# **Data analysis and processing based on Spark：**

# **NBA & Bilibili Data analysis**

**Crouse name:** Data Processing Workshop II

**Crouse Code:** DS 3003

**Team Member:**

**Group14**

Fan Chenwen 2330026038

Tong Xinyi 2330024216

Chen Shaopeng 2330026016

Liu Ziang 2330026108

**Teacher Name:**

Dr. Thomas Canhao XU

[NBA DATA ANALYSIS 4](#_Toc1779161775)

[1. Code environment 4](#_Toc809651597)

[2. Overview 4](#_Toc1358003387)

[3. Original Data implement 6](#_Toc1754296996)

[4. NBA Player Rating System Research Report: From Unified Model to Position-Aware Optimization（Who is MVP） 7](#_Toc1666622109)

[4.1 Data Processing and Feature Engineering 7](#_Toc1288578142)

[4.2 Basis for determining indicator weights: Correlation heat map analysis 8](#_Toc1907736246)

[4.3 Constructing the score 9](#_Toc1392236812)

[4.4 Regression Model Training 10](#_Toc342281572)

[4.5 Output regression score and ranking 11](#_Toc1765173938)

[5. Clustering & Random forests analysis for player win-rate 12](#_Toc1939276308)

[5.1 Motivation 12](#_Toc1057598037)

[5.2 Clustering analysis 13](#_Toc328061640)

[5.3 Random forests analysis for player win-rate 16](#_Toc1141461631)

[6. Optimization of the scoring system - Introduction of location-aware mechanism. .............................................................................................................26](#_Toc1074213566)

[6.1 Background and Problem 26](#_Toc412383433)

[6.2 Optimization of scoring function: Construction of position-aware weighted model 27](#_Toc998629562)

[6.3 Update of Regression Model 28](#_Toc1382347229)

[6.4 Coefficient Analysis and Occupational Interpretation 29](#_Toc1631784557)

[6.5 Conclusion and Value Enhancement 30](#_Toc2036877309)

[BILIBILI DATA ANALYSIS 32](#_Toc760115536)

[1. Background Introduction & Objectives 32](#_Toc2029081196)

[2. Project environment 32](#_Toc727346812)

[3. Code implements 32](#_Toc1040950560)

[4. Project procedures 33](#_Toc1854273458)

[4. 1 Data collection 33](#_Toc491323342)

[4.2 Data Preprocessing 36](#_Toc596786279)

[4.3 Exploratory analysis 39](#_Toc1438359663)

[5. Analyze the UIC data with tree methods 43](#_Toc1274433647)

[5.1 Environment and Dependency Import 43](#_Toc404409951)

[5.2 Initialize Spark session 43](#_Toc132303702)

[5.3 Feature Engineering 44](#_Toc982744869)

[5.4 Feature selection & Model training 47](#_Toc696284206)

[6. Conclusion 50](#_Toc810257739)

# **NBA DATA ANALYSIS**

1. **Code environment**

Python 3.1.2

Jupyter Notebook & Jupyter Lab

Spark: 3.2.0

Java：JDK11

Library: numpy, spark,pandas, seaborn, matlplotlib, sklearn, etc.

1. **Overview**

This project presents a comprehensive data-driven analysis of NBA player performance, aiming to explore the relationship between individual statistics and team success, and to construct an interpretable and role-sensitive scoring system for players.

We begin by collecting and integrating player data from the 2023–2024 NBA season, using sources from the NBA official site and ESPN. The data was preprocessed to ensure consistency, including positional mapping and missing value handling.

The project is divided into three major analytical modules:

**Unified Player Rating System**: We developed a **regression-based** performance scoring model using six engineered features . This system applies **ElasticNet regression** to quantify the impact of each technical indicator, achieving a good fit and high interpretability.

**KMeans Clustering & Random Forest Modeling**: To account for diverse player roles, we clustered players into functional groups and used a **Random Forest** model to predict win rates. We identified plus-minus and teammate quality as key win-rate drivers, highlighting the importance of context beyond raw stats.

**Position-Aware Score Optimization**: Recognizing the limitations of a one-size-fits-all model, we designed a position-sensitive scoring mechanism that dynamically adjusts feature weights based on player positions (C, F, G).

Through these steps, the project not only builds predictive models but also enhances the interpretability and practical value of player evaluation. The methodology and scoring framework can be extended to other seasons, leagues, or basketball research tasks involving team optimization and MVP selection.

### Data Sources

The original code material provides the NBA Players stats data for the 2022-2023 NBA season. According to the original file, the original data is from the well-known data website Kaggle. Based on the introduction of the uploader of this dataset on Kaggle, we inspected the data sources and found that the data comes from two website sources. They are as follows:

NBA official statistics website:

<https://www.nba.com/stats/players/opponent?Season=2023-24>

ESPN News Network:

<https://www.espn.com/nba/stats/player/_/season/2024/seasontype/3>

We obtained two sets of NBA 2023-2024 statistics from this website. Since the NBA 2024-2025 season was not yet completed when we were working, the data set might be incomplete and less referential, so we chose to use the 2023-2024 season's data. After inspecting the info of the data, we found that compared with the original data set, the data fields of the NBA official statistics website were basically the same as those of the original data, but it did not include 'POS', the player's position in the game, while the ESPN data set had 'POS'. Therefore, we adopted a mapping approach to match the 'POS' field of ESPN to the data of the NBA official website. To avoid garbled characters, we blurred the Eastern European letters to facilitate better matching. The following is the implemented code.：

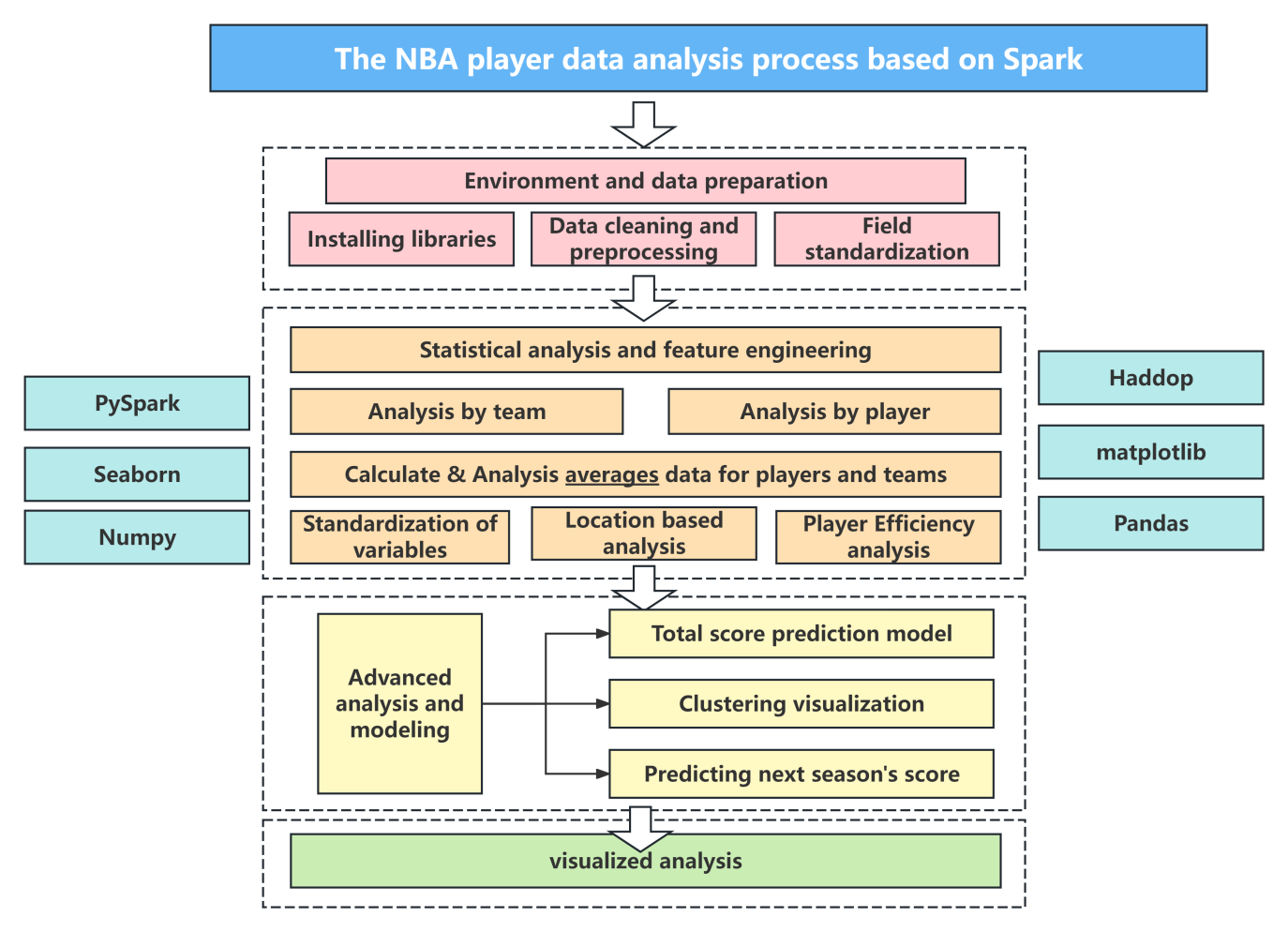


After mapping the attributes of the data, the fields of the new data are exactly the same as those of the original data, thus facilitating code migration. The fields included in the final data are as follows:  
The name of the basketball player PName

* The player's position in the game, including 'N/A' POS
* The abbreviation of the team the player is currently playing for this
* season Team
* The age of the player Age
* The total number of games the player has played in this season GP
* The total number of games won by the player W
* The total number of games lost by the player L
* The total minutes the player has played in this season Min
* The total points made by the player [target] PTS
* The total number of field goals made by the player FGM
* The total number of field goals attempted by the player FGA
* The percentage of successful field goals made by the player FG%
* The total number of 3-point field goals made by the player 3PM
* The total number of 3-point field goals attempted by the player 3PA
* The percentage of successful 3-point field goals made by the player 3P%
* The total number of free throws made by the player FTM
* The total number of free throws attempted by the player FTA
* The percentage of successful free throws made by the player FT%
* The total number of offensive rebounds made by the player OREB
* The total number of defensive rebounds made by the player DREB
* The total number of rebounds (offensive + defensive) made by the player REBThe total number of assists made by the player AST
* The total number of turnovers made by the player TOV
* The total number of steals made by the player STL
* The total number of blocks made by the player BLK
* The total number of personal fouls made by the player PF
* The total number of NBA fantasy points made by the player FP
* The total number of double-doubles made by the player DD2
* The total number of triple-doubles made by the player TD3
* The total difference between the player's team scoring and the opponents' scoring while the player is in the game +/-

1. **Original Data implement**

Following is our random forest and model analysis brief flow chart.



Through this flowchart, we can clearly see that the main focus of the original code lies in the players' scores and the statistics of their various attributes. The original code mainly modeled the prediction scores of the players, mainly using the linear regression model of MLlib. For example, building a linear regression model between rebounds and victories, building a linear regression model between assists and turnovers, and predicting the players' scores for the next season, etc.

In the reproduction of the original code, we added the latest 23-24 data mentioned earlier and carried out code reproduction. Additionally, we changed the file path from Hadoop HDFS to a local save path.

1. **NBA Player Rating System Research Report: From Unified Model to Position-Aware Optimization（Who is MVP）**

**Research Background and Objectives**

With the acceleration of the NBA game pace and the diversification of tactics, the traditional methods of evaluating players' performance based solely on total scores or a single efficiency value have gradually shown their limitations. Therefore, this study aims to construct a scientific and more explanatory regression scoring system. It will comprehensively assess the overall influence of players through multi-dimensional data, and on this basis, further optimize the model to take into account the differences in tactical responsibilities among players of different positions.

**Construction of the Player Rating System**

In this section, we will build a unified player rating system, using data-driven methods to establish a linear regression model based on multiple technical indicators. This rating system aims to provide all players with a fair and interpretable "performance score" for identifying the players with the best performance during the season.

### 4.1 Data Processing and Feature Engineering

The data source is the regular statistics of NBA players for a certain season. We use PySpark for large-scale data processing to ensure the scalability and performance of the model. Since the data we initially attempted to analyze indicated that rating each player based solely on primary data (points, rebounds, assists, etc.) was unreliable and ineffective, our output not only deviated from the facts but also did not perform well in terms of data scoring. Therefore, we attempted to modify the rating indicators and used Spark to process the regular season data of the 2022-2023 NBA.

Based on the understanding of game rules and player roles, we extracted and constructed 6 core derived indicators:

• avg\_points: Average points per game = Total points / Number of games played (PTS / GP)

• avg\_assists: Average assists per game = AST / GP

• avg\_rebounds: Average rebounds per game = REB / GP

• pts\_per\_minute: Points efficiency per minute = PTS / MIN

• win\_rate: Win rate = W / GP

• avg\_plus\_minus: Impact of plus-minus rating = +/- / GP

The above features were normalized using MinMaxScaler to eliminate the differences in scale and unify them for input into the regression model.

At the same time, not only did the data processing take place, but the player's regular season plus-minus rating data was also renamed and added.

**文本, 应用程序

AI 生成的内容可能不正确。**

### 4.2 Basis for determining indicator weights: Correlation heat map analysis

Before constructing the score, we cleaned the data of players with fewer games to ensure its reliability, and used Pandas to conduct correlation analysis on the above six indicators. Through drawing a Pearson correlation coefficient heat map, we observed the linear relationship between each feature and avg\_plus\_minus (net plus-minus score).

• Analysis results show:

-The correlation between win\_rate and avg\_plus\_minus is the highest, indicating that the win rate significantly contributes to the influence of players;

-The correlation between avg\_assists and avg\_rebounds is also strong, reflecting the importance of organization and rebounds in the game;

-pts\_per\_minute shows a certain degree of efficiency representativeness.

By using the new heat map, we compared the correlation data and used the strongly correlated features to model.

**图表

AI 生成的内容可能不正确。**

Therefore, based on the linear correlation strength of each indicator in the heat map with the target performance, we have set the following weight ratios:

* avg\_assists: 18%
* avg\_rebounds: 15%
* pts\_per\_minute: 14%
* avg\_points: 10%
* win\_rate: 25%
* avg\_plus\_minus: 18%

This scoring function aims to comprehensively evaluate a player's basic statistics, performance efficiency and influence on the outcome, and it has good interpretability.

### 4.3 Constructing the score

Based on the weights determined above, we manually construct a scoring function called "score" as the target of the regression model:

* score = Weighted sum (of six indicators × respective weights)
* The weighting is based on the correlation degree reflected in the aforementioned heat map. In the programming implementation, we create separate columns for each standardized feature and use the following weighting logic:

**图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。**

This approach ensures:

* Each indicator is compared on a unified scale;
* The weights are assigned based on the results of statistical analysis and professional understanding of basketball;
* A preliminary simulation of the overall impact of the players on the game is conducted, providing labels for the training of the regression model. We manually construct a scoring function called "score" as the target of the regression model:
* score = weighted sum (six indicators × respective weights)

This score reflects the comprehensive contribution of the players to the outcome and rhythm of the game.

### 4.4 Regression Model Training

To ensure that the constructed scoring system has both predictive capabilities and good interpretability, we selected a linear regression model with the ElasticNet regularization term for training. ElasticNet is a regression method that combines L1 (Lasso) and L2 (Ridge) regularization.

* L1 regularization helps in feature selection by compressing the coefficients of less important features to 0, achieving the goal of sparsity.
* L2 regularization (ridge regression) can prevent the model from overfitting to the training data and improve generalization ability, especially in cases where there is collinearity among features.

In our program, the specific parameters of ElasticNet are set as follows:

* regParam = 0.1: regularization strength
* elasticNetParam = 0.3: controls the balance between L1 and L2 (0 is pure ridge regression, 1 is pure Lasso)

The training process is as follows:

1. First, pack the six normalized features into a feature vector using VectorAssembler for the regression model to use;

2. Construct score as the label column;

3. Use the LinearRegression class for fitting, and the model will adaptively learn the optimal regression coefficients for each feature based on the training data.

The significance of model training lies in:

* Reverse optimization of the scoring function to verify the rationality of the score we constructed;
* Outputting feature weight coefficients, which can quantitatively analyze the influence of each technical indicator on the overall player score;
* Generating regression prediction scores to achieve the ranking of players' comprehensive abilities, providing a basis for subsequent MVP determination.

Model performance:

* R² = 0.8016 (good fit)
* RMSE = 0.056

**图形用户界面, 文本, 电子邮件

AI 生成的内容可能不正确。**

### 4.5 Output regression score and ranking

After the model training is completed, we apply it to the data of all players to predict the regression\_score of each player, which is the comprehensive score derived from the training results.

The specific steps are as follows:

* Use the trained model to predict the features and generate the "prediction" field;
* Rename the predicted values as "regression\_score" and combine them with the original player information;
* Sort by score in descending order, output the top 10 high-scoring players, and conduct visual display.

This process not only verifies the generalization ability of the model but also enables us to evaluate the player performance based on the data, providing quantitative basis for MVP or All-Star selections. We apply the trained model to all player data, output its "regression\_score" as the final score result, sort by score and generate the top 10 player list.

