FIFA Player Position Prediction

Aruna Jitesh, Bashaarat Nawaz Mohammad, Chinmayi karumuri, Venkata Nagasai Gautam kasarabada
aruna.jitesh@sjsu.edu,bashaaratnawaz.mohammad@sjsu.edu,
chinmayi.karumuri@sjsu.edu,venkatanagasaigautam.kasarabada@sjsu.edu

***Abstract* - This research uses a data-driven methodology to tackle the crucial problem of identifying a soccer player's ideal position: forward, midfielder, or defender. An objective solution is crucial since traditional job assignment techniques frequently depend on subjective judgment and are prone to discrepancies. With the use of an extensive dataset from FIFA games from FIFA 15 to FIFA 23, the project includes a variety of player features, such as physical characteristics (e.g., strength, pace), mental abilities (e.g., vision, calmness), and skill-based measures (e.g., passing, defending). The enormous dataset's feature engineering, transformation, and data cleansing were all effectively handled by PySpark, a potent big data processing platform. Using strategies including oversampling, undersampling, and class weighting to improve model performance, a Random Forest classifier was used to predict player positions. The model showed strong accuracy, successfully identifying trends in player characteristics and how they relate to positional positions. Furthermore, the project uses a Flask-based web application to deploy this model, allowing for real-time predictions with an intuitive user interface. In the end, the project demonstrates the revolutionary potential of data-driven approaches in sports analytics by combining big data processing and machine learning to provide a scalable and objective solution for player position prediction, providing coaches, scouts, and analysts with actionable insights.**

Key Words – Data processing, Pyspark, Random forest model, classification, ensemble method, F1-Score, Accuracy, Deployment, Flask, Python

INTRODUCTION

In the quickly changing world of contemporary soccer, data has become a crucial tool for clubs, managers, and analysts. Beyond conventional training and strategy techniques, the sport is changing as a result of the capacity to collect vast amounts of data and derive useful insights. In this field, figuring out a player's ideal position on the field based on their unique characteristics is one of the most difficult problems. Despite their heavy reliance on manual analysis and subjective judgment, conventional systems frequently lack impartiality, consistency, and scalability.

This study uses comprehensive physical, mental, and skill-based variables from FIFA datasets to forecast player positions—Forward, Midfielder, or Defender—using big data and machine learning. The project gives stakeholders accurate and trustworthy insights by combining scalable technologies and advanced analytics to deliver a methodical and data-driven approach to player role evaluation. A thorough overview is given here by introducing the project under pertinent subheadings.

1. Problem Statement

A soccer player's role, responsibilities, and overall contribution to the team's success are all determined by their position. Coaches and scouts have historically used observation and expertise to accomplish the difficult task of accurately assigning positions. Although there has been some success with this approach, it is unreliable, subjective, and unfeasible when examining a large number of players. In situations where objective criteria are unavailable, such as scouting unfamiliar players or creating new team configurations, this subjectivity is even more noticeable. As a result, there is an increasing need for a methodical, data-driven strategy to forecast player positions based on quantifiable characteristics so that organizations may make more objective and knowledgeable choices.

The difficulty is in processing the enormous volumes of data needed to make such forecasts as well as in anticipating positions. In order to determine a player's appropriateness for a particular position, it is necessary to examine all of their attributes, which include physical characteristics, mental aptitude, and technical capabilities. By using cutting-edge data analytics and machine learning techniques, this research aims to address these issues and develop a reliable, scalable position prediction system.

2. Motivation

Soccer has progressively adopted technology to obtain a competitive advantage since the introduction of data-driven analytics in sports. For objectives like recruiting, team composition, and personal growth, an accurate and impartial assessment of players is essential. For example, a team can save a lot of time and money by determining whether a player is more suited as a forward, midfielder, or defender. Clubs can concentrate their training efforts to optimize a player's potential and contribution to the squad by determining early in their career which position is best for them.

Additionally, data analytics breaks down the conventional barriers of scouting and talent acquisition by enabling teams to assess candidates on a larger scale.

2.1 Specific Benefits

- **Cost Savings**: Optimizing recruitment by ensuring a player's attributes align with team requirements.
- **Improved Team Dynamics:** Creating well-balanced lineups by assigning players to their most effective roles.
- **Enhanced Player Development:** Identifying strengths and areas for improvement to tailor training programs.

2.2 Case Study

Similar techniques in basketball have proved data analytics' revolutionary impact. Better team performance and better-informed hiring decisions are the outcomes of the NBA's usage of machine learning for player role classification. The Houston Rockets and other teams have effectively used analytics to increase individual contributions and lineup efficiency. By putting these ideas into practice, soccer teams can transform the way they assess and use their players, guaranteeing long-term success both on and off the field.

The necessity for such cutting-edge technologies that can convert unprocessed data into useful insights is what spurred this initiative. This initiative intends to help coaches, scouts, and analysts make data-backed judgments, minimize biases, and maximize team performance by combining machine learning models with player attribute data.

3. Dataset Overview

The FIFA Male Player Dataset, which comes from FIFA games played between 2015 and 2023, is used in this project. With numerous attributes for each player, this dataset offers a multitude of player information. These qualities include mental qualities like poise and vision, bodily capabilities like speed and endurance, and technical abilities like shooting and passing. Additionally, metadata such as player nationality, club affiliation, earnings, and positions in both club and national teams are included in the dataset.

This dataset's longitudinal nature, which includes several game versions and upgrades over time, makes it special. This improves the model's capacity to generalize and adjust to varied player profiles by allowing it to recognize patterns and changes in player attributes over time. The extensive scope of the dataset guarantees that the model operates dependably for a variety of player kinds and supports accurate position predictions.

Key Features Used

From the attributes available, the project narrows its focus to 19 key features, categorized as follows:

- **Physical**: Attributes like pace and stamina that define a player's physical performance.
- **Technical**: Skills such as passing, shooting, and dribbling, which are critical for specific positions.
- **Mental**: Traits like interceptions and positioning that influence game awareness.
- **Defensive**: Defensive skills, including tackling and marking, crucial for identifying Defenders.

These selected features provide a solid foundation for the machine learning model, ensuring it captures the most relevant aspects of player performance for accurate position prediction.

