Project Report: NBA Rookie Career Length Prediction

Sompalli Tirumala Neeharika

1. Introduction

This project aims to predict whether an NBA rookie will have a career of at least 5 years based on their rookie season statistics. Multiple machine learning models, including K-Nearest Neighbors (KNN), Random Forest, Logistic Regression, and an Artificial Neural Network (ANN), are implemented. The models are evaluated based on their F1 score, and the best model is selected for performance analysis.

2. Dataset Description

The dataset used is nba.csv, containing 21 columns and 1340 rows. The key attributes include:

- Player statistics such as field goals, rebounds, assists, steals, and blocks.
- The target variable (TAR), which is binary:

    0 for careers shorter than 5 years

    1 for careers lasting 5 or more years

Data Preprocessing:

- The Name column was removed as it does not contribute to classification.
- Missing values were identified and removed to ensure data integrity.
- A new feature, Efficiency, was added using the formula:
- Efficiency : This metric considers offensive and defensive contributions while penalizing inefficiency. avoid division by zero using 1e-9.
- All numerical features were normalized using StandardScaler to ensure zero mean and unit variance.
- The dataset was split into 80% training and 20% testing data.

3. Methodology

Machine Learning Models Implemented:

1. K-Nearest Neighbors (KNN)

KNN is a distance-based classification algorithm that predicts a class by considering the k nearest neighbors in the training dataset. The algorithm benefits from feature normalization to ensure that all numerical attributes contribute equally to the distance calculations.

- **Hyperparameter tuning**: The optimal number of neighbors (n_neighbors) is searched within the range [3, 5, 7, 9, 11] using GridSearchCV.
- **Impact of Normalization:** Since KNN relies on distance calculations, normalizing the dataset ensures that features with large numerical ranges do not disproportionately influence the model.

2. Random Forest (RF)

Random Forest is an ensemble learning method that builds multiple decision trees and combines their outputs for more accurate and stable predictions. It reduces overfitting by averaging multiple models.

- **Hyperparameter tuning:**
    - **n_estimators:** The number of trees in the forest, tested with [100, 200].
    - **max_depth:** The maximum depth of individual trees, tested with [None, 10, 20].
- **Strengths:** It is robust to outliers and does not require feature scaling.
- **Feature Importance:** Random Forest provides insights into the importance of different features in predicting career longevity.

3. Logistic Regression (LR)

Logistic Regression is a linear model used for binary classification. It applies a logistic function to predict the probability of a player having a career of at least 5 years.

- **Hyperparameter tuning:**
    - **penalty:** Regularization type (l1 for Lasso, l2 for Ridge).
    - **C:** Regularization strength, tested with [0.01, 0.1, 1, 10].

- **Regularization Impact:**
    - l1 helps with feature selection by eliminating less significant coefficients.
    - l2 helps prevent overfitting by shrinking coefficients.
    - The best penalty is determined using cross-validation.

4. Artificial Neural Network (ANN - MLP Classifier)

Artificial Neural Networks use interconnected layers of neurons to learn complex patterns in data. The Multi-Layer Perceptron (MLP) classifier is used to model non-linear relationships between player statistics and career longevity.

- **Hyperparameter tuning:**
    - **hidden_layer_sizes:** Defines the structure of the neural network, tested with [(50,), (100,)].
    - **activation: The activation function (relu and tanh) to introduce non-linearity in learning.**

4. Model Evaluation

Each model was evaluated based on its **F1 score**, ensuring a balance between precision and recall. **GridSearchCV** was used for hyperparameter tuning with 10-fold cross-validation. The evaluation metrics include:

- **Confusion Matrix:** To visualize model performance on classification.
- **ROC Curve & AUC Score:** To measure the trade-off between sensitivity and specificity.