**图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。**

**图表, 条形图

AI 生成的内容可能不正确。**

## **5. Clustering & Random forests analysis for player win-rate**

### 5.1 Motivation

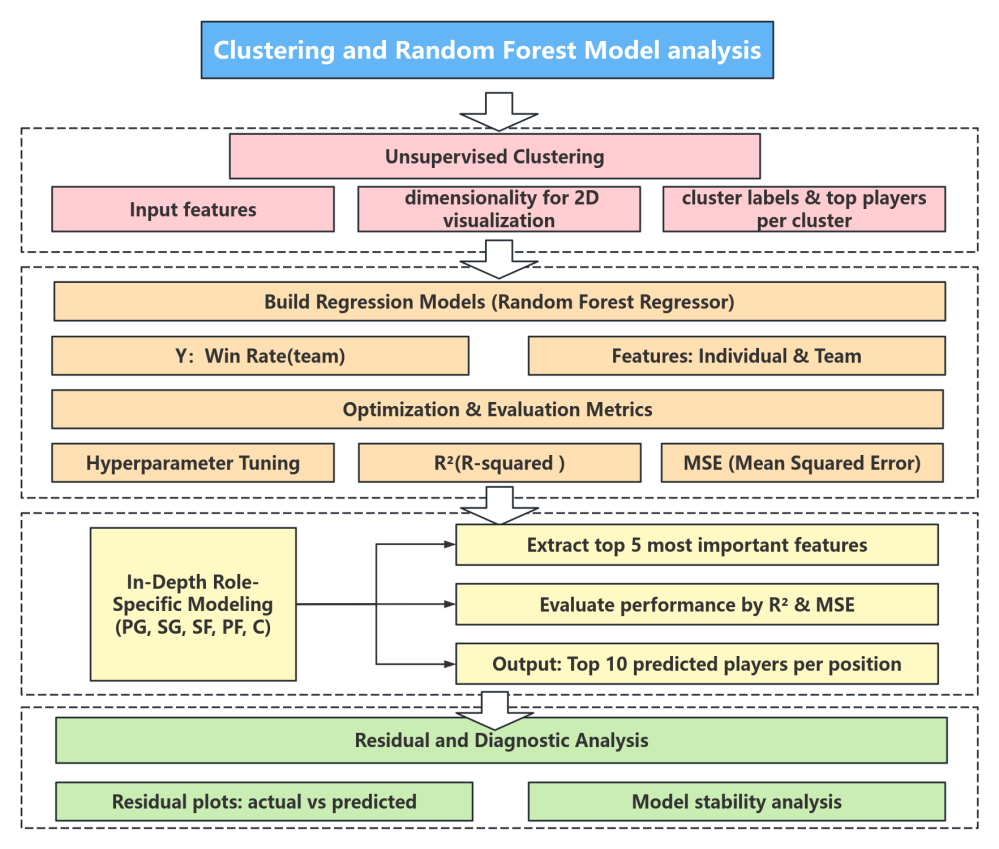
The motivation for this project arises from two key challenges in modern basketball analytics:

Player Evaluation Is Context-Dependent: A player’s raw statistics (e.g., points per game) do not fully reflect their impact on team success. For example, high scorers on weak teams may not be as valuable as balanced contributors on winning teams.

Role Differentiation Matters: Comparing players without considering their functional roles leads to unfair assessments.

Therefore, our approach aims to first segment players into meaningful groups based on their play style and contribution patterns and then evaluate their impact on win rate within their peer groups.

Following is our random forest and model analysis brief flow chart.

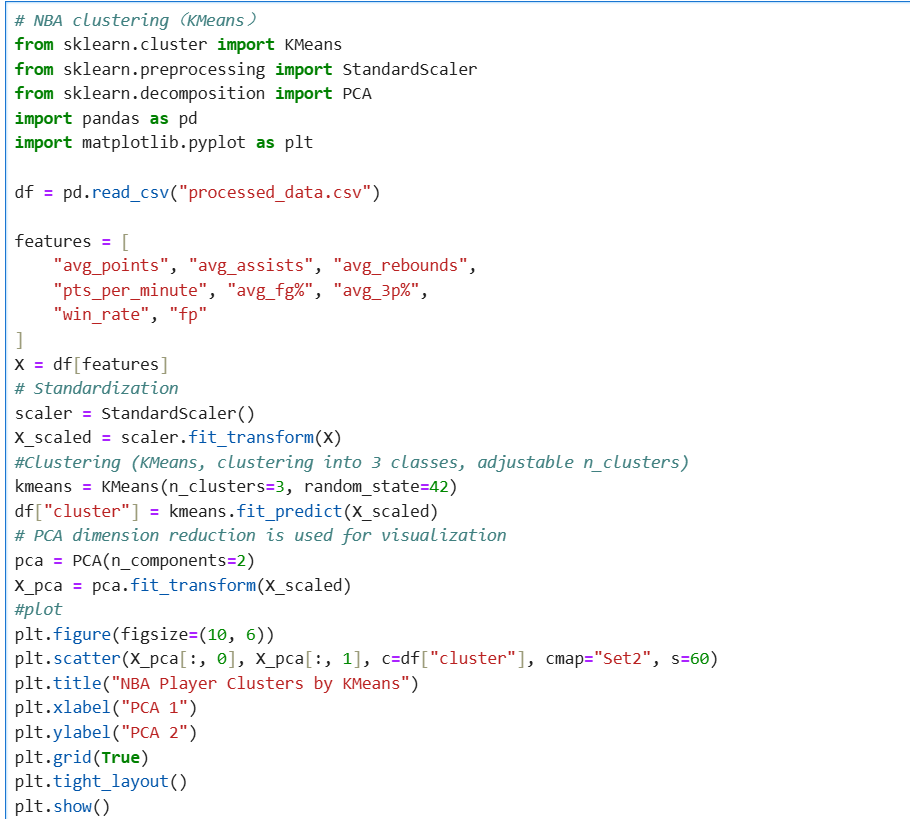


### 5.2 Clustering analysis

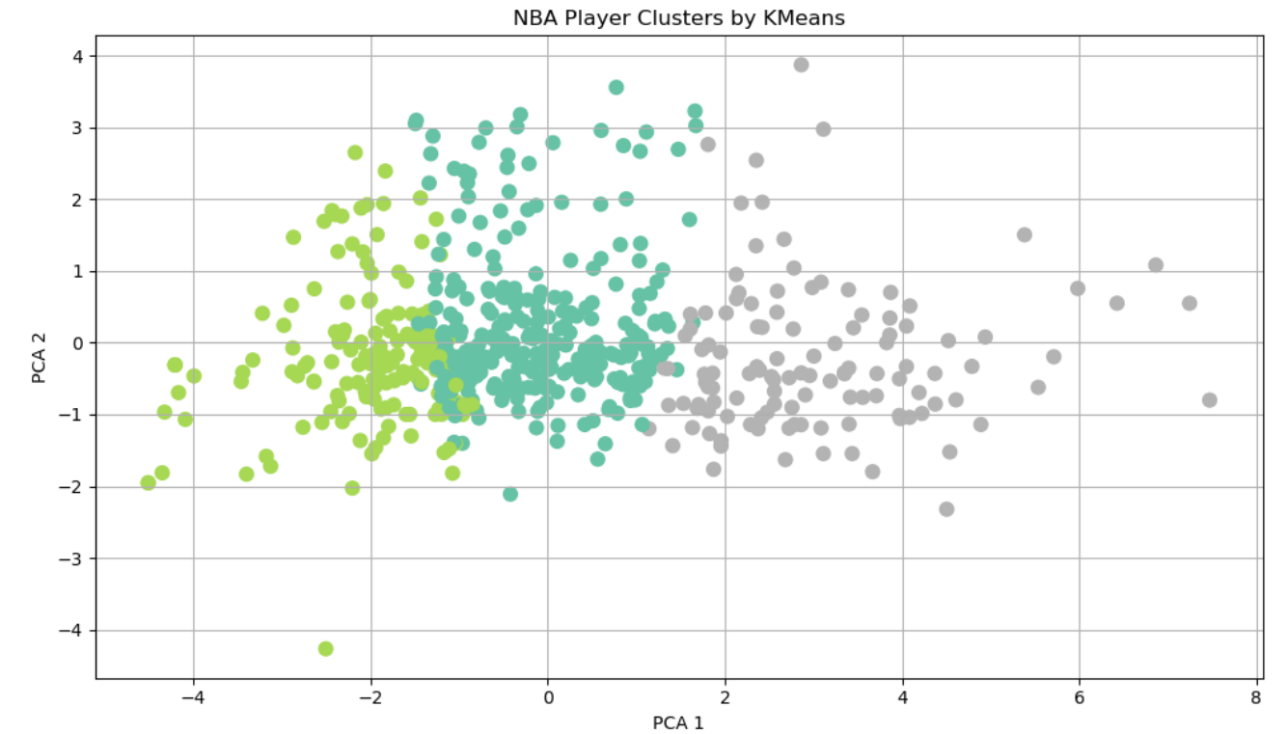
In this section, we first conduct a clustering analysis on the players. The aim of this section is to identify potential player types (such as scoring-oriented, organizing, efficient role players, etc.) from the multiple performance data of the players using the KMeans clustering algorithm, providing a basic reference for subsequent win rate prediction and team optimization.

Our data source is the cleaned NBA player performance dataset, which consists of two separate datasets, one for the 2022-2023 season and the other for the 2023-2024 season, displayed in two different ipynb files. This report is mainly based on our new data, namely the player statistics from the NBA 2023-2024 season. The model for the 22-23 season can be specifically viewed in the 2022-2023 code.

In the original code, the average points per game, three-point shooting percentage, average rebounds per game, and average assists per game for each player were calculated and saved in "processed\_data.csv". Compared to the original data, the new fields added are: avg\_points, avg\_rebounds, avg\_fg%, avg\_3p%. In the subsequent win rate analysis, the data source will be "processed\_data.csv".

The Kmeans clersting code is as follows:  


This is the specific visualization of NBA Player Clusters by KMeans.



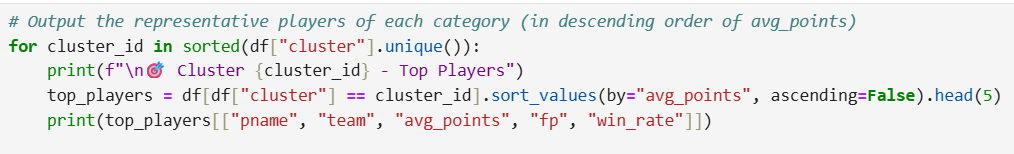
Multiple performance indicators were extracted from the statistics of the NBA 2023-24 season, and the players were cleaned and standardized. The key features used for clustering include:

* avg\_points
* avg\_rebounds
* avg\_assists
* pts\_per\_minute
* avg\_fg%
* avg\_3p%
* win\_rate
* fp

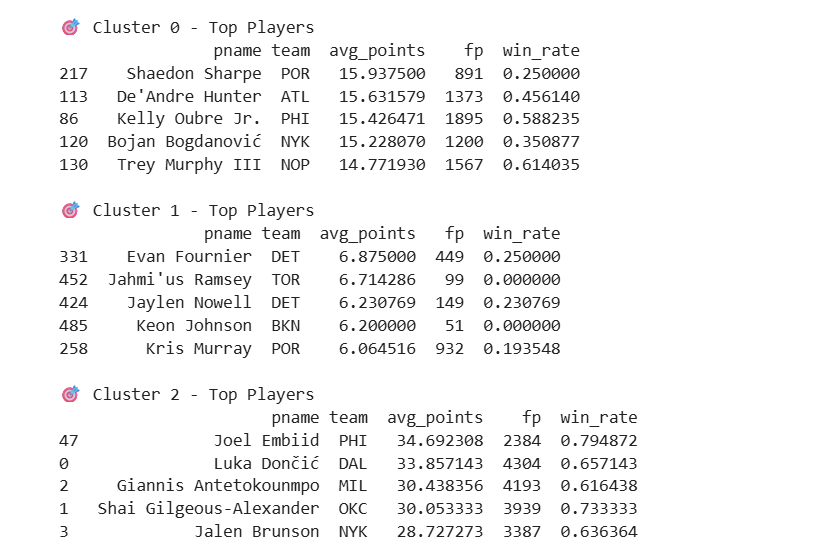
All features are normalized by StandardScaler to ensure that the clustering results are not affected by the scale.

We used the KMeans algorithm to cluster the players, initially setting it to 3 categories (which can be adjusted). Each player was assigned a cluster label to indicate their style group. To enhance visualization and interpretability, we employed PCA (Principal Component Analysis) to reduce the multi-dimensional data to two dimensions and plotted the cluster distribution graph. Different colors in the graph represent different types of players, with clear boundaries, and the clustering results show good separability.

The following code outputs the clustering results and presents the top five players representing each category, sorted in descending order of average points per game.



Here is the output result.



The KMeans clustering algorithm successfully categorized NBA players into three distinct groups based on their scoring, rebounding, assisting, and efficiency metrics:

**Cluster 0: Consistent Scorers**

This group includes players like Shaedon Sharpe and De'Andre Hunter. They average around 15 points per game with moderate Fantasy Points (fp) and a wide range of team win rates. These players often serve as secondary scorers or role-specific offensive contributors. While not the primary stars, they offer stable offensive output and can support winning when placed in the right system.

**Cluster 1: Low-Impact Role Players**

Players such as Evan Fournier and Keon Johnson fall into this category. Their scoring and fp are relatively low, and many of them come from teams with poor win records. This cluster typically represents bench players, development projects, or those on teams undergoing a rebuild. Their direct influence on winning is limited, though they may still hold value in specialized roles or future growth.

**Cluster 2: Franchise Cornerstones**

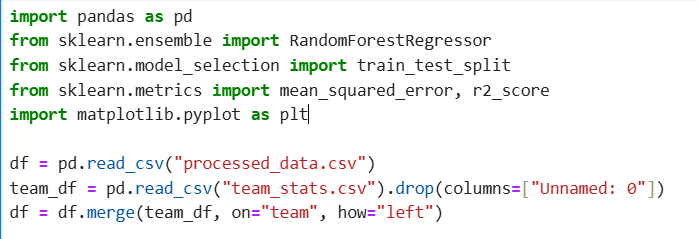
Dominated by superstars like Joel Embiid, Luka Dončić, and Giannis Antetokounmpo, this cluster exhibits extremely high scoring (28–35 PPG), elite fp, and strong win rates. These players are central to their teams’ success, driving both individual performance and team results. The clustering confirms their elite status and highlights their essential impact.

This clustering structure allows us to go beyond traditional position-based classifications and instead group players based on their functional value and impact. Notably, players in Cluster 2 show a strong positive correlation with team success, while those in Cluster 1 show little.

### 5.3 Random forests analysis for player win-rate

**Model construction**

To quantitatively assess how individual performance influences team success, we developed a Random Forest regression model targeting player-associated team win rates. This ensemble learning method is well-suited for handling complex, nonlinear relationships and offers robust interpretability through feature importance scores. Below are the libraries that should be imported and the data to be imported.