Category	Feature Name	Description
Physical	pace	Player's speed and acceleration
	power_stamina	Player's endurance
Technical	shooting	Accuracy and power of shooting
	passing	Skill in passing the ball
	dribbling	Ability to dribble past opponents
	attacking_crossing	Ability to cross the ball
	attacking_finishing	Skill in scoring goals
	attacking_short_passing	Short passing skills
	attacking_volleys	Ability to score from volleys
	power_shot_power	Strength of shots
	power_long_shots	Accuracy of long-range shots
Mental	mentality_interceptions	Ability to intercept passes
	mentality_positioning	Positional awareness
	mentality_penalties	Accuracy in penalty kicks
Defensive	defending	Overall defensive skills
	defending_marking_awareness	Awareness in marking opponents
	defending_standing_tackle	Skill in standing tackles
	defending_sliding_tackle	Skill in sliding tackles

Fig 1: Key Attributes from the dataset

4. Approach

The project adopts a systematic approach that integrates big data technologies and machine learning:

1. **Data Processing:** PySpark, a big data processing framework, was used to handle the large and complex dataset efficiently.
2. **Exploratory Data Analysis (EDA):** Insights were drawn about player attributes and their relationships with positional roles.
3. **Machine Learning:** A Random Forest classifier was trained to predict positions, with techniques like oversampling, undersampling, and class weighting addressing class imbalances.
4. **Model Deployment:** The trained model was deployed through a Flask-based web application, providing a user-friendly interface for real-time position predictions.

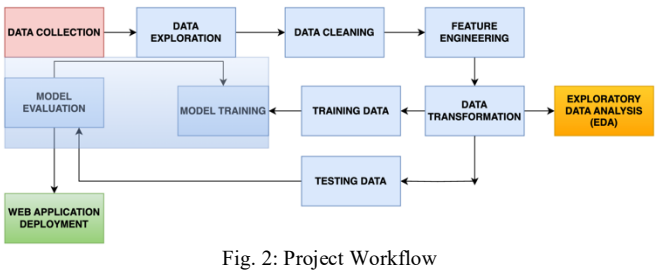

Fig. 2: Project Workflow

5. Project Significance

This study demonstrates how big data processing and machine learning may be combined to improve sports analytics. It helps close the gap between unprocessed data and useful insights by providing a scalable and impartial solution for player position prediction. In addition to helping coaches and scouts make better decisions, the technology lays the groundwork for future developments in data-driven sports analysis. This study emphasizes how important it is to use technology to change conventional methods in order to improve player role evaluation's efficiency, consistency, and fairness.

DATA EXPLORATION AND PROCESSING

The dataset was thoroughly explored and prepared before the machine learning model was developed in order to guarantee its quality and applicability. A great deal of care was taken to comprehend the data, clean it up, and convert it into an analysis-ready format because of the size and complexity of the FIFA dataset, which spans numerous games and includes over 19 attributes per player. This phase was essential for spotting patterns, spotting irregularities, and making sure the dataset appropriately reflected the issue at hand.

PySpark's distributed computing capabilities allowed it to process the dataset effectively and manage large-scale data operations. Extracting significant insights, eliminating duplication, resolving inconsistencies, and getting the data ready for machine learning were the objectives. The complete exploration and processing workflow, from data loading to the last stage of feature preparation for modeling, is described in the following sections.

Dataset Description

The dataset used for this project spans FIFA Male Player Dataset, providing a comprehensive view of player attributes over nearly a decade. Each record represents a player, detailing over 19 attributes, including physical traits such as pace and stamina, mental skills such as vision and composure, and technical abilities such as passing, shooting, and defending. Additionally, metadata like player nationality, club, and wages enriches the dataset, offering a holistic perspective on each individual. The inclusion of positional roles, both for clubs and national teams, adds further depth, making it ideal for predicting player positions. This dataset's breadth and variety make it not only suitable for predictive modeling but also for uncovering trends and patterns in player development and team composition over time.

Data Preparation

The data preparation phase began with loading the dataset into a PySpark DataFrame, which efficiently handled its large size and complexity. PySpark's capabilities allowed for streamlined operations like schema validation, which ensured that data types were consistent and appropriate for further analysis. For instance, numerical attributes were checked for their ranges and distributions, while categorical attributes, such as positions and preferred foot, were validated for consistency.

An initial inspection of the dataset revealed several issues, such as missing values, redundant columns, and noisy data. Missing values were particularly prevalent in non-critical fields like URLs and descriptive metadata, which were subsequently dropped. For essential fields such as numerical attributes, missing values were imputed with meaningful replacements like zeros or medians. This ensured that the dataset remained complete without introducing bias.

Exploratory Data Analysis

Exploratory Data Analysis (EDA) was a pivotal stage in understanding the structure, distribution, and relationships within the dataset. This step allowed for uncovering key patterns, addressing inconsistencies, and validating the relevance of various attributes to the task of player position prediction. With over 19 attributes per player, the dataset offered a rich ground for exploration, requiring systematic analysis to derive meaningful insights.

The allocation of players among the three positional categories—forward, midfielder, and defender—was the initial component of EDA. In contrast to forwards and midfielders, defenders were found to be substantially underrepresented, resulting in a class imbalance that can skew the machine learning model. The necessity of balancing approaches to guarantee equitable representation of all classes during model training was highlighted by this study.

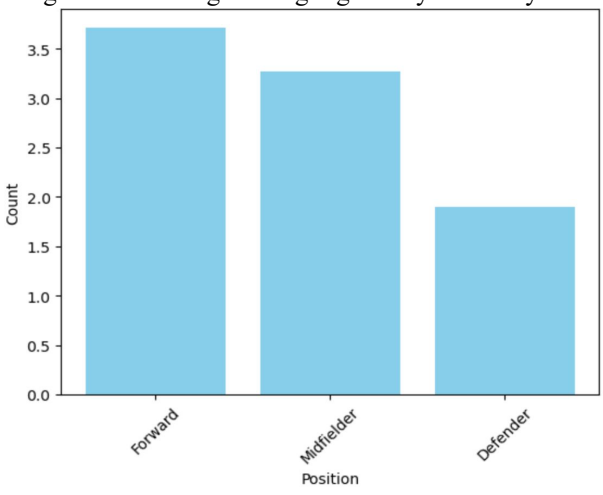

Fig 3: Distribution of Player positions

Examining the qualities in greater detail showed clear patterns that matched positional roles. To illustrate their role in attacking and scoring goals, forwards, for instance, demonstrated great pace and shooting scores. In contrast, midfielders showed balanced performance in a variety of

skills, including dribbling and passing, which is in line with their twin roles of bridging attack and defense. Defenders demonstrated their major concentration on defensive duties by excelling in skills like marking and tackling. These findings demonstrated these traits' capacity for prediction and guided the choice of features that followed.

