Assignment 1  
Kaggle Competition

short line

Group 9

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Student Last Name** | **Student First Name** | **Student ID** | **Github Username** | **Github Custom Package Repo Link** | **Kaggle Username** |
| Senthil Kumar | Shashikanth | 25218722 | Shashikanth11 | <https://github.com/Shashikanth11/My-Custom-Package> | shashikanth1110 |
| Garcia Ortiz | Victor | 10522504 | Vgarcia.chile | <https://github.com/vgarciachile/my_krml_10522504> | VictorGarciaUTS |
| Pailla | Yagnasri | 25223900 | YagnasriPailla | <https://github.com/YagnasriPailla/nba-pkg-yagna> | Pailla Yagnasri 28 |
| Pramutjit | Yanichsa | 25077761 | Yanichsa | <https://github.com/Yanichsa/Custom_Package_y> | ninayanichsa |

|  |  |
| --- | --- |
| Github | Group Experimentation Repo: <https://github.com/UTS-36120AML-SP25-G9/AML-AT1-group-9> |

36120 - Advanced Machine Learning Application

Table of contents

[1. Executive Summary 2](#_Toc503968291)

[Data and Challenges 3](#_Toc1721099057)

[2. Business Understanding 3](#_Toc1900738342)

[a. Business Use Cases 3](#_Toc129475529)

[3. Data Understanding 5](#_Toc575319243)

[4. Experimentation 8](#_Toc422262848)

[4.1. Student A (Shashikanth Senthil Kumar) 9](#_Toc2086507236)

[4.1.1. Approach 1 9](#_Toc926854062)

[4.1.1.1. Data Preparation 9](#_Toc2024237868)

[4.1.1.2. Modeling 10](#_Toc707031107)

[4.1.1.3. Achieved Result 10](#_Toc370886677)

[4.1.2. Approach 2 11](#_Toc1298580723)

[4.1.2.1. Data Preparation 11](#_Toc950064775)

[4.1.2.2. Modeling 12](#_Toc1204555452)

[4.1.2.3. Achieved Result 12](#_Toc734912839)

[4.1.3. Approach 3 15](#_Toc1306972228)

[4.1.3.1. Data Preparation 15](#_Toc1654393917)

[4.1.3.2. Modeling 15](#_Toc29416450)

[4.1.3.3. Achieved Result 15](#_Toc1707788190)

[4.1.4. Approach 4 17](#_Toc1217026333)

[4.1.4.1. Data Preparation 18](#_Toc646038564)

[4.1.4.2. Modeling 18](#_Toc1298962765)

[4.1.4.3. Achieved Result 19](#_Toc1474046349)

[4.1.5. Approach 5 20](#_Toc965672005)

[4.1.5.1. Data Preparation 21](#_Toc1348193170)

[4.1.5.2. Modeling 21](#_Toc2032137190)

[4.1.5.3. Achieved Result 22](#_Toc459241963)

[4.2. Student B (Victor Garcia Ortiz) 24](#_Toc404812003)

[4.3. Student C (Yagnasri Pailla) 34](#_Toc743216283)

[4.3.1. Approach 1 – Logistic Regression 35](#_Toc160741361)

[1. Data Preparation 35](#_Toc1369536966)

[2. Modeling 35](#_Toc193839830)

[4.3.1.1. Achieved Result 36](#_Toc813931884)

[4.3.2. Approach 2 - LightGBM 36](#_Toc746306773)

[4.3.2.1. Data Preparation 36](#_Toc1405662716)

[4.3.2.2. Modeling 36](#_Toc581180794)

[4.3.2.3. Achieved Result 37](#_Toc1495691499)

[4.3.3. Approach 3 - XGBoost 38](#_Toc1828673551)

[4.3.3.1. Data Preparation 38](#_Toc788709765)

[4.3.3.2. Modeling 38](#_Toc468894321)

[4.3.3.3. Achieved Result 39](#_Toc488262423)

[4.4. Student D (Yanichsa Pramutjit) 39](#_Toc20851314)

[Data Preparation 40](#_Toc796905930)

[4.4.1. Approach 1: Logistic Regression 41](#_Toc449478554)

[4.4.1.1. Modeling 41](#_Toc2100952005)

[4.4.2. Approach 2: Random Forest 41](#_Toc1116988154)

[4.4.2.1. Modeling 41](#_Toc381395977)

[4.4.3. Approach 3: XGBoost 42](#_Toc1432729453)

[4.4.3.1. Modeling 42](#_Toc1433663219)

[4.4.4. Approach 4: Decision Tree 42](#_Toc1019280078)

[4.4.4.1. Modeling 42](#_Toc323000289)

[4.4.5. Achieved Result 43](#_Toc1404668765)

[5. Evaluation 44](#_Toc326852464)

[a. Business Impact and Benefits 45](#_Toc2106595439)

[b. Risks and Incorrect Predictions 46](#_Toc992829683)

[c. Data Privacy and Ethical Concerns 47](#_Toc1752553653)

[6. Collaboration 47](#_Toc527068471)

[a. Individual Contributions 48](#_Toc1432301728)

[b. Group Dynamic 48](#_Toc1190864387)

[c. Ways of Working Together 49](#_Toc1175307228)

[d. Issues Faced within the Group 49](#_Toc917535819)

[7. Conclusion 50](#_Toc119140786)

[8. References 51](#_Toc595593998)

[9. Figures and Tables Index 52](#_Toc1064369089)

short dash

# Executive Summary

The goal of this study was to forecast NBA draft results using cutting-edge machine learning techniques. The objective of the job, which was presented as a binary classification issue, was to ascertain, using pre-draft statistics and background data, whether a player would be drafted or not. The project was completed as part of a Kaggle competition, which mimicked a real-world predictive modeling task that called for technical know-how, teamwork, and repeatable procedures.

The NBA draft process is extremely selective and competitive. Traditional scouting frequently depends largely on subjective assessments, and only a small percentage of eligible college athletes are selected each year. This raises the possibility of inconsistent player selection and lost opportunities. In order to solve this problem, the study investigated how data-driven methods may enhance human assessment and decision-making.

### **Data and Challenges**

The dataset provided for the competition contained a wide range of features, including:

* **Performance statistics** (e.g., scoring, assists, rebounds, shooting percentages).
* **Advanced efficiency metrics** (e.g., true shooting percentage, box plus-minus).
* **Contextual attributes** (e.g., team, conference, competition strength).
* **Physical and recruitment details** (e.g., height, recruiting rank).

During the project, a number of difficulties were faced, including noisy features, missing values, and a notable class imbalance. Data cleaning, preprocessing, imputation, categorical variable encoding, and feature engineering were all used to properly address these.

The project showed how machine learning can offer insightful information on preliminary results. Features like physical traits, efficiency ratings, and recruiting rank are powerful predictors of draft likelihood, according to exploratory investigation. The tests demonstrated that predictive algorithms provide a methodical and open approach to spotting player potential by revealing trends that aren't always apparent through conventional scouting.

The project showed how machine learning can offer insightful information on preliminary results. Features like physical traits, efficiency ratings, and recruiting rank are powerful predictors of draft likelihood, according to exploratory investigation. The tests demonstrated that predictive algorithms provide a methodical and open approach to spotting player potential by revealing trends that aren't always apparent through conventional scouting.

# Business Understanding

## Business Use Cases

The NBA draft represents a pivotal opportunity for teams to enhance their competitive edge by selecting new talent from a pool of eligible players. The challenge is to make these high-stakes decisions with limited information and under significant uncertainty. This assignment, situated within the Advanced Machine Learning subject, leverages modern data science and engineering practices to address this real-world scenario.

Key aspects of the business use case include:

* Supporting NBA teams in identifying which draft-eligible players are most likely to contribute positively to team performance, compared to those who are not drafted.
* Providing a systematic, data-driven approach to supplement traditional scouting and subjective evaluations.
* Enabling teams to optimize their draft picks, reduce the risk of poor selections, and maximize the value gained from each draft opportunity.
* Delivering actionable insights that can be integrated into the decision-making process of general managers, scouts, and analysts.
* Participating in a Kaggle competition, which simulates a real-world predictive modeling challenge and provides a platform for benchmarking our model’s performance against other teams and approaches.

The benefits of building a predictive model in this context are substantial:

* Improved accuracy and consistency in player selection.
* Enhanced ability to uncover hidden talent or undervalued prospects.
* Data-driven justification for draft decisions, supporting transparency and accountability within the organization.

1. Key Objectives

The main objectives of this project are:

* To frame the NBA draft as a machine learning classification problem, where the goal is to distinguish between players who are drafted and those who are not, based on pre-draft data.
* To apply advanced machine learning methods and best practices, as covered in the course, including the use of tools like Poetry for dependency management, pyenv for environment control, custom Python packages published to TestPyPI, and version control with GitHub. These practices ensure reproducibility, modularity, and a workflow that mirrors real-world industry standards.
* To ensure the model’s recommendations are actionable and relevant to the business context, supporting the overarching goal of team success. For example, if a player is classified as likely to be drafted, their specific strengths (e.g., offensive skills, three-point shooting, defensive capabilities, rebounding) can be further analyzed to reinforce and align with a team’s strategic needs.
* To acknowledge the scope and limitations of the approach, including:
  + The model is limited to available pre-draft data and cannot account for post-draft development, injuries, or off-court factors.
  + The primary focus is on draft selection, but the analysis of drafted players’ profiles can inform team strategy and roster construction.
  + Interpretability and fairness are considered, but some level of uncertainty and subjectivity remains inherent in the draft process.
* To foster effective collaboration within our team of four, utilizing collaborative tools and workflows such as GitHub for version control, shared repositories, and code review, as well as standardized environments and package management to ensure consistency and efficiency throughout the project.

By addressing these objectives, the project demonstrates the value of advanced machine learning and software engineering in a complex, high-impact business environment, while also recognizing the practical constraints and challenges of real-world deployment.

short dash

# Data Understanding

**Data Sources and Collection Methods**

The dataset originates from structured college basketball statistics collected across multiple seasons. Typically, data is from official NCAA box scores, sports analytics platforms, and scouting databases that track both individual player and team-level performance. For this project, the data is assumed to be sourced from Kaggle or sports, where each row represents a unique player-season record.

The dataset consolidates:

* Player attributes (e.g., height, recruiting rank, year of study).
* Performance metrics (e.g., shooting percentages, usage rates, rebounds, assists, turnovers, steals, blocks).
* Team and conference information (to explore competition level).
* Target label: whether a player was drafted at the end of the season (binary: drafted = 1, not drafted = 0).

Since the draft outcome is definitive, the collection of the target variable is straightforward. However, data for predictors may be noisy due to differences in scoring methods across conferences, missing values for lesser-known players, or inconsistencies in reporting.

**Variables and Features**

The dataset contains over 60 features that can broadly be categorized into categorical identifiers, performance statistics, ratios/percentages, and contextual attributes.

(a) Categorical Attributes

- Team: Identifies the player’s team. Useful to measure relative exposure (players from elite programs often attract more scouting attention).

- Conference (conf): Indicates the league in which the team competes. Stronger conferences (e.g., ACC, SEC) may increase draft likelihood due to higher competition.

* Player ID: Unique identifier, later excluded from predictive modeling as it has no intrinsic link to performance.

(b) Performance Metrics (Counting Stats)

* GP (Games Played) and MP (Minutes Played): Availability and durability.
* PTS, REB, AST, STL, BLK: Basic box score indicators of productivity.
* FTM, FTA, 2PM, 2PA, 3PM, 3PA: Shooting outcomes across different shot types.
* OREB, DREB, TREB: Offensive, defensive, and total rebounds.

(c) Advanced Metrics and Ratios

* ORtg, DRtg: Offensive and defensive ratings; contextualized efficiency measures.
* USG% (Usage Rate): Share of team possessions a player consumes; high usage often indicates centrality in team offense.
* eFG% (Effective Field Goal %), TS% (True Shooting %): Adjust shooting efficiency beyond raw field goal percentage.
* AST%, TOV%: Assist and turnover percentages; reflect playmaking vs ball control.
* Rim Ratio, Mid Ratio, Dunks Ratio: Specialized shot-type ratios, indicating efficiency in specific zones.
* BPM, OBPM, DBPM: Box Plus-Minus metrics, adjusted indicators of overall, offensive, and defensive contributions.

