**FINAL REPORT ON PREDICTION MODELS OF A COLLEGE BASKETBALL PLAYER BEING DRAFTED TO JOIN THE NBA LEAGUE**

Subject Name: Advanced Machine Learning Applications

Author: Vu Ngoc Anh Le

Student ID: 14195743

1. BUSINESS UNDERSTANDING

**Determine business objectives:**

The NBA (National Basketball Association) draft is an annual event in which teams select players from their American colleges as well as international professional leagues to join their rosters. Specifically, the teams that performed the worst in the previous are given the first chances to pick new players for their teams for the next season.

This project is a Kaggle competition and aims to build a machine learning model which can generate the probability of being drafted for each player. The data scientist is required to submit their prediction probability to Kaggle and can see how accurate their outcomes are by watching the accuracy score in the Leaderboard. To do this, a number of machine learning models will be used. Ultimately, the best model with the most accurate probability will be selected.

The results of this project is important to both the teams and the players. Apparently, an accurate model will help improve the performance of the team and accelerate its ranking in the league. Likewise, for the player, if the pick proves to be a good fit, it will be a great beginning of their NBA career.

**Determine data mining goals:**

From a technical data mining perspective, this project is expected to generate a clean and reliable data input with different data preparation steps. In order to achieve that, it is important to check whether the dataset has any duplicated records. In addition, in case of missing values, the data scientist needs to decide whether to remove those records or to fill them with a suitable method. Another essential step is to ensure the dataset is free of outliers, which can be done by investigating the descriptive statistics of the dataset. It is also noticeable that the provided dataset is heavily biased towards target value 0, i.e. not being drafted. The data scientist needs to decide a suitable technique to handle this matter during the data preparation process.

**Produce project plan:**

For phase 2 (Data Understanding), phase 3 (Data Preparation), phase 4 (Modeling), and phase 5 (Evaluation), Python Jupyter Notebook or any Python IDE is the required coding tool to perform the data wrangling tasks. In phase 6 (Deployment), a Python Flask Application could be created and then deployed on Streamlit or Heroku to run a web application. This helps anyone with or without IT technical expertise can utilize it easily.

1. DATA UNDERSTANDING

There are training and testing datasets provided for the prediction task. The datasets contains information about the players and their past performance metrics.

In the training dataset, there are 64 columns, including 63 feature columns and 01 target column. However, in the testing dataset, there is no target column provided. There are 56,091 and 4,970 rows in the training and testing datasets respectively.

