UNIVERSIDAD DISTRITAL
FRANCICO JOSÉ DE CALDAS
Acreditación institucional de alta calidad

Systems Analysis and Design
May 2025

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

David Felipe Ariza Ariza - 20221020029
Oscar Manuel Contreras Gacha - 20221020052
Julián David Méndez Lara - 20241020140

NCAA Basketball Tournament - System Design

1 Summary of the First Workshop

In the First Workshop, an initial analysis of the NCAA basketball tournament system was performed, the objective was to understand the dynamics of results prediction. During this analysis, key elements and relationships were identified that have a significant influence and must be considered in the development of any predictive system. In addition, sensitivity factors and elements of chaos theory that can unpredictably alter tournament outcomes were examined. The relevant elements identified were the following:

- **Key elements:** Teams, season and matches.
- **Sensitive elements:** Courts, coaches, cities *(Home - Away - Neutral)*
- **Chaos elements:** Referee and fans

The above elements with their respective relationships are in the first workshop diagram.

2 Systems Requirements

The system requirements were determined based on the previous workshop specifications. The requirements were delineated to facilitate their identification.

Functional:

FR1: Tournament Outcome Prediction.

- **Description:** The system must predict the outcomes of the 2025 NCAA men's basketball tournament with a high degree of accuracy.
- **Acceptance Criterion:** Prediction accuracy must be greater than 90% on a test or cross-validation dataset.

FR2: Input Data Processing.

- **Description:** The system must be able to ingest, process, and clean relevant data from the dataset provided by the Kaggle competition (or the source defined for the 2025 tournament).
- **Acceptance Criterion:** The system must correctly manage input data formats, identify and address outliers or missing values according to predefined strategies (e.g., imputation, elimination).

FR3: Machine Learning (ML) Model Training.

- **Description:** The system will use processed data to train a machine learning model capable of predicting game outcomes.
- **Acceptance Criterion:** The training process must be reproducible, and the model must be evaluable and versionable.

FR4: Prediction File Generation

- **Description:** The system must export predictions in a CSV file (.csv).
- **Acceptance Criteria:** The CSV file must strictly follow the (ID, Prob) format, where:
  - ***ID:*** A text string consisting of the season (SEASON, e.g., ”2025”), the ID of Team A, and the ID of Team B, concatenated with underscores (e.g., 2025_1101_1102).
  - ***Prob:*** A floating-point numeric value representing the calculated probability of Team A beating Team B (e.g., 0.5).

Non-Functional (measurement requirements):

NFR1: Data Processing Efficiency

- **Description:** The system must process large volumes of historical data (from the Kaggle dataset) efficiently in terms of execution time.
- **Acceptance Criterion (Example):** The total time for data processing and model training must not exceed 5 hours on a defined standard hardware configuration.

NFR2: Adaptability and Generalization

- **Description:** The system design should be adaptable for future NCAA tournament datasets with minimal configuration changes.
- **Acceptance Criterion:** The system must allow configuration of parameters such as the tournament year and the input data file paths.

NFR3: Code Maintainability and Readability

- **Description:** The system's source code must be clear, well-documented, and modularly structured to facilitate understanding, maintenance, and future modifications.
- **Acceptance Criterion:** The code must follow the style guidelines of the chosen programming language (e.g., PEP 8 for Python) and include explanatory comments in complex sections.

NFR4: Reproducibility

- **Description:** The system must be replicable or deployable in similar environments with relative ease, clearly specifying its dependencies and configuration process.
- **Acceptance Criterion:** Provide a dependency file (e.g., requirements.txt for Python) and a README.md with clear installation and execution instructions.

3 High Level Architecture

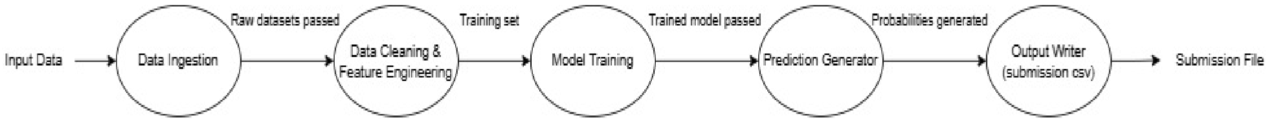


Figure 1: Enter Caption

Module Description:

- **Data Ingestion:** load data from Kaggle (history, seeds, ids, etc.)
- **Cleaning and Feature Engineering:** create variables such as match history, seeds, home/away status.
- **Model Training:** Use models like XGBoost, LightGBM, or Logistic Regression for both training and testing.
- **Prediction Generator:** simulates all encounters and calculates probabilities.
- **Output Writer:** Saves the submission.csv file in the required format.

Systems engineering principles:

The model design has applied the following principles for better performance.

Customer and requirements-centric approach:

The architecture ensures that all modules (ingestion, prediction, writing) align with the specific output and the user's goal: maximizing score.

System Thinking:

Complex interactions between multiple variables consider location, seed, history, chaotic behavior.
The tournament is not model as a fixed sequence, but as an open and dynamic system, sensitive to external conditions (such as referees or fans).

Life cycle approach:

The design covers all stages.

- ***Start:*** Historical data collection.
- ***Development:*** cleanup, training, and evaluation.
- ***Operation:*** Prediction for future games.
- ***Delivery:*** Generation of the required final file.

Interdisciplinarity and collaboration:

The design combines knowledge of:

- Data science and machine learning.
- Software engineer (modularity and scripts design).
- Sports analysis (tournament conditions and knowledge).

Facilitates collaboration between different profiles (data scientists, sports analysts, developers).