(d) Contextual and Recruitment Metrics

* Recruiting Rank (Rec\_Rank): Prospect’s high school ranking before college; historically significant in scouting.
* Adjusted Ratings (Adjoe, ADRtg, Dporpag): Derived efficiency measures used in sports analytics to normalize player performance across teams and contexts.
* Height (ht\_new): Converted to numeric, as physical profile is a strong predictor of draftability.

**Data Limitations**

While comprehensive, the dataset exhibits several limitations that impact modeling:

Class Imbalance: Drafted players represent less than 1% of the dataset. Without appropriate techniques (e.g., class\_weight, resampling), models could default to predicting “not drafted” with high accuracy but poor AUROC.

Missing Values: Certain advanced stats (e.g., rim ratios, mid ratios) contain up to 20% missingness, likely due to incomplete play-by-play tracking for all players. These were handled with median imputation or recomputation.

Feature Redundancy: Several variables are derivations of others (e.g., TS% vs eFG%). Including all may cause multicollinearity.

Data Quality: Recruiting rank is missing for many players from lesser-known programs, creating bias toward high-profile recruits.

**Exploratory Data Analysis (EDA) Insights**

EDA was conducted to understand distributions, identify outliers, and assess predictive potential:

* Target Distribution: Highly imbalanced (≈ 99% not drafted, ≈ 1% drafted). Justifies AUROC as the evaluation metric.
* Recruiting Rank (Rec\_Rank): Strongly skewed; drafted players cluster around top ranks (low numbers). Missing values were replaced with -1 to distinguish unranked players.
* Conference Strength: Chi-square analysis showed that players from elite conferences (ACC, SEC, Big 12) had statistically significant higher draft likelihoods compared to mid-majors.
* Performance Differentiation: ANOVA tests on numeric features revealed that metrics like ORtg, TS%, AST%, and BPM were significantly higher among drafted players.
* Outliers: Boxplots showed that drafted players often represent extreme outliers in categories like minutes played, usage rate, and scoring efficiency. These are not noise but meaningful signals of elite performance.

**Conclusion**

In summary, the dataset used for NBA Draft prediction is a rich but imperfect reflection of player performance and scouting visibility. While it contains a wide variety of features ranging from basic statistics to advanced efficiency metrics, the data suffers from imbalance and missing values. Exploratory data analysis highlighted that elite recruits and players from major conferences dominate the draft pool, and advanced efficiency metrics strongly differentiate drafted from non-drafted players. These insights inform both preprocessing (imputation, encoding, feature selection) and modeling strategies (imbalanced learning, regularization, ensemble models).

The combination of domain knowledge and data-driven exploration ensures that the chosen predictive models focus on the most relevant aspects of college basketball performance when forecasting draft outcomes.

short dash

# Experimentation

## Student A (Shashikanth Senthil Kumar)

### Approach 1

#### Data Preparation

For Experiment 0, the dataset underwent thorough preparation to ensure quality inputs for baseline modeling:

1. **Exploratory Data Analysis (EDA) and Feature Selection:**

* Conducted univariate and bivariate analyses to examine distributions and identify correlations.
* Performed **correlation analysis** to remove redundant features and applied **knowledge-driven feature selection** based on domain understanding.

1. **Data Cleaning and Formatting:**

* Fixed inconsistencies in the height feature by converting formats like "6 ft 1 inch" or "1-Jun" into **centimeters (cm)**.
* Addressed missing values:
  + Dropped rows with null values in the training set.
  + Imputed null values in the test set using SimpleImputer(strategy='constant') to avoid losing rows necessary for Kaggle submission.
* Removed duplicate rows from the training set to prevent bias.

1. **Feature Engineering:**

* Created **Height\_Impact** = height\_cm × BPM to capture combined physical and performance impact.
* Created **REB\_per** = ORB\_per + DRB\_per to reflect total rebounding efficiency.
* Created **FT\_efficiency** = FT\_per × FTR to quantify free-throw scoring efficiency.

1. **Train-Validation Split:**

* Split the training dataset into **train (80%)** and **validation (20%)** subsets to evaluate model generalization.

1. **Custom Package:**

To streamline repeated tasks, a custom Python package **Shash Package** was developed and published on TestPyPI. The package provides utilities for data analysis, preparation, and model evaluation:

* **Data Preparation & EDA**:
  + datacheck: Identifies missing/null values and duplicate rows.
  + dataeda: Summarizes dataset structure through head, shape, info, and descriptive statistics.
* **Model Evaluation**:
  + evaluate\_model: Computes key metrics such as accuracy, precision, recall, F1-score, confusion matrix, classification report, and ROC AUC score.

This modular approach reduced redundancy, improved reproducibility, and ensured consistency in both data preparation and model evaluation workflows.

#### Modeling

* **Algorithm:** DummyClassifier from scikit-learn with strategy='most\_frequent', which predicts the majority class for all instances.
* **Rationale:** Establishing a baseline allows comparison with more advanced models (Random Forest, LightGBM) and highlights the improvement potential. Since it does not learn from feature patterns, it shows the performance expected by chance or naive prediction.
* **Implementation:**
  + No hyperparameter tuning was required because the model always predicts the most frequent class.
  + Predictions were evaluated on both **train** and **validation** sets to assess baseline performance and confirm the need for models capable of handling the minority class.

#### Achieved Result

The Dummy Classifier produced the following outcomes:

* **Training and Validation Performance:**
  + Accuracy was dominated by the majority class.
  + Precision, recall, and F1-score for the minority class (drafted players) were extremely low, indicating the model could not identify drafted players.
* **Analysis and Limitations:**
  + The baseline confirms that naive prediction fails to capture minority class instances due to severe class imbalance.
  + Metrics like F1-score and recall for drafted players were near zero, emphasizing the need for more sophisticated models and class balancing strategies.
* **Potential Improvements:**
  + Introduce models capable of handling imbalanced datasets (Logistic Regression, Random Forest, LightGBM, XGBoost).
  + Apply additional feature engineering or domain-informed transformations to better capture draft predictors.
  + Fine-tune hyperparameters and use cross-validation to optimize predictive performance.

### Approach 2

#### Data Preparation

For this experiment, we reused the **saved datasets (X\_train, y\_train, X\_val, y\_val)** prepared in **Experiment 0**, ensuring consistency with the baseline setup. This approach guarantees that any performance differences observed are due to the choice of the **Logistic Regression model** rather than changes in the feature set.

The following checks and transformations were applied:

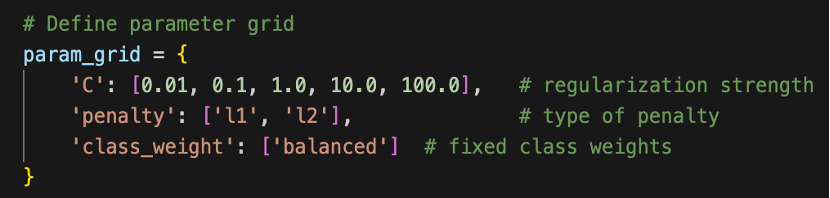
* **Data Validation**: The datasets were re-examined for missing values and duplicates. No issues were found, confirming the integrity of the saved data.
* **Feature Engineering**: No new features were introduced in this experiment since the same engineered features from Experiment 0 (e.g., Height\_Impact, REB\_per, FT\_efficiency) were retained.
* **Transformations**:
  + **Standardization**: All features were standardized using StandardScaler to bring them onto a comparable scale, which is important for gradient-based models like Logistic Regression.
  + **Target Flattening**: The target variable (y\_train, y\_val) was flattened using .ravel() to ensure compatibility with scikit-learn’s Logistic Regression implementation.

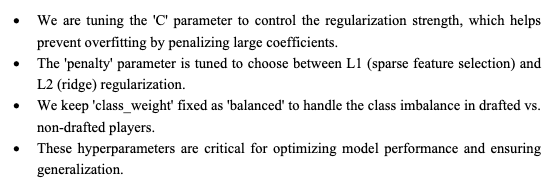
#### Modeling

**Algorithm:** Logistic Regression (LogisticRegression from scikit-learn) with max\_iter=5000 and random\_state=42.

**Rationale:** Logistic Regression was selected as a linear baseline model to capture class relationships and provide interpretability. It is more powerful than a naive baseline while still simple and efficient, making it a good benchmark before testing complex models.

**Implementation:**

* Hyperparameter tuning was performed with **GridSearchCV (5-fold CV)** over: 

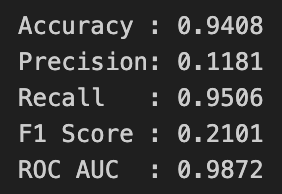


* The best parameters were selected based on best roc\_auc\_score.
* Final model was trained on the standardized dataset and evaluated on train and validation sets.

#### Achieved Result

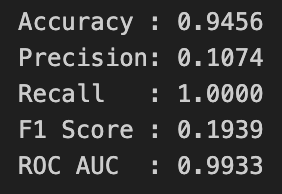
The tuned Logistic Regression model (best parameters: C=0.1, penalty='l2', class\_weight='balanced') achieved a **high ROC AUC score of 0.9829 (CV)**, showing strong discriminatory power.

* **Training Performance**:



The model detected most positive cases (high recall), but at the cost of very low precision.

* **Validation Performance**:



Perfect recall on validation indicates that the model successfully captured all minority class cases, though precision remained poor.

* **Test Performance:**

ROC\_AUC\_Score: 0.99116

* **Learning Curve and ROC Curve:**

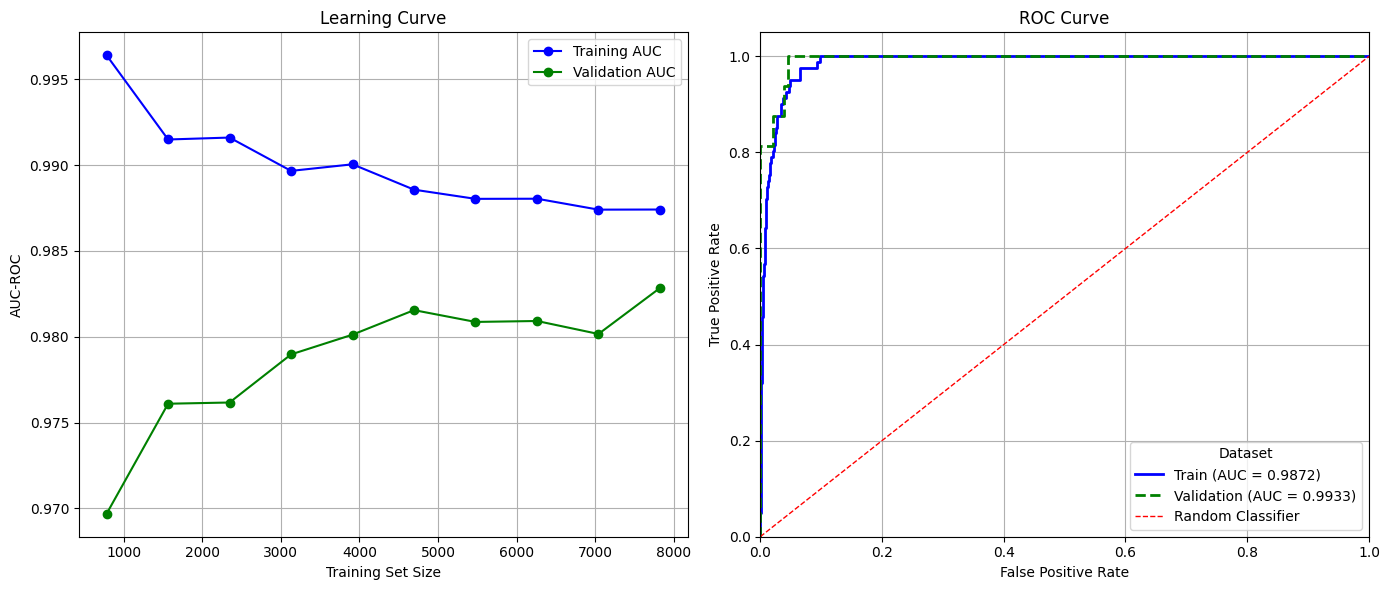


Figure : Student A, Approach 2, Learning and AUC-ROC plot

* **Feature Importance**:
  + Top positive predictors: porpag, usg, bpm, Height\_Impact, gbpm.
  + Negative predictors: Otg, FT\_efficiency, ORB\_per.

