CSE422: ARTIFICIAL INTELLIGENCE
SECTION- 10
FINAL PROJECT REPORT

FIFA22 Position Prediction

MEMBERS:

1. Kazi Mahathir Rahman (23341066)
2. Naser Ahmed Rashed (18101313)

TABLE OF CONTENTS:

Introduction:

In the realm of FIFA22_Position Prediction, the efficacy of machine learning models heavily relies on their training methodologies and underlying algorithms. As the digital landscape of football simulation evolves, so do the techniques employed to predict player positions within the game. Various models, each with its unique architecture and learning approach, contribute to the diverse landscape of predictive analytics in FIFA22. The differences between these training models are profound and nuanced. Some models, such as logistic regression, naive's classifier, XGboost, decision tree, and random tree rely on linear decision boundaries and probabilistic interpretations, making them suitable for simpler classification tasks. On the other end of the spectrum, complex ensemble methods like Random Forest and Gradient Boosting leverage decision trees and ensemble techniques to capture intricate relationships within the data, leading to highly accurate predictions. All these features are how to work in our model that are described in this report. For example, home venues are always in favor of a team. So, the column 'where?' of the dataset can be an important feature for match prediction which we will map to define home as a higher priority. Moreover, before using these features, we will need to have a clean dataset for that reason, we will do pre-processing on our existing dataset by following some methods like null value removal, concat, imputation, and feature scaling.

Motivation:

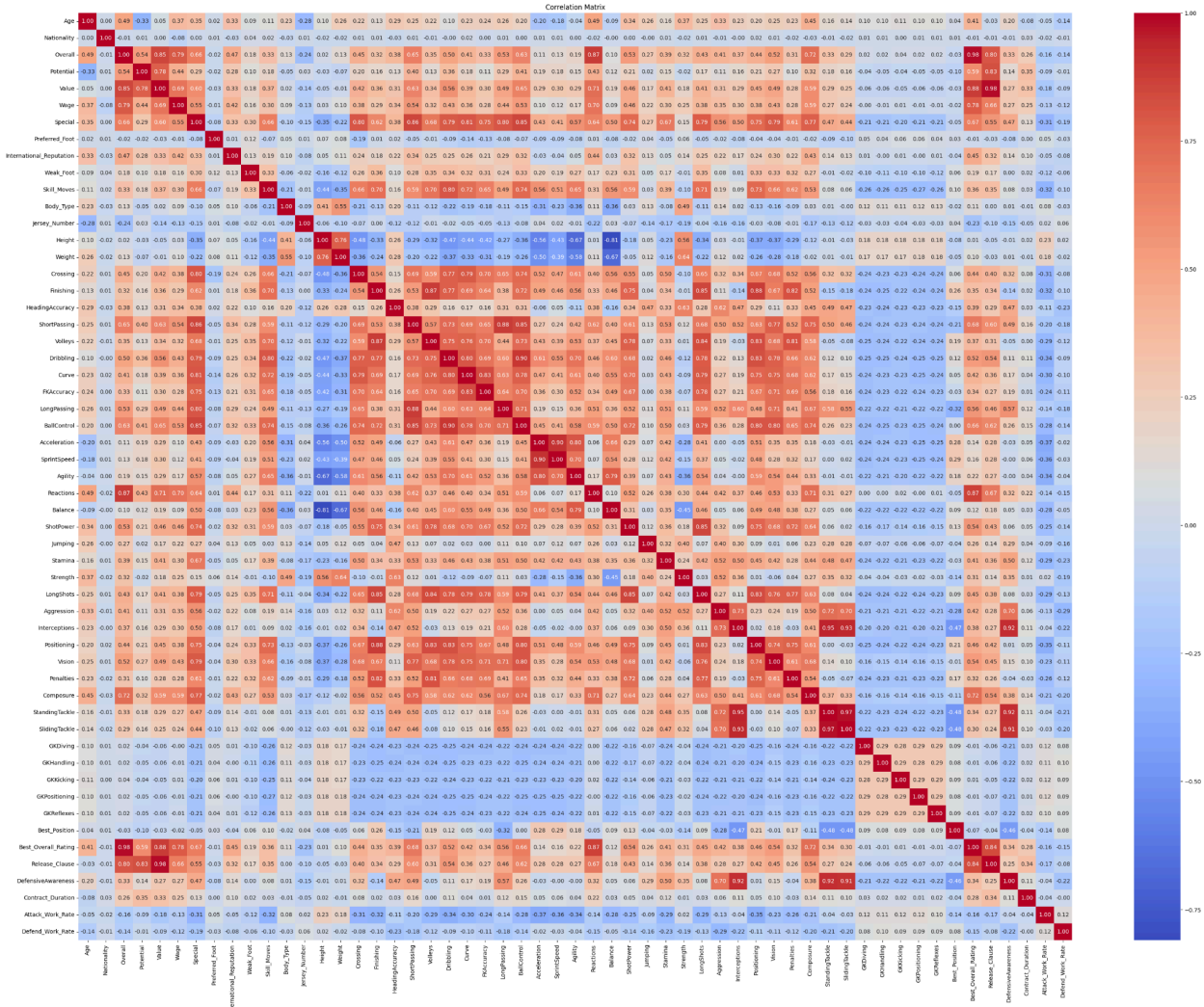
The objective of the FIFA22_Position Prediction study is to augment the gameplay by precisely projecting player locations, refining team composition, and investigating machine learning methodologies. Solving difficulties and restrictions in predictive modeling brought on by the dynamic nature of player movements also advances sports analytics. To stimulate research and innovation in machine learning, sports analytics, and gaming technologies, the study attempts to take advantage of the hype around the FIFA 22 game. The paper seeks to provide useful insights and workable solutions for improving gameplay and developing predictive modeling in gaming by fusing domain experience with state-of-the-art approaches.