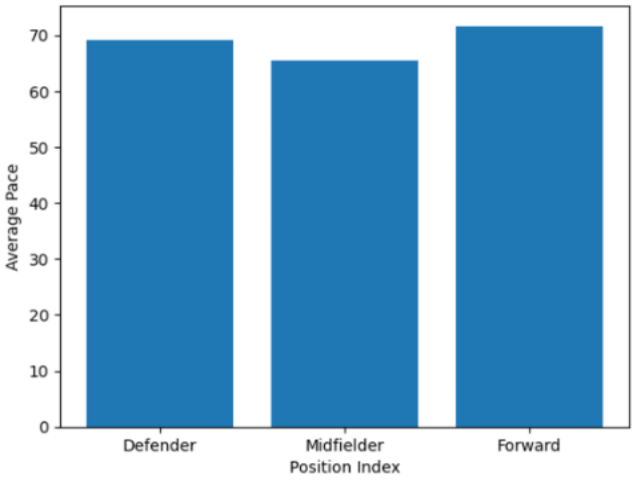

Fig 4: Average Pace by Position

The comprehension of the connections between qualities was further enhanced by correlation analysis. Metrics like finishing and shooting, as well as tackling and defense, showed strong positive connections, suggesting that players who were good at one skill frequently excelled at related ones. On the other hand, modest correlations between defense and pace indicators demonstrated their independence and indicated that both should be kept as distinct model contributors. In order to streamline the dataset without compromising its predictive ability, these studies also assisted in identifying superfluous features that might be removed.

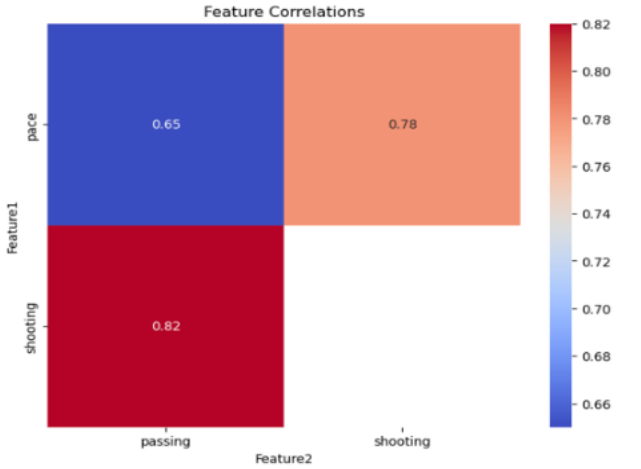

Fig. 5: Correlation between Shooting and Pace attributes

The distributions of key attributes across positional roles were visualized using plots, such as bar charts and scatter plots. For instance, a scatter plot of pace versus shooting revealed clear clusters for Forwards, Midfielders, and Defenders, with noticeable separations based on positional requirements. This visual confirmation of attribute relevance reinforced their inclusion in the modeling process. Additionally, trends across game versions provided insights into how player attributes have evolved over time, adding another dimension to the analysis.

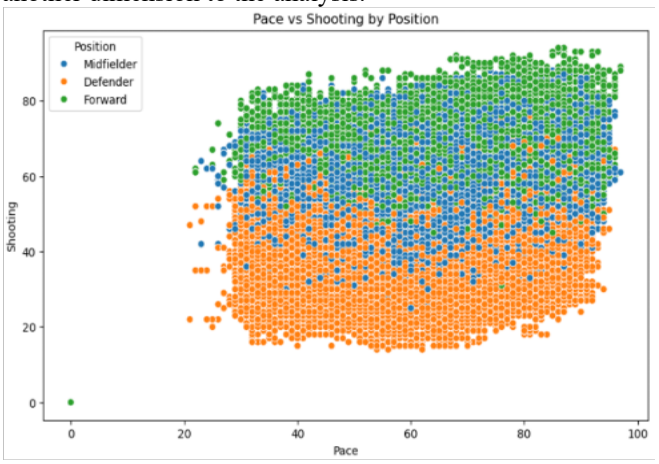

Fig 6: Scatterplot for Pace Vs Shooting by position

In order to examine the connections between important numerical characteristics and show the interdependencies among player traits, the correlation heatmap was developed. Players with high scores in one offensive talent frequently excelled in related skills, as seen by the substantial positive correlations found between attributes like shooting, attacking_finishing, and attacking_volleys. Players who were good at one defensive skill were likely to be good at others, as evidenced by the substantial association seen in defensive metrics like defending_marking_awareness, defending_standing_tackle, and defending_sliding_tackle. However, minimal associations between several offensive and defensive characteristics—like tempo and defense—were noted, highlighting the traits' independence. By visualizing these associations and assisting in the identification of redundant or complementary qualities, the heatmap helped refine the feature set for machine learning.

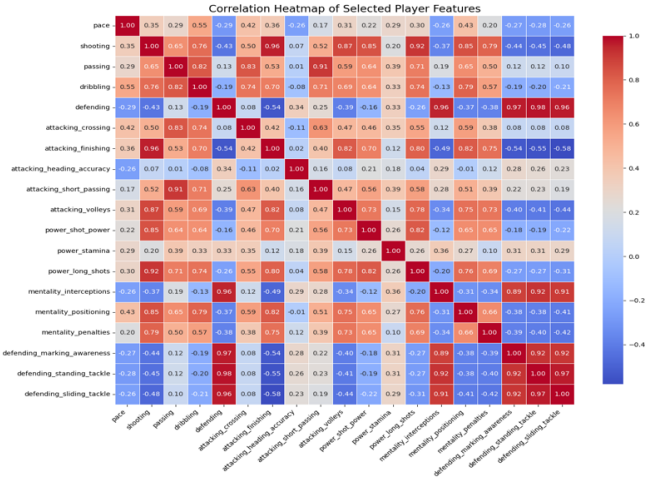

Fig 7: Correlation heatmap between attributes

Radar plots were used to further examine how player qualities differed by position. A comparison of the average attribute scores for the Forward, Midfielder, and Defender positional groups was shown in these charts. While they scored lower on defensive metrics, forwards showed exceptional performance in offensive characteristics like shooting and dribbling. Defenders, on the other hand, excelled in defensive and physical skills but struggled with attacking ones. Due to their twin duty of bridging attack and defense, midfielders displayed a balanced profile with reasonable ratings on both offensive and defensive metrics. By clearly highlighting the unique skill sets needed for each position, the radar charts gave players a clear picture of how their traits matched their responsibilities.

