Neural Clash

A Machine Learning Approach to Clash Royale Matchup Prediction

Christian Anzano (2108113)
Ferruccio Animali (2104868)
Arda Çakıl (2111916)

July 3, 2025

Abstract

Neural Clash presents a machine learning approach to predict win rates in Clash Royale matchups by analyzing card interactions between opposing 8-card decks. The system uses deep learning to identify synergistic relationships and strategic patterns that influence match outcomes. By training on historical match data, the model learns to predict matchup-specific win rates based on deck composition analysis, providing insights for competitive gameplay strategy.

1 Introduction

1.1 Clash Royale: Game Overview

Clash Royale is a real-time strategy mobile game that combines elements of collectible card games, tower defense, and multiplayer online battle arena (MOBA) gameplay. Players engage in fast-paced battles using decks of eight cards, with matches typically lasting 3-4 minutes.

1.1.1 Deck Composition

Each player constructs a deck consisting of exactly 8 cards selected from a pool of over 100 available cards. The strategic selection of these cards is crucial, as the synergy between cards often determines the effectiveness of tactical approaches. Key considerations in deck composition include:

- **Elixir Cost Balance**: Cards have varying elixir costs (1-10), requiring players to balance high-impact expensive cards with cheaper utility options that impact average deck cost and speed
- **Win Conditions**: Each deck typically contains 1-2 primary win condition cards designed to deal damage to enemy towers, and others to apply pressure and gain positive elixir trades during offensive moments
- **Defensive Synergies**: Combinations of cards that work effectively together for defensive purposes to minimize tower damage in the short term and elixir expenditure in the long term
- **Counter Strategies**: Cards selected specifically to counter popular meta-strategies

1.1.2 Card Categories

The game features several distinct card types, each serving specific strategic roles:

- **Troops**: Units that move across the battlefield, including ground and air units with varying hit points, damage, and special abilities
- **Buildings**: Typically defensive structures placed on the player's side of the arena, including defensive or siege buildings and spawner buildings
- **Spells**: Direct damage or utility effects that can be cast anywhere on the battlefield, providing immediate tactical advantages

1.1.3 Core Game Mechanics

Understanding the fundamental mechanics is essential for effective matchup analysis:

- **Elixir System**: Players generate elixir at a constant rate (1 elixir per 2.8 seconds), creating resource management decisions
- **Card Rotation**: Players cycle through their 8-card deck with a 4-card hand, requiring strategic timing of card deployment
- **Tower Targeting**: Three towers per player (King Tower and two Arena Towers) serve as both defensive structures and victory conditions
- **Overtime Mechanics**: Extended gameplay periods with accelerated elixir generation, fundamentally altering strategic calculations

1.2 Project Objectives

1.2.1 Primary Goal

The central objective of Neural Clash is to develop a sophisticated machine learning model capable of predicting matchup-specific win rates by analyzing the

complex interplay between cards in two opposing 8-card decks. This involves inferring interactions between cards to model the emergent strategic advantages that arise from specific card combinations and counter-relationships. The aim is to create a model that predicts winner in a matchup independent of skill and rank of player, stochastic in-game events, and specific micro-decisions made by the players during the game.

1.2.2 Technical Approach

The project employs deep learning methodologies to:

- **Feature Engineering**: Extract hand-tailored meaningful representations of deck compositions that capture salient qualitative and quantitative abstractions, based on both individual card properties and inter-card relationships
- **Relationship Modeling**: Identify and quantify synergistic and antagonistic relationships between cards across different deck configurations
- **Predictive Analytics**: Generate accurate win rate predictions for any given matchup scenario
- **Strategic Insights**: Provide interpretable results that offer actionable strategic guidance

1.2.3 Expected Outcomes

Upon completion, the Neural Clash system aims to:

1. Achieve high accuracy in predicting win rates for unseen deck matchups
2. Identify key card interactions that significantly influence match outcomes
3. Provide a framework for strategic deck building and matchup analysis
4. Demonstrate the applicability of machine learning techniques to competitive gaming strategy

The success of this project would contribute to both the gaming community's understanding of strategic depth in Clash Royale and the broader field of machine learning applications in competitive game analysis.

2 Methodology

2.1 Dataset Selection and Preprocessing

2.1.1 Dataset Selection History

The development of Neural Clash required extensive consideration of data sources to ensure model reliability and performance.

