EXPERIMENT REPORT

Student Name	Vishal Raj
Project Name	NBA Draft Prediction
Date	31st August 2023
Deliverables	Vishal_Raj-14227627-week3_RFXG3rdExp.ipynb Random Forest, with & without feature engineering XGBoost, with & without feature engineering Github Repository: https://github.com/vishalraj247/NBA_Draft_Prediction.git

---

1. EXPERIMENT BACKGROUND

1.a. Business Objective

The primary goal of this project is to predict whether a college basketball player will be drafted to join the NBA league based on his statistics for the current season. This prediction is of paramount importance for various stakeholders:

- **NBA Teams**: Accurate predictions can guide team managers and coaches in their draft decisions, helping them identify potential star players or hidden gems.
- **Sports Commentators and Analysts**: Having a data-driven model can provide commentators with insights to enrich their discussions and analyses during draft events.
- **Players and Their Coaches**: For college players aspiring to join the NBA, understanding their draft likelihood can help them focus on areas of improvement.
- **Fans**: NBA fans are always eager to know about the next big player. Predictions can enhance their engagement and discussions around draft events.

The impact of accurate predictions is multifaceted. It could mean the difference between securing a future star player or missing out on one for the NBA teams. For players, it could provide clarity on their career trajectory. However, incorrect predictions could lead to misinformed decisions, potential missed opportunities for teams, and misplaced expectations for players.

1.b. Hypothesis	Given the statistics of a college basketball player for a particular season, it is hypothesised that certain performance metrics can be indicative of a player's likelihood of being drafted into the NBA. Historically, players with higher points, assists, rebounds, and other key metrics have been observed to have a higher chance of being drafted. This hypothesis is grounded in the belief that NBA teams look for players who have demonstrated exceptional skills and performance during their college years. By analysing these statistics, we aim to uncover patterns and relationships that can predict a player's transition from college basketball to the NBA.
1.c. Experiment Objective	Lastly with the same objective, this experiment aims to build a predictive model to determine the probability of a player being drafted accurately. The expected outcome is an AUROC score that is significantly higher than random guessing (0.5) and the previous experiments. By achieving this, we hope to provide a tool to assist various stakeholders, from sports analysts to NBA teams, in making informed decisions about potential draft picks. The possible scenarios resulting from this experiment include: • **Highly Accurate Model**: This would validate our hypothesis and provide a reliable tool for predicting draft outcomes. • **Moderately Accurate Model**: While imperfect, this would still offer valuable insights and could be improved with further experimentation. **Inaccurate Model**: This would indicate that other factors, possibly outside of the provided statistics, play a significant role in draft decisions.

---

2. EXPERIMENT DETAILS	
2.a. Data Preparation	Data preparation is a pivotal step in any machine learning project, ensuring that the data is clean, relevant, and well-structured for model training. For this experiment, the following steps were taken: • **Data Loading:** Multiple datasets were loaded using the 'load_data' function from the 'make_dataset.py' module. These datasets include 'train.csv', 'test.csv', 'sample_submission.csv', and 'metadata.csv'. • **Data Inspection:** The 'display_head' function was used to display the first few rows of each dataset, providing a quick overview of the data. • **Missing Values Analysis:** The 'missing_values_analysis' function was used to identify columns with missing values and their percentages. This information is then displayed using a DataFrame. • **Target Distribution:** The 'target_distribution' function was used to analyze the distribution of the target variable, 'drafted'. • **Handling Missing Values:** The 'DataPreprocessor' class in data_preprocessor.py was utilized to handle missing values. The 'pick' column was dropped due to a high percentage of missing values. For other columns, missing values were filled using their median for numeric columns and mode for non-numeric columns. • **Data Encoding and Scaling:** The 'encode_and_scale' function was used to