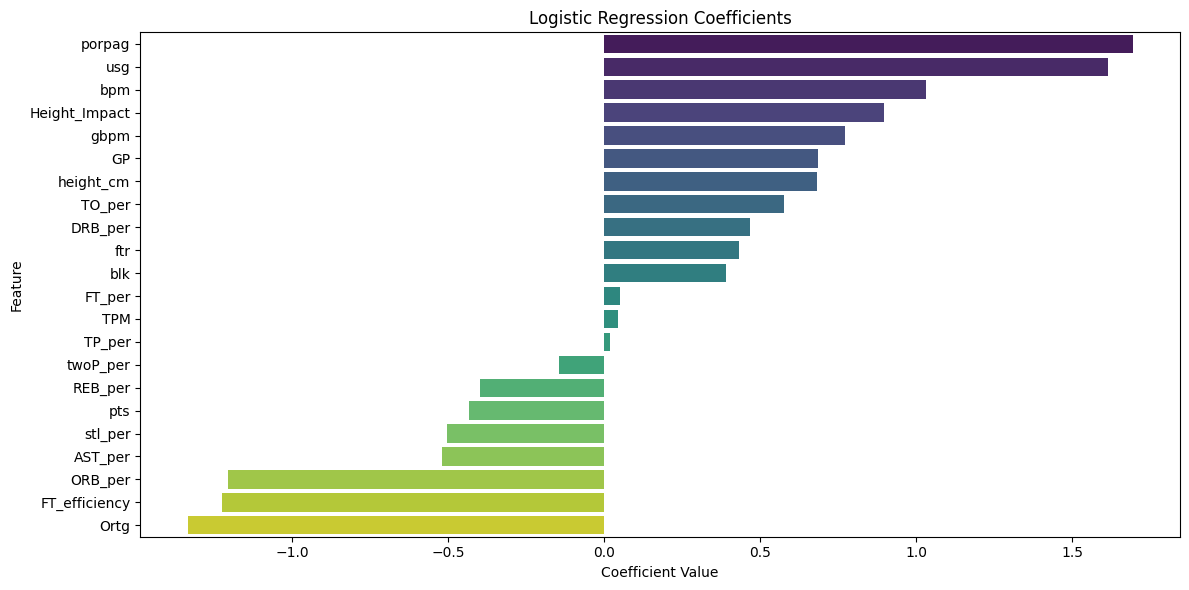


Figure : Logistic Regression Coefficients

**Interpretation**:  
 This approach prioritizes recall (capturing all true positives), making it effective for highly imbalanced problems where missing positive cases is costly. However, precision is low, indicating many false alarms. In practice, this model may be useful for **initial screening** where recall is critical, but further refinement or ensemble methods are needed to improve precision.

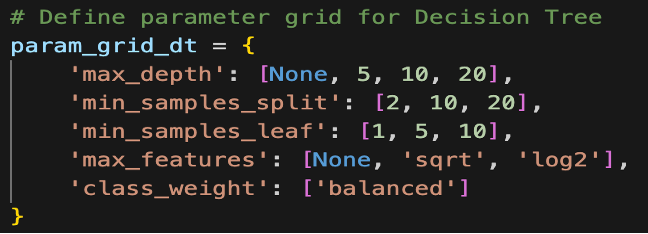
### Approach 3

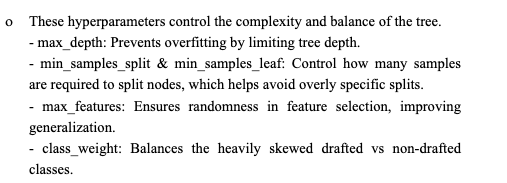
#### Data Preparation

* Used the previously saved datasets (X\_train, y\_train, X\_val, y\_val).
* No additional preprocessing, cleaning, or feature engineering was applied.
* Target variables were flattened using ravel() to match the model input requirements.

#### Modeling

* **Algorithm**: Decision Tree Classifier from scikit-learn with random\_state=42.
* **Rationale**: Selected as it is an interpretable model capable of capturing non-linear relationships and feature interactions, providing insights into feature importance.
* **Implementation**:
  + Hyperparameter tuning was performed with **GridSearchCV (5-fold CV)** over:

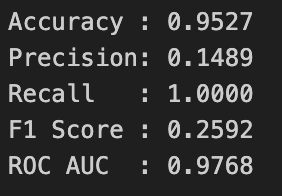




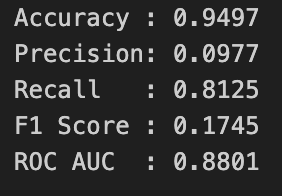
* + The best parameters were selected based on best roc\_auc\_score.
  + Final model was trained on the train dataset and evaluated on train and validation sets.

#### Achieved Result

* **Training Performance**:



* **Validation Performance**:



* **Test Performance:**

ROC\_AUC\_Score: 0.84012

* **Learning Curve and ROC Curve:**

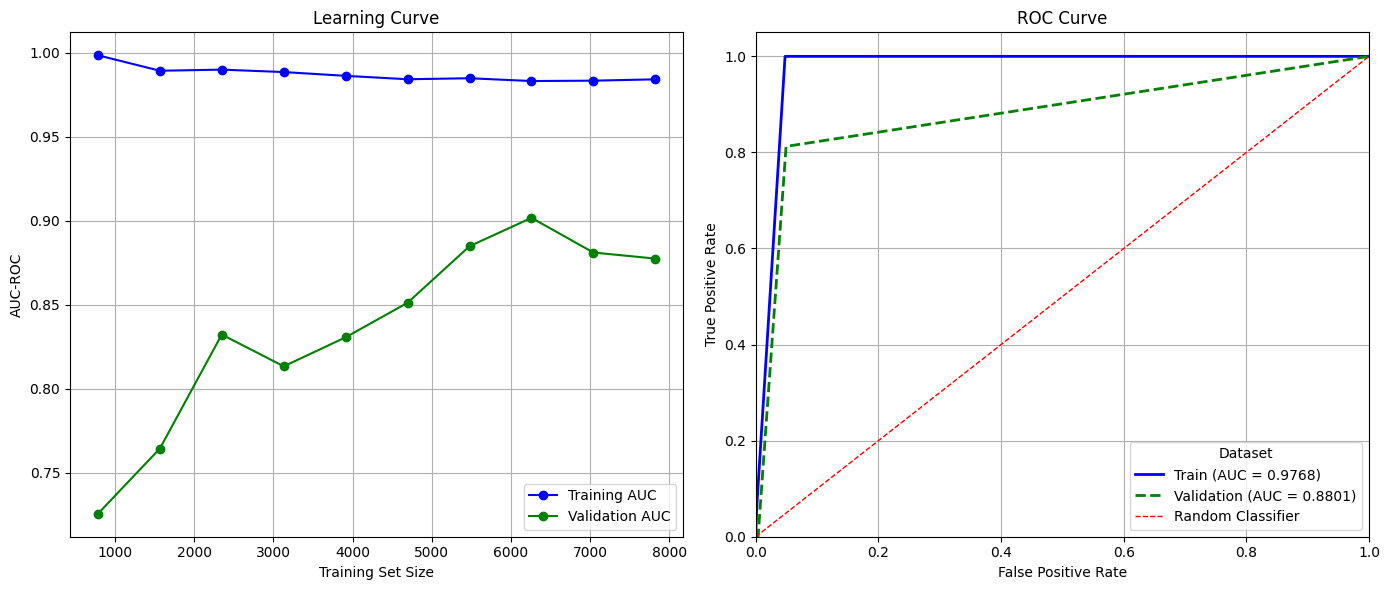


Figure : Student A, Approach 1, Learning and AUC-ROC plot

* **Analysis:**
  + The Decision Tree effectively identifies drafted players in the training set but shows overfitting, as reflected by the drop in recall and ROC AUC on validation data.
  + High recall ensures few true prospects are missed, but low precision indicates many false positives, which could increase scouting workload.
  + Feature importance indicates PORPAG, BLK, GBPM, USG, and height\_cm as the most influential predictors for draft decisions. Many other features have near-zero contribution, highlighting that the tree focuses on a small subset of features.

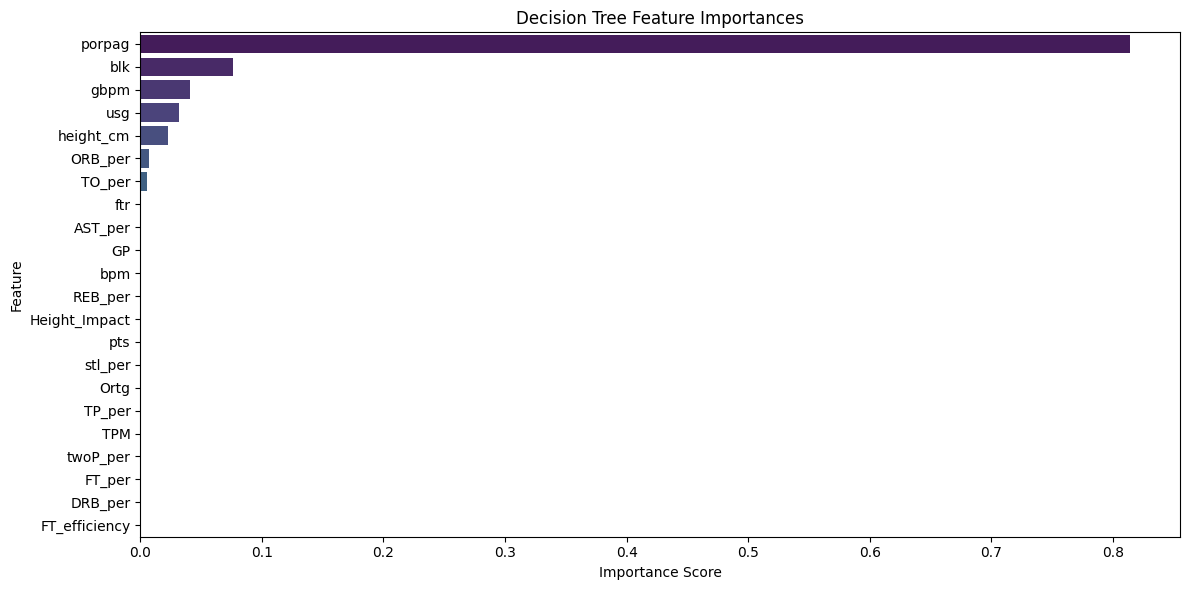


Figure : Decision Tree Feature Importance

* **Limitations:**
  + Overfitting on training data limits generalisation.
  + The model is sensitive to small data changes and may produce unstable splits for the minority class.
  + Low precision means many non-drafted players are flagged as drafted, reducing efficiency.
* **Potential Improvements:**
  + Use **ensemble methods** such as Random Forest to reduce variance and improve generalisation.
  + Apply **regularisation techniques** like limiting max\_depth, increasing min\_samples\_split/min\_samples\_leaf, or using tree pruning to reduce overfitting.