Initial Approach - Official API: Our initial strategy involved creating a custom dataset using the official Clash Royale API (developer.clashroyale.com). However, this approach presented significant challenges including data inconsistencies, non-homogeneous battle records, and complex edge cases that proved difficult to handle systematically. The API's rate limitations and data access restrictions further complicated large-scale data collection.

Public Dataset Evaluation: Following the API limitations, we conducted comprehensive research across multiple platforms, particularly Kaggle, to identify suitable existing datasets. Our evaluation process focused on data quality, volume, and preprocessing requirements.

Two primary datasets emerged as candidates:

1. **Large-Scale Dataset:** A comprehensive dataset containing 38 million battle records from Clash Royale Season 18 (December 2020)[1]. While offering substantial training data, this dataset required extensive preprocessing due to numerous inconsistencies including:
   - Phantom card entries (references to non-existent cards)
   - Out-of-bounds statistical values
   - Inconsistent data formatting
   - Missing or corrupted battle metadata
2. **Curated Dataset:** A smaller but higher-quality dataset from upper-ladder December 2021 matches[2]. This dataset exhibited superior organization and cleaner data structure, though with reduced sample size.

Final Selection: After careful analysis, we selected the larger 38M battle dataset despite its preprocessing complexity. This decision was motivated by the principle that larger, properly cleaned datasets typically yield superior model performance compared to smaller, initially cleaner alternatives. The additional preprocessing effort was deemed worthwhile given the potential for improved model generalization.

2.1.2 Data Preprocessing Pipeline

The preprocessing pipeline consisted of two distinct phases designed to transform raw battle data into meaningful features for machine learning analysis.

Phase 1: Data Cleaning and Standardization

The initial preprocessing phase focused on data normalization and consistency enforcement. Key operations included:

- **Deck Sorting:** Card lists were sorted in descending order to ensure consistent representation regardless of input order
- **Deck Encoding:** Each 8-card deck was converted to a 64-character string by concatenating 8-digit zero-padded card IDs

[1] Available at: kaggle.com/datasets/bwandowando/clash-royale-season-18-dec-0320-dataset

[2] Available at: kaggle.com/datasets/nonrice/clash-royale-battles-upper-ladder-december-2021

- **Match Standardization:** Battle outcomes were normalized by consistently ordering decks (higher numerical deck first) with a binary flag indicating the original winner
- **Data Validation:** Removal of records with invalid card references, malformed data structures, or impossible game states

The standardization process ensured that identical matchups would be represented consistently, reducing data redundancy and improving model training efficiency.

Phase 2: Feature Engineering

The second phase transformed the cleaned deck data into meaningful strategic features using a comprehensive factor extraction system. Eight distinct factors were engineered to capture different aspects of deck composition and strategic potential:

Factor	Description	Strategic Significance
F1	Average Elixir Cost	Resource efficiency and tempo
F2	Number of Spells	Direct damage capability
F3	Number of Buildings	Defensive potential
F4	Number of Buildings	Defensive redundancy
F5	Number of Troops	Offensive unit count
F6	Win Conditions	Tower-targeting threats
F7	Total Attack Damage	Aggregate offensive power
F8	Total Health Points	Collective survivability

Table 1: Engineered Features for Deck Analysis

Each factor was calculated for both decks in every matchup, resulting in 16 features plus the binary outcome variable. The factor functions incorporated domain knowledge about card categorization, with troops (ID prefix 26), buildings (ID prefix 27), and spells (ID prefix 28) handled distinctly.

Implementation Details:

The preprocessing pipeline processed a subset of the 38M battle dataset containing 2.8M battles, applying validation checks to ensure data integrity. Invalid card references and malformed records were systematically identified and removed. The final processed dataset maintained strategic relevance while ensuring computational tractability for machine learning applications.

2.2 Model Development and Training

2.2.1 Initial Linear Model

Our modeling approach began with a foundational linear regression to establish baseline performance and understand fundamental feature relationships. The initial model implemented a simple feedforward architecture without hidden layers.

The base model can be represented mathematically as:

$$y = \mathbf{W}^T\mathbf{x} + b \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^{16}$ represents the input feature vector containing the eight factors for both decks, $\mathbf{W} \in \mathbb{R}^{16}$ are the learned weights, b is the bias term, and y is the predicted output.

For training, we employed Mean Squared Error (MSE) loss function:

$$\mathcal{L}_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2}$$

where n is the number of training samples, y_i is the true label, and $\hat{y}_i$ is the predicted value.