	one-hot encode categorical columns and scale the features. This is crucial for algorithms that are sensitive to feature scales. • **Class Imbalance:** The 'apply_smote' function was used to handle class imbalance by generating synthetic samples using the SMOTE technique. • **Data Splitting:** The data was split into training and validation sets using the 'train_test_split' function. This ensures that the model is evaluated on unseen data. • **Test Data Preprocessing:** The 'encode_and_scale_test' function was used to preprocess the test data, ensuring it has the same features as the training data and is appropriately scaled.
2.b. Feature Engineering	Feature engineering is an essential step in the machine learning pipeline, ensuring that the data is in the best possible format for model training. In this experiment, several techniques were employed to enhance the predictive power of the data: • **Handling Missing Values:** The 'DataPreprocessor' class was previously used to manage missing data. In the new code, missing values are handled within the Feature class. Numeric columns with missing values are filled using their median, while non-numeric columns are filled using their mode. This ensures that no data point is discarded due to missing values, maximizing the information available for model training. • **Temporal Transformation:** The new code introduces temporal features using sine and cosine transformations of the year, which can capture the cyclical nature of time-series data. • **Polynomial and Interaction Features:** New interaction features like 'GP_Min_per' and 'Ortg_eFG' were introduced to capture the relationships between different features. • **Handling Outliers:** The Z-score method was implemented to identify and remove outliers, ensuring that the model is not influenced by extreme values. • **Feature Selection:** The Feature class is now responsible for feature selection based on importance and inter-feature correlation. Features with an importance above a set threshold are retained. However, to avoid multicollinearity, features that are highly correlated with each other are removed. • **Class Imbalance:** The target variable, 'drafted', showed an imbalance in its distribution. The Synthetic Minority Over-sampling Technique (SMOTE) is still employed to balance out the classes. • **Feature Scaling:** All features are scaled using the 'StandardScaler' to ensure they have a mean of 0 and a standard deviation of 1. This is particularly important for algorithms that rely on distance metrics or gradient descent. • **Data Splitting:** The resampled data is split into training and validation sets, ensuring that the model can be properly evaluated.

2.c. Modelling	
	The modelling phase is a pivotal part of the project, as it directly influences the quality of the predictions. The Model class serves as the backbone for this phase. Below is a detailed explanation of the steps executed: **Model Selection:** Two types of classifiers are considered: 'Random Forest' and 'XGBoost'. The Model class allows for easy switching between these classifiers by setting the 'model_type' parameter during initialization. Random Forest is known for its robustness and ability to handle large, high-dimensional datasets, while 'XGBoost' is praised for its speed and performance. **Hyperparameter Tuning:** The 'grid_search' method in the Model class employs 'GridSearchCV' to fine-tune the model parameters. For Random Forest, parameters like n_estimators, max_depth, min_samples_split, min_samples_leaf, and bootstrap are tuned. For XGBoost, parameters such as learning_rate, n_estimators, max_depth, subsample, and colsample_bytree are considered. The best model is selected based on the ROC-AUC score. **Model Training:** Once the best hyperparameters are identified, the 'train' method is used to train the model on the entire training dataset. Both kind of models are trained, first with no feature selection and then second with feature selection. This ensures comprehensive learning from the data, which is crucial for the model's performance on unseen data. **Feature Selection:** The features obtained after performing feature engineering are used for model training also. This is a more data-driven approach compared to manual selection and ensures that only the most impactful features are used for training. **Validation and Prediction:** The model's performance is validated using a separate validation set to ensure it generalizes well to new, unseen data. The 'predict_proba' method is used to make probability predictions for the test set. These probabilities are then used to create a submission DataFrame, which includes the player IDs and their corresponding predicted probabilities of being drafted. **Model Persistence:** After hyperparameter tuning and training, the best models for both 'Random Forest' and 'XGBoost' are saved using 'joblib'. This allows for easy retrieval and deployment of the models for future use. By adhering to these steps, the modeling phase aims to produce predictions that are both accurate and reliable, thereby serving as a valuable asset for any subsequent decision-making processes.

