# **EXPERIMENT REPORT**

|  |  |
| --- | --- |
| **Student Name** | Shivani Nandkishor Nipane\_24622969 |
| **Project Name** | **Assessment1- Binary Classification Project** |
| **Date** | 28/06/2023 |
| **Deliverables** | <Nipane\_ShivaniNandkishor-24622969-week2\_lightgbm.ipynb>  <xgboost> |

|  |  |
| --- | --- |
| 1. **EXPERIMENT BACKGROUND** | |
| Provide information about the problem/project such as the scope, the overall objective, expectations. Lay down the goal of this experiment and what are the insights, answers you want to gain or level of performance you are expecting to reach. | |
| **1.a. Business Objective** | Explain clearly what is the goal of this project for the business. How will the results be used? What will be the impact of accurate or incorrect results?  The main goal of this project is to develop a predictive model that uses a rookie player's statistics to predict if they will last at least 5 years in the NBA league. This will be used by stakeholders such as team managers, talent scouts, analysts, or potentially even sports betting companies to inform their decisions and strategies.  Here's a detailed breakdown of how these results might be used and the impact they could have:  **Team Management and Talent Scouts:** These stakeholders are always looking for promising talent that will have longevity in their careers. If a rookie player is predicted to have a career of at least 5 years in the NBA, they may be considered a more stable and worthwhile investment for the team. This could influence decisions on who to draft, trade, or invest in for training and development.  **Sports Analysts and Commentators:** Predictions about player longevity could be used to inform analyses, articles, and broadcasts. This could generate interesting discussions and content for fans, and analysts who consistently make accurate predictions could improve their reputation.  **Sports Betting Companies or Bettors:** Accurate predictions about player longevity could be used to set betting odds or inform betting decisions.  As for the impact of accurate or incorrect results:  **Accurate Results:** Accurate predictions can provide valuable insights and give a competitive edge to the stakeholders. It can help in making informed decisions that could lead to better team performance, more accurate analyses, and better-informed betting.  **Incorrect Results:** Incorrect predictions can have significant consequences. A team may invest in a player who doesn't perform well or has a shorter career than expected, leading to wasted resources and potential missed opportunities. For sports analysts, incorrect predictions can damage their credibility. For bettors or sports betting companies, it could lead to financial losses.  The predictive model's performance should be continuously monitored and updated to incorporate the most recent data and ensure its reliability and accuracy. Regular maintenance and updating of the model is necessary to ensure it remains effective and accurate over time. |
| **1.b. Hypothesis** | Present the hypothesis you want to test, the question you want to answer or the insight you are seeking. Explain the reasons why you think it is worthwhile considering it,  The primary hypothesis for this project is: "A player's statistics during their rookie season can predict whether they will last at least 5 years in the NBA." Essentially, we want to answer the question: "Can we predict a player's career longevity based on their rookie season performance?"  This hypothesis is worthwhile for several reasons:  1. **Strategic Decisions:** If proven true, the hypothesis can significantly aid strategic decision-making for various stakeholders in the NBA, such as team management and talent scouts. Knowing whether a player is likely to have a long career could influence decisions about recruitment, development, and investment.  2. **Resource Allocation:** Teams could allocate their resources more efficiently by focusing on players likely to have longer careers.  3. **Fan Engagement:** Predicting player longevity can also help engage fans, as it gives them another dimension to consider when following their favorite teams and players. |
| **1.c. Experiment Objective** | Detail what will be the expected outcome of the experiment. If possible, estimate the goal you are expecting. List the possible scenarios resulting from this experiment.  The expected outcome of this experiment is a trained machine learning model, specifically using XGBoost, that can predict whether a rookie player will have a career lasting at least 5 years in the NBA, based on their performance statistics from their first season.  The measure of success for the model will be its performance on the AUROC (Area Under the Receiver Operating Characteristics) score. While it's hard to provide a specific estimate without knowing the data and problem complexity, a common target in binary classification problems is an AUROC above 0.7 for the test set, which could indicate a reasonably good model. However, the higher the AUROC, the better the model is at distinguishing between players who last at least 5 years and those who don't.  Here are the possible scenarios resulting from this experiment:  1. High AUROC Score (> 0.7) : This is the best-case scenario, where the model's predictions are accurate and reliable. This would allow stakeholders to use this model confidently to inform their decisions. If the score is particularly high, say over 0.85, the model is performing excellently.  2. Moderate AUROC Score (0.5 - 0.7) : In this scenario, the model has some predictive power but is not very reliable. The model may need further tuning, or additional, more informative features may be needed to improve the model's performance.  3. Low AUROC Score (< 0.5) : This is the worst-case scenario, where the model's predictions are not better than random guessing. In this case, a different approach may be needed, such as using a different algorithm, adding more features, or redefining the problem. |

