Collegiate basketball tournaments, a complete systemic analysis

Systems Analysis and Design

Workshop N°2

Nicolás Romero Rodríguez - code: 20222020023

Xiomara Salome Arias Arias - code: 20222020028

Carlos Andres Celis Herrera - code: 20222020051

Universidad Distrital Francisco José de Caldas

Bogotá, Colombia

Collegiate basketball tournaments, a complete systemic analysis

Systems Analysis and Design

Workshop N°2

Workshop 1 findings.

There are 64 teams playing in this tournament, once a team loses a match they are instantly eliminated from the tournament. In order to generate a winner for any given match the input also contains extensive data about the team's performance. Including statistics on both teamwide and individual level. Due to the amount of data provided it's necessary to set parameters to identify which data is useful and which data is noise. Once the relevant data is identified it's also necessary to determine which are the factors that contribute the most to determining a match's result.

Match's results can also be affected heavily by their environment, including factors such as player's personal struggles, logistics affecting the tournament's schedules, teams having long travel distances from match to match or referee bias. This factors are impossible to foresee, and in order to generate the predictions many of these factors have to be ignored.

- **Constraints** Rules are distinct for both men's and women's divisions which can cause certain aspects of a team's performance to affect the match's result differently.

  Not all of the info that is fed into the model has the same level of detail, as it's specified that there are some timeframes where data for some teams is not complete, though the information of the most recent years is specified to be complete, this difference in data detail has to be taken into account at the moment of setting parameters for the prediction.

  The tournament takes place over three weekends, which gives little rest for the players between matches, alongside tight travelling windows for teams which can impact player's performance and these metrics aren't really possible to be taken into account in this model.

- **Data Characteristics** The performance information provided for the model is given in various .csv files, each containing a different category in data, competing a total of 37 . csv files that contain information dating from 1984 until 2024 and includes every team's history of seeds, offensive/defensive stats on teamwide and individual level and locations where games were played.

- **Chaos-Theory factors**

  - Personal performance of a player can be affected by a sudden lesion or personal circumstances that are unforeseeable with the data provided and may occur suddenly, impacting the match result.
  - Wrong calls or referee bias has a direct impact on a match's statistics and has a large impact on how a match's result plays out.
  - Unexpected delays on a game's schedule can affect the match dynamic and impact the result of a game without any way of foreseeing such events.
  - Even if a team has a consistently bad or remarkable performance across these tournaments, there's still a chance that there is a sudden change in the teams' performance as a randomness factor of real-world performance that causes unpredictability despite having structured predictions
  - Due to the large amount of data given, and the synergy component of the teams' dynamic, small differences in player's feats can affect the prediction's rate disproportionately.

System Requirements

- The system has to be able to import the information given by the competition organizers, including team statistics, seed history and locations. Not only including teams participating in the NCAA, but also including every eligible team for said tournament, and therefore has to be able to contain and process this volume of information without breaking. With 64 teams, there are 2016 possible

matchups, and taking into account every possible eligible team for the tournament (360) there are 64,620 possible matchups.

- The system must include a prediction engine that generates match probabilities based on the inputs given. These probabilities must be expressed as floating point numbers between 0 and 1.
- The system has to generate predictions of hypothetical matchups that may not happen in the tournament, amplifying stress on the system and increasing running times.
- It's necessary that the system maintains enough uptime to generate the all of the predictions required at least once, subsequent predictions would only be necessary for improved submissions.
- Interpretability must be added to the system in the sense that there must be clarity on what are the parameters that the model values the most at the moment of generating predictions, though the contest doesn't ask for this it's a useful resource for enhancing and maintaining the model for future iterations of the contest.

High level architecture

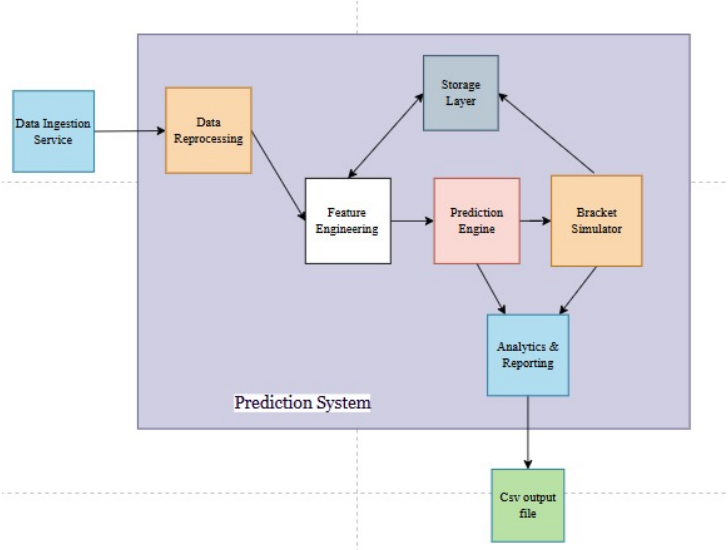

Figure 1. High level architecture of the prediction system.

- **Data Ingestion Service** This module is responsible for collecting the historical data needed to feed the system. The information is provided in multiple .csv files, containing individual and team level statistics from 1984 to 2024. This service acts as the entry point of the system, and its main function is data extraction.