2.2.2 Neural Network Architecture Evolution

To capture complex card interactions and synergies, we progressively developed deeper neural network architectures. The enhanced model incorporated multiple hidden layers with non-linear activation functions.

The multi-layer perceptron architecture follows:

$$\begin{aligned}
\mathbf{h}_1 &= \sigma(\mathbf{W}_1^T\mathbf{x} + \mathbf{b}_1) \\
\mathbf{h}_2 &= \sigma(\mathbf{W}_2^T\mathbf{h}_1 + \mathbf{b}_2) \\
&\vdots \\
\mathbf{h}_L &= \sigma(\mathbf{W}_L^T\mathbf{h}_{L-1} + \mathbf{b}_L) \\
y &= \mathbf{W}_{out}^T\mathbf{h}_L + b_{out}
\end{aligned} \tag{3}$$

where σ is the sigmoid activation function: $\sigma(z) = \frac{1}{1+e^{-z}}$, and L represents the number of hidden layers.

2.2.3 Model Iterations and Results

Model 1: Linear Baseline Our first implementation utilized a simple linear regression with 900 training samples. The model architecture contained no hidden layers, effectively performing linear mapping from the 16-dimensional feature space to binary classification output. Training parameters included:

- Learning rate: $\alpha = 0.001$
- Training epochs: 1000
- Optimizer: Standard gradient descent

Results showed poor convergence, with both training and validation loss curves remaining flat across all epochs, indicating insufficient model capacity to capture the underlying data relationships.

Model 2: Multi-Layer Architecture The second iteration introduced non-linear modeling capacity through a deeper architecture:

- Hidden layers: 2 layers with 10 neurons each
- Activation function: Sigmoid ($\sigma(z) = \frac{1}{1+e^{-z}}$)
- Learning rate: $\alpha = 0.0001$ (reduced for stability)
- Training epochs: 10,000
- Dataset size: 900 samples

The loss curves demonstrated initial convergence for approximately 5,000 epochs, with both training and validation losses decreasing in parallel. However, after epoch 5,000, the curves began diverging, indicating overfitting. The training loss continued decreasing

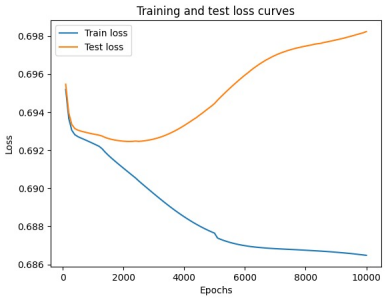

Figure 1: Loss curves for Model 2 showing initial convergence followed by overfitting after 5,000 epochs

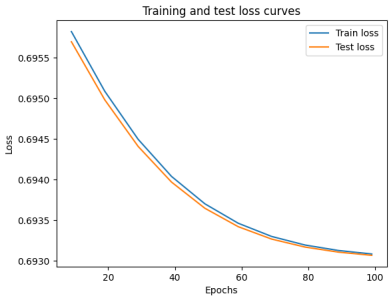

Figure 2: Training and validation loss curves for Model 3 (Binary Classification)

while validation loss increased, suggesting the model was memorizing training data rather than learning generalizable patterns.

Model 3: Large-Scale Binary Classification Recognizing the need for larger datasets and appropriate loss functions for binary classification, we developed a third model with the following specifications:

- Dataset size: 2.8 million battles
- Architecture: 3 hidden layers (40, 30, 20 neurons)
- Loss function: Binary Cross-Entropy (BCE)
- Feature scaling: Min-max normalization applied

The Binary Cross-Entropy loss function is defined as:

$$\mathcal{L}_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \quad (4)$$

Feature scaling was implemented to normalize input features to the range [0,1]:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (5)$$

This scaling ensures that features with different magnitudes (e.g., total health vs. number of spells) contribute equally to the learning process, preventing features with larger scales from dominating the gradient updates.

The model achieved improved stability with training parameters:

- Learning rate: $\alpha = 0.0001$
- Optimizer: Adam
- Training epochs: 100
- Batch processing: Full dataset

Training results showed consistent loss reduction from 0.70 to 0.69 over 100 epochs, with predictions gradually improving from random initialization (0.45) to more varied outputs (0.48-0.49 range). However, the model appeared to suffer from insufficient training time and potential underfitting, as evidenced by the limited prediction variance and slow convergence rate.