Functional and architectonic analysis:

Each module fulfills a clear and well-defined function:

- ***Ingestion:*** Obtains data
- ***Features:*** Transforms and preparation variables
- ***Training:*** Learns the model
- ***Prediction:*** Applies logic to new device pairs
- ***Writer:*** Deliver the required product

A modular pipeline architecture has been developed to enhance traceability and debugging.

Risk Management:

The impact of:

- Missing data → preprocessing handling
- Chaos in games → control variables
- Overfitting → regularization and cross-validation

The modular design allows for component-specific error isolation and resolution.

Continuous Verification and Validation:

The design promotes the use of:

- Cross-validation with metrics such as the Brier Score.
- Testing with known historical sets.
- Comparison with public benchmarks on Kaggle.

Each step can be verified individually (e.g., visual review of features, variable importance analysis)

Flexibility and Adaptability:

The architecture allows.

- Change the model (XGBoost, LightGBM, etc.) without altering the overall flow.
- Add new variables or datasets without redesigning the entire system.
- Reuse the system for other similar competitions.

4 Strategies against Chaos and Sensibility

Proposed Measures:

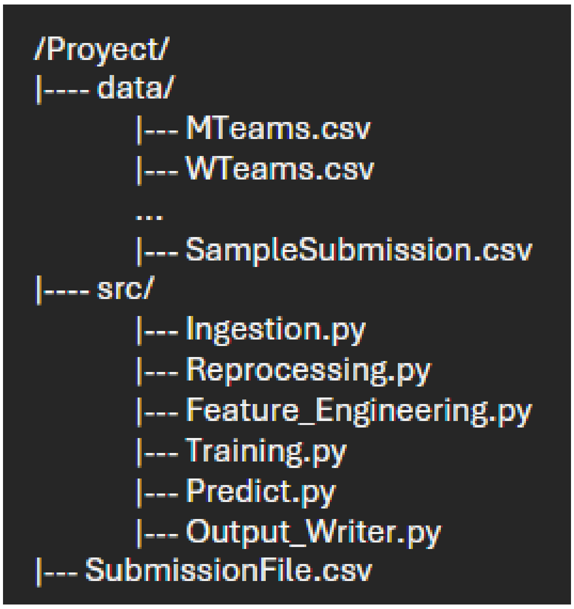
- **Home/Away/Neutral:** Explicitly code whether the game is home, away, or neutral, using dummy variables or ordinal coding as appropriate.
- **Seeds and Slots:** Include official rankings (seeds) and tournament assignments (slots) as predictor variables, along with historical performance statistics by seed.
- **Stats:** Add features that account for randomness, averaging each team's historical statistics (shots, fouls, etc.).

- **Overfitting:** May occur when a predictive model for the NCAA tournament learns specific outcomes or patterns from past tournaments too closely—like upsets or one-time player performances. As a result, it may not predict future tournaments well, especially since teams and players change every year.
- **Monitoring (optional):** Cross-validation with multiple random seeds can help reduce overfitting by testing the model on different data splits. This improves the model's generalization and gives more reliable performance metrics. It also shows if the model is too sensitive to specific training sets.

5 Technologic Stack

Code Design:

- Follow the **Modular Pipeline** design pattern: Because it is ideal for sequential data processing systems, as is the case in a prediction competition like the NCAA Tournament.
- Structured repository with folders: */data, /src*
- Final output generated by: *python OutputWriter.py*, with the probabilities calculated by *Predict.py* → *submission.csv*



```
/Proyect/
|---- data/
          |--- MTeams.csv
          |--- WTeams.csv
          ...
          |--- SampleSubmission.csv
|---- src/
          |--- Ingestion.py
          |--- Reprocessing.py
          |--- Feature_Engineering.py
          |--- Training.py
          |--- Predict.py
          |--- Output_Writer.py
|--- SubmissionFile.csv
```

Figure 2: Project Structure

Components	**Tools**
Language	**Python**: Permite manejar grandes cantidades de datos usando las librerías Pandas y NumPy.
Modeling	**Scikit-learn, XGBoost, Logistic Regression**: Permiten predecir probabilidades, son simples, algunos usan árboles de decisión y son muy potentes.
Intake	**Pandas**: Esencial para manejar datos estructurados (CSV, data frames).
Cleaning and Feature Engineering	**NumPy**: Operaciones vectorizadas rápidas en arrays (por ejemplo, restar métricas entre dispositivos). Conversión a arrays NumPy para alimentar modelos (X_train, y_train). Cálculo directo del Brier Score si es necesario.
Testing	**Brier Score (MSE en clasificación probabilística)**: Métrica que mide la precisión de las predicciones probabilísticas, evaluando la diferencia entre probabilidades predichas y resultados reales.
Output	`.csv` generado por script en Python: Salida con probabilidades en el formato (ID, Prob), requerido por la competencia de Kaggle.
Environment	**VS Code, Kaggle Kernels + GitHub**: VS Code para codificación, Kaggle Kernels para notebooks y despliegue del modelo, y GitHub para control de versiones y documentación del flujo de trabajo.

Table 1: Summary of tools, components and environment

6 Conclusion

This project presents a clear and organized system to predict the results of the NCAA Men's Basketball Tournament. It uses basic ideas from systems engineering like thinking about the full process, dividing the work into parts, and handling risks. Each part of the system—from getting the data, cleaning it, choosing the best features, training the model, and finally showing the results—is well separated. This makes the system easier to understand, evaluate, and improve in the future. Python tools, such as Pandas, NumPy, Scikit-learn, and XGBoost, are essential for data processing and prediction. The system manages complex situations, including referee decisions, fan support, game location, and team seed. These features assist in improving the accuracy of the model. In the end, the system was designed to give reliable and clear results, with over 90