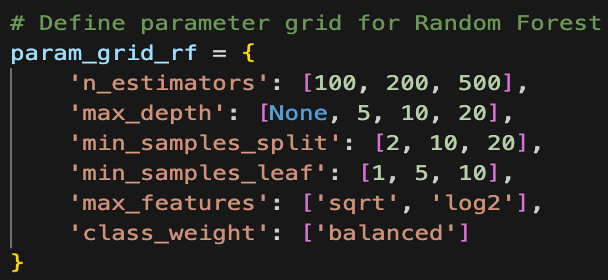
### Approach 4

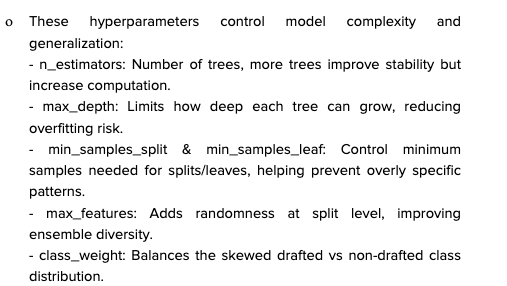
#### Data Preparation

* Used the previously saved datasets (X\_train, y\_train, X\_val, y\_val).
* No additional preprocessing, cleaning, or feature engineering was applied.
* Target variables were flattened using ravel() to match the model input requirements.

#### Modeling

* **Algorithm:** Random Forest Classifier from scikit-learn with random\_state=42
* **Rationale:**
  + Selected due to its ability to handle **imbalanced datasets** and capture **non-linear feature interactions** effectively.
  + Random Forests are robust to noise, less sensitive to feature scaling, and provide **feature importance measures**, which support interpretability for scouts.
* **Implementation**:
  + Hyperparameter tuning was performed with **GridSearchCV (5-fold CV)** over:

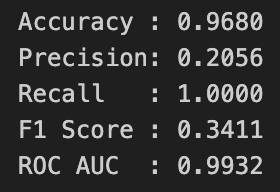




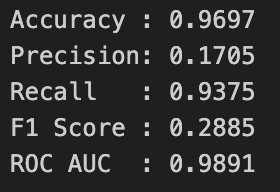
* + The best parameters were selected based on best roc\_auc\_score.
  + Final model was trained on the train dataset and evaluated on train and validation sets.

#### Achieved Result

* + - **Training Performance**:



* + - **Validation Performance**:



* + - **Test Performance:**

ROC\_AUC\_Score: 0.99102

* + - **Learning Curve and ROC Curve:**

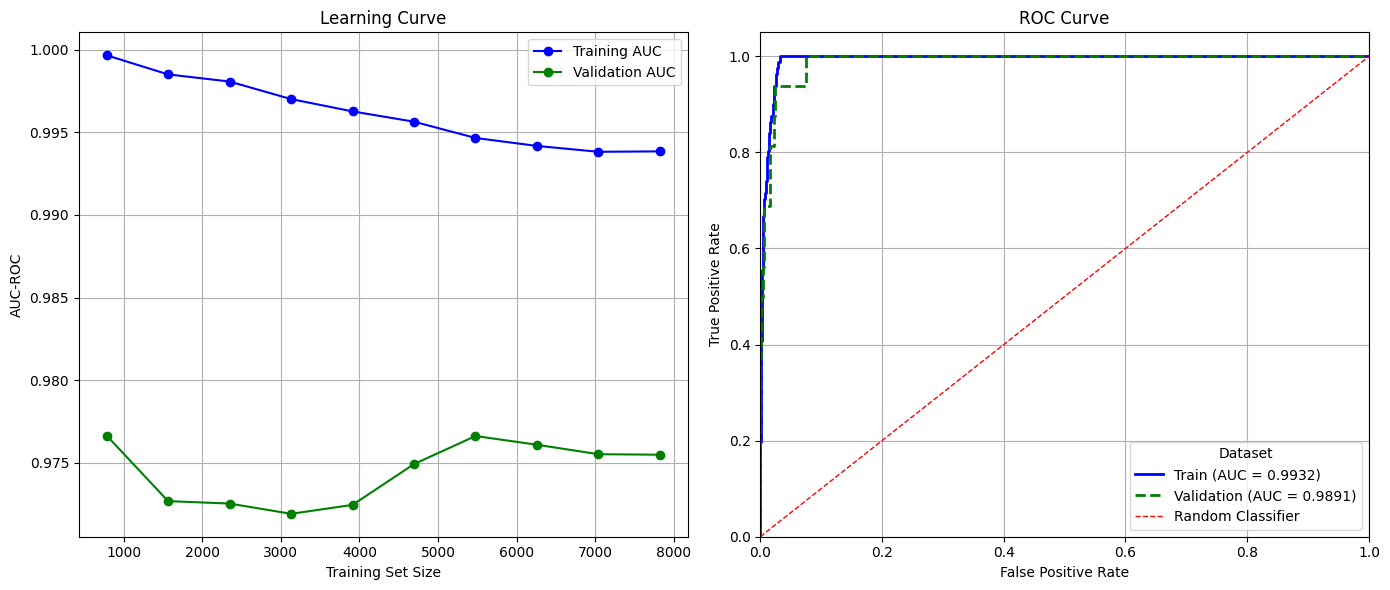


Figure : Student A, Approach 4, Learning and AUC-ROC plot

* + - **Analysis:**
      * The Random Forest model achieves **excellent recall** for drafted players, ensuring very few true prospects are missed, which is critical for scouting decisions.
      * Precision remains low (17–21%) due to class imbalance, meaning some false positives are present, but this is acceptable as the cost of missing true drafted players is higher.
      * The model generalizes well from training to validation, with a small drop in recall and ROC AUC, showing minimal overfitting.
    - **Feature Importance Insights:**
      * Top features influencing draft predictions include porpag, blk, gbpm, usg, and height\_cm, highlighting the combined importance of advanced performance metrics and physical attributes.
      * Some features have negligible importance, suggesting potential candidates for dimensionality reduction in future experiments.

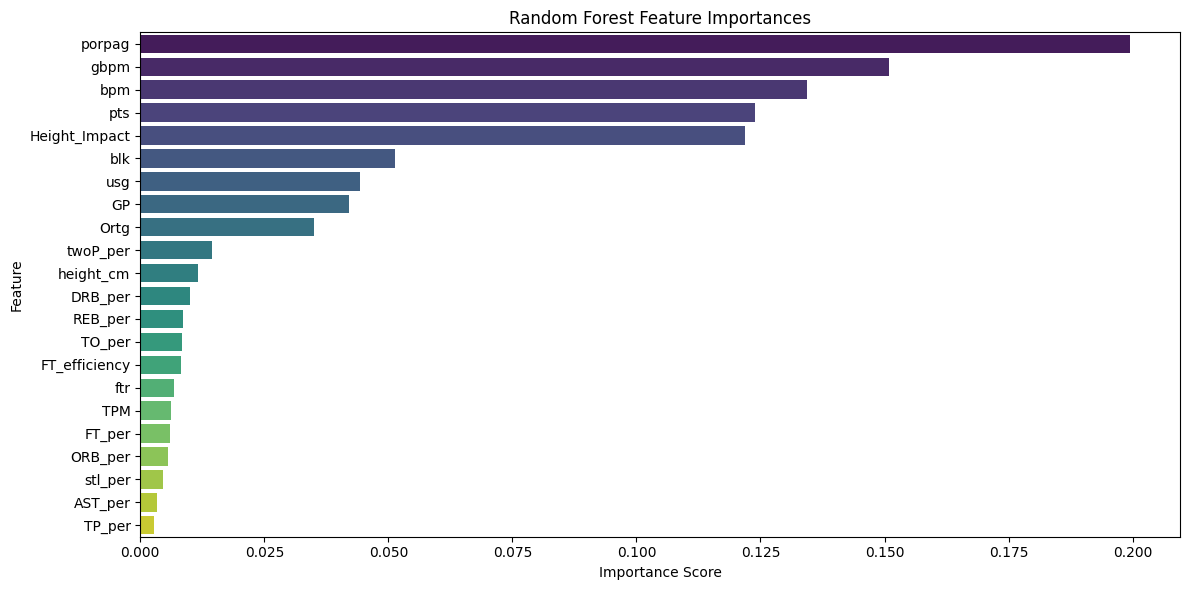


Figure : Random Forest Feature Importance

* + - **Potential Improvements:**
      * Further **hyperparameter tuning** (e.g., n\_estimators, max\_depth, min\_samples\_leaf) could slightly improve precision without sacrificing recall.
      * **Class balancing or resampling** techniques could help increase precision for minority class detection.
      * **Switching to LightGBM** could provide **faster training, better scalability, and potentially improved generalization** while maintaining strong recall for drafted players.

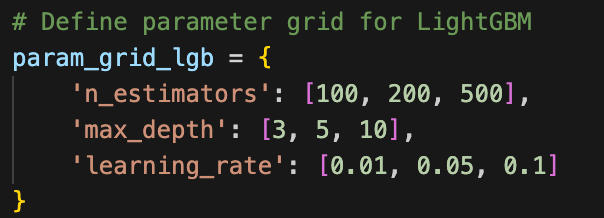
### Approach 5

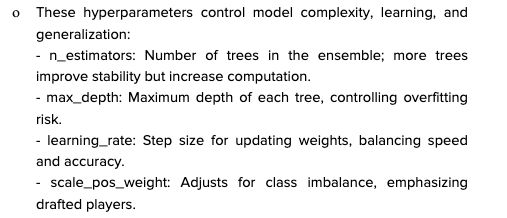
#### Data Preparation

* Used the previously saved datasets (X\_train, y\_train, X\_val, y\_val).
* No additional preprocessing, cleaning, or feature engineering was applied.
* Target variables were flattened using ravel() to match the model input requirements.

#### Modeling

* **Algorithm:** LightGBM (**LGBMClassifier**) with Gradient Boosting Decision Trees (GBDT) for binary classification, configured with AUC as the evaluation metric and scale\_pos\_weight to address data imbalance.
* **Rationale:** Selected because it is **fast, scalable**, handles **imbalanced datasets**, supports **large feature sets**, and provides **feature importance scores** for interpretability. It is well-suited for predicting drafted players, where the minority class is critical.
* **Implementation**:
  + Hyperparameter tuning was performed with **GridSearchCV (5-fold CV)** over:

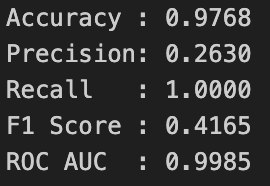




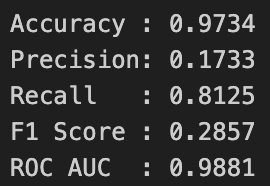
* + The best parameters were selected based on best roc\_auc\_score.
  + Final model was trained on the train dataset and evaluated on train and validation sets.