Due to the small size of the target dataset (572 items), a lightweight pandas library was used. Initially, the input features selected by the model were personal performance, and the selected feature values were:

* min — Total minutes the player has played this season
* fgm — Total number of field goals made
* 3pm — Total number of 3-point field goals made
* reb — Total number of rebounds (offensive + defensive)
* ast — Total number of assists made
* stl — Total number of steals made
* blk — Total number of blocks made
* tov — Total number of turnovers committed
* pf — Total number of personal fouls committed
* dd2 — Total number of double-doubles (10+ in two stats)
* td3 — Total number of triple-doubles (10+ in three stats)
* avg\_fg% — Percentage of successful field goals (FGM / FGA)
* avg\_3p% — Percentage of successful 3-point field goals (3PM / 3PA)
* fp — Total NBA fantasy points earned
* +/- — Point differential while the player is on the court (team score minus opponent score)

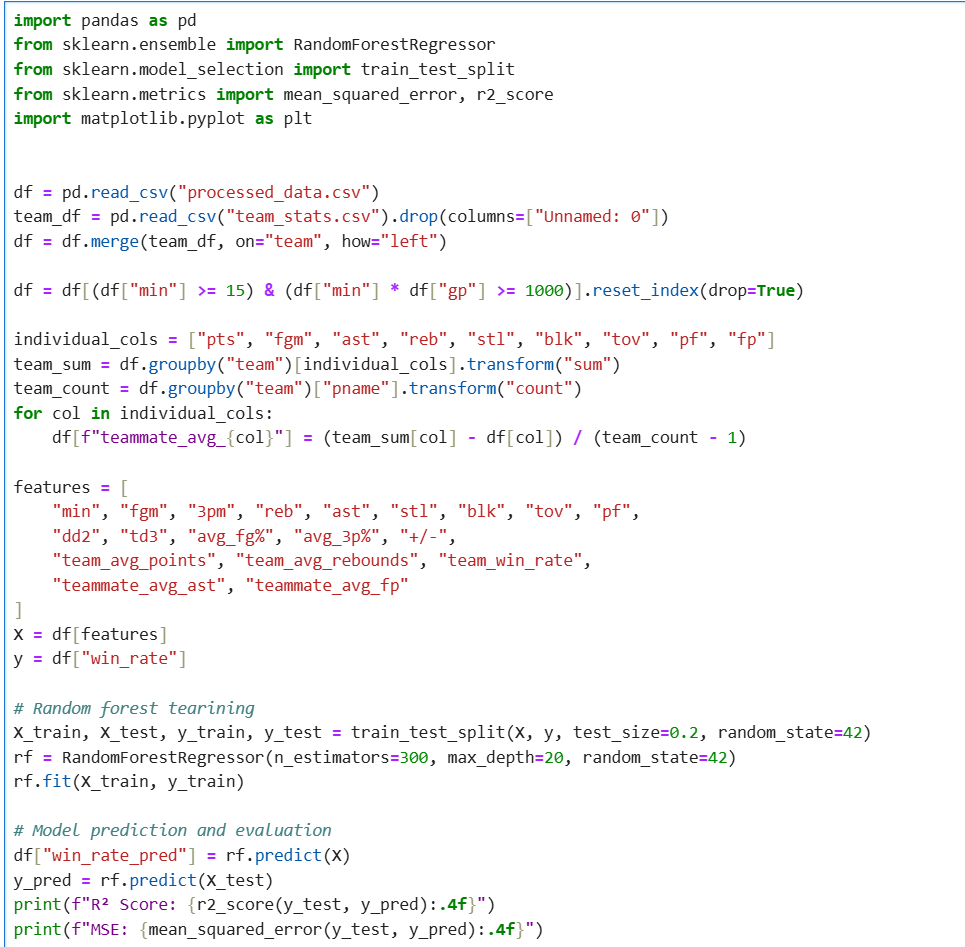
The selected insertion features were all "individual performance". Subsequently, a random forest was constructed by the model. The model architecture utilized 300 decision trees with a maximum depth of 10, selected to balance bias and variance. However, in the subsequent evaluation, R² was only 0.36, indicating that the explanatory power of the model was insufficient, and the parameters needed to be adjusted.



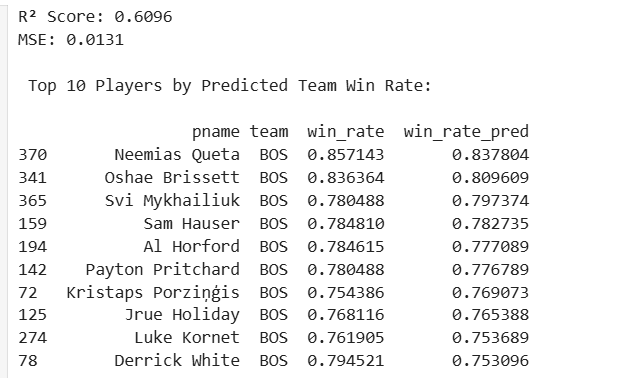
Therefore, we optimized the modeling and parameter selection of this model. The model filtered out players with insufficient game time during the season, and only retained players who averaged more than 15 minutes of playing time per game and those with more than 1000 minutes of playing time. This prevented the appearance of "one-time star" players, which would affect the accuracy of the model. Subsequently, the model calculated the average score of teammates based on the data of the players' teams and teammates. A "team variable" was introduced as an additional feature.

After incorporating "team variables" as features, the explanatory power of the model was enhanced. The R² reached 0.6 and the MSE was only 0.0131. 0.6 is an acceptable R². The NBA league is a league where rankings often change. The winning rate of each player is affected by transfers, team budgets, team draft picks, team budgets, etc. And basketball scores are often large, and the decision-making power of players for the team is also limited. Due to the limitations of data, 60% of the explanatory power is already acceptable. From this, we can see that in the NBA league, individual skills are more important than teamwork.

Following is the implemented code and the output result.

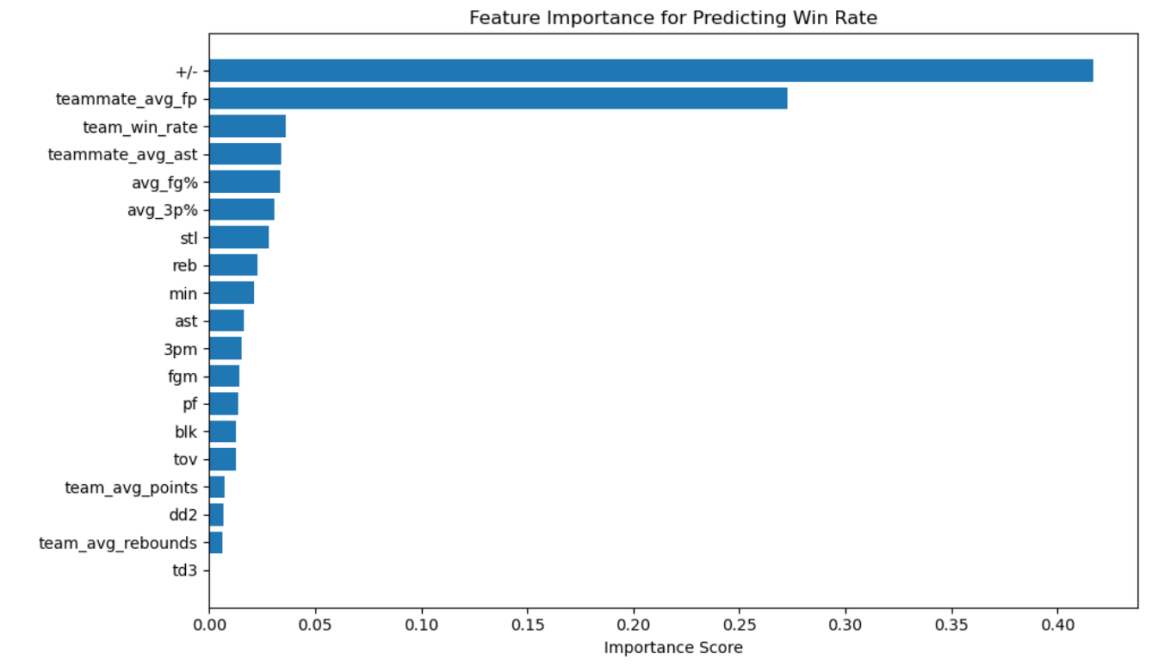


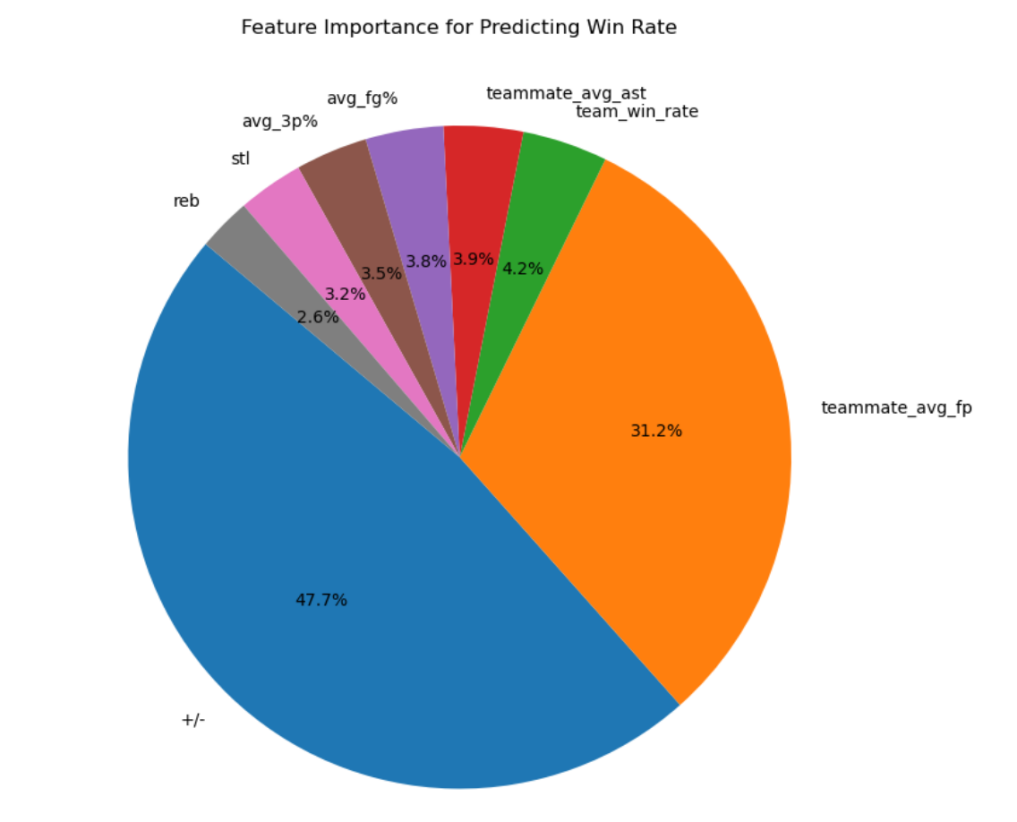




**Feature Importance analysis**

To further explore which player traits most significantly influence the prediction results of win rates, we conducted a feature importance analysis based on the trained random forest model.





As shown in the bar chart and pie chart, the most predictive variable is the plus-minus value (+/-), which alone contributes nearly half of the total feature weight. This metric measures the difference in points between the team and the opponent when the player is on the court, directly reflecting their influence on the outcome of the game.

The second most important variable is the average Fantasy points of teammates (teammate\_avg\_fp), indicating that the overall combat effectiveness of the team a player is in significantly boosts their winning rate. This is highly consistent with the collaborative nature of basketball as a team sport, emphasizing that "the environment shapes value".

In addition, indicators such as the overall team winning rate, the average assists of teammates, and the individual shooting percentages of players (avg\_fg%, avg\_3p%) also demonstrate significant importance. These results collectively reveal that a player's contribution to the winning rate is not determined by a single dimension, but rather shaped by the interaction between individual ability and the team environment.

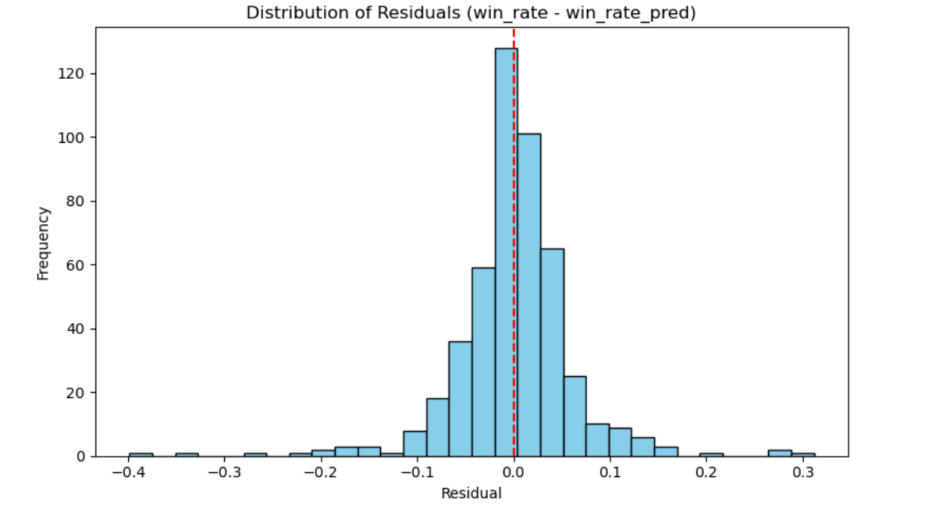
It is worth noting that the importance of traditional statistics (such as points, rebounds, playing time, etc.) in this model is relatively low, suggesting that we should not judge a player's value solely based on the "total amount of data", but rather focus on the "quality of influence".

These analyses help us better identify high-value players who possess both individual contributions and team collaboration abilities in different role positions, and provide important references for player evaluation and team formation.

**Model diagnosis**

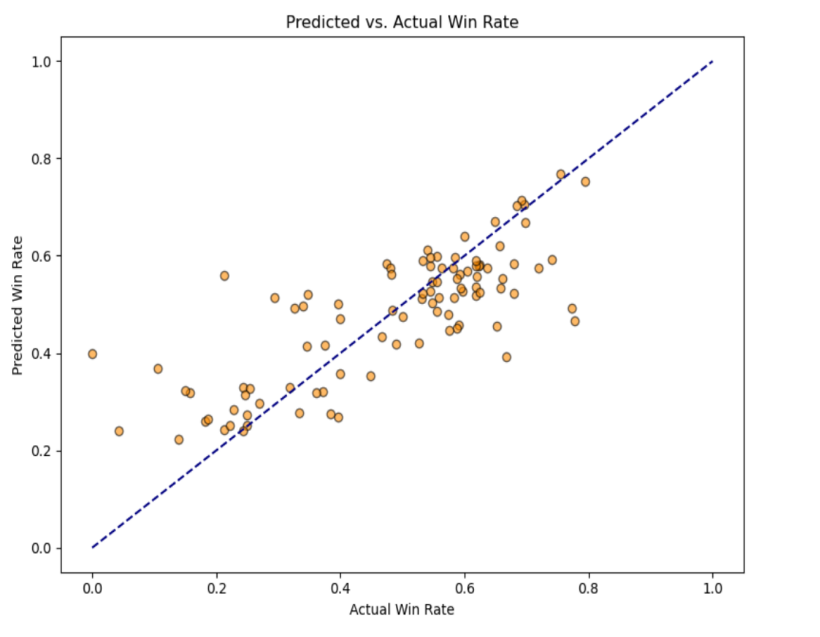
To assess the robustness and reliability of the Random Forest regression model in predicting player-associated win rates, we performed a series of diagnostic evaluations. These included residual distribution analysis, actual vs. predicted comparisons, residual pattern inspection, and cross-validation performance.

**Residual Distribution**



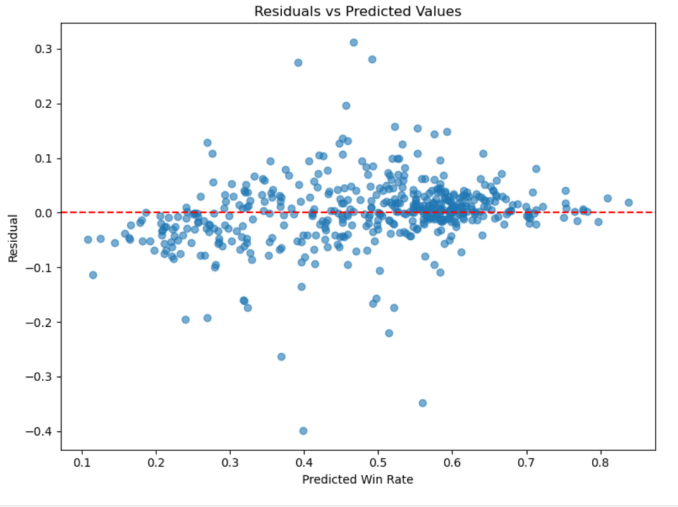
The histogram of residuals (actual win rate minus predicted win rate) exhibits a near-normal distribution centered around zero. This indicates that the model does not systematically overestimate or underestimate predictions and that its errors are symmetrically distributed.

**Actual vs. Predicted Fit**



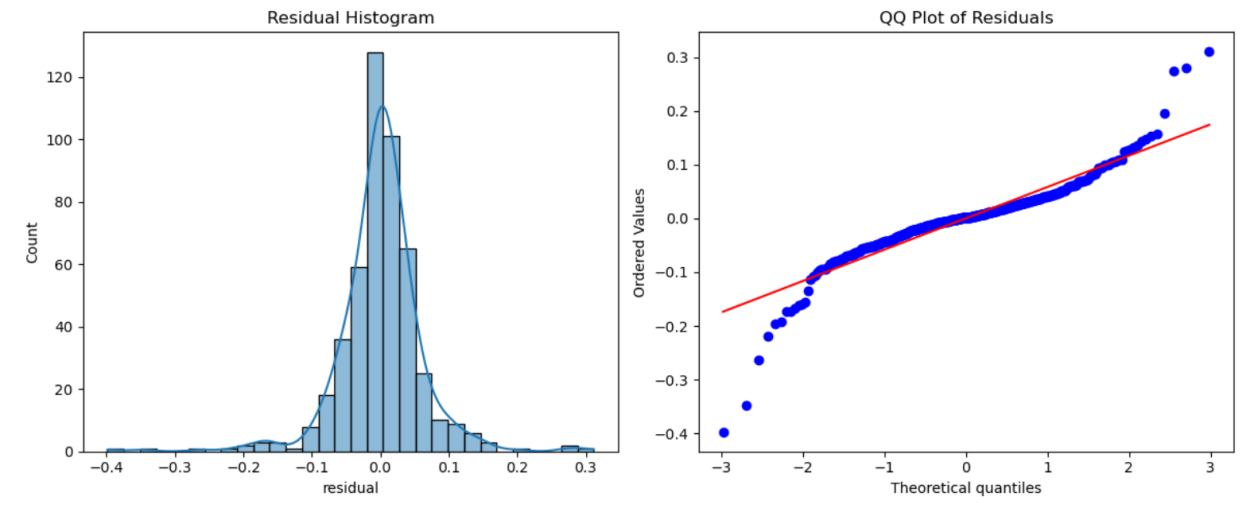
The scatter plot of actual vs. predicted win rates shows that most data points lie close to the diagonal reference line (y = x), suggesting a strong alignment between predicted outcomes and ground truth values across the sample.

