Kien Tran & Alexis Ortiz
CPSCI 307
Professor Chen
Oct 22, 2024

Introduction

In this project, we aim to develop a deep learning model capable of predicting the outcome of boxing matches based on the characteristics of the fighters and the match details. Predicting the winner of a match is a challenging task that involves analyzing multiple factors, such as a fighter's physical attributes, fight history, and past performance. By leveraging machine learning techniques, we aim to identify patterns in these attributes that can lead to accurate predictions of match outcomes.

The primary objective of this phase of the project is to collect and preprocess the data necessary for training our model. Accurate data collection and thorough preprocessing are critical steps in building a successful predictive model, as they ensure that the input data is clean, structured, and ready for analysis. Our goal is to compile a dataset containing information about previous boxing matches, including both fighter-specific attributes (such as age, height, reach, and fight history) and match-specific details (such as rounds fought and fight outcome). This data will form the foundation for training the model to predict future match results.

Data Collection

For the purpose of this project, we gathered data on previous boxing matches from two primary sources: web scraping and Kaggle.

1. Web Scraping: We employed web scraping techniques to extract relevant fight information from various sports and boxing websites. These sites provided detailed data on professional boxing matches, including both fighter-specific attributes and match

outcomes. The process involved writing custom scripts to automatically collect data, ensuring consistency and accuracy during the extraction phase.

2. Kaggle: Additionally, we utilized a Kaggle dataset that contained detailed fight statistics and outcomes. Kaggle provided a structured format for the data, which allowed us to integrate it easily into our training and validation datasets. By combining both sources, we were able to gather a comprehensive dataset that includes detailed information on each fight and boxer involved.

Features Collected

For each fight, we collected 17 features, which represent both fighter-specific and match-specific details. Below is a breakdown of each feature and its significance:

_boxer: The name of the first boxer (fighter A) involved in the match.
s_boxer: The name of the second boxer (fighter B) involved in the match.
f_boxer_result: The result of the fight for the first boxer (win/loss).
f_boxer_age: The age of the first boxer at the time of the fight.
f_boxer_height: The height of the first boxer (in centimeters or inches, depending on the source).
f_boxer_reach: The reach of the first boxer, indicating arm length, which often influences fighting style and performance.
f_boxer_won: The total number of fights won by the first boxer prior to this match.
f_boxer_lost: The total number of fights lost by the first boxer prior to this match.
f_boxer_KOs: The number of knockouts delivered by the first boxer before this match.
s_boxer_age: The age of the second boxer at the time of the fight.
s_boxer_height: The height of the second boxer.
s_boxer_reach: The reach of the second boxer.
s_boxer_won: The total number of fights won by the second boxer prior to this match.
s_boxer_lost: The total number of fights lost by the second boxer prior to this match.
s_boxer_KOs: The number of knockouts delivered by the second boxer before this match.
matchRounds: The total number of rounds scheduled for the match, typically ranging from 4 to 12 rounds.

fightEnd: The method by which the fight ended, such as a knockout (KO), technical knockout (TKO), or decision by judges.
winner: The winner of the fight, indicating whether the first or second boxer was victorious.
These features provide a comprehensive view of both fighters' physical characteristics, fight history, and how the match ended. This level of detail is essential for training our predictive model.

Training and Validation Set Overview

We compiled a total of 120 examples from both web scraping and Kaggle for the purpose of model training and validation.

1. Training Set: The training set consists of 100 examples of past boxing matches. These examples will be used to teach the model how to identify patterns in fighter attributes and match outcomes, allowing it to predict future fight results.
2. Validation Set: The validation set consists of 20 examples, which will be used to evaluate the performance of the model after training. This dataset serves as a benchmark for how well the model generalizes to unseen data.

The features and data collected from these sources will be used in the subsequent stages of data preprocessing and model training to build an accurate boxing match prediction system.

Data Preprocessing

The raw data collected from web scraping and Kaggle required several preprocessing steps to ensure it was suitable for use in training our deep learning model. These steps involved handling missing values, cleaning and formatting the data, normalizing numeric features, and encoding categorical data.