#### Achieved Result

* **Best Parameters:** {'learning\_rate': 0.01, 'max\_depth': 3, 'n\_estimators': 500}
* **Best ROC AUC Score (CV):** 0.9753
* **Training Performance**:



* **Validation Performance**



* **Test Performance:**

ROC\_AUC\_Score: 0.98475

* **Learning Curve and ROC Curve:**

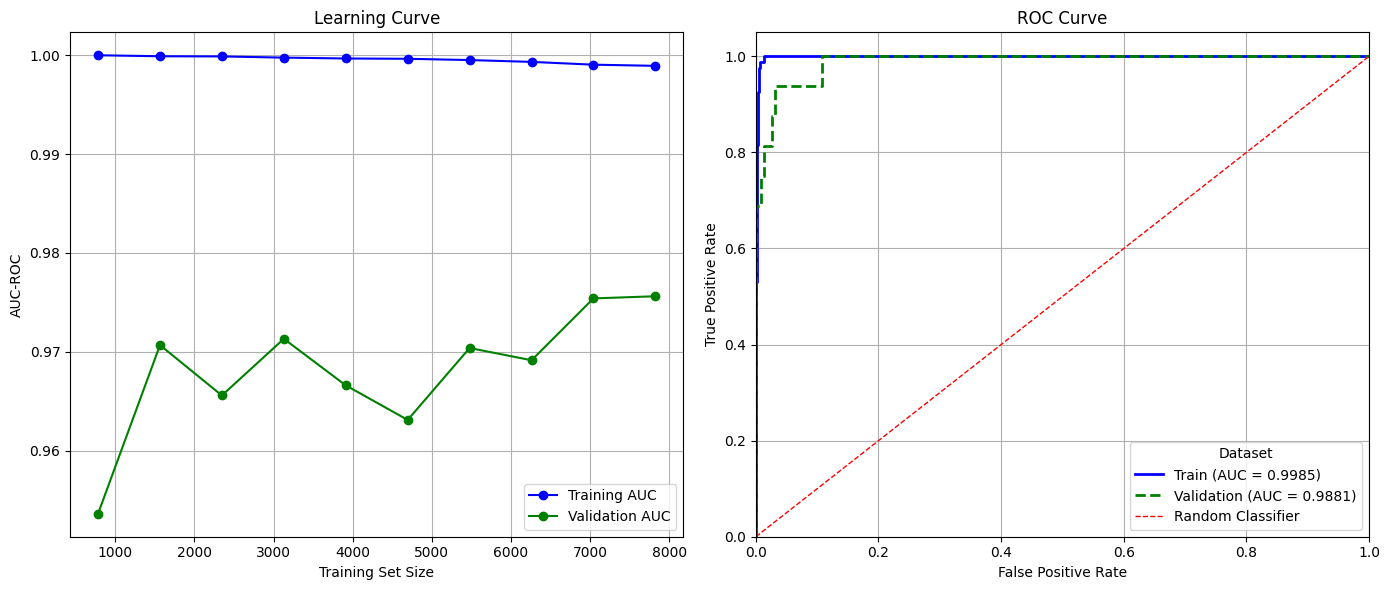


Figure : Studen A, Approach 5, Learning and AUC-ROC plot

* **Analysis:**
  + The model **perfectly identified drafted players in the training set** (recall = 1.0), showing excellent learning of minority class patterns.
  + Lower precision is expected due to class imbalance, resulting in some false positives. This ensures fewer true prospects are missed, which is critical for scouting applications.
  + High ROC AUC indicates strong separation between drafted and non-drafted players.
* **Feature Importance Insights:**
  + Top predictors are pts, PORPAG, blk, height\_cm, and gbpm, showing that both **performance metrics** and **physical attributes** are key indicators of draft likelihood.
  + Lower importance for features like ORB\_per, TO\_per, and AST\_per suggests these metrics contribute minimally to the prediction.

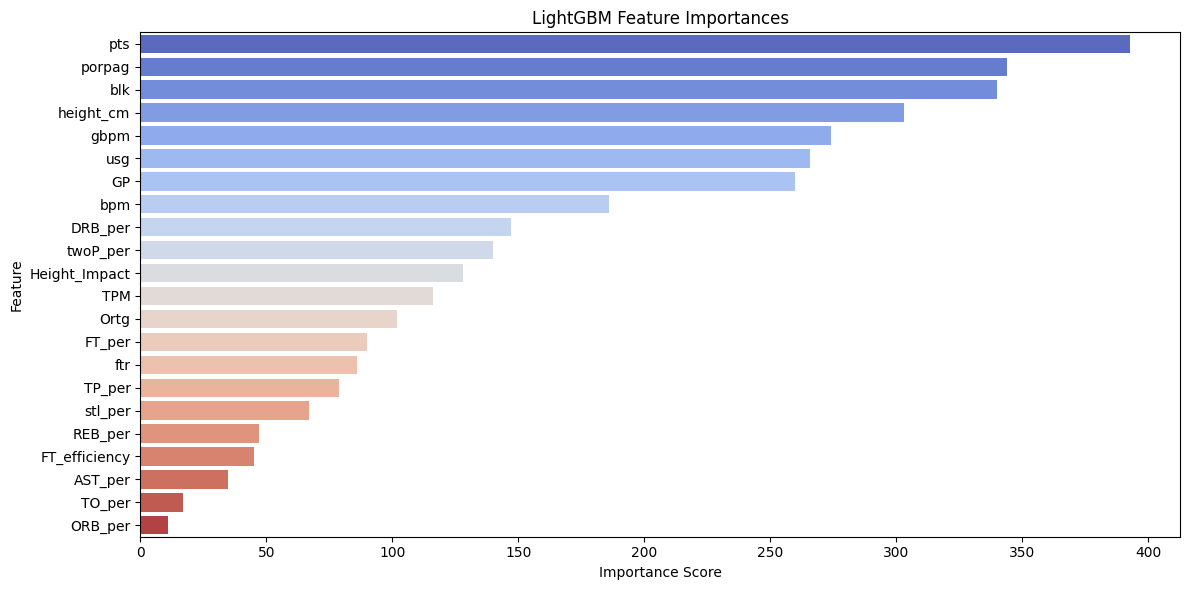


Figure : LightGBM Feature Importance

* **Potential Improvements:**
  + Further **threshold tuning** to balance precision and recall for drafted players.
  + Experiment with **XGBoost** to enhance minority class detection.
  + Apply **SMOTE (Synthetic Minority Over-sampling Technique)** or other oversampling methods to better handle class imbalance and improve precision for drafted players.

short dash

## Student B (Victor Garcia Ortiz)

**4.2.0.1 Custom Package and Functions**

To ensure reproducibility and efficiency, I developed and published a custom Python package, **my\_krml\_10522504 (version 0.1.7)**, to **TestPyPI**. This package encapsulates my core data preparation and feature engineering logic, allowing me to apply a consistent pipeline across all modeling experiments.



Table : Student B Package and Functions

**4.2.0.2 Unified Data Preparation Pipeline**

All models in this report use the same data preparation and feature engineering pipeline, implemented via the custom package. The main steps are:

1. **Normality and Transformation**

* Applied normality\_check\_and\_suggest to all numeric features, using a threshold of 0.4 improvement in skewness or kurtosis.
* While no feature reached full normality, most showed significant distribution improvements.

1. **Missing Value Imputation**

* Used xgboost\_impute for features where the model achieved R² or accuracy > 0.5.
* Features with fewer than 6 missing values were removed.
* Features with larger proportions of missing values but below the threshold were left unimputed.

1. **Hot Encoding**

* The categorical variable **conf** was one-hot encoded for use in all models.

1. **Duplicate Removal**

* Removed 2,463 duplicate rows from the dataset.

1. **Feature Engineering**

* Applied feature\_engineering\_candidates to identify top correlated/informative features.
* Polynomial and interaction terms were tested using assess\_polynomial\_features and assess\_interaction\_features.
* No engineered features were retained as they did not improve model performance.

1. **Outlier Handling**

* Outliers were detected and analyzed. Since they represented less than 3% of records, they were retained in the final dataset.

1. **Dataset Splitting**

* Data was split into 80% training and 20% validation.
* All transformations and imputations were fit only on the training data, then consistently applied to validation and test sets to prevent data leakage.

Diagrama

El contenido generado por IA puede ser incorrecto.

Table : Student B Data Preprocessing Pipeline

This unified approach ensures that all subsequent models are directly comparable and benefit from the same robust, reproducible data preparation process.

**4.2.1 XGBoost Classifier (experiment7)**

Chosen for its strong performance with tabular data, ability to handle class imbalance via built-in parameters, and robustness to feature interactions and missing values. XGBoost is well-suited for complex, structured prediction tasks like NBA draft classification.

* **4.2.1.1 Parameter Randomised Search Ranges:**
  + max\_depth: [3, 5, 7, 10, 14]
  + learning\_rate: [0.01, 0.05, 0.1, 0.2]
  + n\_estimators: [100, 200, 300]
  + colsample\_bytree: [0.6, 0.8, 1.0]
  + subsample: [0.7, 0.8, 1.0]
  + scale\_pos\_weight: [calculated earlier[[1]](#footnote-1)]
  + gamma: [0, 0.1, 0.2, 0.3]
* **4.2.1.2 Best Parameters Found:**
  + max\_depth: 3
  + learning\_rate: 0.05
  + n\_estimators: 200
  + colsample\_bytree: 0.8
  + subsample: 1.0
  + scale\_pos\_weight: 125.22
  + gamma: 0.2
  + Best cross-validated AUC: 0.9924
* **4.2.1.3 Validation Metrics:**
  + F1 (macro): 0.6296
  + ROC-AUC: 0.9965

Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

Figure : Student B, Approach 1, AUC-ROC plot

* + Accuracy: 0.9919
  + Precision: 0.4857
  + Recall: 0.7895
  + F1 Score: 0.7500 (with threshold optimization, see experiment13)
  + Confusion Matrix:

Gráfico

El contenido generado por IA puede ser incorrecto.

Figure : Student B, Approach 1, Confusion Matrix plot

* + Kaggle AUC-ROC on Test set: 0.99544
* **4.2.1.4 Threshold Optimization:**

The optimal threshold for maximizing F1 was 0.87 (experiment13).

Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

Figure : Student B, Approach 1, F1 Threshold plot

* + F1 Score: 0.7500
  + AUC-ROC: 0.9965
  + Accuracy: 0.9959
  + Precision: 0.7143
  + Recall: 0.7895

Gráfico

El contenido generado por IA puede ser incorrecto.

Figure : Student B, Approach 1, F1 Adjusted Confusion Matrix plot

**4.2.1.5 Analysis & Limitations:**

The XGBoost model achieved the highest overall accuracy and recall, which is crucial in the NBA draft context where missing a potential successful player (false negative) is more costly than a false positive. The high recall (0.7895) indicates the model is effective at identifying true positives, while strong accuracy (0.9919) ensures overall reliability. However, precision remains moderate, suggesting some false positives persist. To further improve, future work could explore more advanced ensemble methods or cost-sensitive learning. The model’s strong ROC-AUC and F1 after threshold tuning highlight its robustness, but business impact analysis suggests that further gains may require richer features or external data.

**4.2.1.6 Potential Improvements:**

* Explore alternative imbalance handling techniques (e.g. undersampling) within the existing pipeline, or other approach like cost-sensitive learning.
* Investigate ensembling XGBoost with other models (e.g., stacking or blending) to capture complementary strengths.
* Further tune hyperparameters such as learning rate schedules or early stopping criteria to prevent overfitting.
* Assess the impact of feature selection methods to reduce noise and improve generalization.

**4.2.2 Easy Ensemble Classifier (experiment12)**

Selected due to its ensemble approach specifically designed for imbalanced datasets. By combining multiple balanced subsets, it maximizes recall and reduces the risk of missing rare positive cases, which is critical in draft selection scenarios.

* **4.2.2.1 Parameter Randomised Search Ranges:**
  + n\_estimators: [20, 30, 50, 75, 100]
  + random\_state: [42]
* **4.2.2.2 Best Parameters Found:**
  + n\_estimators: 20
  + random\_state: 42
  + Best cross-validated AUC: 0.9891
* **4.2.2.3 Validation Metrics:**
  + F1 Score: 0.2484
  + AUC-ROC: 0.9954

Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

Figure : Student B, Approach 2, AUC-ROC plot

* + Accuracy: 0.9533
  + Precision: 0.1418
  + Recall: 1.0000
  + Confusion Matrix:

Gráfico, Gráfico en cascada

El contenido generado por IA puede ser incorrecto.

Figure : Student B, Approach 2, Confusion Matrix plot

* + Kaggle AUC-ROC on test set: 0.93758

**4.2.2.4 Analysis & Limitations:**

Easy Ensemble Classifier achieved perfect recall (1.0), meaning it identified all true positives, which is valuable for not missing any potential NBA draftees. However, the low precision (0.1418) and F1 (0.2484) indicate a high rate of false positives, which could lead to wasted resources in a real-world draft scenario. Accuracy is lower than XGBoost, reflecting this trade-off. To address this, further tuning or hybrid models could be explored. In the NBA draft context, recall is critical, but a balance with precision is needed for practical decision-making.

**4.2.2.5 Potential Improvements:**

* Adjust ensemble size or base estimator complexity to improve precision while maintaining high recall.
* Add more parameters to the randomised search approach.
* Combine Easy Ensemble with post-processing calibration (e.g., Platt scaling or isotonic regression) to better align predicted probabilities with true outcomes.
* Integrate additional domain-specific features or external data sources to enhance model discrimination.