|  |  |
| --- | --- |
| 1. **EXPERIMENT DETAILS** | |
| Elaborate on the approach taken for this experiment. List the different steps/techniques used and explain the rationale for choosing them. | |
| **2.a. Data Preparation** | Describe the steps taken for preparing the data (if any). Explain the rationale why you had to perform these steps. List also the steps you decided to not execute and the reasoning behind it. Highlight any step that may potentially be important for future experiments  The data preprocessing steps taken:  1. Splitting the dataset : The provided training set was split into a separate training set and a validation set. This was done to create a separate dataset (validation set) that the model has never seen during training. It's used to evaluate the model's performance and to check for overfitting. As the test set provided didn't contain target labels, it couldn't be used to evaluate the model's performance.  2. Handling Negative Values : Certain columns were found to contain negative values, which doesn't make sense in the context of the dataset (e.g. you can't have negative games played or a negative percentage for free throws). The values in these columns were replaced with their absolute values, under the assumption that the negatives were data entry errors.  3. Skewness : The skewness of all features was calculated. This is an important step as many machine learning algorithms assume that the data follows a Gaussian distribution. High skewness can lead to poor model performance.  4. Creating new features : New features were created that represented the average per game for each statistic (e.g., minutes per game, field goals made per game, etc.). This was done to capture the average performance of the player in each game, as the total values could be influenced by the number of games played.  5. Handling infinite values : Any infinite values resulting from the creation of the new features (possibly due to division by zero when games played was zero) were replaced with NaN.  6. Correlation Matrix : A correlation matrix was generated for the training set to understand the relationships between the different features. This is a crucial step to identify highly correlated features which might need handling to avoid multicollinearity.  Here's a potential preprocessing step that wasn't performed but might be important for future experiments:  - Handling missing values : Although it was stated that the dataset didn't contain missing values, it's always a good practice to check for missing values and handle them appropriately if any are found. Missing values can lead to erroneous results from machine learning models.  In addition, it's worthwhile mentioning that the code presents opportunities for improvement:  - Data Transformation : Some features were found to have high skewness. While the skewness was calculated, no transformation (like logarithm or square root transformation) was applied to reduce the skewness.  - Feature Scaling : No feature scaling (like standardization or normalization) was performed. This step is necessary for many machine learning algorithms that are sensitive to the scale of the features.  - Feature Selection : Although a correlation matrix was created, no explicit feature selection or dimensionality reduction techniques were applied. While it's fine for a first pass model, for more refined models, feature selection might be important to improve performance. |
| **2.b. Feature Engineering** | Describe the steps taken for generating features (if any). Explain the rationale why you had to perform these steps. List also the feature you decided to remove and the reasoning behind it. Highlight any feature that may potentially be important for future experiments  Given the presented code and information, here's the feature generation step taken:  Creating Per Game Features : Features were created to represent per game statistics for various attributes (e.g., minutes per game, field goals made per game, etc.). This step was undertaken to normalize player performance based on the number of games played. For example, a player who has played fewer games might have fewer total points simply because they have had fewer opportunities to score, not necessarily because they are a worse player. Therefore, creating per game features helps to better evaluate player performance on an average game basis.  However, there are a few potential feature generation opportunities that were not pursued in the code, but could be considered in future experiments:  1. Interaction Features : The creation of interaction features (i.e., features that are derived from two or more existing features) could potentially uncover new meaningful insights and improve model performance. For example, an interaction feature combining assists and points could capture how scoring ability interacts with teamwork.  