|  |  |
| --- | --- |
| Column | Description |
| team | Name of team |
| conf | Name of conference |
| GP | Games played |
| Min\_per | Player's percentage of available team minutes played |
| ORtg | ORtg - Offensive Rating (available since the 1977-78 season in the NBA) |
| usg | Usg% - Usage Percentage (available since the 1977-78 season in the NBA) |
| eFG | eFG% - Effective Field Goal Percentage; the formula is (FG + 0.5 \* 3P) / FGA |
| TS\_per | TS% - True Shooting Percentage; the formula is PTS / (2 \* TSA) |
| ORB\_per | ORB% - Offensive Rebound Percentage (available since the 1970-71 season in the NBA) |
| DRB\_per | DRB% - Defensive Rebound Percentage (available since the 1970-71 season in the NBA) |
| AST\_per | AST% - Assist Percentage (available since the 1964-65 season in the NBA) |
| TO\_per | TOV% - Turnover Percentage (available since the 1977-78 season in the NBA) |
| FTM | Free Throws |
| FTA | Free Throw Attempts |
| FT\_per | Free Throw Percentage; the formula is FTM / FTA. |
| twoPM | 2P - 2-Point Field Goals |
| twoPA | 2PA - 2-Point Field Goal Attempts |
| twoP\_per | 2P% - 2-Point Field Goal Percentage; the formula is 2P / 2PA. |
| TPM | 3P - 3-Point Field Goals (available since the 1979-80 season in the NBA) |
| TPA | 3PA - 3-Point Field Goal Attempts (available since the 1979-80 season in the NBA) |
| TP\_per | 3P% - 3-Point Field Goal Percentage (available since the 1979-80 season in the NBA) |
| blk\_per | BLK% - Block Percentage (available since the 1973-74 season in the NBA) |
| stl\_per | STL% - Steal Percentage (available since the 1973-74 season in the NBA) |
| ftr |  |
| yr | Student’s year of study: `Fr` for freshmen, `So` for sophomores, `Jr` for juniors, `Sr` for seniors |
| ht | Height of student |
| num | Player's number |
| porpag | Points Over Replacement Per Adjusted Game |
| adjoe | AdjO – Adjusted offensive efficiency |
| pfr |  |
| year | Season's year |
| type | Type of metrics displayed |
| Rec\_Rank | Recruiting rank i.e. what the player was ranked as a recruit coming out of high school |
| ast\_tov | Ratio Assists against Turnover |
| rimmade | Shots made at or near the rim |
| rimmade\_rimmiss | Sum of Shots made at or near the rim and Shots missed |
| midmade | Two point shots that were not made at or near the rim |
| midmade\_midmiss | Sum of Two point shots that were not made at or near the rim and Shots missed |
| rim\_ratio | Ratio between Shots made at or near the rim against Shots missed |
| mid\_ratio | Ratio between Two point shots that were not made at or near the rim and Shots missed |
| dunksmade | Dunks made |
| dunksmiss\_dunksmade | Sum of Dunks made and Dunks missed |
| dunks\_ratio | Ratio between Dunks made and Dunks missed |
| pick | Order of NBA draft |
| drtg | DRtg - Defensive Rating (available since the 1973-74 season in the NBA) |
| adrtg | Adjusted DRtg |
| dporpag | Adjusted porpag |
| stops | Stops - Stops; Dean Oliver's measure of individual defensive stops |
| bpm | BPM - Estimate the player’s contribution in points above league average per 100 possessions played |
| obpm | Offensive BPM |
| Offensive BPM | Defensive BPM |
| gbpm | BPM 2.0 |
| mp | MP - Minutes Played (available since the 1951-52 season) |
| ogbpm | Offensive BPM 2.0 |
| dgbpm | Defensive BPM 2.0 |
| oreb | ORB - Offensive Rebounds (available since the 1973-74 season in the NBA |
| dreb | DRB - Defensive Rebounds (available since the 1973-74 season in the NBA) |
| treb | TRB - Total Rebounds (available since the 1950-51 season) |
| ast | AST - Assists |
| stl | STL - Steals (available since the 1973-74 season in the NBA) |
| blk | BLK - Blocks (available since the 1973-74 season in the NBA) |
| pts | PTS - Points |
| player\_id | Unique identifier of player |
| drafted | Target - Was the player drafted at the end of the season |

There are a number of columns that have missing values. These missing values were replaced with the mean value of that column. Compared to the dropping method, when applying this ffill method, data was not wasted and could be effectively used to train the model.

It was also noted that the distribution of target values is heavily imbalanced, in which value 0 accounting for more than 99% (Figure 1). In this project, we will address this issue with an oversampling method called SMOTE to balance the datasets.

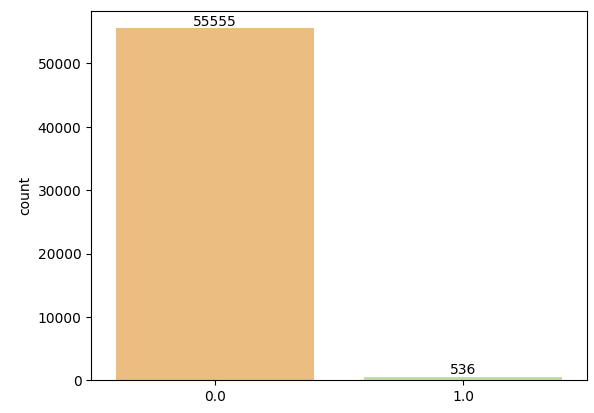


Figure 1 – depicts imbalanced distribution of target values

1. DATA PREPARATION

To clean the datasets, below steps were performed:

1. Drop the type, player\_id,Rec\_Rank, dunks\_ratio, pick columns:

* Column type: There is only one value in this column, which is 'all'. Hence there is no need to keep this column.
* Column player\_id: This column does not play any role in the prediction task.
* Columns Rec\_Rank, dunks\_ratio, pick: These columns contain missing values > 50% of the total values

1. Column num:

* Replace any non-numeric values with NaN values

1. Apply **LabelEncoder** technique to categorical columns to convert them into numerical values, including team, conf, yr
2. **Fill missing values** in the dataframe with their mean values
3. Apply **SMOTE** oversampling technique to address the imbalanced dataset
4. **Feature elimination with correlation matrix** to select the most relevant features of the dataset
5. **Scale data**:
   * Apply StandardScaler method on X\_train, X\_val, X\_test
6. **Split the dataset**:
   * As the testing dataset had no target column, we could not use this dataset to evaluate performance of the models. Therefore, it is essential to split the training dataset into training and validation sets with the test size of 20%

* Split X\_data and y\_data 🡪 X\_train, X\_val, y\_train, y\_val with test size of 20%

1. MODELING

There are a total of 3 experiments implemented to find out the best performing one.

1. Logistic Regression Classifier

* No feature selection performed
* No SMOTE oversampling technique performed
* Train Model with default hyperparameters

1. Adaboost Classifier

* Feature selection performed with Correlation Matrix Heatmap
* SMOTE oversampling technique performed
* Train Model with Default Hyperparameters
* Improve Model Performance with Hyperparameter Tuning
  + Hyperparameter Tuning with learning\_rate=0.05

1. RandomForest

* Feature selection performed with Correlation Matrix Heatmap
* SMOTE oversampling technique performed
* Train RandomForest Model with Default Hyperparameter
* Improve Model Performance with Hyperparameter Tuning
  + n\_estimators Hyperparameter
  + max\_depth Hyperparameter
  + min\_samples\_leaf Hyperparameter
  + max\_features Hyperparameter

1. EVALUATION

The AUROC scores were used to evaluate model performances. After conducting 3 models, **RandomForest model with default hyperparameters proved to be the most effective machine learning model amongst the other ones as it is able to generate predictions having highest recall score compared to the actual values, and with minimal overfitting. Its AUROC scores on the training and validation datasets are 1.0 and 0.999873430745366 respectively.**

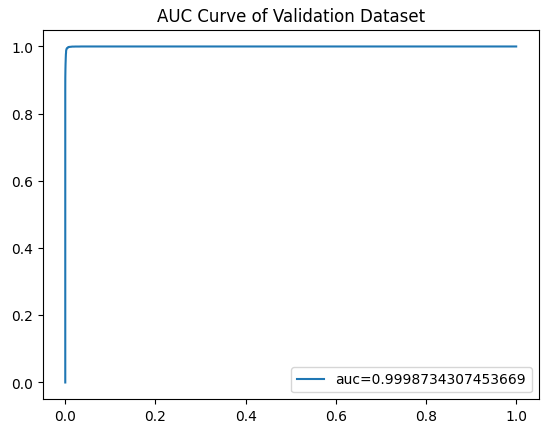
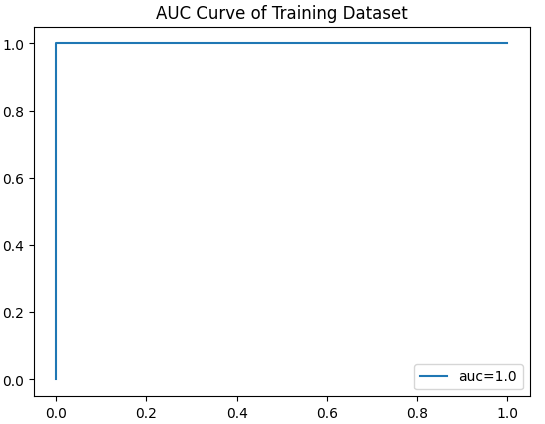
|  |  |  |
| --- | --- | --- |
| ****Experiment 1 (Logistic Regression Classifier)**** | | |
| ****Train model with default hyperparameters**** | **Training** | 0.9891360804890494 |
| **Validation** | 0.9874001668651553 |

