ML Model Research

Wednesday, November 12, 2025 5:19 PM

Core Libraries for Data Handling and Scientific Computing:

- **NumPy:**
  Provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. It forms the foundation for many other ML libraries.
- **Pandas:**
  Offers powerful data structures like DataFrames, making data manipulation, cleaning, and analysis highly efficient.
- **SciPy:**
  Builds on NumPy and provides modules for scientific and technical computing, including optimization, integration, interpolation, and signal processing, which are valuable in machine learning.

General-Purpose Machine Learning Libraries:

- **Scikit-learn (sklearn):** A widely used library for classical machine learning algorithms. It offers a comprehensive set of tools for classification, regression, clustering, dimensionality reduction, model selection, and preprocessing.

Deep Learning Frameworks:

- **TensorFlow:**
  open-source framework for building and training deep learning models, known for its flexibility and scalability.
- **PyTorch:**
  open-source deep learning library favored for its ease of use, dynamic computation graphs, and strong community support.
- **Keras:**
  high-level API that runs on top of TensorFlow (and other backends), designed for rapid prototyping and simplified deep learning model development.

Specialized Libraries:

- **XGBoost, LightGBM, CatBoost:**
  Libraries for gradient boosting, known for their high performance and accuracy in various machine learning tasks.
- **Matplotlib & Seaborn:**
  Libraries for data visualization, enabling the creation of various plots and charts to understand and present data.

Classical ML Models used with Sports Prediction

Logistic Regression

- The model looks at various **game statistics** — like field goal percentage, rebounds, assists, or turnovers.
- It combines these stats into a single **score** that reflects how likely a team is to win.
- This score is passed through a **sigmoid function**, which converts it into a **probability between 0 and 1**.
  - A value **close to 1** means a **high chance of winning**.
  - A value **close to 0** means a **high chance of losing**.
- Based on a threshold (usually 0.5), the model then predicts either a **"Win"** or **"Loss."**
- Logistic Regression in NBA predictions helps determine **how likely a team is to win** based on their performance stats from past games.

Random Forest

- A **Random Forest** builds **many individual decision trees**, each trained on random parts of the data (like shooting stats, rebounds, turnovers, etc.).
- Each tree makes its own **prediction** about whether the team will win or lose.
- The **final prediction** is made by **majority vote** — whichever outcome (win or loss) most trees agree on becomes the model's decision.
- Because it uses **many trees instead of one**, the model avoids overfitting and gives **more reliable, balanced predictions**.
- A Random Forest predicts NBA game outcomes by **combining the opinions of many decision trees**, leading to **more accurate and stable predictions** about whether a team will win or lose.

Support Vector Machines (SVM)

- SVM looks at game stats (like field goals, rebounds, assists, turnovers, etc.) and **plots them as points** in a multi-dimensional space.
- It then finds the **best boundary (called a hyperplane)** that separates **winning games** from **losing games** as clearly as possible.
- The model focuses on the **data points closest to this boundary**, called **support vectors**, which are most important for making accurate predictions.
- When a new game's stats are entered, the SVM checks **which side of the boundary** they fall on — that decides whether it predicts a **win or loss**.
- A Support Vector Machine predicts NBA game outcomes by **drawing the best possible line** that separates wins from losses based on team performance statistics

Neural Networks

- **DNN (Deep Neural Network)**
  - A DNN is made up of many **layers of interconnected nodes (neurons)** — similar to how the human brain processes information.
  - Each layer takes in stats (like shooting percentages, rebounds, assists, and turnovers) and learns to recognize **hidden relationships** between them.
  - As the data passes through these layers, the network learns **deeper and more abstract features** — for example, how combinations of stats often lead to wins or losses.
  - In the end, the DNN outputs a **probability** of whether the team will win (close to 1) or lose (close to 0).
  - A Deep Neural Network predicts NBA game outcomes by **learning complex patterns and interactions** among many statistics, allowing it to make **highly accurate win/loss predictions** from large, detailed datasets.