Data description:

Source:

Link: https://www.kaggle.com/datasets/stefanoleone992/fifa-22-complete-player-dataset
Reference: https://www.kaggle.com/code/aidinkiany/fifa22-position-prediction

In total, there are 31 features. The label “**Best_Position**” which is the player's best position based on their attributes. The "**Best_Position**" column having 15 unique classes converted to 8, suggests that it's a multi-class classification problem. Each class represents a different position that a player can play in. Classification problems involve predicting the category or class of a given input based on its features. In this case, given the various attributes of a player, the goal would be to predict which position suits the player best among the 8 available options. Therefore, it fits the definition of a classification problem, specifically a multi-class classification one. In total, the data instances are 16767. The Correlation-

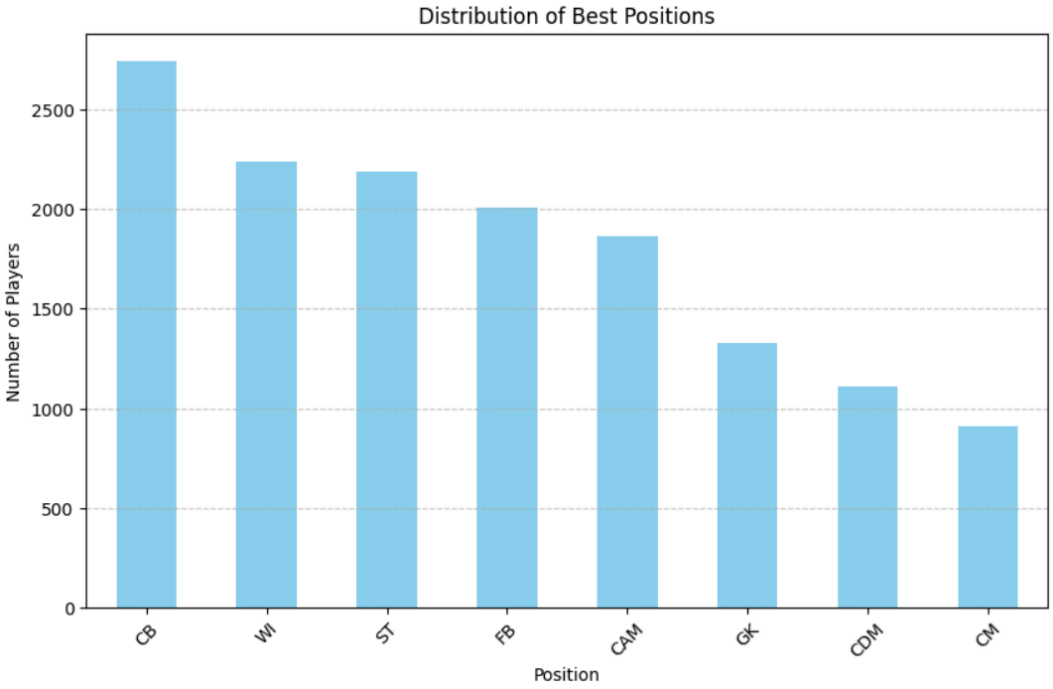


A small description of the features-

- ID: Unique identifier for each player.
- Name: Name of the player.
- Age: Age of the player.
- Photo: Image of the player.
- Nationality: Nationality of the player.
- Flag: Flag representing the player's nationality.
- Overall: Overall rating of the player's skills and abilities.
- Potential: Potential rating indicating the player's expected future performance.
- Club: Current club of the player.
- Club Logo: Logo of the player's current club.
- Value: Market value of the player.
- Wage: Weekly wage of the player.
- Special: Special attributes or abilities of the player.
- Preferred_Foot: Player's preferred foot for playing.