2. Polynomial Features : Polynomial features could be created to capture more complex relationships between features. This is especially relevant for linear models, which cannot capture non-linear relationships between features.  3. Domain Specific Features : Considering domain specific knowledge to create new features could also be valuable. For example, in basketball, a popular statistic is the Player Efficiency Rating (PER), which is a measure of a player's per-minute productivity. It's a complex formula that takes into account various aspects of a player's performance.  Lastly, while a correlation matrix was generated, no explicit features were removed. In the future, this could be a useful step in feature selection, especially for features that are highly correlated with each other, to avoid the issue of multicollinearity. |
| **2.c. Modelling** | Describe the model(s) trained for this experiment and why you choose them. List the hyperparameter tuned and the values tested and also the rationale why you choose them. List also the models you decided to not train and the reasoning behind it. Highlight any model or hyperparameter that may potentially be important for future experiments  For this experiment, the LightGBM model was used. LightGBM is a gradient boosting framework that uses tree-based learning algorithms and is known for its fast training speed and high efficiency. It can handle the large size of data and takes lower memory to run, making it suitable for this project.  Here are the hyperparameters that were tuned:  1. n\_estimators : This is the number of boosting stages to perform. Too many can lead to overfitting. We tested 100, 200, 300, 400, and 500 to find an optimal trade-off between model complexity and performance.  2. max\_depth : This is the maximum depth of a tree. Deeper trees can model more complex relationships by adding more nodes, but as depth increases, the model becomes more prone to overfitting. We tested depths of 3, 4, 5, 6, and 7 to find an optimal depth.  3. learning\_rate : This is the step size at each iteration while moving toward a minimum of a loss function. Lower values are generally preferred as they make the model robust to the specific characteristics of tree and thus allowing it to generalize well. We tested learning rates of 0.01, 0.05, and 0.1.  After the hyperparameter tuning, we achieved an AUROC of 0.6961 which is a slight improvement over the initial AUROC of 0.6681.  Models not trained in this experiment:  We didn't train models like SVM, Random Forests, or Neural Networks in this experiment. The reasons could be due to the computational efficiency of gradient boosting models over these models, especially when dealing with large datasets. Also, gradient boosting models often outperform other models on a variety of datasets.  Potential future experiments:  Further tuning of hyperparameters, even introducing new ones like `min\_child\_samples` or `subsample`, could possibly lead to better model performance. Also, techniques like early stopping could be explored to prevent overfitting. Moreover, it would be worthwhile to try out different types of models, like CatBoost or even Neural Networks, to see if they can provide better results. |

|  |  |
| --- | --- |
| 1. **EXPERIMENT RESULTS** | |
| Analyse in detail the results achieved from this experiment from a technical and business perspective. Not only report performance metrics results but also any interpretation on model features, incorrect results, risks identified. | |
| **3.a. Technical Performance** | Score of the relevant performance metric(s). Provide analysis on the main underperforming cases/observations and potential root causes.  For the LightGBM model, we are using the Area Under the Receiver Operating Characteristic curve (AUROC) as our performance metric. After hyperparameter tuning, the AUROC score of the model improved from 0.6681 to 0.6961. While this is a reasonable improvement, the score is still less than the perfect AUROC score of 1, indicating that there is room for model performance to be enhanced.  Underperforming cases might be attributed to the following reasons:  1. Feature Representation : The current set of features used to train the model may not adequately capture the underlying patterns in the data to predict the target variable. Additional features could be engineered or feature selection could be conducted to identify the most relevant features.  2. Model Complexity : While hyperparameters were tuned, it's possible that the model is still too simple to capture complex relationships within the data or too complex leading to overfitting. Further tuning or regularization techniques could be applied.  3. Data Quality : If the dataset contains noise, outliers, or errors, these can negatively affect the model's performance. Further data cleaning and preprocessing steps could be beneficial.  4. Imbalanced Classes : The target classes might be imbalanced. If one class significantly outnumbers the other, the model might be biased towards the majority class. Techniques such as oversampling the minority class, undersampling the majority class, or using different performance metrics that are better suited to imbalanced datasets could be considered. |