- **LSTM (Long Short Term Memory)**
  - Type of recurrent neural network
  - LSTMs are designed to handle **time-series data** or sequences, so they can learn from the **order and history of games**.
  - They take in stats from **previous games** (like points, rebounds, assists, turnovers) and remember important patterns over time.
  - Unlike regular neural networks, LSTMs can **retain long-term dependencies**, meaning they understand how past performances affect future outcomes.
  - The network outputs a **probability** of winning or losing for an upcoming game based on these learned patterns.
  - LSTMs predict NBA game outcomes by **learning from sequences of past games**, capturing both recent and long-term trends to make **more informed win/loss predictions**.

K-Nearest Neighbors (KNN)

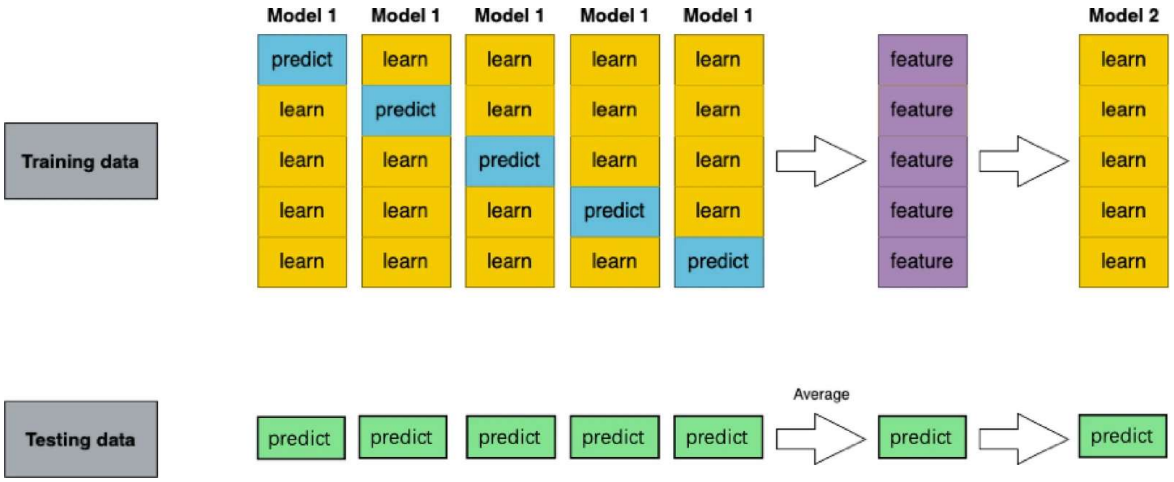
- **Supervised learning algorithm** used for both **classification** and **regression** tasks.
- It is **non-parametric**, meaning it doesn't assume any specific data distribution.
- It is **instance-based**, meaning it makes predictions using the actual training samples rather than a learned model.
- **Main idea:** Predicts outcomes based on **similarity** between data points.
- **Process for classification:**
  i. When a new sample needs to be classified, the algorithm finds the **k closest data points** from the training set.
  ii. The closeness is measured using a **distance metric**, such as:
    - **Euclidean distance**
    - **Manhattan distance**
    - **Minkowski distance**
  iii. The **majority class** among these k neighbors is assigned as the predicted label for the new sample.