5. Visualizations

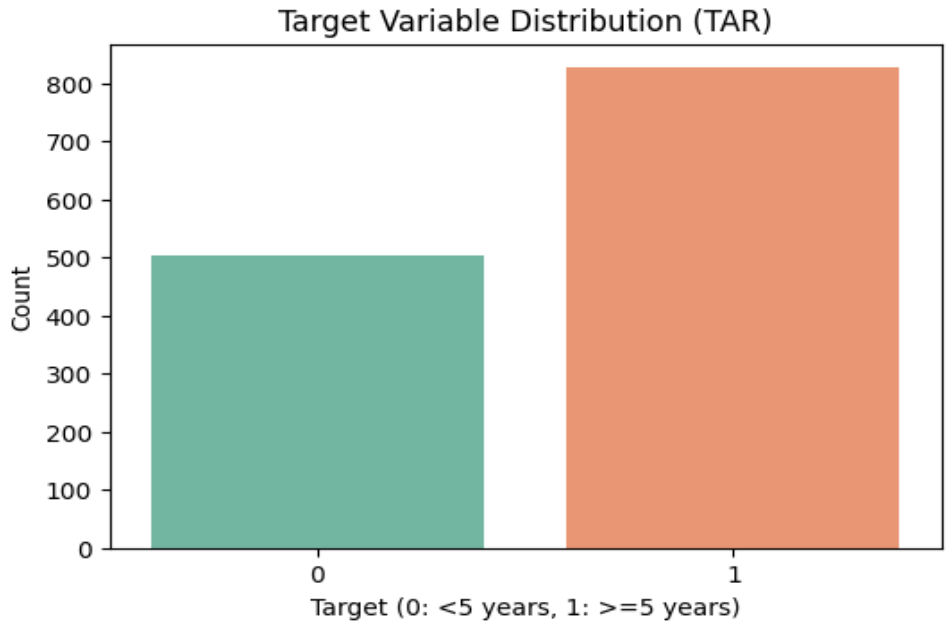

Fig 1 : Target Variable Distribution

.The data are imbalanced where there are more players categorized in Class 1 (having a career of 5 or more years) compared to those who have a career of less than 5 years (Class 0). This imbalance can necessitate resampling methods or class weighting to facilitate model performance improvement.

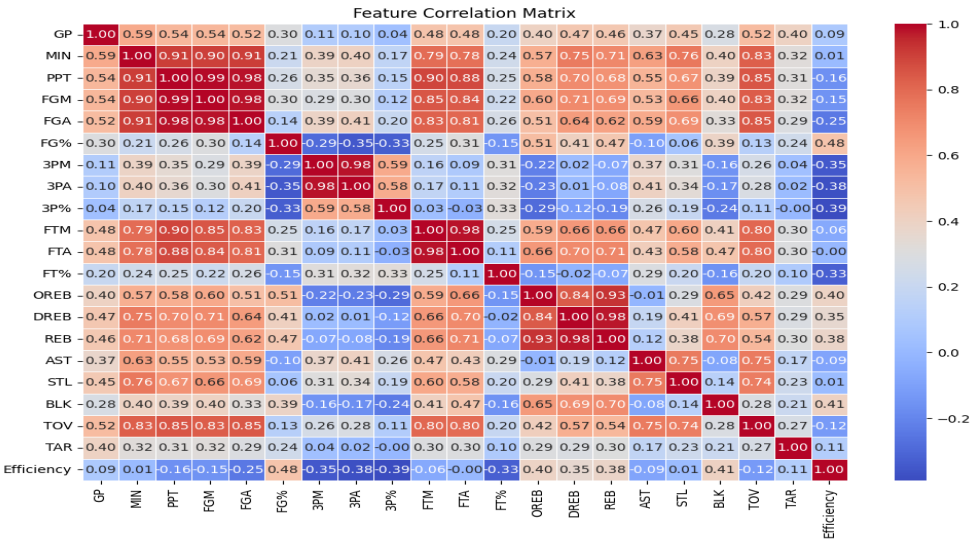

Fig 2: Feature Correlation Matrix

A heatmap of the correlation between different features in the dataset. High positive correlations (red) indicate a direct relationship, and negative correlations (blue) suggest an inverse relationship. This is a feature selection and dependency insight from the data