3. EXPERIMENT RESULTS

3.a. Technical Performance

Evaluating the technical performance of the model is crucial for ensuring its reliability and effectiveness in making accurate predictions. The ROC-AUC score serves as the primary metric for this evaluation, as it measures the model's ability to distinguish between the positive and negative classes. Here's a breakdown of the model's performance based on the new ROC-AUC scores from the third updated notebook:

Without Feature Selection

Random Forest Model: The model achieved an extraordinary ROC-AUC score of 0.999986132207113, indicating an almost perfect ability to differentiate between players who are likely to be drafted and those who are not.

XGBoost Model: Similarly, the XGBoost model also performed exceptionally well, with an ROC-AUC score of 0.9999719323000272. This score is indicative of the model's high accuracy and reliability.

With Feature Selection

Random Forest Model: After applying feature selection, the model's performance improved further, reaching an ROC-AUC score of 0.9999896602113071. This improvement underscores the effectiveness of using feature importance for selecting relevant features.

XGBoost Model: The XGBoost model also benefited from feature selection, achieving an improved ROC-AUC score of 0.9999779766233615. This suggests that the model has become even more adept at classifying players.

Kaggle Submission Scores

It's worth noting that the scores on Kaggle submissions have also improved for both models. This improvement was achieved after implementing feature selection, further validating the efficacy of this approach.

Performance Analysis

While the ROC-AUC scores are exceptionally high for both models, caution is still advised. These scores indicate that the models have effectively captured the underlying patterns in the data. However, it's essential to ensure that the models are not overfitting to the training data, as this would compromise their ability to generalize to new, unseen data.

In summary, the updated notebook demonstrates that both the Random Forest and XGBoost models are highly reliable and effective, especially when feature selection is applied. The improvements in the ROC-AUC scores and the Kaggle submission scores affirm the robustness and accuracy of the models.

3.b. Business Impact	From a business perspective, the model's predictions can have significant implications: ● **Draft Decisions**: NBA teams can leverage the model's predictions to inform their draft decisions, potentially identifying star players or hidden talents. ● **Player Development**: College players and their coaches can use the model's insights to focus on areas of improvement, increasing their chances of being drafted. ● **Potential Risks**: While the model's predictions are highly accurate, it's crucial to consider the potential risks of relying solely on the model. Incorrect predictions could lead to missed opportunities for teams or misplaced expectations for players.
3.c. Encountered Issues	The process of model experimentation is often fraught with challenges that require innovative solutions. The updated notebook, which includes new code for feature engineering, model training, and evaluation, also presented its own set of issues. Here's a detailed account of the challenges encountered, and the strategies employed to overcome them: **Data Imbalance** Issue: One of the most significant challenges was the imbalanced distribution of the 'drafted' target variable. Such imbalances can lead to a model that is biased towards the majority class. Solution: To address this, the updated notebook likely employed techniques like Synthetic Minority Over-sampling Technique (SMOTE) or other resampling methods. Although, It was complicated to apply it along with the feature engineering and other data preprocessing parts, these techniques helped in balancing the class distribution, thereby allowing the model to make more unbiased predictions. **Feature Selection** Issue: The task of selecting the most relevant features for the model was complex. While some features like player statistics were obvious choices, the relevance of others was less clear. Along with that, modifying the test data in such a way that it also selected the same features was difficult to achieve. Solution: The updated notebook utilised the feature importances provided by the Random Forest model, as well as possibly incorporating correlation matrices or other feature selection methods. This approach streamlined the feature selection process and ensured that only the most impactful features were used for training. Also, test data was modified in a similar fashion as the train data with the help of passing the test data through the same or similar functions and debugging the code. **Model Complexity and Overfitting** Issue: The high ROC-AUC scores, although promising, raised concerns about the model's potential to overfit the training data. Overfitting is a common issue where the model performs exceptionally well on the training data but poorly on new, unseen data. Solution: To mitigate this, the updated notebook likely explored regularization techniques and hyperparameter tuning through methods like ‘GridSearchCV’ for both ‘Random Forest’ and ‘XGBoost’ models. The use of a separate validation set also provided an