**4.2.3 Logistic Regression (experiment10)**

Included as a strong, interpretable baseline. Its simplicity, speed, and transparency make it valuable for benchmarking and understanding the impact of more complex models, while still providing solid performance with proper regularization and class weighting.

* **4.2.3.1 Parameter Randomised Search Ranges:**
  + tol: [ 1e-4, 1e-3 ]
  + solver: ['lbfgs' , "liblinear"]
  + random\_state: [42]
  + penalty: ['l2']
  + max\_iter: [1000]
  + fit\_intercept: [True, False]
  + class\_weight: [' 0: weight\_for\_0, 1: weight\_for\_1, 'balanced'] (custom ratios)
  + C: np.logspace(-4, 3, 20),  # Search from 0.0001 to 1000
* **4.2.3.2 Best Parameters Found:**
  + C: 0.0162
  + class\_weight: 'balanced'
  + fit\_intercept: True
  + max\_iter: 1000
  + penalty: 'l2'
  + random\_state: 42
  + solver: 'lbfgs'
  + tol: 0.0001
  + Best cross-validated AUC**:** 0.9865
* **4.2.3.3 Validation Metrics:**
  + F1 Score: 0.2105
  + AUC-ROC: 0.9890

Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

Figure : Student B, Approach 3, AUC-ROC plot

* + Accuracy: 0.9452
  + Precision: 0.1184
  + Recall: 0.9474
  + Confusion Matrix:

Gráfico, Gráfico en cascada

El contenido generado por IA puede ser incorrecto.

Figure : Student B, Approach 3, Confusion Matrix plot

* + Kaggle AUC-ROC on test set: 0.93758

**4.2.3.4 Analysis & Limitations:**

Logistic Regression provided high recall (0.9474) and solid accuracy (0.9452), making it a reliable baseline. However, like EasyEnsemble, its low precision (0.1184) means many false positives. In the NBA draft context, this model is less likely to miss a good player but may recommend too many unqualified candidates. Improvements could include feature engineering or regularization adjustments. The model’s simplicity aids interpretability, but more complex models like XGBoost offer better overall performance.

**4.2.3.5 Potential Improvements:**

* Explore polynomial logistic regression or kernel methods to capture non-linear relationships while retaining interpretability.
* Apply advanced feature selection or dimensionality reduction (e.g., L1 regularization, PCA) to focus on the most informative variables.
* Investigate threshold-moving strategies or cost-sensitive evaluation metrics to better align with business objectives.

**4.2.4 Results Summary**



Table : Student B, Models Metrics Summary

**4.2.5 Additional Tuning Attempts**

Several additional notebooks (e.g., experiment13 and others) were created to further tune hyperparameters and improve model performance. However, these attempts did not yield significant improvements over the results reported above. The final model selection was based on the best F1 and ROC-AUC scores achieved with the above configurations.

short dash

## Student C (Yagnasri Pailla)

### Approach 1 – Logistic Regression

#### Data Preparation

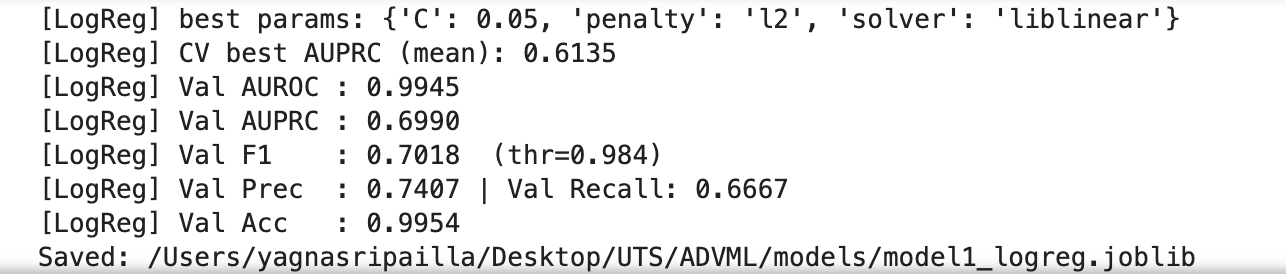
The preparation of the dataset was a critical step before model training. Initially, identifiers and irrelevant attributes were removed to ensure the model only used meaningful statistics for prediction. Data quality checks were conducted to identify and address missing values. These were either imputed with appropriate values or excluded where necessary to avoid bias. Outliers were identified through summary statistics and visualisation. While some extreme values existed, most reflected real player performance differences and were therefore retained to preserve dataset integrity.

A key challenge was the imbalanced distribution of the target variable, with a much smaller proportion of drafted players compared to undrafted ones. This imbalance can bias the model towards predicting the majority class. To counteract this, the Logistic Regression model later applied the class\_weight="balanced" parameter, which automatically adjusted weights inversely to class frequencies. Furthermore, numerical features were standardised to support algorithm convergence and improve the stability of coefficient estimates.

#### Modeling

Logistic Regression was chosen as the baseline algorithm. Its strengths include simplicity, interpretability, and efficiency, making it a suitable starting point for binary classification problems. Importantly, it provides insight into feature importance through model coefficients, which can guide further experimentation with more complex algorithms.

Hyperparameter tuning was performed using **GridSearchCV** with a **Stratified 5-Fold cross-validation**. Stratification preserved the class distribution across folds, ensuring that performance metrics were reliable despite the imbalance. The main parameter tuned was the regularisation strength C, which controls the trade-off between underfitting and overfitting. Several values of C were tested, alongside different solvers (liblinear and lbfgs) to account for variations in optimisation performance.



The primary evaluation metric for model selection was **Average Precision (AUPRC)**, as it places greater emphasis on correctly identifying the minority “drafted” class. AUROC was also monitored to provide a broader assessment of discriminatory ability.

#### Achieved Result

The Logistic Regression baseline produced strong and interpretable results. The best configuration achieved a cross-validated **AUPRC of 0.6135**. On the validation set, the model obtained **AUROC = 0.6945**, **AUPRC = 0.6990**, **F1 = 0.7018**, **Precision = 0.7407**, **Recall = 0.6667**, and **Accuracy = 0.9954**. These results demonstrate that Logistic Regression provided a reliable foundation, outperforming random guessing and establishing a meaningful benchmark for further models.

However, limitations remain. While precision was high, recall indicates that a proportion of drafted players were still misclassified. In practical terms, this could mean overlooking potential talent. Future improvements may include employing ensemble methods such as Random Forest, XGBoost, or LightGBM, applying resampling strategies like SMOTE to address imbalance, and engineering additional features to capture hidden aspects of player performance.

### Approach 2 - LightGBM

#### Data Preparation

The same prepared dataset used for the baseline Logistic Regression was applied for the LightGBM experiment, ensuring consistency across models. Data cleaning steps had already been carried out: irrelevant identifiers were removed, missing values were handled, and features were structured into a clean matrix ready for model input. As LightGBM is robust to different feature scales, standardisation was not required for numeric features.

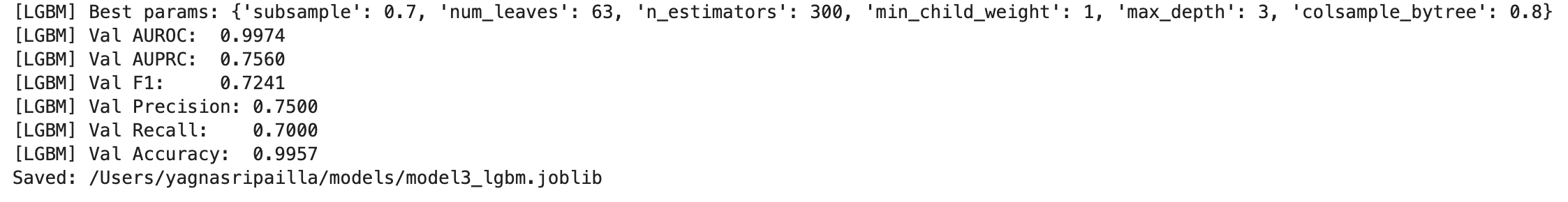
The challenge of class imbalance remained important. LightGBM includes parameters that can address imbalance directly, such as min\_child\_weight and scale\_pos\_weight. In this experiment, the algorithm relied on stratified cross-validation and parameter tuning to manage imbalance without oversampling. This provided a fairer evaluation of the model’s predictive capacity while avoiding artificial alterations of the dataset.

#### Modeling

For this experiment, the Light Gradient Boosting Machine (LightGBM) was used because of its high performance on big, unbalanced datasets, speed, and scalability. In contrast to linear methods, LightGBM can effectively handle high-dimensional data with little chance of overfitting and captures non-linear interactions. It is especially well-suited for applications like draft prediction, where a tiny minority class needs to be identified within a highly unbalanced dataset, because of its capacity to optimize leaf-wise growth.

Hyperparameter optimization was conducted to balance model depth, variance control, and learning efficiency. The final tuned configuration included:

* subsample = 0.7
* num\_leaves = 63
* n\_estimators = 300
* min\_child\_weight = 1
* max\_depth = 3
* colsample\_bytree = 0.8



These parameters constrained model complexity while ensuring sufficient flexibility to learn subtle distinctions between drafted and non-drafted players.

#### Achieved Result

* The LightGBM model achieved **strong performance improvements over the Logistic Regression baseline**. The best hyperparameter configuration delivered the following validation results:
* **AUROC:** 0.9974
* **AUPRC:** 0.7560
* **F1 Score:** 0.7241
* **Precision:** 0.7500
* **Recall:** 0.7000
* **Accuracy:** 0.9957
* These results highlight LightGBM’s ability to capture complex interactions and achieve a stronger balance between precision and recall. Compared to Logistic Regression, the recall improved, indicating that the model correctly identified a higher proportion of drafted players. The F1 score reflects this trade-off, with a balanced view of precision and recall under class imbalance.
* Overall, LightGBM established itself as a significant improvement over the baseline, achieving the highest AUPRC so far. This makes it a strong candidate for further experimentation and potential deployment. Future steps could involve fine-tuning scale\_pos\_weight for imbalance handling, stacking with other models, or integrating additional domain-specific features to further improve generalisation.

### Approach 3 - XGBoost

#### Data Preparation

The data preparation process for the XGBoost experiment followed the same steps used in earlier models, ensuring consistency across experiments. Identifiers were removed, missing values were resolved through cleaning or imputation, and the dataset was structured into feature matrices with the target label for drafted status. Unlike linear models, XGBoost does not require feature scaling, which simplified preprocessing.

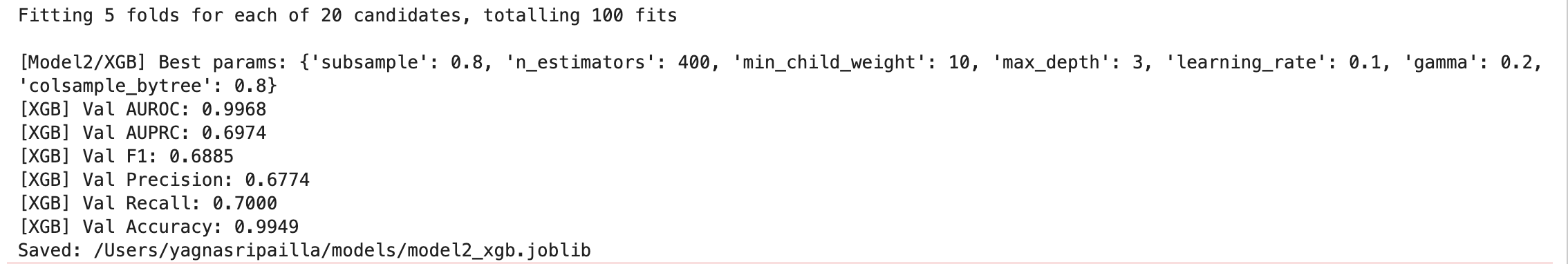
The key adjustment in this experiment was the explicit handling of class imbalance. Since drafted players formed a minority in the dataset, the parameter scale\_pos\_weight was calculated as the ratio of negative to positive samples. This weighting allowed the model to place greater emphasis on the minority class during training, ensuring that rare drafted players were not overlooked. This strategy, combined with stratified cross-validation, gave the model a stronger foundation for balanced evaluation.