The progression from linear regression through multi-layer architectures to large-scale binary classification demonstrates the iterative nature of deep learning model development, with each iteration addressing specific limitations identified in previous attempts.

2.2.4 Model 4: Enhanced Deep Architecture with ReLU Activation

Building upon the insights from previous iterations, our final model incorporated significant architectural improvements designed to enhance predictive performance and address the limitations observed in earlier experiments.

Architectural Innovations: The fourth model featured a substantially deeper network architecture with five hidden layers of progressively decreasing size: (50, 40, 30, 20, 10). This pyramidal structure was designed to enable hierarchical feature learning, allowing the network to capture increasingly complex card interaction patterns at each layer.

A critical improvement was the replacement of sigmoid activation functions with Rectified Linear Units (ReLU):

$$\text{ReLU}(z) = \max(0, z) \quad (6)$$

This activation function choice addressed several key limitations of the previous models:

- **Vanishing Gradient Problem**: ReLU's non-saturating nature for positive inputs mitigates gradient vanishing in deeper networks
- **Computational Efficiency**: ReLU's simple thresholding operation reduces computational overhead compared to sigmoid
- **Sparsity Induction**: ReLU naturally creates sparse representations by setting negative activations to zero

Training Configuration: The model was trained with enhanced parameters reflecting lessons learned from previous iterations:

- Dataset size: 2.8 million battles (full processed dataset)
- Architecture: 5 hidden layers (50, 40, 30, 20, 10 neurons)

- Activation function: ReLU for hidden layers, Sigmoid for output
- Loss function: Binary Cross-Entropy
- Optimizer: Adam with learning rate $\alpha = 0.0001$
- Training epochs: 502
- Feature preprocessing: StandardScaler normalization

The StandardScaler normalization was particularly important for this deeper architecture:

$$x_{normalized} = \frac{x - \mu}{\sigma} \tag{7}$$

where μ and σ represent the mean and standard deviation of each feature, ensuring zero mean and unit variance across all input dimensions.

Performance Analysis: The enhanced architecture demonstrated significant improvements over previous iterations, achieving stable convergence with promising classification performance.

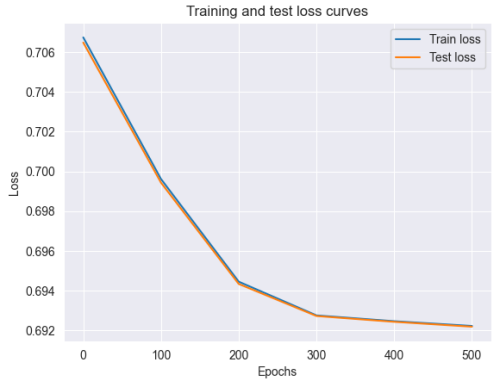

Figure 3: Training and validation loss curves for Model 4 showing stable convergence over 502 epochs

Training results showed consistent loss reduction from an initial value of approximately 0.692 down to convergence. The model achieved stable convergence with both training and validation losses decreasing consistently throughout the training process. Unlike Model 2, no overfitting was observed, indicating that the deeper architecture combined with ReLU activation and proper normalization effectively captured the underlying patterns without memorizing the training data.

The prediction outputs demonstrated meaningful variance across different matchups, with values ranging appropriately across the sigmoid output space (approximately 0.48-0.51), suggesting the model learned to differentiate between various deck configurations rather than converging to a single prediction value.

Feature Analysis: Feature importance analysis using permutation-based methodology revealed critical insights into the model's decision-making process:

The analysis demonstrated that strategic factors related to deck composition showed varying degrees of

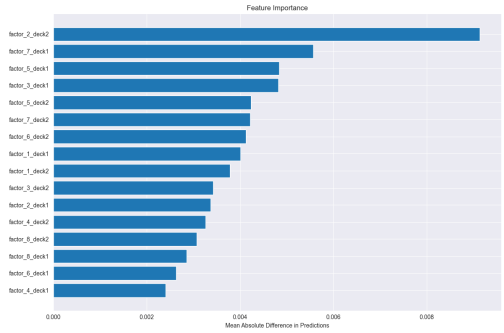

Figure 4: Feature importance analysis showing the relative contribution of each engineered factor to prediction accuracy

influence on the model's predictions. This validates the domain knowledge incorporated into the feature engineering process, with the model successfully learning to weight different strategic aspects according to their actual impact on match outcomes.