**Residuals vs. Predicted Values**



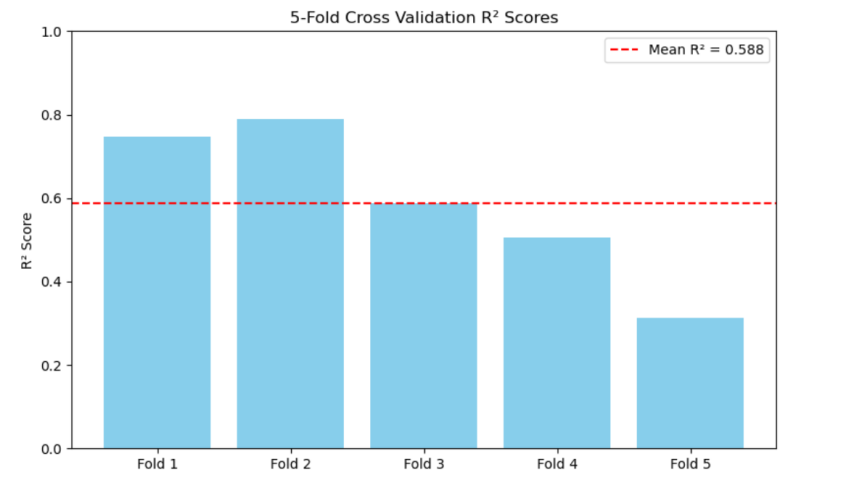
The residuals plotted against predicted values appear evenly scattered without any discernible funnel shape or curvature, implying that the model does not suffer from significant heteroscedasticity and maintains consistent variance across predictions.

**Normality Check**



The Q-Q plot of residuals mostly follows the 45-degree reference line, with mild deviations at the tails. This pattern supports the assumption that the residuals are approximately normally distributed, validating further statistical inference.

**5-Fold Cross-Validation**



The model's R² scores across five folds range from 0.28 to 0.79, with a mean R² of 0.588. While there is some variation between folds, the average performance indicates a moderate to good generalization capability. Notably, the model performed best in Fold 1 and Fold 2.

**Position-Specific Modeling and Interpretation**

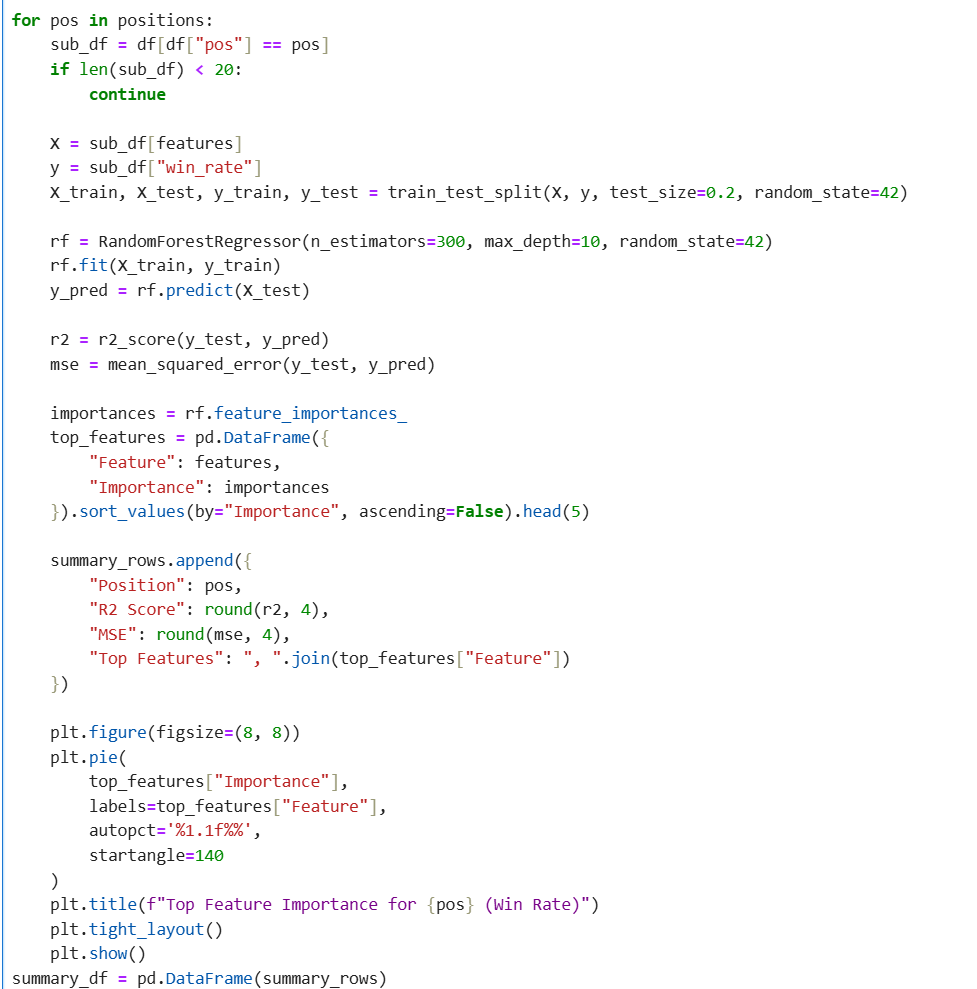
To further explore how different types of players contribute to team success, we trained separate Random Forest regression models for each player position (e.g., PG, SG, SF, PF, C). This approach accounts for the diverse functional roles and responsibilities across positions, allowing for a more tailored evaluation of win rate predictors.

For each position group:

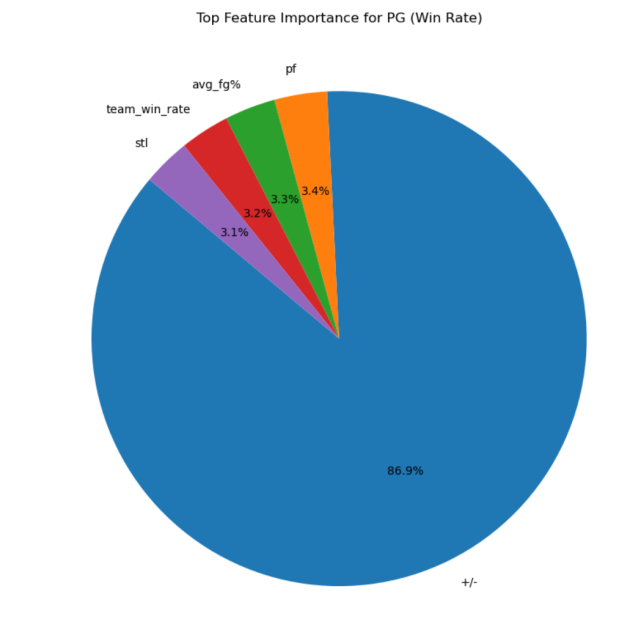
* Only players with at least 15 minutes per game and a total playing time over 1000 minutes were included to ensure statistical validity.
* We trained position-specific models using the same feature set as before.
* We extracted the top 5 most important features for each position and visualized their relative contributions using pie charts.

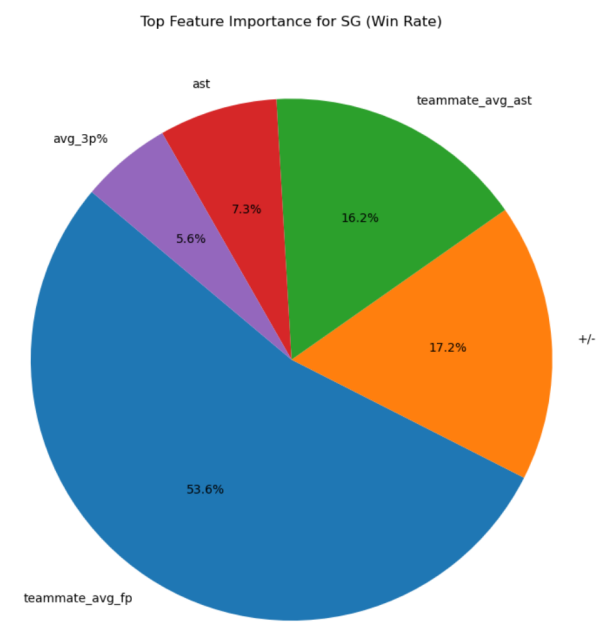
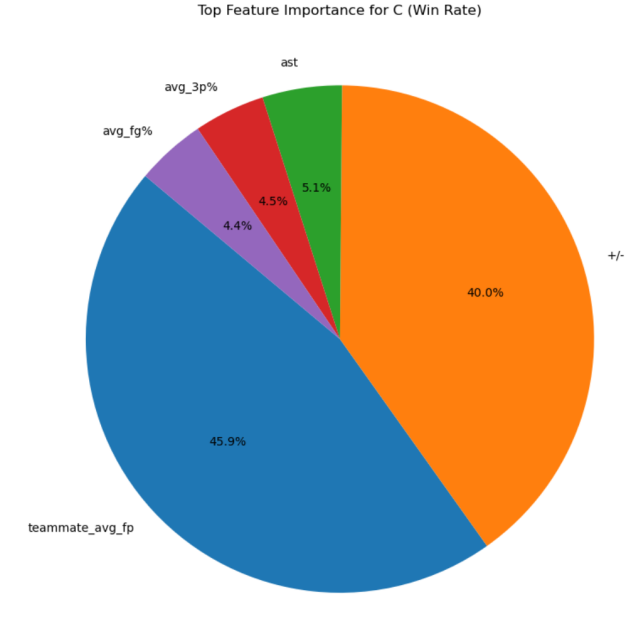
Following is the implement code.

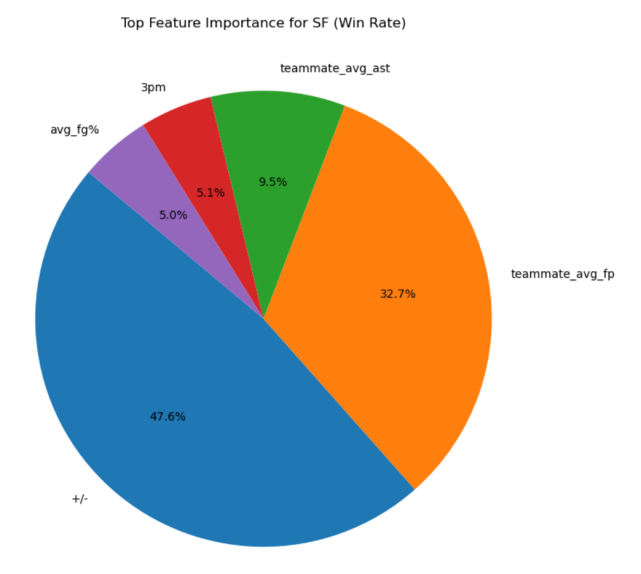




The output result is as follows. For ease of presentation, it is shown in the form of a pie chart.







The resulting feature importance distributions reveal notable differences across positions:

* Point Guards (PG):

For point guards, the model revealed a dominant reliance on plus-minus (+/-), accounting for 86.9% of the total importance. This suggests that the net impact when a PG is on the court overwhelmingly determines win rate, reflecting their central role in orchestrating the offense. Other features such as personal fouls (pf), field goal percentage (avg\_fg%), and team win rate contributed marginally.

* Shooting Guards (SG):

In contrast, shooting guards displayed a more balanced set of influences. Teammate average fantasy points (teammate\_avg\_fp) ranked highest (53.6%), followed by plus-minus (17.2%) and teammate average assists (16.2%). This indicates SGs often rely on off-ball movement and team support to generate scoring opportunities, making teammate quality a critical factor.

* Small Forwards (SF):

For small forwards, both plus-minus (47.6%) and teammate\_avg\_fp (32.7%) were the most significant drivers. This suggests SFs contribute through well-rounded performance that depends not only on their on-court presence but also on how effectively they synergize with the team. Assists, 3PM, and FG% played secondary roles.

* Power Forwards (PF):

PFs demonstrated a diverse distribution, with plus-minus (49.2%) and teammate\_avg\_fp (23.0%) again leading, but also notable contributions from teammate\_avg\_ast (11.9%) and avg\_fg% (10.9%). This indicates PFs play hybrid roles, contributing both offensively and defensively within the team system.

* Centers (C):

For centers, the top features were teammate\_avg\_fp (45.9%) and plus-minus (40.0%), together accounting for nearly 90% of total importance. Other features like assists, 3P%, and FG% had minor influence, emphasizing that centers' contributions are deeply tied to team context and their direct on-court impact.

Across all positions, plus-minus (+/-) and teammate average fantasy points consistently emerge as the most influential predictors of win rate. However, their relative weights vary substantially by position, illustrating the importance of role-specific evaluation. Traditional stats like scoring and rebounding carry more relevance for big men, while assist-based metrics are more prominent for guards. This position-aware analysis supports more accurate player valuation and targeted tactical development.

## **6. Optimization of the scoring system - Introduction of location-aware mechanism. (Interesting discovery)**

### 6.1 Background and Problem

When we were predicting MVPs, we found that the construction of features had a significant impact on the model. Therefore, we attempted to design new scoring models for each player in different positions to see if this could lead to a more excellent scoring system model. Under a unified model, each player uses the same set of scoring logic. However, in reality, players in different positions (center C, forward F, guard G) have significant differences in their responsibilities and tactical functions. A unified standard may cause unfairness for players with different role responsibilities.

For example:

• The center focuses more on defense and rebounds

• The guard emphasizes organization and scoring efficiency

• The forward also takes on responsibilities such as offense and assisting

### 6.2 Optimization of scoring function: Construction of position-aware weighted model

To address the deviation caused by a unified scoring standard for players in different positions, we designed and implemented a position-aware scoring system. The core lies in constructing the function weighted\_score\_expr(pos), which dynamically adjusts the scoring weights of each feature based on the player's position.

(1) Function design concept:

This function receives the player's position field POS. It uses the when().otherwise() structure to determine the position type and assigns a customized feature weighting scheme for each type of position. The design concept is as follows:

• Center (C): Bears key responsibilities on the defensive end. Therefore, the model enhances the weights of avg\_rebounds and avg\_plus\_minus to reflect the influence of rebounds and overall defense.

• Guard (G): Focuses on organization and offensive efficiency. Therefore, the weighting emphasizes avg\_assists and pts\_per\_minute to measure their passing ability and scoring efficiency per minute.

• Forward (F): As a versatile tactical player, we assign relatively balanced intermediate weights to six features to reflect the integrated effect of multiple abilities.

(2) Actual implementation code:

**图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。**

**文本

AI 生成的内容可能不正确。**

Among them, weights\_c, weights\_g, and weights\_f are predefined dictionary structures that record the corresponding weighting coefficients for each feature at different positions. The function's purpose is to weight and sum all the indicators based on the player's POS field, and output the position-aware score field named "score".

(3) Benefits and Advantages:

• Enabled the customization of the scoring function, eliminating the "one-size-fits-all" approach;

* Enabled the regression model to learn the coupling relationship between position and characteristics;
* Enhanced the sensitivity of the scoring system to the tactical responsibilities of players, improving fairness and interpretability.

### 6.3 Update of Regression Model

The ElasticNet regression model was re-trained using the position-weighted scores as the target. (The model performed well in handling complex variables.)

Performance of the updated model:

• R² = 0.6610 (The fit degree slightly decreased)

• RMSE = 0.1215 (The error slightly increased)

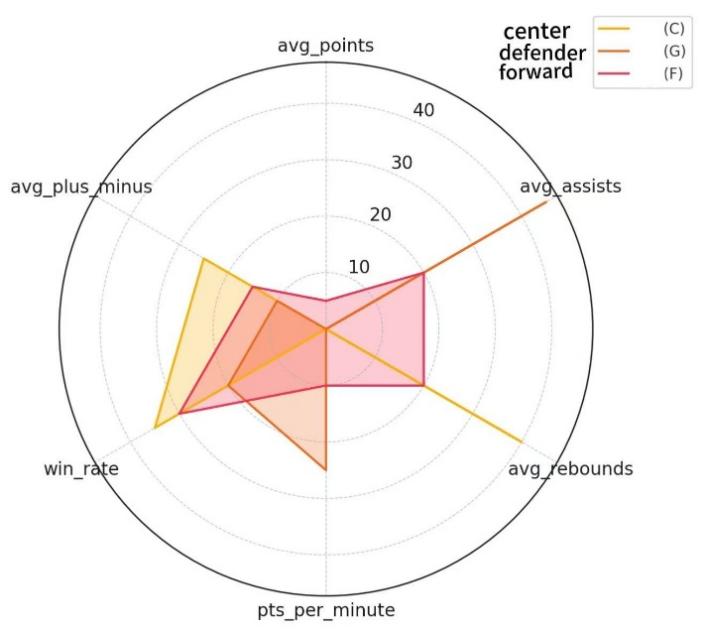
Although the indicators slightly declined, the model showed significant improvements in interpretability and fairness in employment.

