binary vector with around 1300 columns.

We separate the dataset into training and testing sets using an 80% / 20% split. We use stratified splitting in order to keep the same proportion of wins and losses in both sets. Scaling is not required at this step because One-Hot Encoding only produces binary values.

5 Phase 1: Baseline Models

We test three baseline models: Logistic Regression, Decision Tree, and K-Nearest Neighbors.

The dataset is strongly imbalanced, so most models tend to predict victories much better than defeats. The KNN model almost always predicts Player 1 wins, which gives a high recall for the winning class but very poor detection of losses. This shows that KNN mostly copies what it sees and does not learn strong decision rules.

The Decision Tree reaches an accuracy around 0.62. It predicts very few defeats correctly and misses most losses. This model follows the global bias of the dataset and is not reliable in this context.

Logistic Regression reaches an accuracy around 0.63 to 0.67 depending on the parameters. It is the only baseline model that really learns useful patterns inside the decks. It does not only follow the dataset bias but also uses the card combinations to improve predictions.

6 Hyperparameter Tuning

We use GridSearchCV to test several combinations of hyperparameters automatically. This allows us to select the best parameters for each model.

After tuning, the best cross-validated scores are around 0.72 to 0.75. However, the test accuracy remains around 0.69. This shows that tuning helps slightly, but deck composition alone is still not enough to reach high performance.

7 Phase 2: Dataset Enrichment

In Phase 2, we enrich the dataset using the second dataset from Kaggle. This dataset contains information about each card such as elixir cost, rarity, win rate and usage rate.

We normalize the card names in both datasets and merge the information for each card slot of both players. After merging, some values are missing because not all cards exist in the second dataset. We fill these missing numerical values using the mean of each column.

Then we remove all useless columns such as card names and normalized names.

8 Advanced Feature Engineering

After merging, we create new aggregated features for each deck: the average elixir cost, the average card win rate, the average usage rate, and the average rarity for both players.

We also compute the number of identical cards between both decks, called deck overlap.

These features represent the general strength and style of each deck instead of focusing only on individual card names.

9 Phase 2 Models

We train again Logistic Regression, Decision Tree and KNN using these new features. The performances increase slightly compared to Phase 1.

We then test more advanced models: Random Forest, XGBoost, and CatBoost. These models reach better results. CatBoost gives the best performance with an accuracy around 0.73 and a ROC-AUC around 0.60.

We also test dimensionality reduction with PCA. With only 8 principal components, Logistic Regression still reaches an accuracy around 0.72. This shows that most of the information is contained in a small number of strategic features.

Finally, we use a voting ensemble combining Logistic Regression, Random Forest and XGBoost. This ensemble reaches an accuracy around 0.724. The gain remains limited.

10 Discussion

Even with advanced models and better features, the overall performance stays limited. This is mainly due to missing gameplay information. The dataset does not contain card levels, player skill, elixir management, timing, or placement decisions. Two identical decks can produce very different results depending on the player.

The dataset imbalance also makes the prediction of defeats difficult. The models naturally learn that Player 1 wins most of the time.

11 Conclusion

In this project, we showed that it is possible to predict the result of a Clash Royale match using only the deck composition. We started with simple models and then added more strategic features and advanced algorithms.

However, the performance remains limited by the lack of real gameplay information. Future work could include real-time match data, player levels, and elixir usage in order to significantly improve prediction accuracy.