- International_Reputation: Reputation of the player on the international stage.
- Weak_Foot: Rating of the player's weaker foot ability.
- Skill_Moves: Rating of the player's skill moves.
- Work_Rate: Player's work rate indicating defensive and offensive work.
- Body_Type: Body type of the player.
- Real_Face: Indicator of whether the player has a real face in the game.
- Position: Primary playing position of the player.
- Jersey_Number: Jersey number was worn by the player.
- Joined: Date when the player joined their current club.
- Loaned_From: Club the player is loaned from (if applicable).
- Contract_Valid_Until: Expiry date of the player's contract.
- Height: Height of the player.
- Weight: Weight of the player.
- Various skills attributes (e.g., Crossing, Finishing, HeadingAccuracy, etc.): Ratings for different skills and attributes of the player.
- Best_Overall_Rating: Overall rating for the player's best position.
- Release_Clause: Release clause amount in the player's contract.
- DefensiveAwareness: Rating indicating the player's defensive awareness.

The label has a total of 8 classes and they don’t have the same number of instances.

```
8
['CAM' 'CM' 'ST' 'FB' 'CDM' 'CB' 'WI' 'GK']
```


Distribution of Best Positions
Number of Players
0
500
1000
1500
2000
2500
CB
WI
ST
FB
CAM
GK
CDM
CM
Position

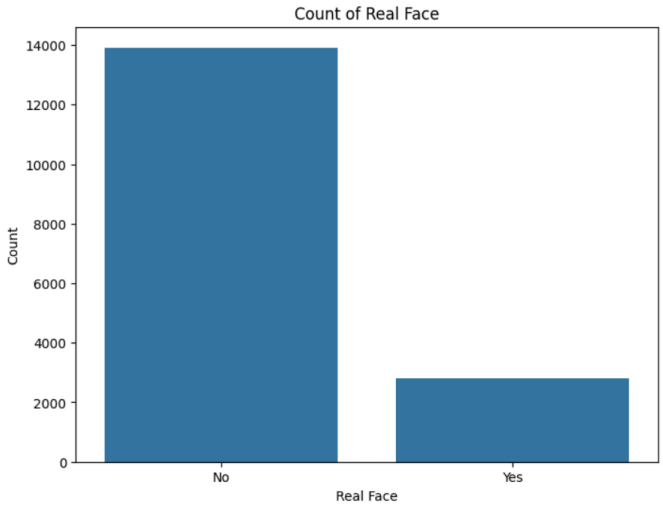
Data Pre-Processing:

Step by step -

- **Problem_1** - “Loaned_From” and “Marking” have null values of more than 90%. So dropped two columns.

```
Loaned_From -- 93.22418738049714 -- object
Marking -- 94.66419694072657 -- float64
```

- Problem_2 - ‘Real_Face’ have was too imbalance which make it model biased. So dropped the column.