**图形用户界面, 文本

AI 生成的内容可能不正确。**

### 6.4 Coefficient Analysis and Occupational Interpretation

Through the regression analysis by position grouping (training models for C, G, and F respectively), we have obtained the following occupational interpretations:

****

### 6.5 Conclusion and Value Enhancement

In this study, we completed the entire process of constructing from the "unified scoring system" to the "position-aware differentiated scoring model" through a systematic modeling process and model optimization path. This was not only an expansion of the scoring model structure but also an enhancement of the combination of data interpretability and practical professional logic.

Firstly, based on the heatmap analysis, we constructed a unified weighted scoring system and trained it using the ElasticNet regression model, obtaining a regression scoring system with good fitting effects. At this stage, we verified the intrinsic correlation between technical statistical indicators and player performance (such as net margin, winning rate), and achieved quantitative analysis of feature interpretability.

Subsequently, in the fifth part, we further introduced the "position-aware scoring mechanism", combined with the actual basketball tactical responsibilities, and designed scoring weight logic for players in three categories: center, forward, and guard. On this basis, we redefined "score" and trained the regression model, maintaining the rigor of statistical modeling while making the scoring model more in line with the professional attributes of the basketball arena.

The greatest contribution of this stage lies in two aspects:

1. Dynamic reflection of feature dimension importance: The position of each player will directly affect the evaluation criteria. We used function mapping to make the same technical indicator have different scoring weights in different positions, thereby achieving true "based on responsibilities" ability evaluation.

2. Optimization of model structure and interpretability: Although the position-aware model has a slightly lower R² compared to the unified model, its interpretability and professional rationality have significantly improved. We believe that the effectiveness of the scoring model not only lies in the numerical fitting accuracy but also in whether it can truly reflect the player's game responsibilities and roles.

Ultimately, this study presents a complete data modeling approach:

• Start from data-driven modeling to construct the initial scoring system;

• Achieve weight reverse learning and interpretable output through regularization regression;

• Then, based on business logic, optimize by role to achieve refined modeling.

This scoring system provides a clear and scalable solution path for player performance evaluation, MVP attribution analysis, and quantification of positional responsibility impact, and has strong transfer and adaptability, which can be used in other seasons, leagues, and tactical type analyses.

This report not only has theoretical significance but also provides a demonstration framework for the practical research on the combination of data and sports.

**图表

AI 生成的内容可能不正确。**

1. **Conclusion**

This project presents a multi-stage analytical framework for evaluating NBA player performance using data-driven methods. Starting from unified regression-based scoring, we progressed to clustering-based player role identification and ultimately introduced a position-aware scoring mechanism. Each phase emphasized not only predictive accuracy but also interpretability and fairness—key factors in real-world sports analytics.

Our findings highlight that traditional statistics alone are insufficient for capturing true player value. Contextual features like plus-minus and teammate quality significantly influence win rates, and role-specific evaluation yields deeper insight than uniform scoring models. The methodology developed in this study offers strong potential for application across different seasons, leagues, or even other team sports, laying the groundwork for future research in performance modeling and tactical analysis.

# **BILIBILI DATA ANALYSIS**

1. **Background Introduction & Objectives**

**Bilibili** is one of the most familiar comprehensive video platforms among young users in China, featuring a large amount of high-quality content and creators. Users can participate in interactions through various methods such as comments, likes, and forwards, creating the unique charm of this platform. Against the backdrop of increasingly rich platform content, how to identify videos with "high quality" characteristics from the vast number of videos is of great significance for optimizing the platform's recommendation mechanism and improving user experience.

This project focuses on the videos related to UIC on the BilbBili platform for research. Firstly, a **crawler program** was written to collect all the video data containing the keywords and tags related to "UIC", including video descriptions, creator influence, release time. Subsequently, based on the **Spark** big data processing framework, the original data was cleaned and analyzed to identify key feature variables. Finally, a **random forest model** was used to build a prediction model for the "**quality**" of the videos, attempting to determine whether a certain video is likely to become popular and highly disseminated high-quality content.

The objectives of this project are as follows:

(1) Utilize **web scraping technology** to collect all videos related to **UIC** on the Bilibili platform and the corresponding data of the UP hosts.

(2) Clean the raw data, including **field filtering**, **deduplication**, and **handling missing values**.

(3) Conduct a preliminary exploratory analysis of the processed data.

(4) Use the **Spark framework** to complete the storage and analysis of the data, build a **random forest model**, and use multi-dimensional features as input to predict the video quality classification results.

(5) Present some **interesting findings** during the project process.

1. **Project environment**

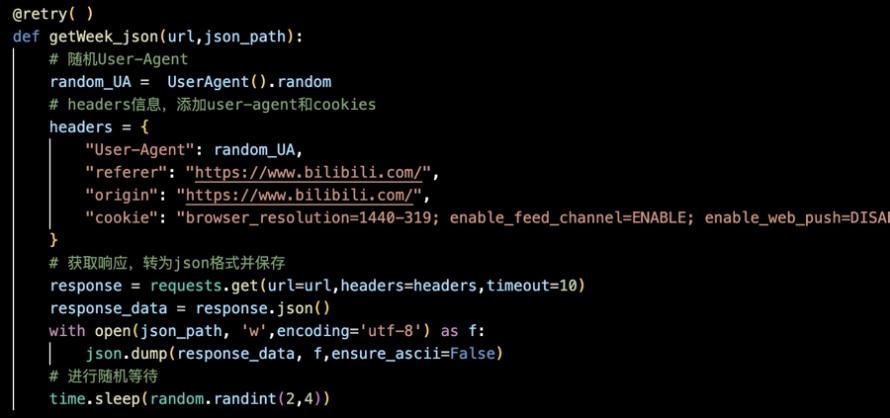
Python: 3.12.7

Apache Spark: 3.2.0

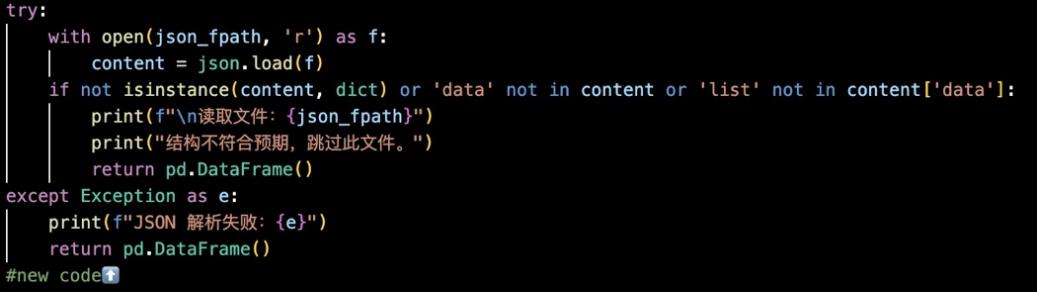
Java: JKD 11

Library: pandas , numpy, spark, seaborn, matlplotlib.ect

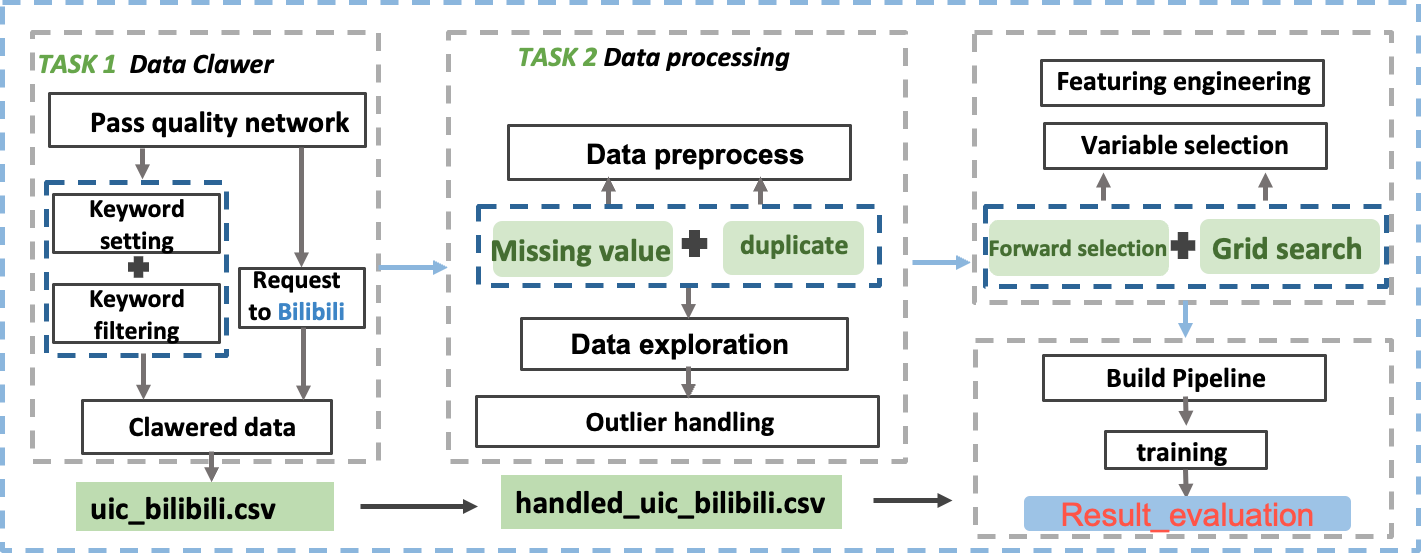
1. **Code implements**
2. For file **bilibili\_weekly\_new.py**. To circumvent Bilibili's **anti-crawling mechanism**, **cookies** were added, and the **crawling time was changed** from the original 1 second to a random 2 to 4 seconds. The code we modified is in the red box in the picture.



1. For file **data\_preprocess\_new.py.** By checking the data obtained from the crawler, it was found that the must-read values for each issue of issue 105 were empty. So, the code was added to skip Issue 105. The code is as follows:



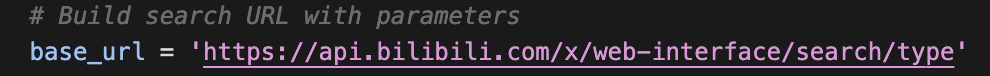
1. **Project procedures**

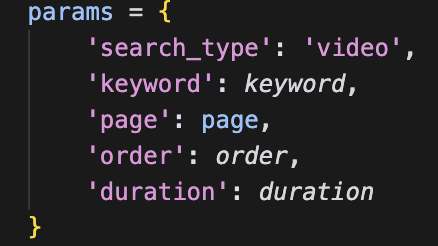


### 4. 1 Data collection

The data collection in this project is implemented in the script 爬虫.py, aiming to collect video content related to "**UIC**" (Beijing Normal University-Hong Kong Baptist University United International College) on Bilibili. Unlike the official "Weekly Must-See" section provided by Bilibili, this project proactively gathers data across the entire platform using **keyword-based search**.

The crawler utilizes the following **official Bilibili API**:

This interface returns a list of video search results based on parameters such as search type: video, keyword, page, duration, and time range.



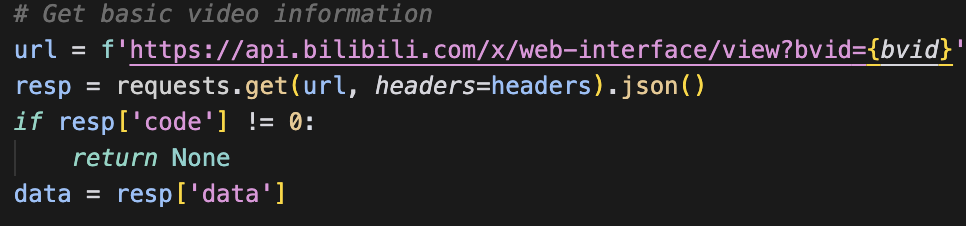
**The data collection process consists of the following steps:**

**(1) Keyword Setting and Filtering**  
The script defines a set of inclusion keywords (e.g., “UIC”, “BNBU”, “联合国际学院”) and exclusion keywords (e.g., “University of Illinois”, “UIC Chicago”) to ensure the accuracy of collected data. A custom filtering function is\_valid\_video is used to examine each video's title, description, and tags to determine if it meets the criteria.

**(2) Request Construction and Anti-Crawling Strategy**  
The crawler is built using the requests library. Each request is equipped with a randomly generated User-Agent (via the fake\_useragent library) to mimic real browser behavior. Cookies are also included to gain access to complete data. To avoid triggering Bilibili’s anti-crawling measures, a random delay of 1–2 seconds is introduced between consecutive requests.

**(3) Fetching Video Details**  
For each search result, the crawler extracts the video’s BVID and further queries detailed information from the following APIs:

* Video details:



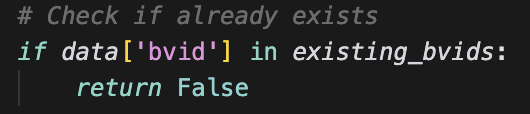
* Uploader’s follower count:



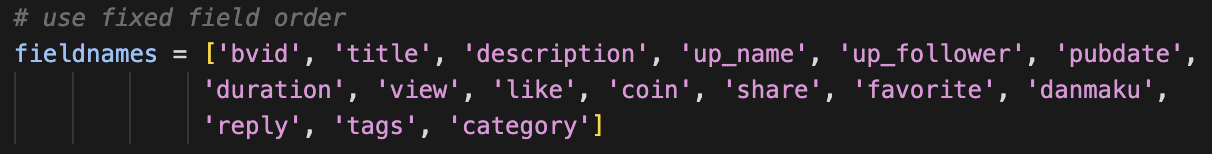
* Video tags:

**(4) Data Storage and Deduplication**

Collected data is saved to a local CSV file named uic\_videos.csv. Before saving, the script checks whether the BVID has already been stored to avoid duplication.

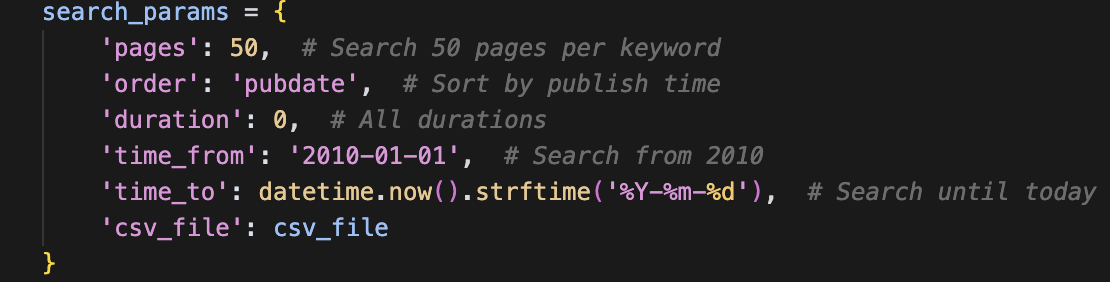


The saved fields include:



* BVID, title, description, uploader name, number of followers, publication time, duration
* Views, likes, coins, shares, favorites, danmaku, comments
* Tags (separated by Chinese comma "、"), and video category

The script iterates through all predefined keywords, with a maximum of 50 pages searched per keyword. The full qualified video information is printed to the console and saved to the CSV file for further processing.



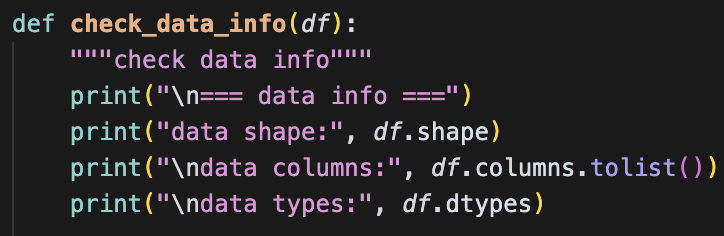
Data collection is complete.

### 4.2 Data Preprocessing

The data preprocessing step was implemented in the clean.py script. It includes loading the original CSV dataset, performing missing value handling and feature engineering, generating data quality reports, and exporting the cleaned data. After preprocessing, the dataset becomes suitable for further analysis and modeling.