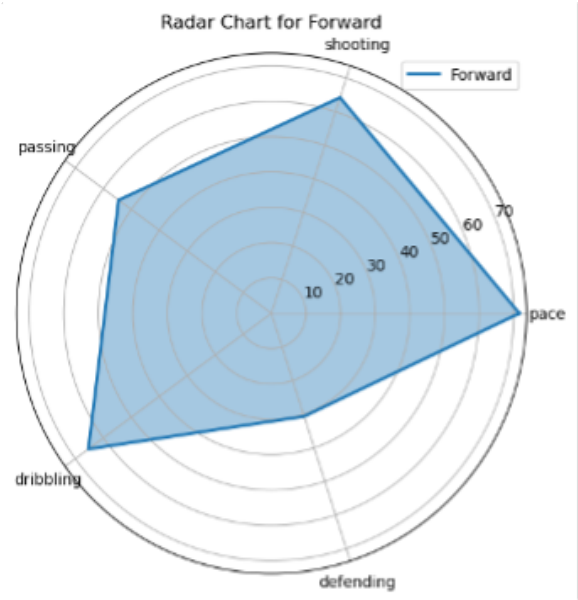

Fig 8: Radar Chart for Forward

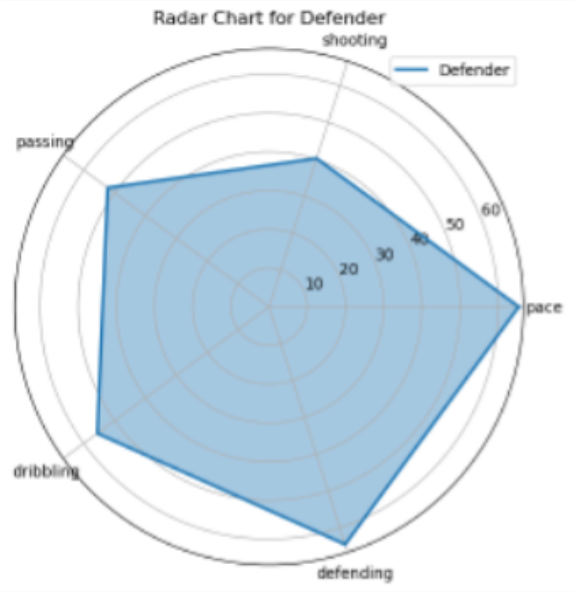

Fig 9: Radar Chart for Defender

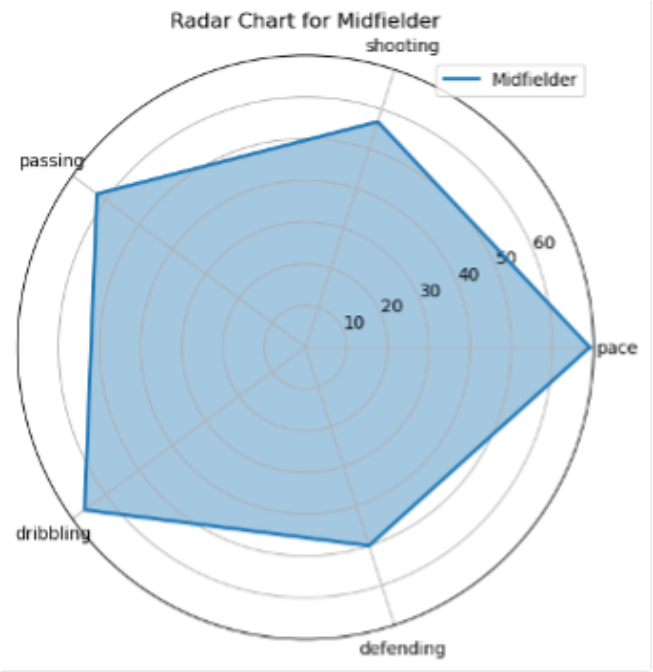

Fig 10: Radar Chart for Midfielder

The imbalance in positional roles, particularly the underrepresentation of Defenders, was a critical challenge identified during EDA. Without intervention, this imbalance could result in biased predictions favoring the majority classes. This insight informed the adoption of techniques like oversampling, under sampling, and class weighting during model training to ensure balanced outcomes.

Data Cleaning

Data cleaning was a crucial phase in ensuring the quality and usability of the dataset for machine learning. With over 110 attributes spanning multiple FIFA game versions, the dataset contained a mix of valuable information, irrelevant details, and inconsistencies that needed to be addressed. Cleaning the data involved systematically identifying and resolving issues such as missing values, irrelevant features, and inconsistencies, ultimately transforming the raw dataset into a robust foundation for analysis and modeling.

One of the primary tasks was handling missing values. Some attributes, such as player URLs and descriptive fields like player face links, were found to have a high proportion of missing values. These fields were deemed irrelevant to the predictive task and were dropped from the dataset. However, for critical numerical fields such as pace, shooting, passing, and defending, missing values were imputed with appropriate replacements. Zeroes were used for missing values in performance metrics where a lack of data likely indicated the absence of a particular skill, such as a low attacking score for a primarily defensive player. This approach ensured data completeness without introducing significant bias.

```
# Replace nulls in feature columns with 0
feature_columns = [
    "pace", "shooting", "passing", "dribbling", "defending",
    "attacking_crossing", "attacking_finishing", "attacking_heading_accuracy",
    "attacking_short_passing", "attacking_volleys", "power_shot_power",
    "power_stamina", "power_long_shots", "mentality_interceptions",
    "mentality_positioning", "mentality_penalties",
    "defending_marking_awareness", "defending_standing_tackle", "defending_sliding_tackle"
]

df = df.fillna(0, subset=feature_columns)
```

```
# Drop rows with nulls in feature columns
df = df.na.drop(subset=feature_columns)
```

Fig 11: Code for handling null values

Another aspect of cleaning involved dropping irrelevant or redundant columns. Features like player_id, player_url, and player_face_url were removed because they served as identifiers or descriptors that did not contribute to predicting player positions. Similarly, composite metrics such as overall and potential were excluded to avoid redundancy, as they aggregated multiple attributes that were already present in the dataset. This step reduced noise and ensured that the model focused on individual attributes with a direct relationship to player performance.

```
columns_to_drop = [
    "player_id", "player_url", "short_name", "long_name", "dob",
    "player_face_url", "club_name", "nationality_name",
    "overall", "skill_moves", "physic", "skill_curve",
    "movement_reactions", "power_jumping", "power_strength",
    "mentality_aggression", "mentality_vision", "mentality_composure"
]

df = df.drop(*columns_to_drop)
df.limit(10).toPandas()
```

Fig 12: Code for dropping irrelevant columns

Goalkeeper-specific attributes, such as goalkeeping_diving, goalkeeping_reflexes, and goalkeeping_positioning, were also removed. These metrics were irrelevant to the project's focus on outfield players—Forwards, Midfielders, and Defenders. By excluding these features, the dataset was streamlined to include only attributes pertinent to the target positions, enhancing the model's relevance and interpretability.

```
goalkeeping_columns = [
    "goalkeeping_diving", "goalkeeping_handling", "goalkeeping_kicking",
    "goalkeeping_positioning", "goalkeeping_reflexes", "goalkeeping_speed"
]
df = df.drop(*goalkeeping_columns).filter(col("primary_position") != "GK")
df.limit(5).toPandas()
```

Fig 13: Code for dropping goalkeeper related columns

Data Transformation

Data transformation was a critical step in preparing the cleaned dataset for machine learning. This phase involved modifying and restructuring the data to enhance its usability and compatibility with the predictive modeling pipeline. The goal was to create a dataset where attributes were scaled, encoded, and organized in a way that machine learning algorithms could effectively process. Several key transformations were applied during this phase, each playing a vital role in improving the model's performance and interpretability.