- **Data Preprocessing** This is where the cleaning, normalization and organization of the raw data from the ingest is performed. Incomplete data is handled, formats are homogenized and duplicates or irrelevant records are eliminated. This module is essential to ensure that the data are consistent and useful for subsequent stages.

- **Feature Engineering** In this stage, the transformation of the processed data into meaningful variables for the prediction model is carried out. Key features such as offensive/defensive efficiency, historical performance, match location, among others, are selected and constructed. The "noise information" that does not provide predictive value is also discarded.

- **Storage Layer** This module stores all intermediate and final data generated during processing. It stores the clean datasets, selected features, prediction and simulation results, allowing traceability and information retrieval. It acts as the system database.

- **Prediction Engine** This component is the heart of the system: a machine learning model trained to predict the outcome of a match between two teams. It uses the variables generated in the previous stage and makes decisions based on historical patterns, ignoring non-modelable chaotic factors such as injuries, referee biases or logistical delays.

- **Bracket Simulator** Once the outcome of a single match can be predicted, this module allows the entire tournament to be simulated following the logic of direct elimination. The simulator makes round-by-round decisions based on the prediction engine and advances the winning teams to determine the tournament champion.

- **Analytics and Reporting** This module generates visual analysis and numerical summaries of model performance and simulations. It allows observing patterns, validating predictions and drawing useful conclusions. Its objective is to provide information that is understandable to both technical and non-technical users.

- **CSV Output File** Finally, the system generates a .csv file with the results of the tournament simulation, including predictions for each round, relevant metrics and winning team. This file can be used for external analysis or integration with other systems.

- **Systems engineering principles** This module separation aims to leave only one responsibility to each module so they can be easily modifiable without having to alter other functions. Despite this, all of the modules are deeply dependent on the rest of the modules for the system to work properly and the failure of one module would cause the rest of the system to fail.

  The way the bracket generator was designed aims to make it scalable so that if the format and amount of themes of the tournament changes this design can be easily updated to match their criteria

  Additionally the streamlined approach of the data handling process aims to make this changes easily traceable so that the data processing can be modified or expanded easily if needed.

  Metrics implemented to measure the quality of the prediction aim to create a data centric design that values the data and output quality.

Sensitivity and chaos

- **Feature noise:** Minor changes in stats like a 0.01 difference in offensive rating can affect predictions. Fort his purpose the features can be normalized.

- **Unplayed matchups:** Many team pairs don't play often, which may affect the accuracy of the prediction. Tp mitigate this the system can focus on transitive comparisons and results with common opponents.

- **Environment:** Weather scenarios that affect the location of the match can lead to cancellations and delays, these are ignored in the system at the time of forecasting.
- **Game results:** The results of the games are sensitive since they are used to make future predictions and are used as feedback from the bracket.
- If there are gaps in the data information for any of the equipment, they are filled in so as not to bias the engine predictions.
- If failure of a team to show up for a scheduled game leads to delays or changes in the match, thus, teams that do not show up for games will automatically forfeit.
- If the data are very old, the model prediction may become inaccurate, so it is decided to take more updated data on the team.

Technical stack

- The Pandas library in python can be used in the data handling process due to its functionalities for handling .csv files. This would be used on the file data loader module with the funcion read.csv.
- Panda would also be used in the preprocessing module for merging the .csv files by using the concat function. This module could also use use scikit-learn.preprocessing for normalizing the data in order to mitigate noise
- The matchup generator could be made using Pandas for reading the content of the now processed csv files and the itertools library to create each pair efficiently by using the combinations function.
- The prediction engine could use logistic regression as a baseline model, using features like seed difference, win rate and average scoring. Logistic regression could be implemented by using the sklearn library in python, which contains the LogisticRegression function, along with tools to train the model and generating metrics to evaluate the model's accuracy, with the only requirement being that

the data used for the model needs to be completely organized so that the model can work properly, also this model would have trouble handling chaotic factors, and those would have to be handled in the preprocessing module. The logistic regression process begins with asking a specific question, such as, “Do rainy days affect our monthly sales?” Then relevant historical data is collected and the factors involved are identified. From this data, a model is trained that relates the independent variables to a categorical dependent variable, using a logistic function rather than a straight line, as in linear regression. This model allows predictions to be made on new data, estimating the probability of occurrence of a given category or event. This model was chosen due to its capability to generate predictions in terms of probabilities given a data set of independent variables by using logarithmic equations.

Implementation and Integration: The data loading module would insert the csv files into the system using the read_csv function of the pandas library in python. For this purpose the files would be located in a directory related to the project.

The preprocessing model would remove the data of these files related to years previous to 2015 using the pandas filtering function. Once this data is cut we can group and normalize it, grouping could be done using the merge function from panda after entering the filtered data into an array. Once this is done we can normalize the data using the Standard_scaler function from sklearn.

For the organization of the teams by the bracket we are going to use the concepts of the “abstract factory” design pattern with the purpose of creating sets of objects with some variations among them. Since the bracket is fed back with each result it is useful when changing the number of teams in competition.

The features engineering module would rearrange and compare the metrics for each bracket using the group_By function and data filtering. Once this final filtering is done the resulting datased would then pass to the prediction engine, where the logistic regression function from sklearn would be used to generate and train the machine learning module. And finally the results of the predictions would go to the export

module where this data would be turned into a csv using the pandas.to_csv function to generate the final output.