| **3.b. Business Impact** | Interpret the results of the experiments related to the business objective set earlier. Estimate the impacts of the incorrect results for the business (some results may have more impact compared to others)  The business objective in this case appears to be the prediction of whether a player will play at least 5 years in the NBA, based on a set of data from their initial period in the league.  Our experimentation with the XGBoost and LightGBM models produced AUROC scores of approximately 0.6985 and 0.6961, respectively, after hyperparameter tuning. These scores suggest that the models have some predictive power, but they are not perfect. They provide some level of confidence in identifying players who will have at least a 5-year career in the NBA, but the scores also indicate that there will be a number of incorrect predictions.  Incorrect predictions can have two forms:  1. False positives : The model predicts a player will last at least 5 years in the NBA when they will not. The impact of such an error could be significant, particularly if contracts or investments are made based on these predictions. It could lead to unnecessary spending or resource allocation for a player who will not have a long tenure.  2. False negatives : The model predicts a player will not last at least 5 years in the NBA when they actually will. In this case, a team might overlook or undervalue a player who could turn out to be a valuable asset.  The potential impacts of these errors need to be balanced against the utility of the model. Even an imperfect model can provide valuable insights and help inform decisions. However, it's crucial to interpret the model's predictions with an understanding of its limitations, and not to rely solely on the model for decision making.  For further improvements, we might consider collecting more data, engineering additional features, trying different models or algorithms, or using ensemble methods to improve prediction accuracy. Additionally, in real-world scenarios, the model's predictions would ideally be combined with expert knowledge in the field for the most informed decision making. |
| **3.c. Encountered Issues** | List all the issues you faced during the experiments (solved and unsolved). Present solutions or workarounds for overcoming them. Highlight also the issues that may have to be dealt with in future experiments.  During the execution of the experiments, the following issues were encountered:  1. Data Preprocessing : We found that some features in the dataset had negative values where it didn't make sense, for example, games played (GP) or free throw percentage (FT%). For GP and FT%, we replaced the negative values with their absolute values assuming these were typos. However, for other features with negative values, we chose to keep them, reasoning that they might not be typos and could carry some meaningful information.  Future Considerations : Depending on the domain knowledge, one might want to handle these negative values differently. It would be important to understand what these negative values mean in the context of the features they are present in.  2. Feature Engineering : We introduced new features representing average values per game. This step was straightforward, however, it led to the creation of infinite values when the games played (GP) was zero. These infinite values were replaced with NaN.  Future Considerations : It would be worth exploring more feature engineering steps that might help improve the model's performance. Also, it might be interesting to see if removing those players with zero games played from the dataset could lead to a better performing model.  3. Model Selection and Hyperparameter Tuning : We used XGBoost and LightGBM for our experiments and performed hyperparameter tuning using GridSearchCV. While we were able to get some improvements with hyperparameter tuning, the process was computationally expensive and time-consuming.  Future Considerations : For more efficient hyperparameter tuning, methods like RandomizedSearchCV or Bayesian Optimization could be used. Additionally, other models and ensemble methods could be experimented with.  4. Class Imbalance : Our target variable was imbalanced with most players having a career of at least 5 years.  Future Considerations : Techniques to handle class imbalance such as SMOTE, ADASYN, class weighting, etc., could be used to potentially improve model performance.  5. Model Evaluation : We evaluated our model using the AUROC score. While this metric is a good measure for binary classification tasks, it does not provide a full picture of the model performance.  Future Considerations : In future experiments, using additional metrics like Precision, Recall, F1-score, and the confusion matrix could provide more comprehensive insight into the model's performance. |

|  |  |
| --- | --- |
| 1. **FUTURE EXPERIMENT** | |
| Reflect on the experiment and highlight the key information/insights you gained from it that are valuable for the overall project objectives from a technical and business perspective. | |
