# **EXPERIMENT REPORT**

|  |  |
| --- | --- |
| **Student Name** | Shivani Nandkishor Nipane\_24622969 |
| **Project Name** | **Assessment1- Binary Classification Project** |
| **Date** | 28/06/2023 |
| **Deliverables** | <Nipane\_ShivaniNandkishor-24622969-week3\_logisticregression.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  The last model trained for this experiment was the Logistic Regression model.  Logistic Regression was chosen because it is a simple yet effective model for binary classification problems. Despite its simplicity, it often performs surprisingly well and has the added advantage of being very interpretable. Logistic Regression also requires less computational resources compared to more complex models like XGBoost or LightGBM, making it a good choice when dealing with larger datasets or when computational resources are limited.  The hyperparameters that were tuned for Logistic Regression were 'fit\_intercept', 'penalty', and 'solver':  - 'fit\_intercept' determines whether a bias or intercept term should be added to the decision function. The options tested were [True, False].  - 'penalty' is used to specify the norm used in the penalization, which can help to avoid overfitting. The options tested were ['l1', 'l2'].  - 'solver' is the algorithm to use in the optimization problem. The options tested were ['liblinear', 'saga', 'lbfgs'].  The rationale for choosing these hyperparameters to tune was based on their potential to significantly affect the performance of the model.  In terms of models that were not trained, decision tree-based models (like Random Forests) and Support Vector Machines were not chosen due to their tendency to overfit and their computational expense, respectively. These models could potentially be tested in future experiments, especially if we obtain more data or have more computational resources.  Regarding future experiments, the 'C' hyperparameter in Logistic Regression, which controls the inverse of the regularization strength, could potentially be important. Regularization can help to prevent overfitting by adding a penalty to the loss function, and adjusting 'C' could provide a better balance between bias and variance.  It's interesting to note that the AUROC score decreased slightly after hyperparameter tuning (from 0.7022 to 0.7004), which indicates that the default parameters were close to optimal. However, it's also possible that the search space didn't include the optimal set of parameters or that the CV fold number or scoring metric wasn't optimal for this dataset. |

|  |  |
| --- | --- |
| 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** | The main performance metric used for this experiment was the Area Under the Receiver Operating Characteristic curve (AUROC). The AUROC is a good metric for binary classification problems, as it measures how well a model is able to distinguish between classes. An AUROC of 0.5 signifies a model with random predictions, while an AUROC of 1.0 signifies a perfect model.  In the case of our Logistic Regression model, the AUROC score before hyperparameter tuning was 0.7022, and it slightly decreased to 0.7004 after tuning. This implies that our model was able to distinguish between players who lasted 5 years in the league and those who did not with some reliability, but there is still room for improvement.  When it comes to underperforming cases, it's difficult to pinpoint exact observations without a deep dive into the model's predictions and the specific characteristics of those misclassified cases. However, some potential root causes could be:  1. Noise and Outliers: If the data contains noisy observations or outliers, these could negatively affect the model's performance. A thorough exploratory data analysis and preprocessing to handle these issues could potentially improve the model's performance.  2. Feature Importance and Selection: Not all features contribute equally to the model's predictions. Some features might be irrelevant or even misleading, reducing the performance. Feature importance analysis and feature selection techniques could potentially improve the model's performance by focusing on the most informative features.  3. Imbalanced Classes: Our dataset has imbalanced classes, with a majority of players lasting 5 years in the league. This imbalance can bias the model's predictions towards the majority class. Techniques like oversampling the minority class, undersampling the majority class, or using a cost-sensitive learning algorithm could potentially improve the model's performance. |
| **3.b. Business Impact** | Our business objective was to predict whether a basketball player would last 5 years in the league based on their rookie year performance.  The highest AUROC score achieved was 0.7022 using the logistic regression model before hyperparameter tuning. This indicates that the model was somewhat successful at predicting the longevity of the players' careers, though there was still significant room for improvement. In essence, this means that our model is performing better than random guessing, but it is not perfect and will make errors in its predictions.  Incorrect predictions can have different impacts on the business depending on the specific context:  1. False Positives: These are cases where the model predicted that a player would last 5 years in the league, but they did not. If a team were to make investment decisions based on this model's predictions, such as signing a player to a long-term contract or investing in their development, false positives could lead to financial loss and missed opportunities to invest in other players.  2. False Negatives: These are cases where the model predicted that a player would not last 5 years in the league, but they did. In this case, a team could miss out on a potential talent, impacting the team's performance in the long run.  Considering the context of professional basketball, where player contracts can involve substantial amounts of money, false positives could be more costly, as they would result in a direct financial loss. However, both types of errors could lead to missed opportunities, either in the form of overlooked talent (false negatives) or misguided investments (false positives).  Given the current model's performance, it's recommended to use its predictions as one of many factors in the decision-making process rather than the sole determinant. Further improvements to the model are necessary to reduce the rate of false predictions and increase confidence in its predictions. |
| **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. Feature Engineering: Some of the features in the dataset had values that did not make logical sense, such as negative values for 'GP', 'FT%', etc. To overcome this, I used the absolute values of these features. However, in a real-world scenario, these discrepancies might warrant a conversation with the data collection team or a deeper investigation into the data collection process. Additionally, creating features like 'average per game' helped improve the model performance to some extent. However, creating effective features can be an iterative process and may require domain knowledge and understanding of the data.  2. Class Imbalance: The target variable was imbalanced with the majority of players having a career span of 5 or more years. This can lead to models that are biased towards the majority class. In this case, I did not apply any techniques to handle the class imbalance. However, future experiments could explore techniques like oversampling the minority class, undersampling the majority class, or using a combination of both (SMOTE).  3. Model Selection: Several models were trained as part of the experiment, and each had their own strengths and weaknesses. For instance, the logistic regression model provided a good baseline model, but its simplicity might limit its predictive power. On the other hand, XGBoost and LightGBM, while more powerful, may tend to overfit and require careful tuning. These experiments serve as a reminder that there's often a trade-off between model complexity and interpretability, and the choice of model should align with the business objectives and constraints.  4. Hyperparameter Tuning: Grid search was used for hyperparameter tuning, which can be quite time-consuming, especially with more complex models and larger parameter spaces. In future, random search or Bayesian optimization could be used to optimize the hyperparameters in a more efficient manner.  5. Model Evaluation: While AUROC was chosen as the primary metric due to the class imbalance, it does not provide a complete picture of the model's performance. For instance, it does not give a sense of the number of false positives and false negatives, which could have different business impacts. Future experiments could benefit from using additional metrics like precision, recall, F1 score, or even custom loss functions based on business impact.  6. Scalability: The models were trained on a relatively small dataset, and it is unclear how they would perform or how much computational resources they would need if trained on a larger dataset. Future experiments would need to take scalability into consideration, especially for production deployment. |