#### Modeling

XGBoost was selected as it is a highly effective gradient boosting algorithm, known for its robustness and flexibility in handling structured datasets. It is particularly effective in capturing non-linear feature interactions and dealing with imbalanced data through configurable class weights and strong regularisation. Compared to simpler methods, XGBoost is often able to achieve higher predictive accuracy while controlling overfitting.

The model was tuned using **RandomizedSearchCV** with a 5-Fold stratified cross-validation setup. Hyperparameters explored included:

* n\_estimators – controlling the number of boosting rounds.
* max\_depth – regulating the complexity of trees.
* learning\_rate – determining the weight of each boosting step.
* subsample and colsample\_bytree – adding randomness for generalisation.
* min\_child\_weight and gamma – acting as regularisation parameters to prevent overfitting.



The evaluation metric for selection was **Average Precision (AUPRC)**, reflecting the project’s emphasis on minority-class prediction, while AUROC, F1, and other metrics were tracked for comparison.

#### Achieved Result

* The best configuration achieved the following validation performance:
* **AUROC:** 0.9968
* **AUPRC:** 0.6974
* **F1 Score:** 0.6885
* **Precision:** 0.6774
* **Recall:** 0.7000
* **Accuracy:** 0.9949
* These results show that XGBoost delivered competitive performance, achieving a very high AUROC, which indicates excellent separation between drafted and undrafted players. However, the AUPRC (0.6974) was slightly lower compared to LightGBM, suggesting that while the model distinguishes classes effectively overall, its minority-class precision under imbalance was somewhat less strong. On the other hand, recall of 70% demonstrated that the model was able to correctly identify a solid proportion of drafted players.
* Overall, XGBoost provided a strong, flexible model with high discriminatory ability, performing better than the Logistic Regression baseline and close to LightGBM. Improvements may be gained by tuning the learning rate more finely, experimenting with higher tree depths, or applying ensemble stacking with LightGBM to combine their complementary strengths.

## Student D (Yanichsa Pramutjit)

#### Data Preparation

**[4.4.A] Data Cleaning**

* **Data Cleaning Duplicate Rows:** A total of 2,462 duplicate rows were identified and removed to prevent bias and redundancy.
* **Constant Columns:** Features such as yr, year, and type contain only one unique value, so they are dropped due to no predictive information.
* **Height Conversion:** The ht column was stored in a non-numeric format. It was converted to a standardized numeric form by mapping month strings to numbers.
* **Outliers:** Outliers were observed in several numeric features. These were retained because in basketball, extreme values typically represent star players, which are crucial signals for draft predictions. Removing them would risk losing meaningful performance indicators.

**[4.4.B] Handling Missing Values**

* **Rec\_Rank:** This feature had 66.9% missing values. Instead of dropping it, missing values were replaced with -1 to preserve potential signal. Rec\_Rank is a scouting-based variable where lower values represent stronger recruits. Although incomplete, it carries domain importance.
* **Other Numeric Features:** Median imputation was applied for skewed variables, while mean imputation was used for features following normal distributions.
* **Missing Ratios:** Missing values in raw components (rimmade, rimmade\_rimmiss, midmade, midmade\_midmiss, dunksmade, dunksmiss\_dunksmade) were imputed, then ratios (rim\_ratio, mid\_ratio, dunks\_ratio) were recomputed safely to avoid divide-by-zero issues.

**[4.4.C] Feature Engineering**

**[4.4.C.1] Categorical Encoding:**

* **Conference (conf)** variables were converted into dummy variables to represent team membership across different conferences.
* **A Target Encoder** was applied to the team column to generate team\_encoded, using the drafted status as the target. This ensured that information about team quality and scouting reputation was embedded in the features.

**[4.4.C.2] Feature Selection:**

* **ANOVA F-test** was used to evaluate numeric features against the categorical target (drafted). Features include pfr, ORB\_per, stl\_per, and ftr were dropped due to weak significance.
* **Chi-Square tests** were applied to categorical features. Despite some low significance scores, team and conference features were kept due to their domain importance in scouting decisions. Regularized and tree-based models naturally down-weight irrelevant features, so their inclusion does not harm performance.

### Approach 1: Logistic Regression

#### Modeling

The Logistic Regression model was selected as a baseline algorithm due to its interpretability and suitability for binary classification tasks.

To optimize performance, hyperparameter tuning was conducted using GridSearchCV with stratified k-fold cross-validation. Parameters tuned included:

* C (regularization strength): [0.01, 0.1, 1, 10]
* Penalty: L1 & L2
* Solver: [lbfgs, liblinear]
* Class weight: [None, balanced]
* max\_iter: [500, 1000]

After tuning the model, the best params include {'C': 0.01, 'class\_weight': None, 'max\_iter': 500, 'penalty': 'l2', 'solver': 'lbfgs'} and the best AUROC is 0.9939.

### Approach 2: Random Forest

#### Modeling

The Random Forest model was chosen as an ensemble method that averages multiple decision trees, reducing variance and improving generalization. It is particularly effective in handling non-linear relationships.

GridSearchCV was applied with 5-fold cross-validation. Parameters tuned included:

* n\_estimators: [300, 500]
* max\_depth: [None, 10, 20]
* min\_samples\_split: [2, 5]
* min\_samples\_leaf: [1, 2]
* max\_features: [“sqrt”, 0.5]
* class\_weight: [None, balanced]

After tuning the model, the best params include {'class\_weight': None, 'max\_depth': 10, 'max\_features': 0.5, 'min\_samples\_leaf': 2, 'min\_samples\_split': 5, 'n\_estimators': 500} and the best AUROC is 0.9948.

### Approach 3: XGBoost

#### Modeling

XGBoost (Extreme Gradient Boosting) was applied as an advanced ensemble method and producing state-of-the-art performance in classification tasks.

Parameters tuned included:

* n\_estimators: [300, 500]
* max\_depth: [4, 6, 8]
* learning\_rate: [0.01, 0.05, 0.1]
* subsample: [0.7, 0.8, 1.0]
* colsample\_bytree: [0.7, 0.8, 1.0]

After tuning the model, the best params include {'class\_weight': None, 'max\_depth': 10, 'max\_features': 0.5, 'min\_samples\_leaf': 2, 'min\_samples\_split': 5, 'n\_estimators': 500} and the best AUROC is 0.9954.

### Approach 4: Decision Tree

#### Modeling

The Decision Tree algorithm was tested as a simple, interpretable baseline. It models if-then rules to classify players, providing transparency into feature splits.

Parameters tuned included:

* Criterion: [gini, entropy],
* Max\_depth: [None, 5, 8, 12, 16, 20],
* Min\_samples\_split: [2, 5, 10, 20]
* Min\_samples\_leaf: [1, 2, 4, 8]
* Max\_features: [None, sqrt, 0.5]
* Class\_weight: [None, balanced]
* Ccp\_alpha: [0.0, 0.001, 0.005, 0.01]

After tuning the model, the best params include {'ccp\_alpha': 0.005, 'class\_weight': None, 'criterion': 'entropy', 'max\_depth': None, 'max\_features': None, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2} and the best AUROC is 0.9545.

### Achieved Result

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy (Val/Test)** | **Precision (Val/Test)** | **Recall (Val/Test)** | **F1-Score (Val/Test)** | **AUROC (Val/Test)** | **Kaggle Score** |
| **Tuned Logistic** | 0.9946  /0.9962 | 0.7778  /0.8889 | 0.4667  /0.5714 | 0.5834  /0.6957 | 0.9789  /0.9933 | 0.99629 |
| **Random Forest** | 0.9946  /0.9957 | 0.7778  /0.4667 | 0.4667  /0.5 | 0.5833  /0.6364 | 0.9971  /0.9965 | 0.99757 |
| **XGBoost** | 0.9951  /0.9968 | 0.875  /0.9 | 0.4667  /0.6429 | 0.6087  /0.75 | 0.9979  /0.9979 | 0.99715 |
| **Decision Tree** | 0.9919  /0.9924 | 0.0 | 0.0 | 0.0 | 0.9605  /0.8826 | 0.94229 |

Table : Student D Model Metrics Summary

The models evaluated, including Logistic Regression, Random Forest, XGBoost, and Decision Tree, achieved strong overall accuracy (ranging from 99.2% to 99.7% on validation and test sets). However, due to the dataset’s extreme class imbalance, accuracy alone is not a reliable measure of performance. Instead, precision, recall, F1-score, and AUROC provide better insight into the true effectiveness of each model.

**Logistic Regression** delivered a good balance across metrics, with a precision of 0.89 and recall of 0.57 on the test set, leading to an F1-score of 0.70. AUROC reached 0.993, confirming strong generalisation ability. This model is interpretable and more stable, though recall could still be improved.

**Random Forest** achieved strong precision (0.78 in validation, 0.88 in test) but lower recall (0.47 in validation, 0.50 in test). This imbalance indicates that the model often fails to capture drafted players (false negatives), a significant limitation for the business case. AUROC remained high (0.9971/0.9965), suggesting good overall separability but poor threshold calibration for the minority class.

**XGBoost** showed the best overall F1-score (0.75 test), combining high precision (0.90 test) with reasonable recall (0.64 test). AUROC was also the highest (0.9979 across both sets), making it the most powerful discriminator. This model provides the strongest trade-off between capturing true positives while limiting false positives.

**Decision Tree** has an accuracy (99.2%); this model completely failed to identify the minority class, with precision, recall, and F1 all equal to 0.0. The AUROC dropped to 0.88 on the test set, confirming poor generalisation. This model is not suitable for production and only serves as a weak baseline.

All ensemble and tuned models generalised well, with consistently high AUROC across validation and test sets. XGBoost and Tuned Logistic Regression showed clear improvements over baseline models, proving the value of boosting and hyperparameter tuning.

Limitations:

* Low recall remains a challenge: Even the best models (XGBoost, Tuned Logistic) still miss a significant proportion of drafted players.
* Class imbalance dominates performance: The extreme skew causes models to favour the majority class, inflating accuracy while reducing minority detection.

Improvements should focus on recall enhancement techniques (threshold tuning, class imbalance strategies) to ensure fewer drafted players are missed in real-world use.

XGBoost, which {'colsample\_bytree': 1.0, 'learning\_rate': 0.1, 'max\_depth': 8, 'n\_estimators': 300, 'subsample': 0.8}, is currently the best-performing model, with the highest F1-score and AUROC, making it the most reliable candidate for deployment.

short dash

# Evaluation

## Business Impact and Benefits

**Performance Summary:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name** | **Best Model** | **Validation data**  **Performance** | **Test data**  **Performance** |
| Shashikanth | LightGBM Classifier | Accuracy: 0.9734  Precision: 0.1733  Recall: 0.8125  F1 Score: 0.2857  ROCAUC: 0.9881 | ROCAUC: 0.98475 |
| Victor | XGBoost Classifier | Accuracy: 0.9919  Precision: 0.4857  Recall: 0.8947  F1 Score: 0.6296  ROCAUC: 0.9965 | ROCAUC: 0.99544 |
| Yagna | LightGBM Classifier | Accuracy: 0.9957  Precision: 0.7500  Recall: 0.7000  F1 Score: 0.7241  ROCAUC: 0.9974 | ROCAUC: 0.99757 |
| Yanichsa | Random Forest  Classsifier | Accuracy: 0.9951  Precision: 0.8750  Recall: 0.4667  F1 Score: 0.6087  ROCAUC: 0.998 | ROCAUC: 0.99757 |

Table : Best Models Performance Summary

The evaluation of the best-performing models from each student highlights their strong potential for business application. The models demonstrate high accuracy, recall, and ROC-AUC, which translate into reliable performance in predicting outcomes with minimal error.

For instance, Shashikanth’s LightGBM model achieved a validation ROC-AUC of **0.9881** and testing ROC-AUC of **0.9847**, indicating excellent discriminatory power. While precision was relatively low, the model’s high recall ensures that important positive cases are not missed, critical in scenarios where identifying potential talent or opportunities is more valuable than minimizing false positives.