**(1) Data Loading and Initial Inspection**

Using the pandas library, the original dataset uic\_videos.csv was loaded. The script checks the basic structure of the data, including dimensions, null values, duplicate rows, and data types:

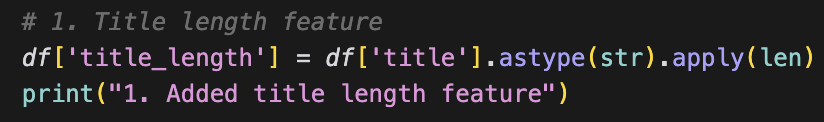


**(2) Feature Engineering**

To enhance the representational power of the dataset, over 20 new features were generated based on video content and metadata. These features fall into the following categories:

**Text-based features**

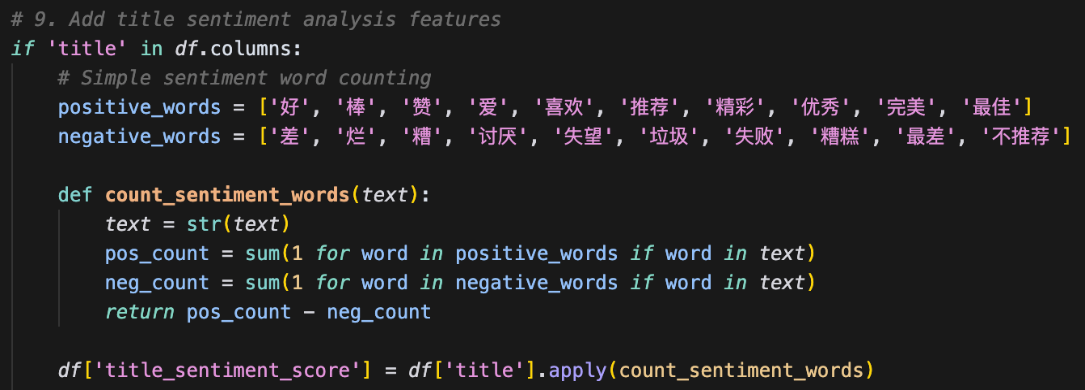
* Title length (title\_length)



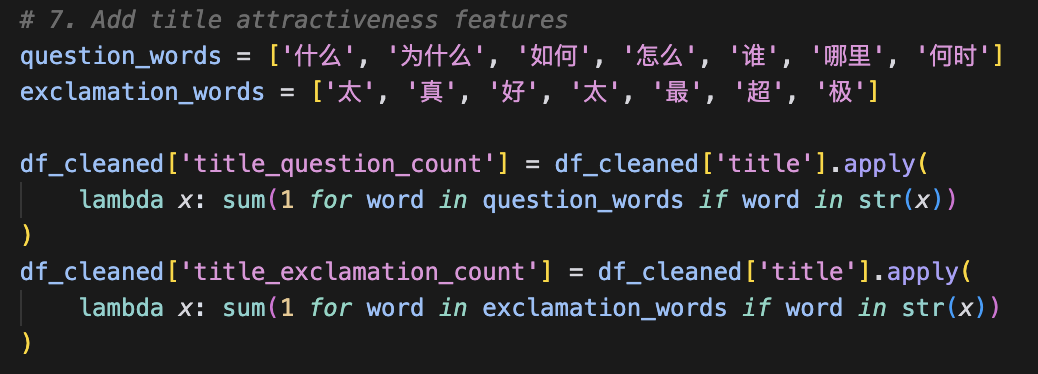
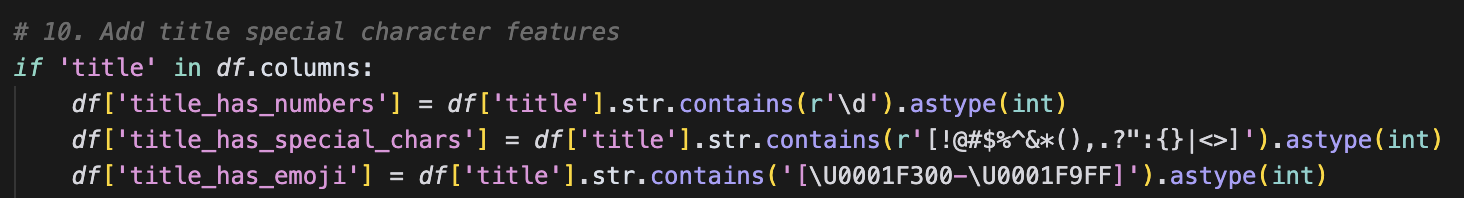
* Title complexity (punctuation, digits, English letters, etc.)



* Sentiment score based on presence of positive/negative words

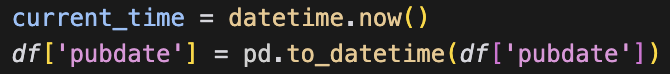


* Presence of special characters, emoji, question/exclamation words

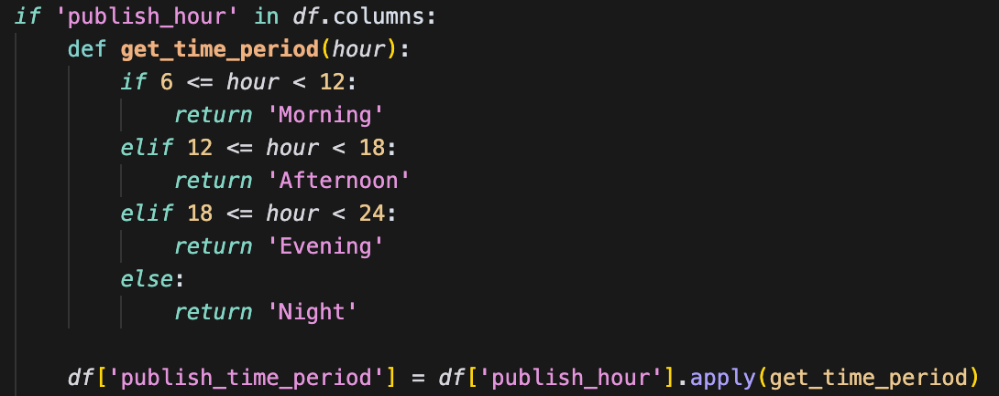


* **Time-Based Features**

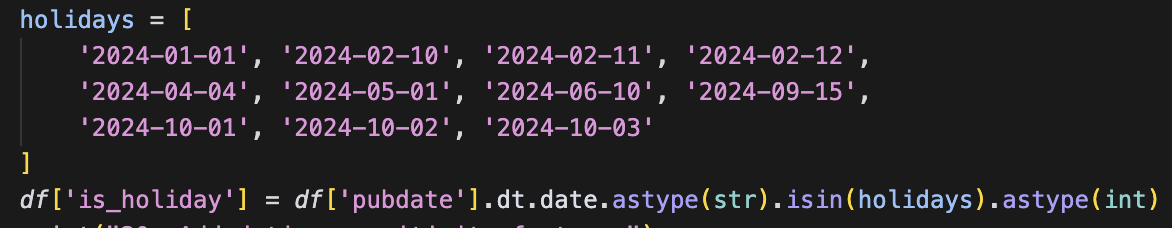
Convert Publication Date to Datetime



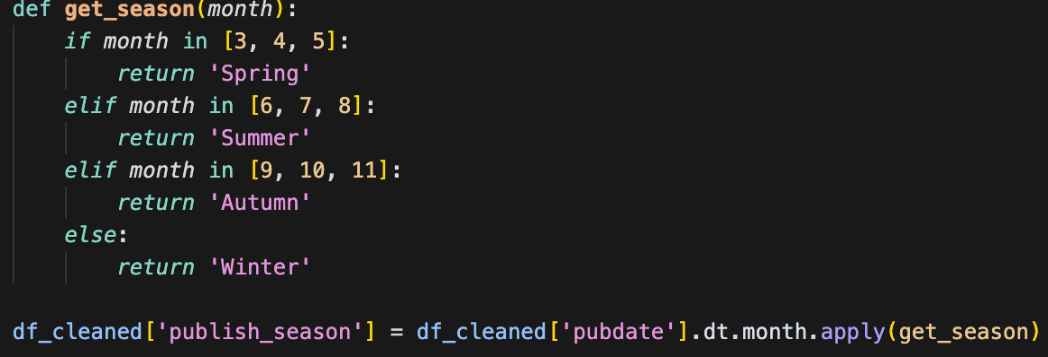
Extract Hour and Time Period of Publication



Weekend & Holiday Indicator



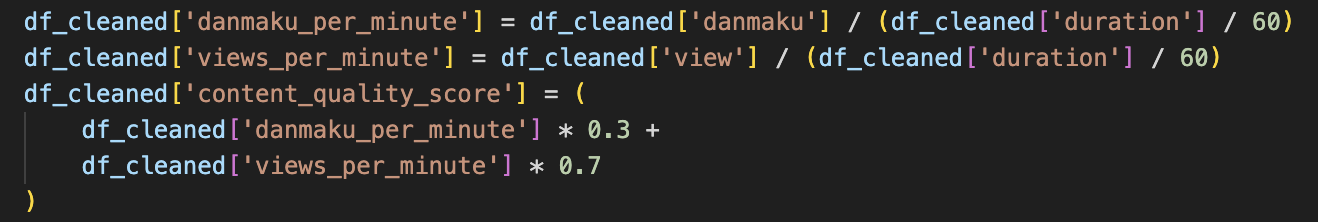
Season of Publication



* **Engagement & Popularity Metrics**

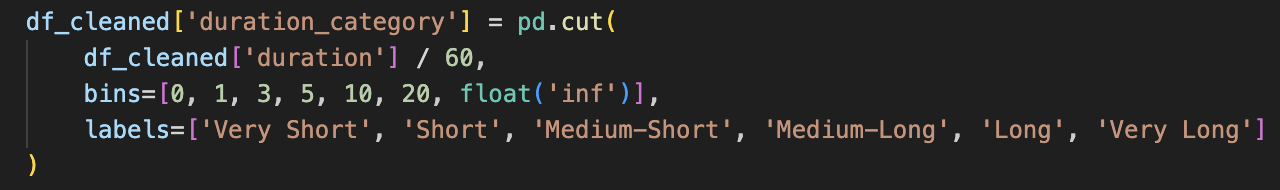
Interaction Rate and Weighted Heat Score

Content Quality Score

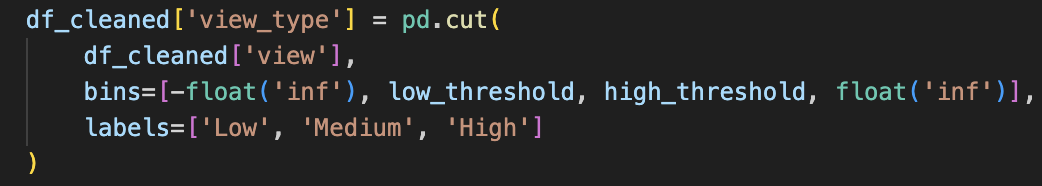


* **Video Type Classification**

Duration-Based Category

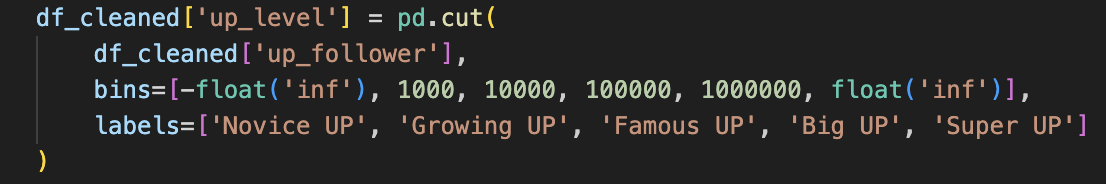


View Count Segmentation

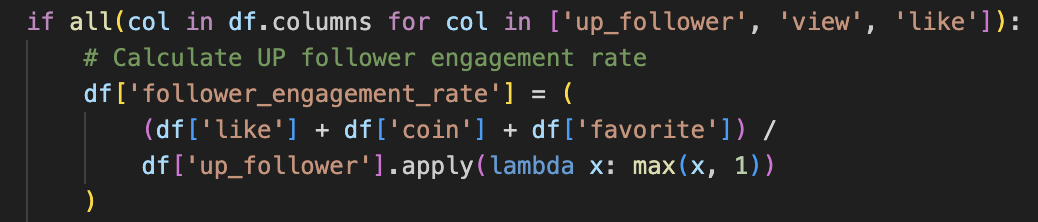


* **Uploader-Related Features**

UP Level Classification



Influence Score (Engagement per Follower)



Data preprocessing is complete

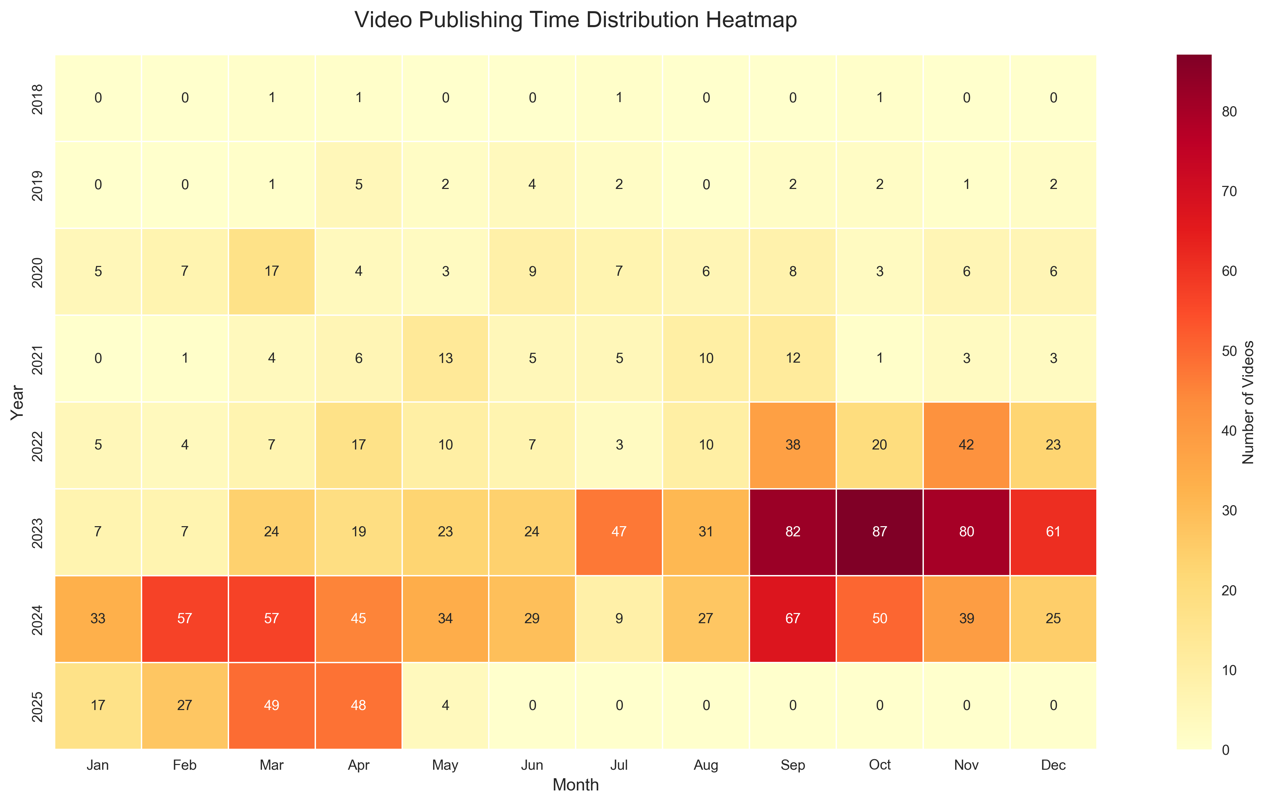
### 4.3 Exploratory analysis

This section presents an **in-depth exploration** of the cleaned **Bilibili video** related with **UIC** dataset using Python visualization tools.

The script analyze\_bilibili\_data.py was used to analyze **video trends, categories, content creators, interaction metrics**, and to generate **visual summaries** such as heatmaps, bar charts, and word clouds.