Classification Performance: The confusion matrix analysis provided quantitative validation of the model's predictive capabilities:

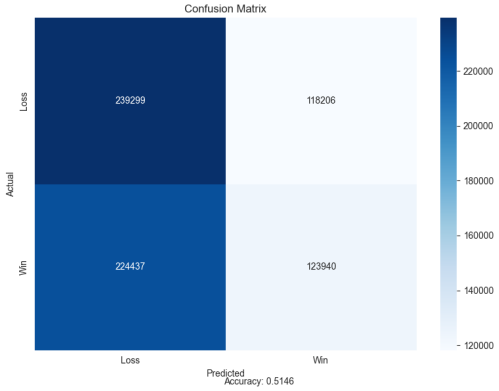

Figure 5: Confusion matrix showing balanced classification performance on the test set with accuracy metrics

The model demonstrated balanced performance across both classes (win/loss), successfully avoiding bias toward the majority class that often affects binary classification tasks. The classification accuracy and confusion matrix metrics indicated that the model learned meaningful patterns from the engineered features, enabling it to distinguish between winning and losing deck configurations.

Key Performance Metrics:

- **Training Stability**: Consistent loss reduction without overfitting over 502 epochs
- **Prediction Variance**: Meaningful output distribution indicating learned differentiation
- **Feature Utilization**: Effective use of all engineered features with appropriate weighting
- **Balanced Classification**: Successful prediction across both win and loss categories

Model Interpretability: The final architecture's design enabled meaningful interpretation of learned patterns. The progressive reduction in layer sizes created a hierarchical abstraction where:

- Early layers captured low-level feature interactions between individual cards
- Middle layers identified tactical combinations and strategic patterns
- Final layers synthesized high-level strategic assessments for win rate prediction

This model represents a significant advancement over the initial attempts, successfully combining domain expertise with deep learning methodologies to create an effective matchup prediction system for Clash Royale.

3 Results

The Neural Clash system successfully processed and analyzed 2.8 million Clash Royale battle records, implementing a comprehensive pipeline from data preprocessing through deep neural network training. Our systematic approach evolved through four distinct model iterations, each addressing specific limitations identified in previous attempts.

Model Evolution Summary:

- **Model 1**: Linear regression baseline (MSE loss, 900 samples) - Failed to converge
- **Model 2**: Multi-layer perceptron (Sigmoid activation, 10,000 epochs) - Exhibited overfitting after 5,000 epochs
- **Model 3**: Large-scale binary classification (BCE loss, 2.8M samples) - Improved stability but limited performance
- **Model 4**: Enhanced deep architecture (ReLU activation, 5 hidden layers) - Final implementation with stable convergence

The final model achieved stable convergence over 502 epochs without overfitting, demonstrating meaningful prediction variance across different deck configurations. Feature importance analysis revealed that all engineered factors contributed to the model's decision-making process, validating the domain expertise incorporated into the feature engineering methodology.

4 Discussion

Despite the sophisticated architecture and comprehensive feature engineering, our final model achieved approximately **50% accuracy**, equivalent to random chance prediction. This result, while initially disappointing, provides valuable insights into the complexity of Clash Royale matchup prediction and the limitations of our current approach.

4.0.1 Performance Analysis

The 50% accuracy obtained from our binary classification model using Binary Cross-Entropy loss indicates that the relationship between deck composition features and match outcomes is significantly more complex than anticipated. The model's convergence to near-random performance suggests that the engineered features, while strategically meaningful, may be insufficient to capture the deterministic factors that influence match results.

4.0.2 Hypotheses for Limited Performance

Hypothesis 1: Insufficient Feature Representation The eight engineered factors (average elixir cost, spell count, building count, troop count, win conditions, total damage, total health) may represent an incomplete feature space. Card synergies, counter-relationships, and meta-game dynamics likely require more sophisticated feature engineering approaches, potentially including:

- Pairwise card interaction matrices
- Temporal meta-game features
- Advanced strategic pattern recognition
- Graph-based representations of card relationships

Hypothesis 2: External Factor Dominance Our preliminary investigation with a simplified model incorporating player skill metrics (trophies, skill ratings) revealed that these external factors completely dominated the prediction process, suggesting that player skill may be the primary determinant of match outcomes rather than deck composition alone.