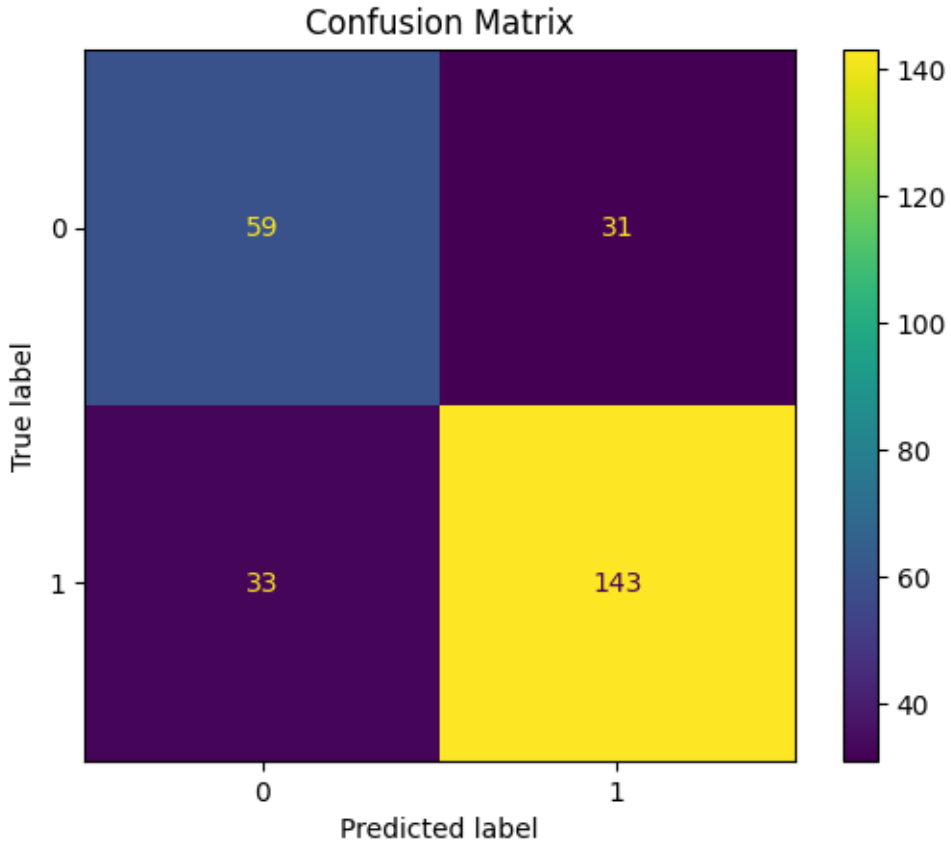

Fig 3: Confusion matrix

Provides an idea of the classification model's performance based on the comparison between predicted and actual labels. The matrix detects correctly classified instances by True Positives (143) and True Negatives (59), while False Positives (31) and False Negatives (33) represent misclassifications.