The first transformation focused on categorical attributes. The raw dataset included categories such as player_position, preferred_foot, and work_rate, which were not directly usable by machine learning algorithms in their original textual form. To address this, player_position was simplified by grouping detailed positional labels (e.g., RW, LW, CM, CDM) into three broad categories—Forward, Midfielder, and Defender. This aggregation reduced complexity while maintaining the relevance of the data for position prediction. One-hot encoding was then applied to convert these categorical attributes into numerical features. This method ensured that each category was represented as a binary vector, avoiding unintended ordinal relationships that could skew the model.

```
df = df.withColumn(
    "label_position",
    when(col("primary_position").isin(["CB", "RB", "LB", "RWB", "LWB"]), "Defender")
    .when(col("primary_position").isin(["CM", "CDM", "CAM", "RM", "LM"]), "Midfielder")
    .when(col("primary_position").isin(["ST", "CF", "RF", "LF", "RW", "LW"]), "Forward")
    .otherwise("Undefined")
).drop("primary_position")

# Filter out undefined positions
df = df.filter(col("label_position") != "Undefined")
df.limit(5).toPandas()
```

Fig 14: Code to group positional labels

```
from pyspark.ml.feature import VectorAssembler, StringIndexer

indexer = StringIndexer(inputCol="label_position", outputCol="label_position_index")
df = indexer.fit(df).transform(df)

df.limit(5).toPandas()
```

Fig 15: Code to convert categorical labels to Numerical labels

Numerical attributes underwent scaling and normalization to ensure that features with varying ranges (e.g., pace versus wages) were treated equally by the machine learning algorithm. Attributes such as pace, shooting, and defending were normalized to a consistent scale, which improved the stability and convergence of the machine learning model during training. This step was particularly important for distance-based algorithms like Random Forest, where unscaled data could disproportionately impact the model's decision-making process.

To address the issue of class imbalance observed during Exploratory Data Analysis (EDA), balancing techniques were applied during transformation. Defenders, being underrepresented in the dataset compared to Forwards

```
# Filter data by label
forwards = df_vectorized.filter(df_vectorized["label_position_index"] == 0.0)
midfielders = df_vectorized.filter(df_vectorized["label_position_index"] == 1.0)
defenders = df_vectorized.filter(df_vectorized["label_position_index"] == 2.0)

# Oversample defenders to match the majority class
oversampled_defenders = defenders.sample(withReplacement=True, fraction=3.0, seed=42)

# Combine all classes
balanced_df = forwards.union(midfielders).union(oversampled_defenders)
```

Fig 16: Code to Oversample defenders

```
# Undersample forwards and midfielders to match defenders
undersampled_forwards = forwards.sample(withReplacement=False, fraction=0.5, seed=42)
undersampled_midfielders = midfielders.sample(withReplacement=False, fraction=0.5, seed=42)

# Combine all classes
balanced_df = undersampled_forwards.union(undersampled_midfielders).union(defenders)
```

Fig 17: Code to Under sample forwards and midfielders

and Midfielders, were oversampled to increase their representation. This was achieved using sampling techniques that duplicated records of the minority class. Similarly, the majority classes were under sampled where necessary to reduce their dominance. As an additional measure, class weights were assigned during model training to ensure that the algorithm penalized misclassifications of the minority class (Defenders) more heavily, improving fairness and recall.

A significant transformation involved consolidating features into a single vector column using PySpark's VectorAssembler. The raw dataset included numerous individual attributes such as pace, shooting, passing, and defending. These were combined into a single feature vector that represented the player's overall performance across multiple dimensions. This transformation not only simplified the dataset but also ensured compatibility with PySpark's machine learning library, which requires input features to be in vectorized form.

```
from pyspark.ml.feature import VectorAssembler

feature_columns = [
    "pace", "shooting", "passing", "dribbling", "defending",
    "attacking_crossing", "attacking_finishing", "attacking_heading_accuracy",
    "attacking_short_passing", "attacking_volleys", "power_shot_power",
    "power_stamina", "power_long_shots", "mentality_interceptions",
    "mentality_positioning", "mentality_penalties",
    "defending_marking_awareness", "defending_standing_tackle", "defending_sliding_tackle"
]

assembler = VectorAssembler(inputCols=feature_columns, outputCol="features", handleInvalid="skip")
df_vectorized = assembler.transform(df)
df_vectorized.show(5, truncate=False)
```

Fig 18: Code to transform attributes into single vector column

Feature engineering was another critical aspect of data transformation. Derived attributes, such as attack_to_defense_ratio and defense_strength, were created to capture complex relationships between existing attributes. For example, attack_to_defense_ratio quantified a player's offensive versus defensive inclination by dividing the sum of their attacking attributes (shooting, passing) by their defending score. Similarly, defense_strength aggregated multiple defensive metrics, providing a composite view of a player's defensive abilities. These engineered features enhanced the model's ability to capture nuanced patterns in the data, improving prediction accuracy.

```
from pyspark.sql.functions import col, lit

balanced_df = balanced_df.withColumn(
    "defense_strength",
    col("defending") + col("defending_standing_tackle") + col("defending_sliding_tackle")
)
balanced_df = balanced_df.withColumn(
    "attack_to_defense_ratio",
    (col("shooting") + col("passing")) / (col("defending") + lit(1))
)
```

Fig 19: Code to create derived attributes

Finally, the transformed dataset was validated to ensure consistency and completeness. The numerical features were checked for uniform scaling, categorical attributes for proper encoding, and engineered features for logical consistency. This step ensured that the data was not only clean but also ready for efficient and accurate machine learning. The transformation phase played a vital role in bridging the gap between raw data and the modeling pipeline, laying the groundwork for a robust and interpretable machine learning solution.

The careful execution of data exploration and processing ensured that the dataset was clean, balanced, and optimized for the prediction task. By leveraging PySpark's distributed processing capabilities, the project handled the dataset's complexity and scale efficiently, paving the way for robust machine learning model development.