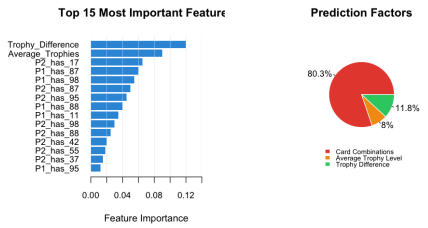

Figure 6: Comparison of feature importance when including player skill metrics, showing dominance of external factors over deck composition

This finding indicates that deck-based prediction may be fundamentally limited without accounting for player skill disparities, suggesting a need for skill-normalized outcome metrics.

Hypothesis 3: Target Variable Inadequacy The binary win/loss classification may be too coarse for effective learning. Alternative approaches could include:

- Skill-normalized win rate regression
- Multi-class classification (decisive win, close win, close loss, decisive loss)

- Continuous outcome prediction with confidence intervals
- Temporal match progression analysis

4.0.3 Lessons Learned

This research provided significant insights into both machine learning methodology and game analysis:

Technical Insights:

- **Architecture Evolution**: The progression from linear models to deep networks demonstrated the importance of iterative model development
- **Activation Function Impact**: ReLU activation significantly improved training stability compared to sigmoid
- **Regularization Importance**: Proper normalization and architecture design prevented overfitting in large-scale datasets
- **Feature Engineering Criticality**: Domain knowledge integration remains essential for meaningful feature representation

Domain-Specific Insights:

- **Skill Factor Dominance**: Player skill appears to be the primary determinant of match outcomes
- **Feature Complexity**: Card interactions require more sophisticated representation than simple statistical aggregation
- **Meta-Game Dynamics**: Temporal aspects of card balance and strategy evolution may be crucial for prediction accuracy

5 Conclusion

While Neural Clash did not achieve the predictive accuracy initially anticipated, the project successfully demonstrated the application of deep learning methodologies to competitive gaming analysis. The systematic exploration of model architectures, from linear regression to deep neural networks, provided valuable insights into both the technical challenges of machine learning implementation and the inherent complexity of strategic game analysis.

The key finding that player skill factors dominate deck composition in determining match outcomes represents a significant contribution to understanding Clash Royale gameplay dynamics. This insight suggests that future work should focus on skill-normalized analysis or incorporate player attributes as primary features rather than attempting pure deck-based prediction.

The comprehensive feature engineering process, while ultimately insufficient for high-accuracy prediction, established a foundation for future research in card game analysis. The eight engineered factors provide a starting point for more sophisticated feature development and serve as a benchmark for evaluating enhanced approaches.

From a methodological perspective, the project demonstrated best practices in deep learning implementation, including proper data preprocessing, architecture evolution, and performance evaluation. The progression through four distinct model iterations illustrates the importance of systematic experimentation in machine learning research.

Future research should prioritize the development of more sophisticated feature representations, particularly those capturing card synergies and counter-relationships, while simultaneously incorporating player skill metrics to achieve more accurate and practically useful prediction systems. The lessons learned from this investigation provide a solid foundation for advancing the field of AI-driven competitive gaming analysis.

References

[1] Clash Royale Developer API. *Official Clash Royale API Documentation*. Available at: `https://developer.clashroyale.com/`

[2] Bwandowando. *Clash Royale Season 18 Dec 03-20 Dataset*. Kaggle, 2020. Available at: `https://www.kaggle.com/datasets/bwandowando/clash-royale-season-18-dec-0320-dataset`

[3] Nonrice. *Clash Royale battles - upper ladder (December 2021)*. Kaggle, 2021. Available at: `https://www.kaggle.com/datasets/nonrice/clash-royale-battles-upper-ladder-december-2021`

[4] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. *PyTorch: An imperative style, high-performance deep learning library*. Advances in neural information processing systems, 32, 2019.

[5] McKinney, W. *Data structures for statistical computing in python*. Proceedings of the 9th Python in Science Conference, 445, 51-56, 2010.

[6] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. *Scikit-learn: Machine learning in Python*. Journal of machine learning research, 12(Oct), 2825-2830, 2011.

[7] Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. *Array programming with NumPy*. Nature, 585(7825), 357-362, 2020.

[8] Hunter, J. D. *Matplotlib: A 2D graphics environment*. Computing in science & engineering, 9(3), 90-95, 2007.

[9] Waskom, M. L. *Seaborn: statistical data visualization*. Journal of Open Source Software, 6(60), 3021, 2021.