|  |  |  |
| --- | --- | --- |
| ****Experiment 2 (Adaboost Classifier)**** | | |
| ****Train model with default hyperparameters**** | **Training** | 0.9965191666385046 |
| **Validation** | 0.9958602138308584 |
| ****Train model with learning\_rate=0.05**** | **Training** | 0.9965191666385046 |
| **Validation** | 0.9958602138308584 |

|  |  |  |
| --- | --- | --- |
| ****Experiment 3 (RandomForest Classifier)**** | | |
| ****Train model with default hyperparameters**** | **Training** | 1.0 |
| **Validation** | 0.9998734307453669 |
| ****Train model with**** n\_estimators=90 | **Training** | 1.0 |
| **Validation** | 0.999869927420505 |
| ****Train model with**** n\_estimators=90, max\_depth=30 | **Training** | 0.9999965275615045 |
| **Validation** | 0.9996575246817134 |
| ****Train model with**** n\_estimators=90, max\_depth=18 | **Training** | 0.9994360241472875 |
| **Validation** | 0.9987519901315267 |
| ****Train model with**** n\_estimators=90, max\_depth=18, min\_samples\_leaf=2 | **Training** | 0.9995628033374554 |
| **Validation** | 0.9988653520539349 |
| ****Train model with**** n\_estimators=90, max\_depth=18, min\_samples\_leaf=6, max\_features=12 | **Training** | 0.9991857853148767 |
| **Validation** | 0.9983015638064566 |

Figure 2 and Figure 3 illustrate the AUC Curve on traning and validation datasets of the RandomForest model with default hyperparameters.

**Figure 2 Figure 3**



From the **best model, feature importance was calculated (Figure 4). The 3 most important features are Points Over Replacement Per Adjusted Game (porpag), adjusted porpag, and player’s estimated contribution in points above league average per 100 possessions played.**

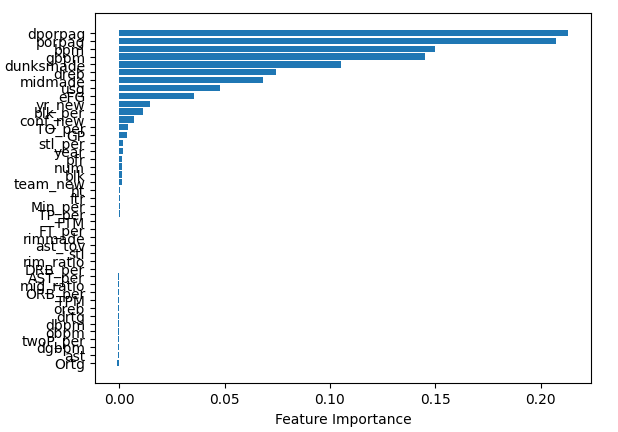


Figure 4 – illustrates feature importance of the best model

1. DEPLOYMENT

When it comes to deployment, it is critical to ensure not only the effectiveness but also the efficiency of the model are achieved. In other words, the model needs to produce accurate predictions, and be optimized in terms of execution speed as well, especially when the size of the dataset is built up over time.

As the end users normally do not have technical knowledge or applications to run the machine learning model, it is essential to build an web app with questionnaires covering all important factors as the data input. Python Flask App and Streamlit or Heroku are suggested solutions to build this web app. When the end users click the ‘Submit’ button on the web app, the back-end system will run the model and generate the predictions accordingly. As regards to maintenance and monitoring of the deployment, it is essential for machine learning engineers to frequently update the data and retrain the model when needed.

This project contains little to none potential data privacy risk as there is no sensitive or personal information in the dataset.