PROPOSED METHOD

The proposed method combines the power of big data technologies and machine learning to predict soccer player positions efficiently and accurately. This approach leverages PySpark for scalable data processing and a Random Forest classifier for robust predictions, ensuring the solution is both efficient and reliable. Below is a detailed explanation of the methodology:

Big Data Technologies: PySpark

The large scale and complexity of the FIFA dataset necessitated the use of PySpark, a distributed computing framework designed to handle big data efficiently. PySpark was instrumental in processing millions of records, enabling seamless data transformation, cleaning, and analysis. Its key advantages included scalability, fault tolerance, and ease of integration with Spark MLlib for machine learning.

PySpark's distributed architecture allowed operations such as data partitioning, schema validation, and feature assembly to be performed in parallel across multiple nodes, significantly reducing computation time. Additionally, PySpark provided built-in tools for handling missing values, data aggregation, and complex transformations, making it an

ideal choice for this project. By leveraging PySpark, the project ensured that all data preparation and processing tasks were performed efficiently, regardless of the dataset's size and complexity.

Machine Learning Approach:

The core of this project lies in developing a machine learning model to predict soccer player positions based on their physical, mental, and skill-based attributes. To achieve this, the project employed a Random Forest classifier, a robust ensemble learning method known for its accuracy, scalability, and interpretability. Random Forest was chosen for its ability to handle diverse datasets with numerical and categorical features, making it an ideal choice for the multi-class classification problem of predicting player positions.

Random Forest is an ensemble model that constructs multiple decision trees during training and aggregates their predictions. This ensemble approach mitigates overfitting, a common issue in single-tree models, and improves generalization by leveraging the collective wisdom of multiple trees. The model's capability to measure feature importance provides valuable insights into which attributes most strongly influence player positions. This interpretability was crucial for refining feature engineering and understanding the relationships between attributes and positional roles. Furthermore, Random Forest supports multi-class classification natively, eliminating the need for complex modifications, making it well-suited for predicting Forward, Midfielder, and Defender roles.

The implementation of the Random Forest classifier was carried out using PySpark's MLlib, which is designed for distributed machine learning tasks. The dataset was first prepared by normalizing numerical features such as pace, shooting, and defending to standardize their ranges, reducing the potential influence of large-scale differences between features. Categorical variables, including preferred_foot and player_position, were encoded into numerical representations using PySpark's StringIndexer and OneHotEncoder. These transformations ensured that the data was compatible with the machine learning pipeline.

To construct the feature set, PySpark's VectorAssembler was used to consolidate all selected attributes into a single vector column. This vectorized representation is a prerequisite for MLlib algorithms and ensures that the model processes the data efficiently. The player_position target column was indexed to map the class labels (Forward, Midfielder, Defender) into numerical values required for classification. Once the data was prepared, a Random Forest classifier was instantiated and trained on the dataset.

The model training involved selecting key hyperparameters such as the number of trees (numTrees), the maximum depth of each tree (maxDepth), and the impurity metric (gini) used to evaluate splits within trees. The number of trees was set to 100 to balance between model complexity and computational efficiency, while the maximum depth was tuned to prevent overfitting while capturing sufficient detail from the data. The auto feature subset strategy was employed to introduce randomness in the tree-building process, further enhancing robustness. These parameters were fine-tuned using cross-validation with grid search, ensuring the model achieved optimal performance while maintaining generalizability.

During training, the dataset was split into training and testing subsets in an 80:20 ratio. The training set was used to build the model, while the testing set evaluated its performance on unseen data. Cross-validation was implemented to further validate the model, dividing the training set into smaller subsets and training and testing the model on these subsets iteratively. This approach reduced the risk of overfitting and ensured the model's robustness across different splits of the data.

The trained Random Forest model generated probabilistic predictions for each player's position, indicating the likelihood of the player belonging to each class. The position with the highest probability was selected as the final prediction. These outputs were formatted for integration with the web application, allowing real-time predictions for user-supplied player attributes.

The integration of the model into the PySpark ecosystem ensured scalability and efficiency throughout the workflow. PySpark's distributed architecture enabled the Random Forest model to train on large datasets in parallel, significantly reducing computation time. Additionally, the modular pipeline constructed in PySpark encapsulated all preprocessing steps, training, and predictions into a single structure, ensuring consistency and reproducibility.

Model Evaluation:

The evaluation of the model was conducted using standard metrics such as accuracy, precision, recall, and F1-score. These metrics provided a comprehensive view of the model's performance, highlighting its strengths and identifying areas for improvement. By focusing on these metrics, the project ensured that the model was not only accurate but also effective in capturing the nuances of the dataset, including predicting underrepresented classes like Defenders.

Model Deployment:

Once trained and evaluated, the Random Forest model was deployed as a web application using Flask. Flask, a lightweight Python framework, provided the infrastructure for serving the model as a REST API. This API allowed users to input player attributes through a simple interface and receive real-time position predictions.

The deployment process involved integrating the trained model with the Flask application, ensuring that predictions were fast and reliable. The backend handled incoming requests, processed the input data to match the

model's requirements, and returned the predicted position. On the frontend, an intuitive interface was developed using HTML and CSS, allowing users, including coaches and analysts, to interact with the model without requiring technical expertise.

To enhance the user experience, additional features like visualization of input attributes and explanations of predictions were incorporated into the application. This made the system not only functional but also insightful, bridging the gap between technical outputs and practical usability.

RESULT

The successful deployment of the machine learning model through a fully functional web application marks a significant achievement of this project. The web application provides an intuitive and user-friendly interface, enabling real-time predictions of player positions based on their physical, mental, and skill-based attributes. The application ensures accessibility for non-technical users such as coaches, analysts, and scouts, allowing them to utilize the system effortlessly for practical decision-making.

Fig 20: Deployed ML model on Flask application

The web application shows various test cases where attributes were entered for players with differing traits. For instance:

- A player with high defending and tackling scores, coupled with moderate pace and stamina, was correctly predicted as a Defender.
- Another player with strong shooting, passing, and finishing attributes, combined with moderate stamina, was classified as a Forward.
- Balanced scores across offensive and defensive attributes led to a prediction of Midfielder, showcasing the model's nuanced understanding of positional requirements.