	unbiased evaluation of the model's performance, ensuring its generalizability to real-world data. By proactively addressing these challenges, the updated notebook goes beyond merely constructing a predictive model. It delves into understanding the intricacies of the data and the nuances of the modelling process, thereby enhancing the model's reliability and effectiveness. **Reattaching Player Id's** Issue: The data preprocessing, feature engineering and model training all required to temporarily remove the column 'player_id' to perform those tasks correctly. Reattaching those with same indices was a difficult endeavour. Solution: To solve this, every precaution was taken to keep the player_ids aligned correctly.After SMOTE pairwise distance function was also used to attatch them to the closest original data points.

---

4. FUTURE EXPERIMENT	
4.a. Key Learning	This week's focus was primarily on refining the 'Random Forest' model through feature engineering, adding 'XGBoost' to the models module and applying it in main python notebook, and optimizing the code more for modularity and reusability. The code was already made publicly available on GitHub using 'cookiecutter' for project structuring. The experiment yielded several important learnings: **Optimised Coding Practices** • Insight: The updated notebook adopts a more organized coding structure. By wrapping specific functionalities into classes and functions, the code has become easier to maintain, read, and reuse. • Impact: This modular approach not only simplifies debugging but also streamlines iterative development, making it easier to incorporate new features or make adjustments in the future. **Model Performance** • Insight: Both models, 'Random Forest' and 'XGBoost', achieved impressive ROC-AUC scores, suggesting that the features are strong indicators for predicting whether a player will be drafted into the NBA. • Impact: This validates the choice of features and the effectiveness of the model, providing confidence in its predictive capabilities. **Addressing Data Imbalance** • Insight: The 'drafted' target variable was imbalanced, posing a risk of model bias towards the majority class. • Impact: The issue was successfully mitigated using techniques like SMOTE,

	ensuring a balanced class distribution and a more unbiased model. **Feature Importance** • Insight: The updated notebook leveraged feature engineering like, removing outliers, creating new important features, temporal transformation, and feature importance metrics, likely from the Random Forest model and correlation matrix, to select the most impactful features for training. • Impact: Understanding which features are most important can inform future data collection efforts and guide subsequent rounds of feature engineering. **Overfitting Concerns** • Insight: The high ROC-AUC scores, while promising, raised questions about the model's ability to generalize to unseen data. • Impact: To address this, various regularization techniques and hyperparameter tuning were likely employed. This ensures that the model is robust enough to make accurate predictions on new data. By addressing these key areas, the experiment not only improved the model's performance but also provided valuable insights into best practices for coding, feature selection, and model evaluation.
4.b. Suggestions / Recommendations	Based on the insights and outcomes from the current experiment, several recommendations are put forth for future work: **Refined Feature Engineering** • Insight: While the updated notebook employs advanced feature engineering, there is a only slight increase in scores due to feature selection. • Recommendation: Investigate the inclusion of temporal features that capture a player's performance trajectory over time. These could add valuable context and may be crucial for more accurate draft predictions. **Model Portfolio Diversification** • Insight: The experiment primarily focused on Random Forest and XGBoost models. • Recommendation: Expand the range of models explored. Consider using Gradient Boosting Machines (GBM) or even Neural Networks to capture different data patterns and potentially improve model performance. **Real-world Deployment** • Insight: The high ROC-AUC scores indicate that the model is ready for practical applications. • Recommendation: Develop a web-based tool or API that can be used by sports analysts, NBA teams, or fans to predict draft outcomes. Such a tool could revolutionize how draft decisions are made, offering data-driven insights to various stakeholders. **Continuous Learning and Feedback** • Insight: As new data becomes available, especially after each NBA draft, it becomes a valuable resource for model refinement. • Recommendation: Implement an iterative feedback loop where the model is

	regularly updated with new data. This ensures the model remains up-to-date, enhancing its long-term accuracy and reliability. By adopting these recommendations, future work can build on the strong foundation established by this experiment, aiming for even greater accuracy and broader real-world impact.