| **4.a. Key Learning** | Reflect on the outcome of the experiment and list the new insights you gained from it. Provide rationale for pursuing more experimentation with the current approach or call out if you think it is a dead end.  The outcome of the experiment has been insightful in a number of ways:  1. Model Choice : The experiment demonstrated the effectiveness of gradient boosting models like XGBoost and LightGBM in dealing with this type of predictive problem. They both provided fairly good results, which reaffirms the power of these models for tasks of binary classification.  2. Feature Engineering : The creation of new features, like the average per game metrics, showed the potential of feature engineering in improving the performance of our models. Although the improvement wasn't drastic, it was a step in the right direction.  3. Data Preprocessing : The presence of negative values in certain features, where logically they shouldn't exist, stressed the importance of thoroughly understanding and cleaning the dataset before model training.  4. Hyperparameter Tuning : The exercise of hyperparameter tuning taught us about the trade-off between model performance and computational resources. Significant improvements were observed after tuning, but it was also a resource-intensive task.  Given these insights, it does not seem that the current approach is a dead end. However, there are definitely opportunities for improvements and alternative strategies that could be explored:  1. More advanced Feature Engineering : This includes trying out more complex transformations, interactions between features, or even dimensionality reduction techniques like PCA.  2. Addressing Class Imbalance : Techniques to handle class imbalance such as SMOTE, ADASYN, or class weighting should be experimented with.  3. Ensemble Modeling : Combining the predictions of multiple models to make the final prediction could potentially yield better results.  4. Exploring Other Models : Other types of models, such as neural networks or SVMs, could also be experimented with. Different models have their own strengths and weaknesses and it may be worthwhile to try a different approach.  5. Better Evaluation Metrics : Finally, exploring additional evaluation metrics beyond AUROC could give a more holistic view of the model's performance.  Overall, I believe more experimentation with this approach is warranted given the potential areas for improvement listed above. |
| **4.b. Suggestions / Recommendations** | Given the results achieved and the overall objective of the project, list the potential next steps and experiments. For each of them assess the expected uplift or gains and rank them accordingly. If the experiment achieved the required outcome for the business, recommend the steps to deploy this solution into production.  Given the results of the experiment and the goal of predicting whether NBA rookies will last at least 5 years in the league, here are potential next steps and experiments:  1. Address Class Imbalance (High Priority) : Our target variable is imbalanced, which might lead to a biased model that mostly predicts the majority class. Using techniques like SMOTE, ADASYN or even class weighting during model training can help address this issue.  Expected uplift : Moderate to High. This could significantly improve our model's ability to correctly classify minority class instances, thereby improving overall model performance.  2. Advanced Feature Engineering (High Priority) : We can create more complex features, try feature interactions, or use dimensionality reduction techniques. We might also try to gain more insights from the existing data by creating more meaningful metrics.  Expected uplift : Moderate. The success of this step largely depends on our creativity and domain knowledge. Done right, it can uncover hidden patterns in the data and significantly boost performance.  3. Ensemble Modeling (Medium Priority) : This involves training multiple models and combining their predictions. For instance, we could try a weighted average of predictions from XGBoost and LightGBM, or even include other types of models.  Expected uplift : Moderate. Ensemble methods are known to enhance prediction accuracy, but they also add complexity and computational expense.  4. Exploring Other Models (Medium Priority) : We can try other machine learning models like Neural Networks, SVMs, or Random Forests.  Expected uplift : Low to Moderate. While other models might work better, they also might require more tuning and computational resources.  5. Hyperparameter Tuning (Low Priority) : While we have done some basic tuning, there's still room for more exhaustive hyperparameter tuning, potentially using methods like random search or Bayesian optimization.  Expected uplift : Low. Further tuning may yield some improvement, but likely it will be marginal, and it's also computationally expensive.  Once we have a model that meets the business requirements in terms of performance and prediction accuracy, we can proceed with deploying the model to a production environment. This would involve:  - Finalizing the model : This includes final training on the entire dataset, saving the model parameters or the entire model using libraries like joblib or pickle, and writing scripts for pre-processing new incoming data in the same way as the training data.  - Deployment : This could be done on-premise or on the cloud (like AWS, Azure, or GCP) depending on the company's infrastructure. The model could be served as a RESTful API for easy integration with other services.  - Monitoring : After deployment, the model's performance should be monitored over time to ensure that it's still providing value and to catch any "model drift".  - Maintenance : Depending on the monitoring results, the model might need to be retrained or fine-tuned periodically, using updated data. |