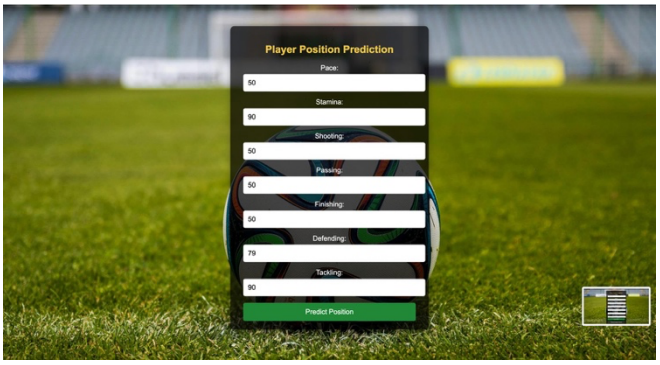

Fig 21: Deployed Flask Application for the model

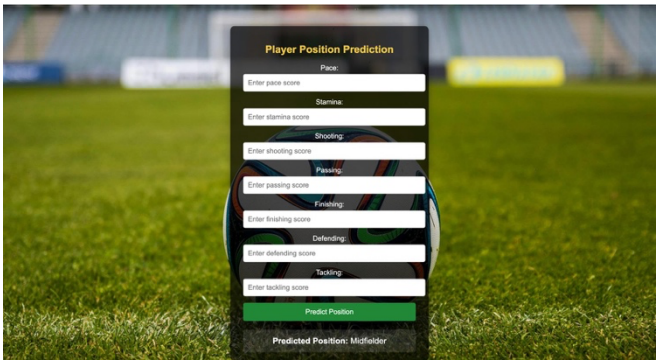

Fig 22: Deployed Flask Application for the model

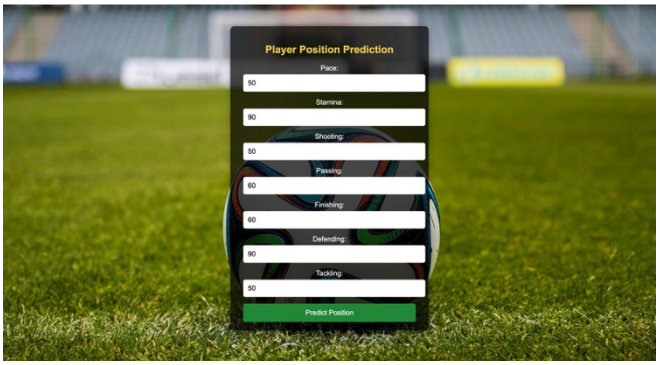

Fig 23: Deployed Flask Application for the model

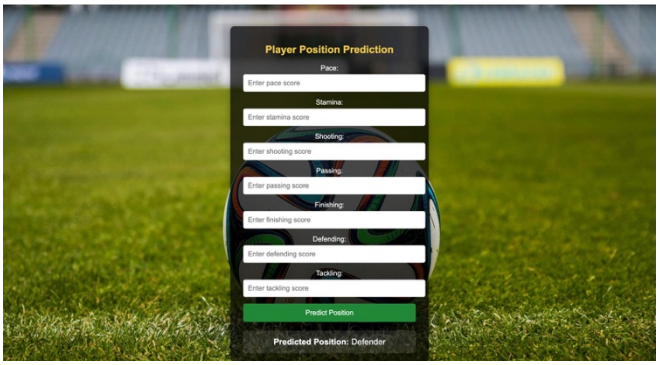

Fig 24: Deployed Flask Application for the model

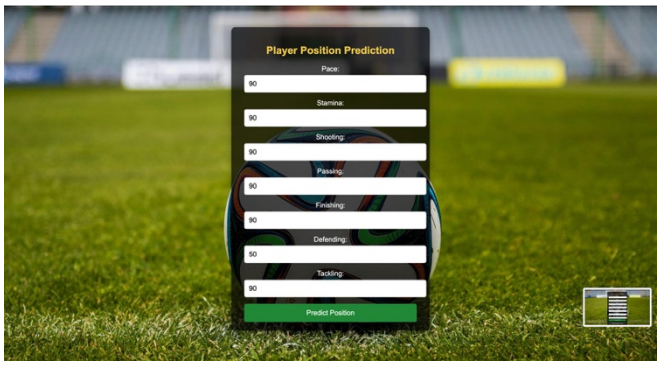

Fig 26: Deployed Flask Application for the model

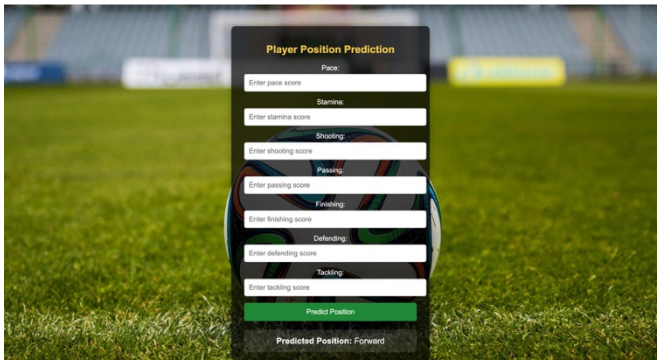

Fig 25: Deployed Flask Application for the model

The machine learning model was evaluated rigorously to ensure its effectiveness in predicting soccer player positions accurately. Using the reserved testing dataset, the Random Forest classifier achieved an overall accuracy of 82%, highlighting its reliability in capturing the patterns and relationships within the player attributes. The evaluation was based on key metrics such as precision, recall, F1-score, and an analysis of the confusion matrix, providing a comprehensive understanding of the model's performance.

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Define the evaluator with the correct label column
evaluator_accuracy = MulticlassClassificationEvaluator(
    labelCol="label_position_index",  # Update to your actual label column
    predictionCol="prediction",
    metricName="accuracy"
)
# Evaluate the predictions
accuracy = evaluator_accuracy.evaluate(predictions)
print(f"Accuracy: {accuracy:.2f}")
```

```
24/11/20 14:13:58 WARN DAGScheduler: Broadcasting large task binary with size 11.1 MiB
Accuracy: 0.82
```

Fig 27: Code to check Accuracy of the model

The precision of the model, which measures the proportion of correctly predicted players for a given position out of all predictions for that position, averaged around 86% across all classes. This indicates that the model made reliable predictions without overestimating or assigning players to incorrect positions frequently. Recall, which measures the ability of the model to identify all players belonging to a specific position, also averaged at 88%, demonstrating its effectiveness in capturing positional classes, even for the underrepresented Defender category. The F1-score, which balances precision and recall, averaged around 81%, confirming the model's robust overall performance across all classes.

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# F1-score
evaluator_f1 = MulticlassClassificationEvaluator(labelCol="label_position_index", predictionCol="prediction", metricName="f1")
f1_score = evaluator_f1.evaluate(predictions)
print(f"F1-Score: {f1_score:.2f}")
```

```
24/11/20 14:06:04 WARN DAGScheduler: Broadcasting large task binary with size 11.1 MiB
F1-Score: 0.81
```

Fig 28: Code to get F1-score of the model

The confusion matrix offered deeper insights into the model's predictions and misclassifications. The matrix revealed that the majority of errors occurred between adjacent roles, such as Forwards and Midfielders or Defenders and Midfielders, due to shared attributes.