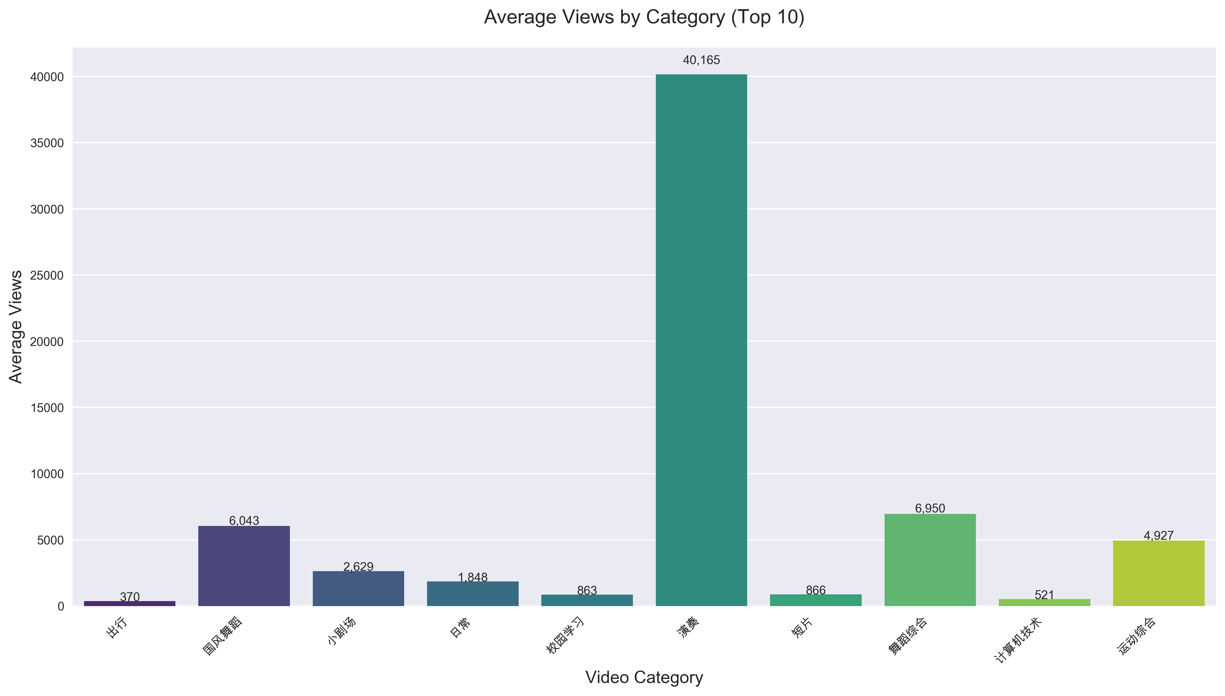
### (1) Temporal Distribution Analysis

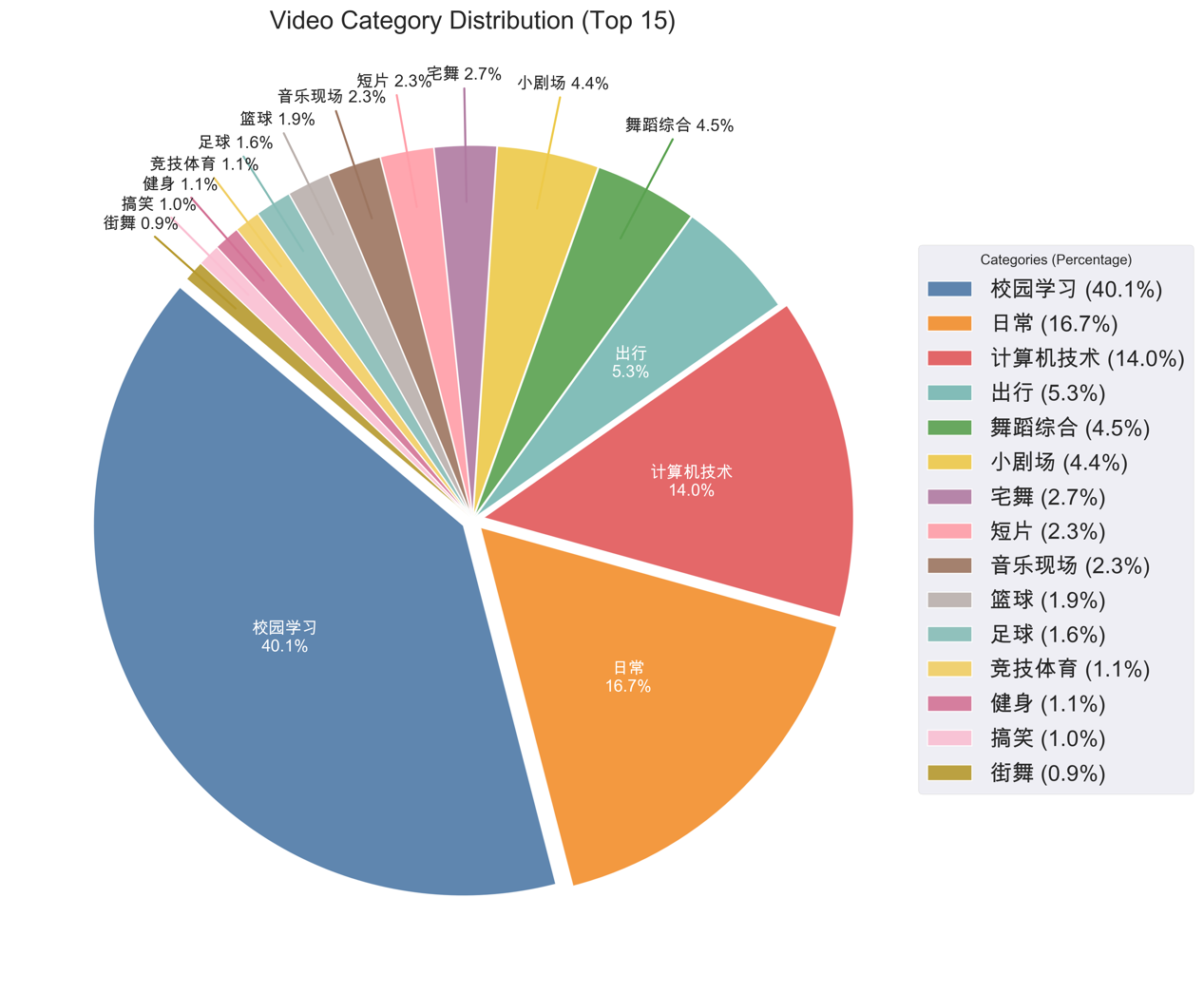
The dataset was first analyzed for its **temporal distribution** across years and months. A pivot table was created to count the number of videos published per month in each year, visualized via a **heatmap**:



The heatmap reveals a clear growth trend in UIC-related video production over the years, with a **significant surge beginning in 2023**; notably, video uploads peak during the **back-to-school season** （September to November）, which may reflect increased campus activity and viewer engagement during this period.

**(2) Video Category Analysis**

**The top 15 most common video categories** were analyzed and visualized via an enhanced **pie chart**. Additionally, the average view count per category (top 10) was displayed using a **bar chart**. 

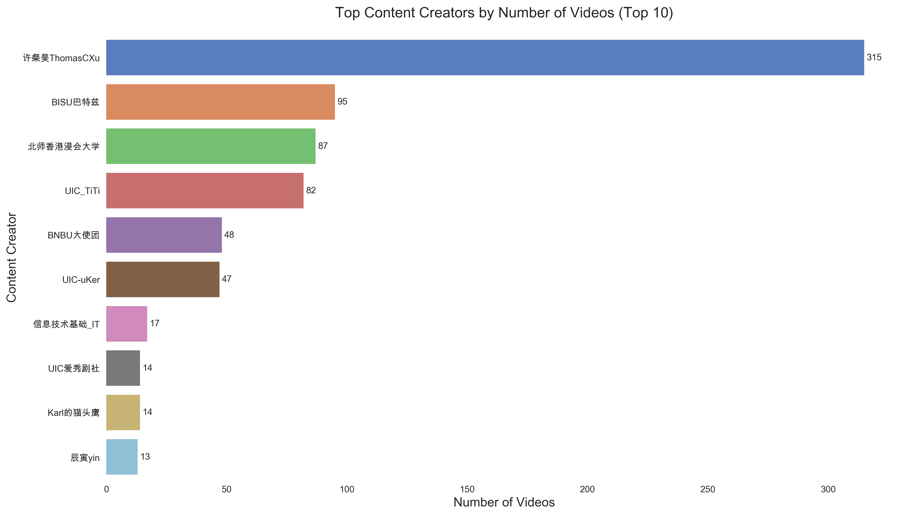


The charts show that while “Campus Learning” is the most frequently uploaded category, videos in the “Performance” category receive the highest average views, suggesting that artistic content, though less common, attracts greater audience engagement.

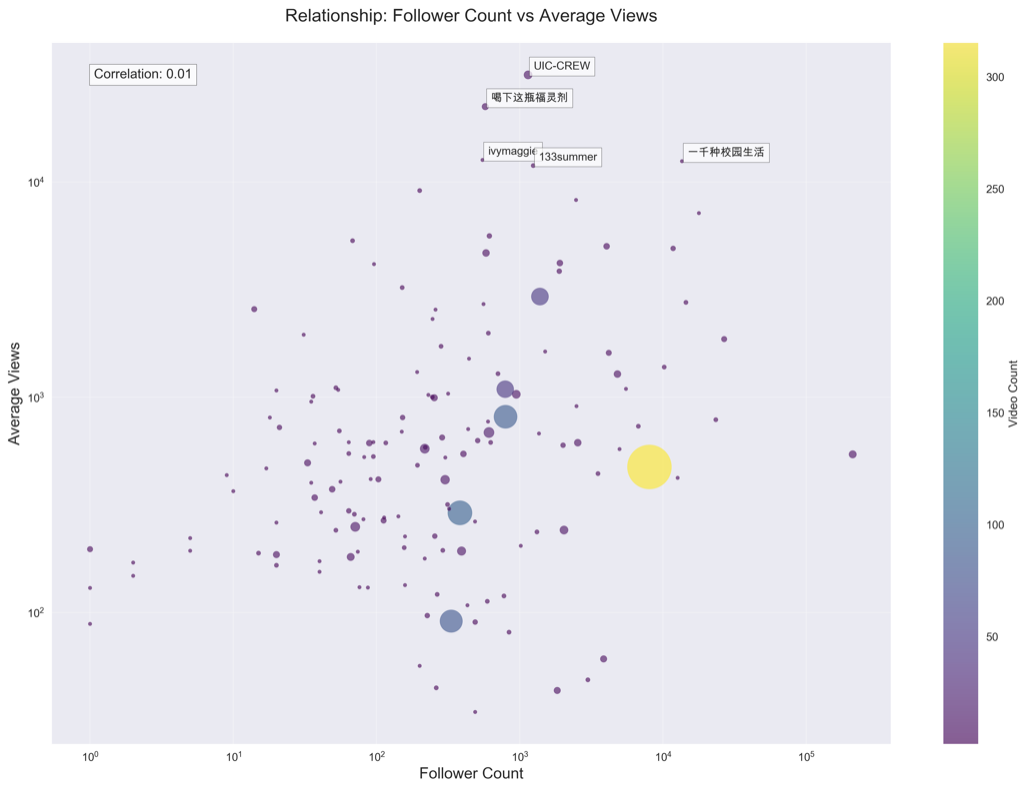
**(3) Uploader Analysis**

The script also explored UP (uploader) behavior and influence:

* Top 10 uploaders by video count were plotted horizontally.
* A scatter plot of **follower count vs average view count** was generated, with circle size representing the number of videos.



This bar chart highlights the top 10 content creators by number of videos, with professor ThomasCXu" standing out significantly by publishing over 300 videos

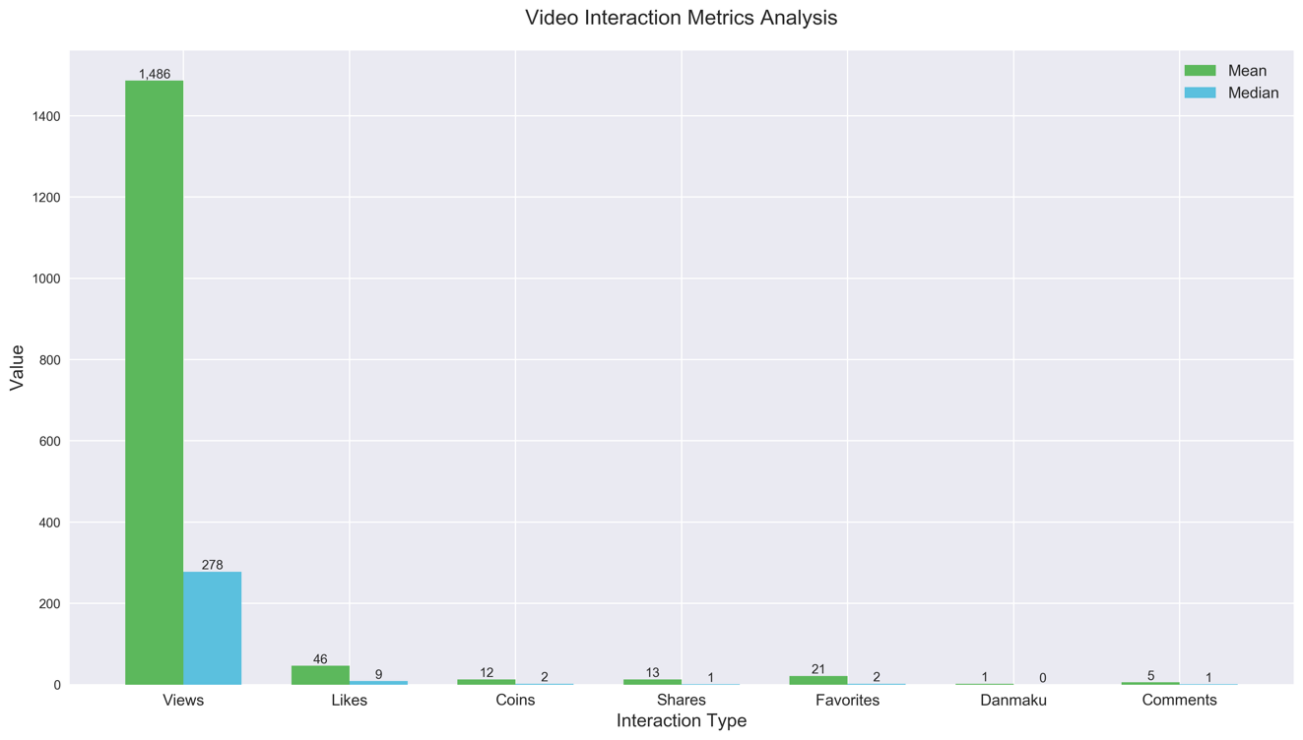


This scatter plot shows that there is virtually no correlation between follower count and average views (correlation = 0.01), indicating that having more followers does not necessarily lead to higher video viewership.

**(4) Interaction Metrics Analysis**

To evaluate user engagement, the following metrics were analyzed and compared in terms of mean and median:

Views, Likes, Coins, Shares, Favorites, Danmaku (real-time comments), Replies



This bar chart shows that all interaction metrics—especially views—exhibit strong right-skewed distributions, where the mean is significantly higher than the median, indicating that a small number of viral videos drive most of the engagement.

### (5) Word Cloud Generation

Three customized word clouds were generated to visually summarize:

* **Popular video categories** (based on top 10 videos)
* **Frequently appearing keywords** in video titles



The word clouds reveal that “Campus Learning” and “Daily Life” are the most dominant video categories, while frequently appearing keywords in titles include “Thomas,” “北师港” , “Programming,” and “Data Analytics,” reflecting the strong academic and technical focus of UIC-related content.

1. **Analyze the UIC data with tree methods**

**The codes for this part are all contained in the file "** **spark\_video\_quality\_analysis 1.py".**

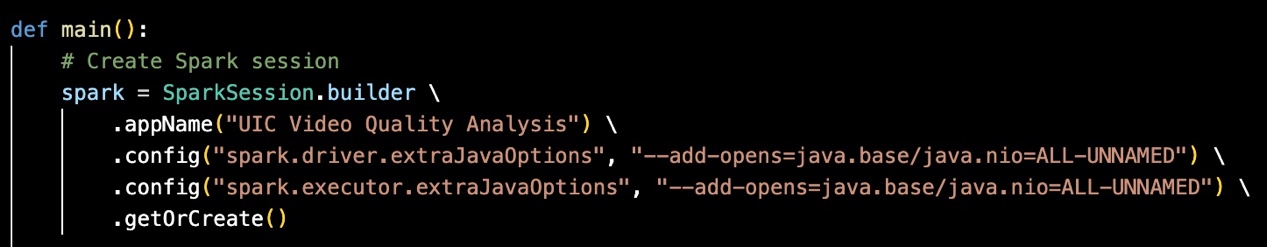
Using **Spark**, feature engineering and statistical analysis were conducted on the cleaned data; subsequently, a video quality scoring index was constructed, and prediction modeling was carried out using the **random forest model** based on **Spark MLlib**. Moreover, the project also completed correlation analysis, feature selection heat maps, and feature importance visualization by combining **Pandas**, **Matplotlib**, and **Seaborn**, forming a complete data-driven modeling process.

### 5.1 Environment and Dependency Import

1. from pyspark.sql import SparkSession
2. from pyspark.sql.functions import col, count, avg, sum, min, max, stddev, when, isnan, isnull, log, corr
3. from pyspark.sql.functions import hour, dayofweek, month, year
4. from pyspark.ml.feature import VectorAssembler, StringIndexer, StandardScaler
5. from pyspark.ml.regression import RandomForestRegressor
6. from pyspark.ml.evaluation import RegressionEvaluator
7. from pyspark.ml import Pipeline
8. import pyspark.sql.functions as F
9. import matplotlib.pyplot as plt
10. import seaborn as sns
11. import pandas as pd
12. import numpy as np

This part imports all the necessary libraries for **data processing (Spark)**, **feature engineering (MLlib)**, **model evaluation**, **visualization (matplotlib/seaborn)**, and **data analysis (pandas/numpy)**.

### 5.2 Initialize Spark session



Create a **SparkSession**. To prevent PySpark from encountering errors due to module access restrictions in the new version of the Java environment, we have added compatibility configuration when creating the Spark session.