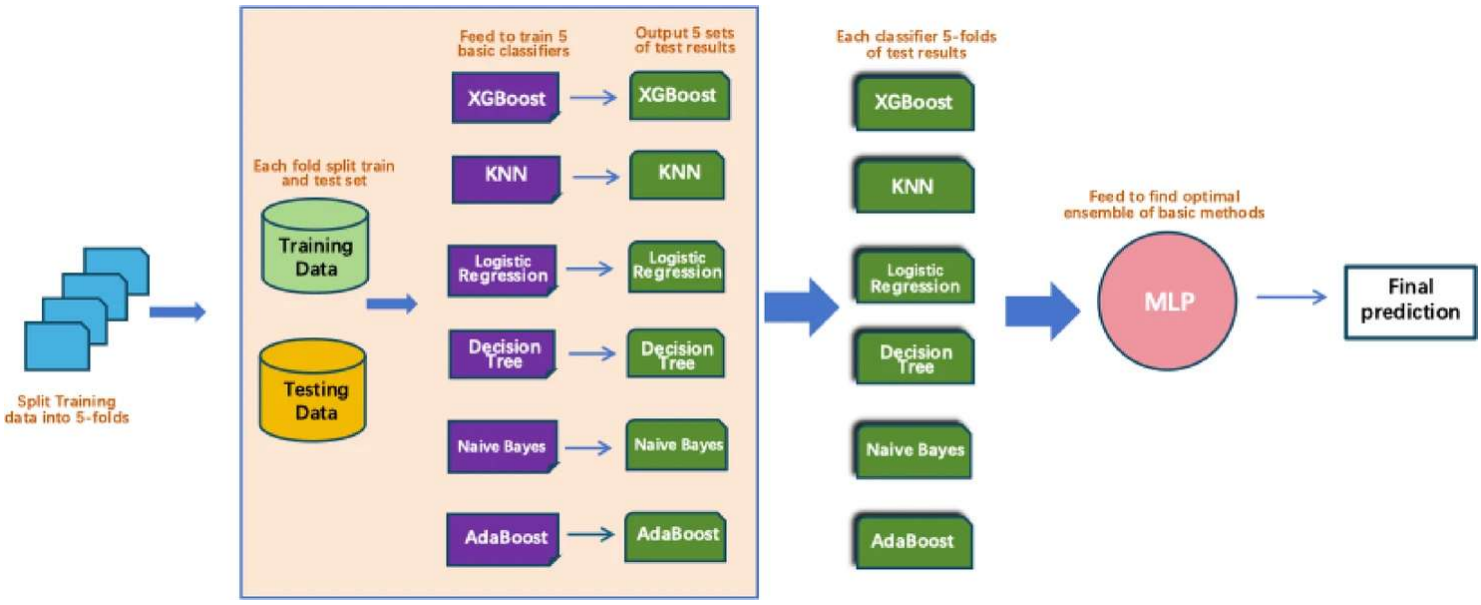
AdaBoost

- Ensemble learning method that combines multiple weak learners to create a strong classifier.
- Each iteration focuses more on instances misclassified in previous rounds.
- Subsequent classifiers are guided to concentrate on more challenging samples.
- Functions as a meta-estimator by first training a base learner on the original dataset.
- In later iterations, the base learner is retrained with adjusted sample weights.
- Weights of misclassified instances are increased to improve performance on difficult examples.
- Progressively enhances its accuracy with each iteration.

Stacked Ensemble Model

- https://www.nature.com/articles/s41598-025-13657-1#Sec3
- Test accuracy: 83%
- Implemented on a machine powered by an 8th-generation Intel Core i7 CPU running at 3.1 GHz with 16 GB of RAM (about the same specs as my current laptop).
- Enhance overall performance by integrating predictions from multiple base models.
  - Combine outputs for more robustness and accuracy
  - Hyperparameter optimization using **GridSearchCV**





Classic	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	ROC (%)
XGBoost	81.03	82.36	80.18	81.26	90.82
KNN	80.15	81.18	79.79	80.48	87.92
Decision Tree	74.19	75.07	74.37	74.72	79.87
AdaBoost	81.10	81.53	81.64	81.58	89.97
Logistic Regression	80.49	81.90	79.52	80.70	90.28
Naïve Bayes	76.56	79.15	73.71	76.33	85.67
Sacking (MLP)	83.27	84.46	82.56	83.50	92.13

- **Base Models (6):** XGBoost, KNN, AdaBoost, Naive Bayes, Logistic Regression, Decision Tree
- **Meta-Learner:** Multi-Layer Perceptron (MLP) with 2 hidden layers × 50 neurons → combines base model outputs
- **Training:** 5-fold cross-validation
  - 4 folds train base models, 1 fold validates
  - Repeat 5× so every fold validates once
- **Meta Input:** Concatenate out-of-fold predictions from base models → input to MLP
- **Test Predictions:** Average base model predictions → feed MLP → final output
- **Key Idea:** Base models capture patterns; MLP learns **how to combine them for best accuracy**

XGBoost and SHAP (2024 Publication)

- https://pmc.ncbi.nlm.nih.gov/articles/PMC11265715/#sec016
- **Critical Features**
  - Field goal percentage, defensive rebounds, and turnovers are critical factors influencing game outcomes throughout all stages.
  - Assists are important indicators of game outcomes during the first two quarters.
  - Offensive rebounds and three-point shooting percentages become more influential in the later stages of the game.