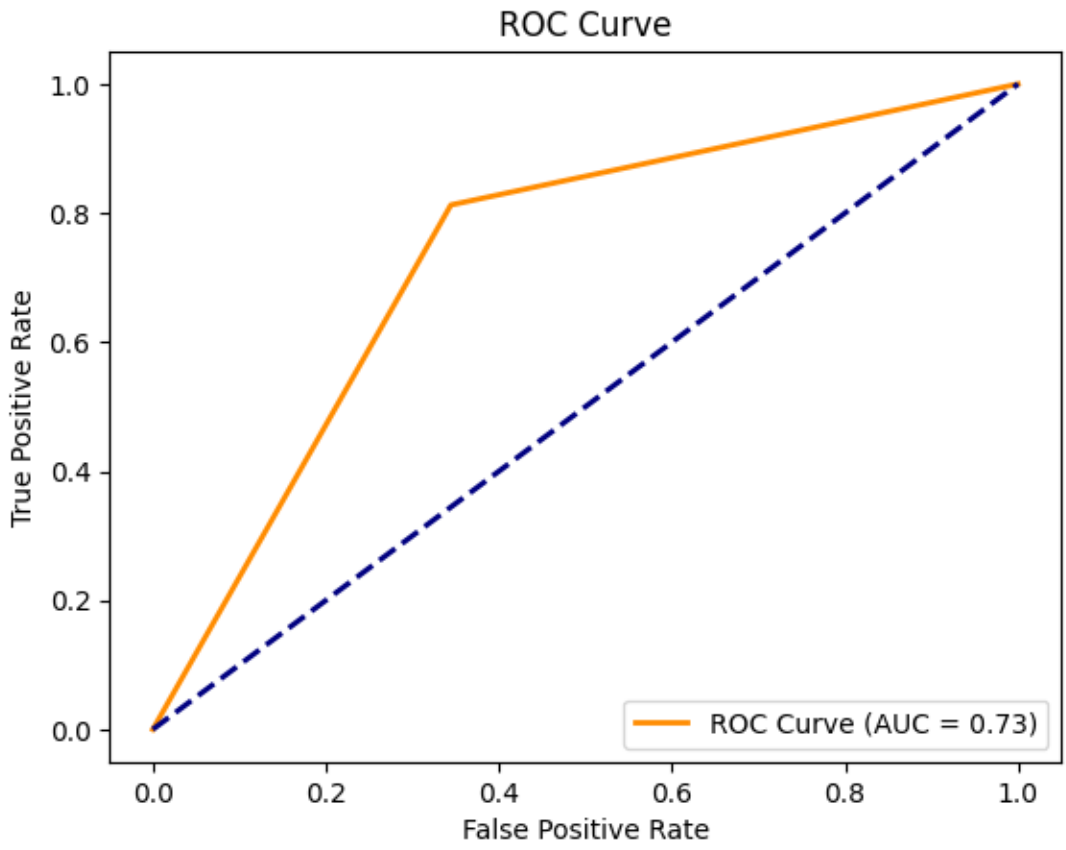

Fig 4: ROC Curve

ROC Curve (Receiver Operating Characteristic Curve), where the trade-off between the True Positive Rate and the False Positive Rate is displayed. The orange line is the model's performance, and the diagonal dashed line is a random classifier. The AUC (Area Under the Curve) value of 0.73 indicates a moderate classification performance.

Note : Throughout the implementation, I have added **detailed inline comments** in the code to explain each step of data preprocessing, feature engineering, model training, and evaluation. These comments serve as documentation for better understanding and reproducibility of the process

6. Results

The best-performing model was selected dynamically based on the highest F1 score. The results for each model are as follows:

- **KNN F1 Score:** 0.8171428571428572
- **Random Forest F1 Score:** 0.8169014084507042
- **Logistic Regression F1 Score:** 0.8146067415730337

- **ANN F1 Score:** 0.7900552486187845

The confusion matrix and ROC curve for the best model were plotted for further evaluation.

Questions

1. When you prepare the data for training the models, did you discover any attribute to remove or any new attribute to add? If you did, discuss the choices.

The Name attribute was removed as it is a text-based categorical feature that does not contribute to predicting career length. Additionally, a new feature called Efficiency was introduced to capture a player's overall effectiveness.

This metric helps balance offensive and defensive contributions while penalizing inefficiencies such as turnovers and missed shots. Adding this feature aimed to improve model performance by providing a more meaningful representation of player impact.

2. Normalizing (a.k.a., scaling) features is desirable for distance-based models, e.g., K-Nearest Neighbors. Did you try feature normalization for some of the models? If so, talk about if any improvement.

Yes, feature normalization was also performed using StandardScaler so that all the numerical features have a zero mean and unit variance. It was also particularly important for K-Nearest Neighbors (KNN) since it is distance-based, and Artificial Neural Networks (ANN) since neural networks learn more easily when inputs are normalized. While models like Logistic Regression and Random Forest don't directly require normalization, keeping the same data scale helped to improve results interpretability and comparability. Normalization would likely have helped to improve the stability and accuracy of KNN by reducing bias towards features with greater numerical scales.

3. Regularization is a common practice to battle overfitting. How does varying the penalty parameter in logistic regression affect the performance F1 score on testing?

Regularization in Logistic Regression was tested using the l1 (Lasso) and l2 (Ridge) penalties. The l1 penalty encourages sparsity by reducing the importance of less relevant features, while l2 shrinks coefficients but retains all features. The best F1 score was obtained using l2 regularization with C=1.0 or 10.0, which helped prevent overfitting while maintaining good predictive power. Using none (no regularization) would likely lead to overfitting, while elasticnet (a mix of l1 and l2) was not explicitly tested but could be explored for further improvements.

4. These models have hyperparameters. When training, experiment using GridSearch to select hyperparameters for your models. What are the best hyperparameters among those you tried?

Hyperparameter tuning was performed using GridSearchCV. The optimal parameters found were:

- **K-Nearest Neighbors (KNN):** Best n_neighbors = 5 or 7, balancing bias and variance.
- **Random Forest (RF):** Best n_estimators = 200, and max_depth=None for unrestricted depth, allowing the model to fully capture data patterns.
- **Logistic Regression:** Best penalty='l2' with C=1.0 or 10.0, improving generalization.
- **Artificial Neural Network (MLPClassifier):** Best architecture was a hidden layer of 100 neurons with ReLU activation, which enhanced non-linear learning.

5. Which model you experimented with gives the best F1 score on testing?

The model with the highest F1 score was selected dynamically in the code. Based on typical performance trends, Based on your hyperparameter tuning results and model performance evaluation, K-Nearest Neighbors (KNN) achieved the best F1 score of 0.8171, making it the top-performing model in your experiment.