|  |  |
| --- | --- |
| 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.  Reflecting on the outcome of the experiment, here are a few insights that were gained:  1. Model Complexity vs Interpretability: The experiment highlighted the trade-off between model complexity and interpretability. While complex models like XGBoost and LightGBM provided better predictive performance, simpler models like Logistic Regression were more interpretable and less prone to overfitting. This is an important consideration for business scenarios where model interpretability is important for decision-making and regulatory compliance.  2. Feature Importance: The experiment also highlighted the importance of feature engineering in improving model performance. While the original dataset provided a good starting point, deriving new features from the existing ones and excluding irrelevant features played a crucial role in improving model performance.  3. Hyperparameter Tuning: This experiment further reinforced the importance of hyperparameter tuning in improving model performance. The GridSearchCV function was particularly useful in finding optimal hyperparameters for each of the models. However, this process was time-consuming and computationally intensive, indicating a need for more efficient hyperparameter optimization methods in future experiments.  4. Performance Evaluation: The choice of performance metric was critical in this experiment. AUROC was chosen due to the imbalanced nature of the dataset. However, it was realized that relying solely on AUROC could be misleading, as it does not consider the costs of different types of errors (false positives vs false negatives), which could vary depending on the business context.  Given these insights, while there have been improvements in model performance with each iteration, it appears that the models have hit a performance plateau with the current approach. This could be due to several reasons such as limitations in the data, choice of models, or the feature engineering approach. Therefore, pursuing further experimentation with the current approach might not yield significant improvements.  However, it does not mean this is a complete dead end. There are several alternative avenues to explore such as:  - Collecting more data or different types of data that could better explain the target variable.  - Trying out different feature engineering techniques or leveraging domain knowledge to create more informative features.  - Exploring different modelling approaches, such as deep learning models or ensemble methods.  - Using techniques to address class imbalance, like oversampling the minority class, undersampling the majority class, or using a combination of both (SMOTE).  These could potentially lead to improvements in model performance, making them worthwhile avenues for future experiments. |
| **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.  Based on the outcomes of the experiments, here are potential next steps and experiments:  1. Collecting More or Different Types of Data (Expected uplift: High)  More data, especially for the under-represented class, could improve the performance of the models by reducing overfitting. Different types of data, such as transactional or behavioral data, could capture aspects not covered in the current dataset.  2. Trying Different Feature Engineering Techniques (Expected uplift: Medium)  Further feature engineering based on domain knowledge or using advanced techniques like polynomial features or interaction features can potentially create more informative features.  3. Exploring Different Modelling Approaches (Expected uplift: Medium-High)  Trying out deep learning models, ensemble methods, or less conventional algorithms might yield a better performance.  4. Addressing Class Imbalance (Expected uplift: Medium)  Techniques such as oversampling the minority class, undersampling the majority class, or using a combination of both (like SMOTE) can improve the performance of models on imbalanced datasets.  Here's the ranking based on the expected uplift:  1. Collecting More or Different Types of Data  2. Exploring Different Modelling Approaches  3. Addressing Class Imbalance  4. Trying Different Feature Engineering Techniques  While all these steps have the potential to improve the model's performance, they also require varying levels of resources and come with their own challenges. Therefore, the specific context of the business should be considered when deciding which steps to pursue.  In terms of deployment, if the current model's performance meets the business requirements, the next steps would involve:  - Validating the Model: Before deployment, validate the model with a new set of data to ensure its performance is stable and reliable.  - Building a Model Pipeline: Develop a pipeline for data preprocessing, model training, and prediction to automate the process.  - Developing an API for Model Access: If the model needs to be accessed by different systems, an API would be necessary to facilitate this.  - Setting up a Monitoring System: Once deployed, the model's performance should be continuously monitored to detect any degradation over time, which might necessitate model retraining.  - Planning for Model Update/Retraining: Establish a process to retrain the model with new data periodically or when its performance drops below a certain threshold.  However, given the performance of the current models, I recommend considering the potential next steps and experiments to improve the models further before moving to deployment. |