### 5.3 Feature Engineering

**Construction of target vector**

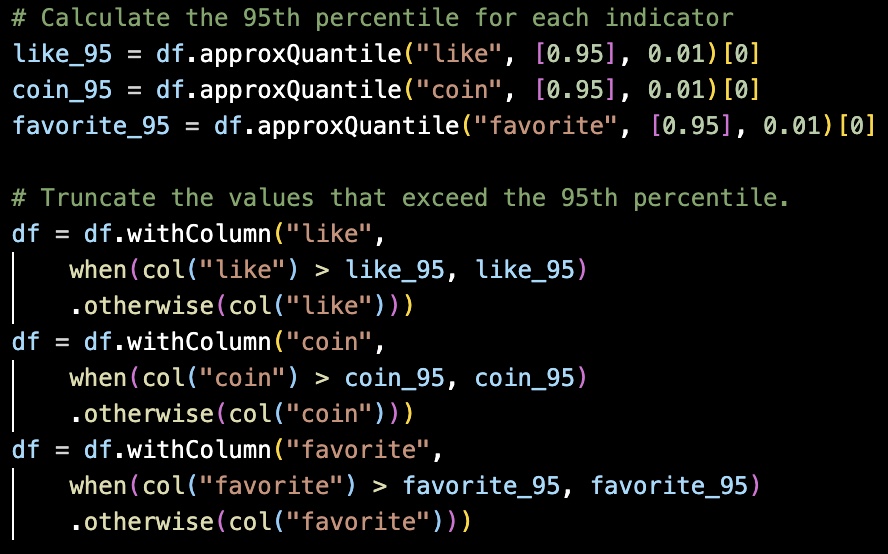
After reading the data related to UIC, several fields including **['view', 'like', 'coin', 'favorite']** were selected as the criteria for evaluating the quality of the videos. Calculate the **Pearson correlation coefficient** for each pair of indicators, and check whether the direction of the coefficients is consistent. The decision on whether to combine them into a single target variable.

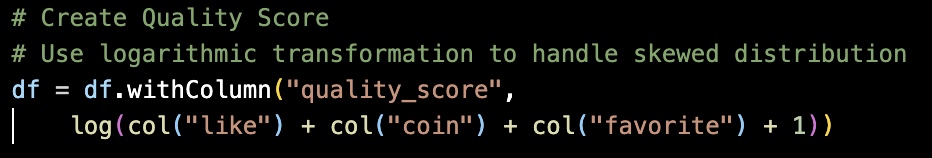




The Pearson correlation coefficient shows that there is a **high positive correlation** between each pair of groups. This indicates that it is a reasonable approach to combine these indicators into a **single target variable**, as they reflect different aspects of the same video quality. This enables a more comprehensive evaluation of video quality and reduces the errors caused by a single indicator.

To reduce the influence of extreme values on data prediction, we employ the **Windsorization method**. Specifically, all values that exceed the **95th percentile** threshold are replaced with that threshold value instead of being deleted. The code is presented as follows：



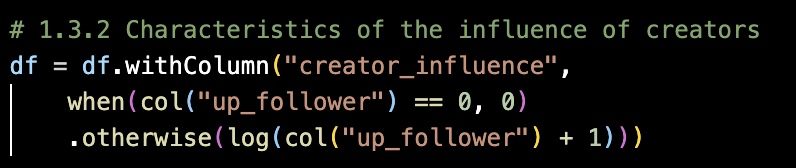
****

After conducting a correlation analysis of these three sets of data and handling the outliers, the target variable "**quality score**" was finally constructed.

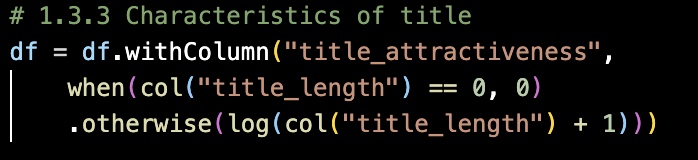
**Construction of input variables**

To enhance the model's predictive ability for video quality, we have constructed a series of new numerical and coding features based on the original data obtained by web crawlers, as follows:

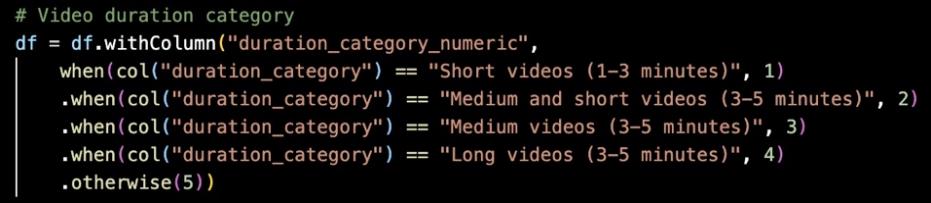
* creator\_influence：



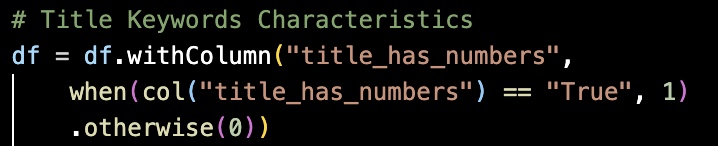
* title\_attractiveness：



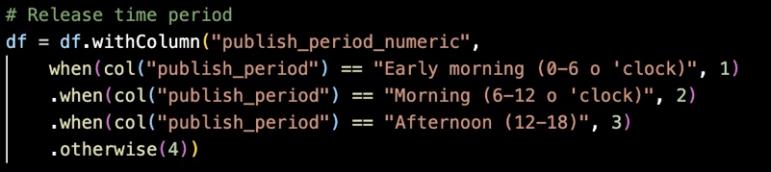
* duration\_category\_numeric:



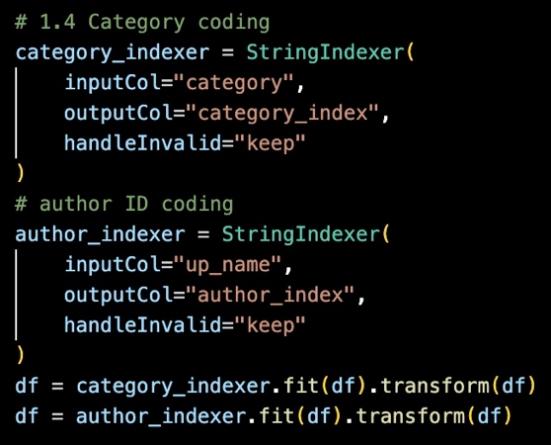
* title\_has\_numbers:



* publish\_period\_numeric:

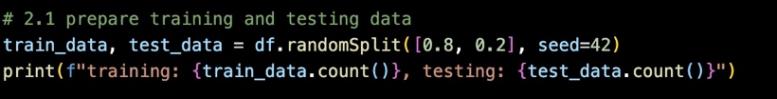


* Category & Author coding:



### 5.4 Feature selection & Model training

In order to facilitate the subsequent training and verification using the machine learning model, the data is divided into the training set and the verification set in an 8:2 ratio:

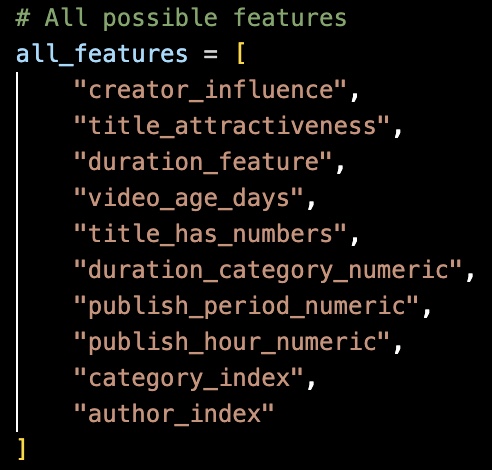


At this point, the number of samples in the training set and the validation set is:

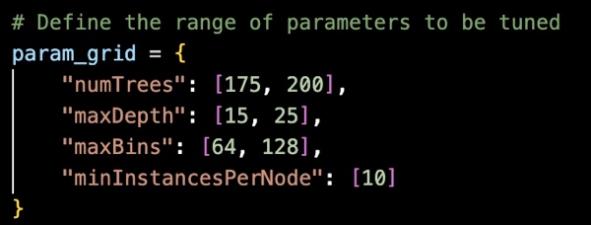


To enhance the predictive ability of the model and reduce the influence brought by **redundant features**, we adopted the **Forward Feature Selection method** to gradually add and evaluate 10 features and simultaneously conduct **grid search** of the random forest model. This process builds the model on the training set, takes the **R2 value** as the index on the test set, and selects the **optimal features** and r**andom forest parameters** for each round.

A list of all possible features in the model：



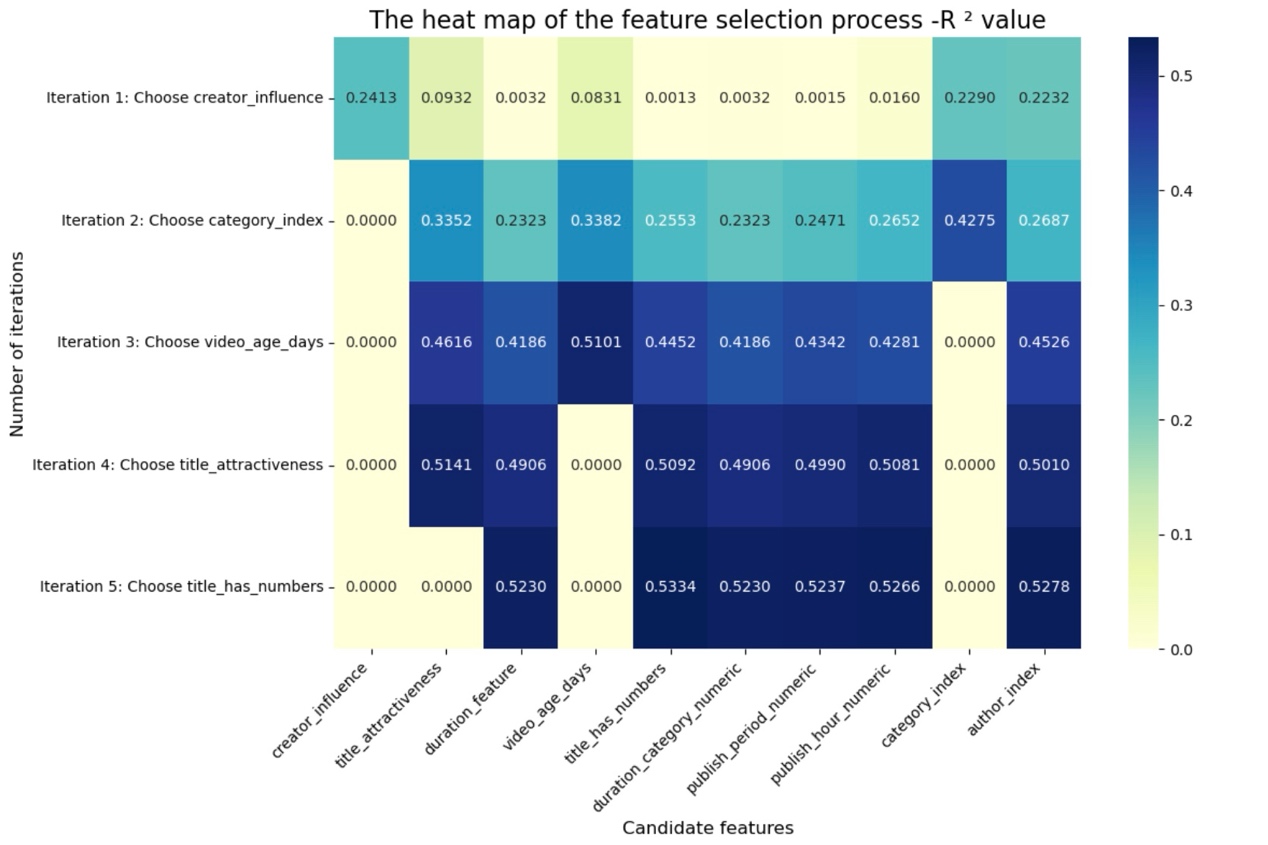
The parameters provided by **grid search：**



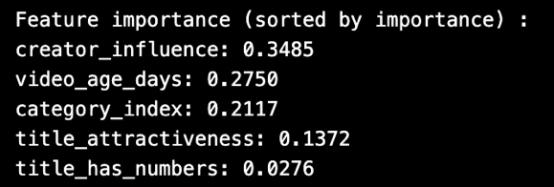
There are a total of 10 features. One new feature is attempted to be added in each iteration. It stops when there is no significant improvement for 3 consecutive times (**no\_improvement\_count < 3**).

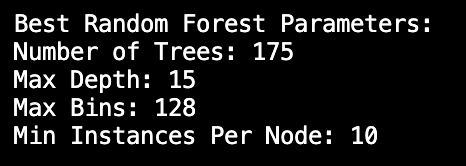
So feature selection will undergo up to 10 iterations at most. For each feature, **eight combinations of parameters** will be attempted. In this random forest model, the **maximum total number of iterations** 10 × 8 = 80 times.

For the **Forward Selection method**, the performance results of the test set **R2** in each round are visualized as a **heat map**. As shown in the figure, the horizontal axis in the figure represents 10 features, and the vertical axis represents the iterative process of each round. In each round, we add the features that have not yet been selected to the current model one by one and evaluate their **R2** performance on the test set. The darker the color, the better the model performance brought by the corresponding feature. The best among them is **R2 = 0.5334**:



After **variables selection** and **grid search**, the selected best variables and optimal parameters are respectively:

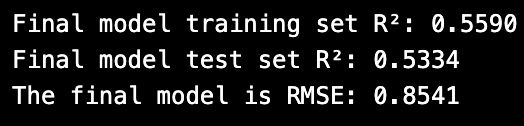




The influence of the **UP host** and the r**elease time of the video** are the two most crucial factors, which play a decisive role in predicting the video quality score. Content **categories** are also very important, reflecting the differences in audiences and content preferences. The features related to the title (such as attractiveness and whether it contains numbers) have some effect, but they are far less effective than the previous three.

This is in line with practical experience: bloggers with a large number of followers, videos in popular categories, and those released at appropriate times are more likely to achieve high-quality scores.

Utilize the selected variables and the output results of the optimal parameters：



The test set **R²** is approximately **0.53**34, indicating that the model can explain the variance of approximately 53% of the target variable (video quality score). Although this **R²** is not high, based on our data with a **small sample size** and **high randomness**, it is still a good result.

Use the **first five** videos as examples for prediction：

文本

AI 生成的内容可能不正确。

图表

AI 生成的内容可能不正确。

Take the **previous five video**s as examples for prediction demonstration and visualize the **category** as the identification tag.

**Horizontal coordinate:** Video category.

**Vertical coordinate:** Video quality score.

**Blue pillar:** Actual mass fraction.

**Pink pillar:** The mass fraction predicted by the model.

**Dotted line:** y = x reference line, indicating the ideal situation where "predicted value = actual value".

The difference between the predicted value and the actual value can be visually observed. The model's predictions for most categories are relatively accurate, especially for the Computer Technology category, which is very close to the dotted line in the figure on the right.

1. **Conclusion**

In this project, we completed the quality score prediction for all the data related to **UIC on Bilibili** from data acquisition to system modeling.

Its significance lies in:

* **Content creation optimization:** By predicting the video quality score, content creators can understand which factors (such as the create influence, the release time of the video, the content category, the attractiveness of the title, etc.) have the greatest impact on video performance, and thus optimize the content in a targeted manner to improve the quality score of the video.
* **New video performance forecast:** For newly released or soon-to-be videos, the model can predict the quality score, helping UP owners and platforms assess their potential and adjust promotion strategies in a timely manner.
* **Promoting data-driven strategies:** Through model prediction, managers can speak with data and formulate more reasonable content operation strategies, such as focusing on supporting certain categories, optimizing the release time window, and guiding UP hosts to enhance their influence.

In conclusion, this project has crawled and analyzed the video content related to UIC, and also brought a new method for predicting video quality by establishing a mathematical model, which has strong application value and promotion prospects.

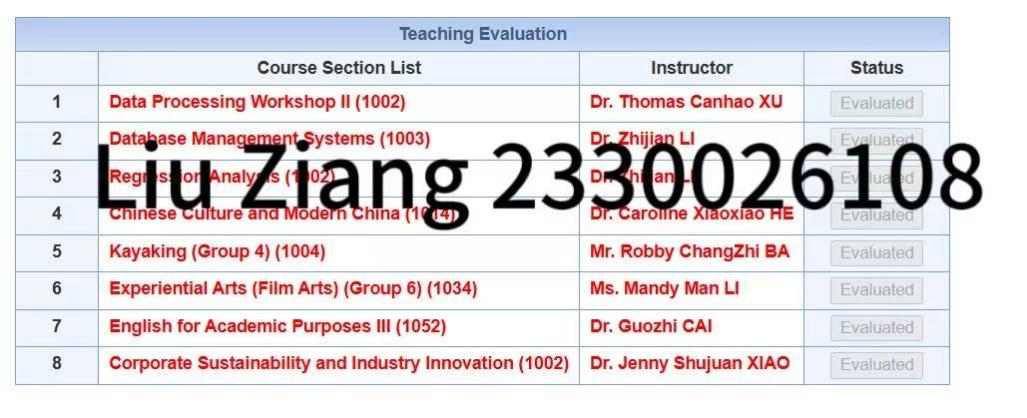
**CFQ Part:**

**图形用户界面

AI 生成的内容可能不正确。图片包含 日历

AI 生成的内容可能不正确。图形用户界面, 文本

AI 生成的内容可能不正确。**

****