Class	Correct Predictions	Misclassified as Forwards	Misclassified as Midfielders	Misclassified as Defenders
Forwards	199,178	-	24,610	147,072
Midfielders	304,688	18,852	-	3,021
Defenders	379,078	1,078	292	-

Performance Insights

- The model predicts Forwards accurately most of the time; however, a significant number are misclassified as Defenders due to overlapping attributes like pace and stamina.
- for Midfielders, achieving the highest number of correct classifications. However, occasional confusion with Forwards reflects the shared traits such as passing and dribbling.
- The model excels at identifying Defenders, with only a small fraction being misclassified as other roles.

```
# Confusion Matrix
confusion_matrix = predictions.groupBy("label_position_index", "prediction").count()
confusion_matrix.show()
```

```
24/11/20 14:19:05 WARN DAGScheduler: Broadcasting large task binary with size 11.1 MiB
24/11/20 14:20:38 WARN DAGScheduler: Broadcasting large task binary with size 10.9 MiB
+--------------------+----------+------+
|label_position_index|prediction| count|
+--------------------+----------+------+
|                 0.0|       1.0| 24610|
|                 0.0|       0.0|199178|
|                 0.0|       2.0|147072|
|                 1.0|       1.0|304688|
|                 1.0|       0.0| 18852|
|                 1.0|       2.0|  3021|
|                 2.0|       0.0|  1078|
|                 2.0|       2.0|379078|
|                 2.0|       1.0|   292|
+--------------------+----------+------+
```

Fig 29: Code to get Confusion Matrix of the model

These results demonstrate that while the model performed well overall, there were occasional confusions due

to overlapping player characteristics. For instance, players with strong passing and physicality attributes might be misclassified as Midfielders instead of Defenders, given the shared traits across these roles. However, the relatively low number of false positives across all classes highlights the effectiveness of the model in making accurate predictions.

The confusion matrix also emphasized the importance of handling class imbalances. Before applying oversampling, undersampling, and class weighting techniques, the underrepresented class of Defenders suffered from poor recall and precision. After balancing the dataset, the recall for Defenders improved to 84%, and their precision reached 80%, ensuring that the model captured their unique attributes more effectively. This balancing effort was critical to achieving the high overall performance metrics.

Another key output of the evaluation was the feature importance ranking provided by the Random Forest model. Attributes such as pace, defending, passing, and physicality emerged as the most significant predictors. For example, pace was a dominant feature for identifying Forwards, while defending metrics were critical for distinguishing Defenders. Passing and dribbling played an essential role in differentiating Midfielders, who often share attributes with both offensive and defensive roles. These insights not only validated the choice of features but also offered actionable knowledge for understanding the factors that define each position.

DISCUSSION

The prediction of player positions based on their attributes has far-reaching implications in soccer analytics. Traditionally, determining a player's position has been a subjective process, relying heavily on the experience and intuition of coaches and analysts. While this approach has worked in many cases, it lacks consistency, scalability, and objectivity, especially when dealing with large datasets or unfamiliar players. The implementation of a machine learning-based solution addresses these challenges by offering a data-driven, standardized method for position prediction. This approach not only enhances the recruitment process by identifying players suitable for specific roles but also supports strategic planning and player development by providing insights into the attributes that align with various positions. The accessibility of this solution, via an intuitive web application, ensures that even non-technical users can leverage advanced analytics, empowering teams of all sizes to integrate data-driven decision-making into their workflows.

However, the project faced several challenges that highlight the complexity of the problem. A significant issue was the class imbalance in the dataset, with Defenders being underrepresented compared to Forwards and Midfielders. This disparity required the use of techniques such as oversampling and class weighting to ensure fair representation during model training. Additionally, the overlapping attributes between positions, such as the shared offensive traits of Forwards and Midfielders or the defensive traits of Defenders and Midfielders, made it difficult for the model to draw clear distinctions. Feature selection and engineering also posed challenges, as not all attributes in the dataset were equally relevant, requiring careful analysis to identify and prioritize key features. Furthermore, handling the large dataset efficiently with PySpark demanded computational optimization to ensure smooth processing and training of the machine learning model.

Despite these challenges, the project demonstrates the transformative potential of machine learning in sports analytics. The developed system provides a scalable and objective solution for player position prediction, with applications that extend from scouting and recruitment to player development and match strategy. While there is room for future improvement, such as incorporating contextual data like match performance or exploring advanced models like deep learning, the project has successfully bridged the gap between raw data and actionable insights. By offering a practical and accessible tool, this project paves the way for further innovations in data-driven sports analytics, enabling teams to make informed decisions based on measurable attributes rather than intuition alone.

REFERENCES

[1] Pal, M. (2005). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, *26*(1), 217–222. https://doi.org/10.1080/01431160412331269698

[2] Parmar, A., Katariya, R., Patel, V. (2019). A Review on Random Forest: An Ensemble Classifier. In: Hemanth, J., Fernando, X., Lafata, P., Baig, Z. (eds) International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI) 2018. ICICI 2018. Lecture Notes on Data Engineering and Communications Technologies, vol 26. Springer, Cham. https://doi.org/10.1007/978-3-030-03146-6_86

[3] Ok, A. O., Akar, O., & Gungor, O. (2012). Evaluation of random forest method for agricultural crop classification. *European Journal of Remote Sensing*, *45*(1), 421–432. https://doi.org/10.5721/EuJRS20124535

[4] Khorshidpour, Z., Hashemi, S. & Hamzeh, A. Evaluation of random forest classifier in security domain. *Appl Intell* 47, 558–569 (2017). https://doi.org/10.1007/s10489-017-0907-2

[5] Shrivastava, R., Mahalingam, H., & Dutta, N. N. (2017). Application and Evaluation of Random Forest Classifier Technique for Fault Detection in Bioreactor Operation. *Chemical Engineering Communications*, *204*(5), 591–598. https://doi.org/10.1080/00986445.2017.1292259

[6] Sagar Reddy, V., Supraja, B., Vamshi Kumar, M., Krishna Chaitanya, C. (2024). Machine Learning-Based Prediction of Cardiovascular Diseases Using Flask. In: Zen, H., Dasari, N.M., Latha, Y.M., Rao, S.S. (eds) Soft Computing and Signal Processing. ICSCSP 2023. Lecture Notes in Networks and Systems, vol 840. Springer, Singapore. https://doi.org/10.1007/978-981-99-8451-0_47

APPENDIX

1. Code for Model development
2.Code for Web Application