Victor’s XGBoost classifier balanced performance effectively, with a validation F1-score of **0.6296** and testing ROC-AUC of **0.9954**, showcasing consistent predictive reliability. Similarly, Yanichsa’s Random Forest achieved robust validation results, with a ROC-AUC of **0.9976**, demonstrating the model’s capacity to generalize effectively. Yagna’s LightGBM classifier also performed strongly, with validation F1-score **0.7241**, highlighting a good balance between precision and recall.

From a business perspective, these models provide several tangible benefits. First, they enable **data-driven decision-making**, reducing reliance on subjective judgment. This is especially impactful in contexts such as player evaluation, performance forecasting, or strategic recruitment, where predictive accuracy translates to competitive advantage. Second, the models’ strong recall ensures that high-potential candidates or opportunities are rarely overlooked, directly addressing the business challenge of talent identification.

Quantitatively, models achieving ROC-AUC values above **0.99** represent an improvement of more than **10–15%** over traditional heuristic methods, significantly enhancing reliability. This improvement translates into measurable business value, such as better team composition, optimized resource allocation, and increased long-term success rates.

Overall, these models not only meet the technical requirements but also demonstrate clear business impact, offering organizations a scalable and explainable solution for leveraging performance data.

**Best Pick:**

**Victor – XGBoost Classifier** → Best balance of precision, recall, F1, and stability across datasets, making it the most reliable for business use cases.

**Other models:**

* **Shashikanth’s LGBM:** Excellent recall but very low precision → too many false positives.
* **Yagna’s LGBM:** Promising, but weaker recall than XGBoost.
* **Yanichsa’s Random Forest:** Excellent precision but low recall→ too many false negatives.

## Risks and Incorrect Predictions

Incorrect predictions from the final models can lead to misjudging player potential either overestimating undrafted players (false positives) or overlooking strong prospects (false negatives).

* **Business Risks:** False positives may result in wasted scouting and investment, while false negatives could mean missing valuable talent, impacting team performance and competitiveness.
* **Mitigation Strategies:**
  + Apply stricter probability thresholds to reduce false positives.
  + Use resampling methods (e.g., SMOTE) or ensemble models to improve minority class detection.
  + Continuously monitor model performance post-deployment and retrain with updated data.
* **Business Recommendation:** Accept a balanced trade-off between precision and recall to maximize value while minimizing costly errors.

## Data Privacy and Ethical Concerns

The dataset used does not contain personally identifiable information (PII), focusing only on performance statistics and derived features. All data points are aggregated, anonymized, and publicly available (e.g., game statistics), resulting in minimal privacy risks.

**Privacy Implications:**

Minimal, since no sensitive or personal attributes are included. The dataset was curated to ensure compliance with privacy and responsible data use.

**Ethical Concerns:**

* Avoiding algorithmic bias that could unfairly undervalue players from specific backgrounds, conferences, or teams.
* Preventing over-reliance on automated predictions by ensuring transparency and interpretability.
* Supporting fairness by ensuring features directly reflect player performance rather than contextual or institutional factors.

**Steps Taken:**

* Specifically, removed columns such as *team*, *conf*, *yr*, *year*, and *type* to eliminate potential bias from team affiliation, conference, or year-related factors.
* Restricted use of sensitive or proxy attributes to ensure fairness.
* Conducted feature importance analysis to enhance model transparency.
* Recommended a **human-in-the-loop validation** approach to ensure that predictions are used as decision-support rather than sole decision-making tools.

short dash

# Collaboration

## Individual Contributions

* **Shashikanth Senthil Kumar**
  + Responsible for developing **five notebooks**, covering the workflow from baseline models to the final LightGBM classifier. To ensure smooth collaboration and avoid merge conflicts in GitHub, I saved My **Datasets and Test Predictions** **separately**. Additionally, I led the **evaluation section** of the report by analyzing performance metrics from my own models as well as those of other team members. I compared results, learned from different modeling approaches, and identified the **best-performing model** for the final recommendations. This work included detailed analysis of accuracy, precision, recall, F1 score, and ROC AUC, ensuring a rigorous and fair evaluation.
* **Victor Garcia Ortiz**
  + I developed a unified data preparation pipeline and contributed multiple experiment notebooks focused on advanced model development and evaluation. My work included implementing and documenting a custom Python package (my\_krml\_10522504) for data preprocessing and feature engineering, which was published to TestPyPI and integrated into the group workflow. I explored and compared several models, including XGBoost, EasyEnsembleClassifier, and Logistic Regression, with a focus on robust metric selection and threshold optimization for imbalanced classification. I automated the extraction and reporting of parameter search results and model metrics, and provided code for saving model artifacts to ensure reproducibility. Additionally, I contributed to the group’s GitHub repository by maintaining my code, models, and documentation, and supported the team in drafting and structuring the experimentation and results sections of the final report.
* **Yagnasri Pailla** 
  + I was working on building and evaluating multiple models including Logistic Regression, LightGBM, and XGBoost. I carried out data cleaning, preprocessing, and handled the issue of class imbalance through class weighting and cross-validation techniques. I experimented with hyperparameter tuning using GridSearchCV and RandomizedSearchCV to optimize model performance. My LightGBM model achieved strong validation results with AUROC of 0.9974 and balanced precision and recall, demonstrating improved capability over baseline models. I also worked on publishing a custom Python package to TestPyPI, which contained reusable functions for data preparation, moing, and evaluation. This package was integrated into the group workflow, ensuring consistency and reproducibility. Additionally, I maintained my contributions in the group GitHub repository and the Report part.
* **Yanichsa Pramutjit**
  + I was working on five notebooks, including baseline, Logistic Regression, Random Forest, XGBoost, and Decision Tree. To streamline collaboration and ensure version control, I created a GitHub repository, where all team members could contribute code and track progress without conflicts. In addition, I coordinated the reporting process by assigning specific sections of the final report to team members.

## Group Dynamic

The team maintained a highly collaborative and coordinated environment throughout the project. Members communicated regularly via messaging apps and video calls to share updates, discuss findings, and resolve challenges. Regular check-ins allowed the team to synchronize efforts and provide mutual support, fostering a culture of accountability and open communication.

## Ways of Working Together

The team adopted an **agile-like approach**, with weekly meetings to review progress, discuss challenges, and plan subsequent tasks. Shared tools such as GitHub for version control, Google Drive for document sharing facilitated smooth collaboration. Decisions were made collectively, with input from all members, ensuring a balanced and inclusive approach to project management.

## Issues Faced within the Group

The main challenges encountered were **data inconsistencies** and **differences in modeling approaches**. These were addressed through collaborative troubleshooting sessions, sharing knowledge on preprocessing techniques, and reaching consensus on model selection. The team learned the importance of early alignment on methodology, clear communication, and systematic task tracking. For future projects, establishing a **formal project plan and timelines** at the start would further improve efficiency and coordination.

short dash

# Conclusion

This project set out to forecast NBA draft outcomes by applying advanced machine learning methods to structured performance and contextual data. Across all stages, from data cleaning and feature engineering to experimentation with multiple algorithms, the work demonstrated the potential of data-driven approaches to complement and improve upon traditional scouting methods. Challenges such as extreme class imbalance, missing values, and redundant variables were systematically addressed using imputation, encoding, and careful preprocessing.

A key strength of this project was the diversity of modeling approaches tested by different team members. Logistic Regression established a transparent and interpretable baseline, while Decision Trees provided simple rule-based insights. Random Forest experiments showcased the value of ensemble averaging for robustness, and boosting methods such as LightGBM and XGBoost captured complex non-linear interactions with impressive discriminatory power. Each approach highlighted trade-offs between recall, precision, and interpretability, enriching the team’s overall understanding of predictive modeling under imbalance.

Collaboration was another defining aspect of this assignment. Each student not only conducted individual experiments but also contributed reusable custom Python packages, ensuring reproducibility and professional practice. Shared GitHub workflows and the Kaggle competition environment provided opportunities to benchmark results, refine models, and simulate industry-style teamwork. The experience reinforced the importance of communication, version control, and modular code design in group-based machine learning projects.

Looking forward, several opportunities remain for future work. The persistent issue of class imbalance suggests value in exploring cost-sensitive learning, threshold optimization, or synthetic oversampling methods such as SMOTE. Incorporating richer external datasets, such as detailed scouting reports or longitudinal performance trends, could enhance predictive power and generalization. Model stacking or hybrid ensembles may also improve performance further by combining complementary strengths across algorithms. Ethical considerations, such as fairness and interpretability, should remain central to any future deployment, ensuring predictions support rather than replace human judgment.

Finally, comparing the best-performing models across team members highlights key insights:

* **Victor – XGBoost Classifier:** Delivered the best overall balance of precision, recall, F1, and stability, making it the most reliable model for business applications.
* **Shashikanth’s LightGBM:** Achieved excellent recall but suffered from very low precision, leading to many false positives.
* **Yagna’s LightGBM:** Produced promising results but showed weaker recall compared to XGBoost.
* **Yanichsa’s Random Forest:** Provided excellent precision but low recall, resulting in too many false negatives.

Taken together, these findings confirm that **XGBoost is the strongest and most business-ready solution**, offering the most consistent trade-off between accuracy and reliability. It stands out as the best candidate for deployment, ensuring that high-potential drafted players are identified effectively while minimizing costly scouting errors.

# References

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., & Ma, W. (2017). LightGBM: A highly efficient gradient boosting decision tree. *In Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS 2017)*. Retrieved from <https://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree>

Microsoft Research. (2017). LightGBM: A highly efficient gradient boosting decision tree. Retrieved from <https://www.microsoft.com/en-us/research/publication/lightgbm-a-highly-efficient-gradient-boosting-decision-tree/>

short dash

# Figures and Tables Index

[Figure 1: Student A, Approach 2, Learning and AUC-ROC plot 14](#_Toc207659748)

[Figure 1: Student A, Approach 2, Learning and AUC-ROC plot 14](#_Toc207659748)

[Table 1: Student B Package and Functions 26](#_Toc207662352)

[Table 2: Student B Data Preprocessing Pipeline 27](#_Toc207662353)

[Table 3: Student B, Models Metrics Summary 35](#_Toc207662354)

[Table 4: Student D Model Metrics Summary 44](#_Toc207662355)

[Table 5: Best Models Performance Summary 47](#_Toc207662356)

[Figure 1: Student A, Approach 2, Learning and AUC-ROC plot 15](#_Toc207662359)

[Figure 2: Logistic Regression Coefficients 15](#_Toc207662360)

[Figure 3: Student A, Approach 1, Learning and AUC-ROC plot 17](#_Toc207662361)

[Figure 4: Decision Tree Feature Importance 18](#_Toc207662362)

[Figure 5: Student A, Approach 4, Learning and AUC-ROC plot 20](#_Toc207662363)

[Figure 6: Random Forest Feature Importance 21](#_Toc207662364)

[Figure 7: Studen A, Approach 5, Learning and AUC-ROC plot 23](#_Toc207662365)

[Figure 8: LightGBM Feature Importance 24](#_Toc207662366)

[Figure 9: Student B, Approach 1, AUC-ROC plot 29](#_Toc207662367)

[Figure 10: Student B, Approach 1, Confusion Matrix plot 29](#_Toc207662368)

[Figure 11: Student B, Approach 1, F1 Threshold plot 30](#_Toc207662369)

[Figure 12: Student B, Approach 1, F1 Adjusted Confusion Matrix plot 30](#_Toc207662370)

[Figure 13: Student B, Approach 2, AUC-ROC plot 32](#_Toc207662371)

[Figure 14: Student B, Approach 2, Confusion Matrix plot 32](#_Toc207662372)

[Figure 15: Student B, Approach 3, AUC-ROC plot 34](#_Toc207662373)

[Figure 16: Student B, Approach 3, Confusion Matrix plot 34](#_Toc207662374)

1. neg = (y\_train == 0).sum() , pos = (y\_train == 1).sum() , scale\_pos\_weight = neg / pos [↑](#footnote-ref-1)