Handling Missing Data

One of the primary issues with the dataset was the presence of missing or incomplete values. Many cells, especially in fighter-specific attributes like reach or height, contained the label "not

listed." It is essential to address these missing values to avoid introducing bias or errors during model training. The following strategies were employed:

1. Numeric Features: For numeric features such as age, height, reach, wins, losses, and KOs, missing values were replaced with more meaningful placeholders:
    a. NaN (Not a Number): If the feature could reasonably be missing (e.g., a boxer not having a listed reach), we replaced missing values with NaN. This allows certain models, like decision trees, to handle missing data more effectively.
    b. 0: In some cases, such as the number of knockouts (KOs), missing values were assumed to be zero, especially when the fighter is new or less experienced.
2. Categorical Features: Categorical features such as f_boxer_result and fightEnd were handled differently. If a value was missing or "not listed," we applied specific rules:
    a. For f_boxer_result, we assigned a missing result as "undetermined," which could be treated as its own category.
    b. For fightEnd, we also treated missing values as "unknown" and encoded it as a separate category.

This method allowed us to retain as much of the original data as possible while making it suitable for model training.

Data Cleaning

After handling missing values, the next step was to clean and format the data. This included:

1. Data Type Conversion: Ensuring all numeric features were appropriately cast as integers or floats. Features like age, height, and matchRounds were converted to integers, while win/loss records and reach were converted to floats where necessary.
2. Inconsistent Values: We also identified and resolved inconsistencies in the data, such as negative ages or heights outside a reasonable range for human boxers, which were either corrected or marked as outliers for further analysis.

Normalization and Standardization

For certain numeric features, normalization or standardization was required to ensure that the scale of the data did not negatively impact the performance of the deep learning model.

1. Normalization: We normalized features such as height and reach to bring all values into a range between 0 and 1. This is particularly important for models that rely on gradient-based optimization, as it helps prevent features with larger ranges from disproportionately influencing the model.
2. Standardization: For other features like age and win/loss records, we applied standardization by subtracting the mean and dividing by the standard deviation. This ensures that these features follow a normal distribution, which can help certain machine learning algorithms converge faster during training.

Handling Categorical Data

Many of the features in the dataset were categorical, and they needed to be encoded numerically for the model to interpret them. Below are the key categorical features and how they were handled:

1. f_boxer_result: This feature indicates whether the first boxer won or lost the match. Since this is a binary categorical variable, we used label encoding to map "won" to 1 and "loss" to 0.
2. fightEnd: This feature captures how the fight ended, with possible values such as "KO/TKO" (knockout or technical knockout), "UD" (unanimous decision), "RTD" (retirement), and "DQ" (disqualification). Since this is a multi-class categorical variable, we applied one-hot encoding. Each unique value was transformed into its own binary feature, allowing the model to distinguish between different fight endings without assigning arbitrary numerical values.

For example:

"KO/TKO" → [1, 0, 0, 0]

"UD" → [0, 1, 0, 0]

"RTD" → [0, 0, 1, 0]

"DQ" → [0, 0, 0, 1]

winner: This feature indicates the winner of the match, either the first or second boxer. We encoded it as a binary variable: 1 for the first boxer and 0 for the second boxer. This will serve as the target variable for our prediction task.

Feature Scaling

After handling missing values, cleaning the data, and encoding categorical features, the next step was to scale all features to ensure uniformity. We used Min-Max scaling for numeric features like age, height, and reach, mapping their values to a 0-1 range. This ensured that no single feature would dominate the training process due to differences in scale.

Train-Validation Split

Lastly, we maintained our original separation of 100 examples in the training set and 20 examples in the validation set. The validation set remained untouched during the preprocessing phase to serve as an unbiased benchmark for evaluating the model's performance.

References

Data Source

- Spiridonov, Kiril. "Boxing Dataset." Kaggle, https://www.kaggle.com/datasets/kirilspiridonov/boxing. Accessed [Oct 22, 2024].

Tools and Libraries

- Pandas: Used for data manipulation and preprocessing, including handling missing data, cleaning, and feature engineering.
- NumPy: Utilized for numerical operations and array manipulations, particularly during the normalization and standardization of features.
- Scikit-learn: Used for handling categorical data (label encoding and one-hot encoding) and for applying scaling methods like Min-Max scaling.