- **XGBoost**
  - Based on Bayesian optimization and grid search
  - Good accuracy, gets better with real-time updates during game
  - Multiple base learners, *typically decision trees*
  - Significantly faster computational speed
  - Improved generalization performance
  - Scalability
  - Well-suited for large-scale and high-dimensional machine learning tasks

- **SHAP**
  - SHapley Additive exPlanations
  - Good at interpretation and highlights key factors influencing outcome

🔍 **SHAP Analysis**

- **Goal:** To make the MLP-based stacking model (a "black-box" model) more interpretable.
- **Method:** SHAP (SHapley Additive exPlanations) values are used to measure how each feature influences predictions.
- **Sample:** 300 test cases were analyzed to see each feature's contribution and direction of impact.
- **Benefit:** Helps identify which stats most affect game outcomes and whether they push predictions toward wins or losses.

🌍 **Global Feature Importance (Overall Model Behavior)**

- **Visualization:**
  - Figure 8 shows the *mean SHAP values* — features ranked by average influence on predictions.
  - **X-axis:** average impact size.
  - **Y-axis:** features sorted by importance.
- **Most influential features:**
  - **2PA (two-point attempts)**
  - **FG (field goals made)**
  - **2P (two-point makes)**
  - **TRB (total rebounds)**
- **Least influential:**
  - **AST (assists)**
- **Meaning:**
  - Shooting frequency, shooting accuracy, and rebounding strongly influence game outcomes.

📊 **SHAP Summary Plot (Feature Directional Effects)**

- **Visualization:** Figure 9 shows how high and low feature values affect predictions.
  - **Red = high values**, **Blue = low values**.
  - **SHAP value (x-axis):** measures how much each feature pushes prediction toward win/loss.
- **Findings:**
  - **Higher FG and TRB → increase win probability.**
  - **Higher FGA and 2PA → decrease win probability.**
- **Interpretation:**
  - Teams taking **too many two-point shots** but **with low efficiency** tend to lose.
  - High 2PA may mean **inefficient offense** or **desperation shots** when trailing.
  - Frequent two-point attempts might also suffer from **defensive pressure near the rim**.
  - Thus, **high 2PA without high FG% = negative signal** for winning.

Individual SHAP Explanation (Single Prediction Example)

- **Example Case:** Model predicted a **loss** (output f(x) = −1.058).
- **Top feature contributions (Figure 11 – Waterfall Plot):**
  - **2PA = 74 → SHAP = −4.39:** Too many inefficient shots (big negative impact).
  - **2P = 42 → SHAP = +3.00:** Good number of successful two-point makes (positive impact).
  - **FG = 46 → SHAP = +1.25:** Decent scoring success.
  - **TRB = 47, ORB = 17 → Positive impact:** Strong rebounding performance helps.
  - **FGA = 95, 3P = 4 → Negative impact:** Too many shot attempts and poor 3-point shooting hurt performance.
- **Takeaway:** SHAP shows exactly which stats made the model predict a loss.

Combined Global + Individual Insights

- **Most influential stats overall:**
  - Two-point attempts (2PA), Field goals (FG), Total rebounds (TRB), Field goal attempts (FGA).
- **Visualization tools:**
  - **Waterfall and Force plots** show how each feature raises or lowers prediction scores.
- **Feature Interactions:**
  - Strong synergy found between **defensive rebounds (DRB)** and **offensive rebounds (ORB)** → shows model captures teamwork-related effects.

Practical Insights

- Improves **model transparency** and helps **coaches or analysts** understand which stats matter most.
- Supports **lineup optimization** and **strategic planning** (e.g., balance between shot attempts and efficiency).

Limitations & Future Directions

1. **Interaction Strength:**
   - SHAP shows interactions but doesn't *quantify* how strong they are.
   - Future improvement: use **Graph Neural Networks (GNNs)** or **self-attention** to capture these better.
2. **Data Type Limitations:**
   - Current model only uses **structured stats** (e.g., shot counts, rebounds).
   - Could improve by adding **unstructured data** like player movement or in-game tactics.
3. **Generalizability:**
   - Model trained on **NBA data** only.
   - Needs testing on **other leagues (CBA, NCAA)** or **other sports (soccer, volleyball)**.