- Problem_3 - The difference between ‘Contract_Valid_Untill’ and ‘Joined’ gives us information about a player and how long will play for a club. A new column was made named ‘Contract_duration’ which contains the difference between ‘Contract_Valid_Untill’ and ‘Joined in days.

```
df['Contract_Duration']
✓ 0.0s

0         5.0
1         8.0
2         2.0
3        10.0
4         4.0
        ...
16731     0.0
16732     4.0
16733     2.0
16734     2.0
16735     6.0
Name: Contract_Duration, Length: 16736, dtype: float64
```

- Problem_4 - ‘Name’ defines a player uniquely. If ‘Name’ column has a duplicate, that decreases the model quality. So drop the duplicate values from ‘Name’.

```
Number of rows with duplicate names: 647
```

- Problem_5 - Now drop the unnecessary categorical columns which is not needed to train in the model. 'ID','Name','Photo', 'Flag', 'Club Logo', 'Position', 'Joined','Contract_Valid_Until' are dropped.

```
to_drop = ['ID','Name','Photo', 'Flag', 'Club Logo', 'Position', 'Joined','Contract_Valid_Until']
df.drop(columns=to_drop, inplace=True)
```

- Problem_6 - Dropped the rows which contains null values. After drop the data instances number was 14385.

```
(14385, 55)
```

- Problem_7 - ‘Value’, ‘Wage’, and ‘Release_clause’, these three columns have a currency data type. These columns were converted to numeric values.

```
def convert_currency_to_number(x):
    output = x.replace('€','')
    if 'M' in x:
        output = output.replace('M','')
        output = float(output) * 1000000
    elif 'K' in x:
        output = output.replace('K','')
        output = float(output) * 1000
    output = int(output)
    return output

df['Value'] = df['Value'].apply(convert_currency_to_number)
df['Wage'] = df['Wage'].apply(convert_currency_to_number)
df['Release_Clause'] = df['Release_Clause'].apply(convert_currency_to_number)
```

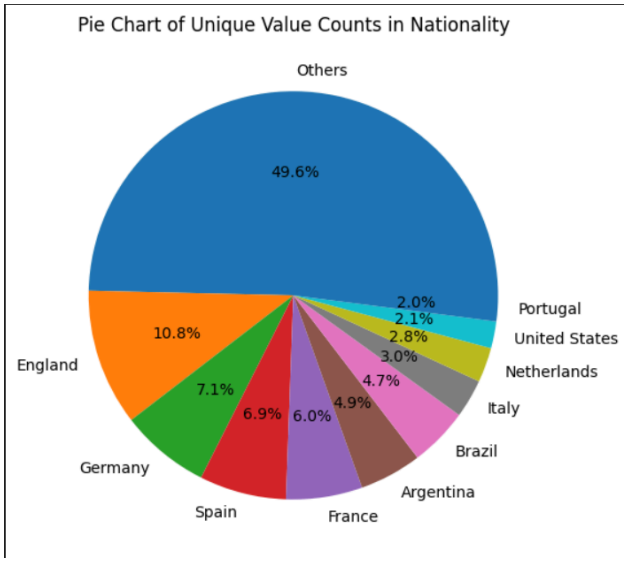
- Problem_8 - ‘work_rate’ column values have two values split by ‘/’. Two new columns named ‘att_work_rate’ and ‘def_work_rate’ were created from this column.

```
def convert_work_rate(x):
    attack_work_rate, defend_work_rate = x.split('/')
    return attack_work_rate, defend_work_rate
```

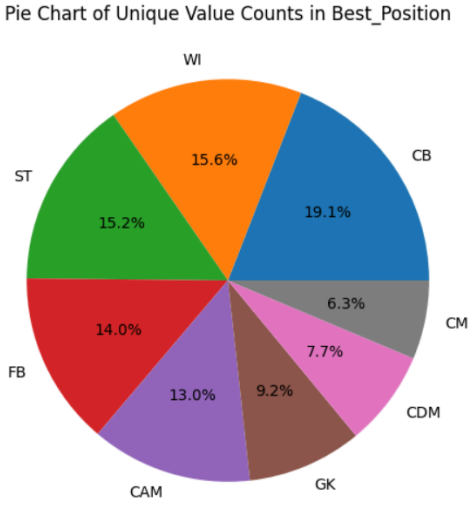
- Problem_9 - Dropped ‘Club’ as it has a lot of unique values.

```
832
```

- Problem_10 - The ‘Notionality’ column has 166 unique values. The values was converted to 11.

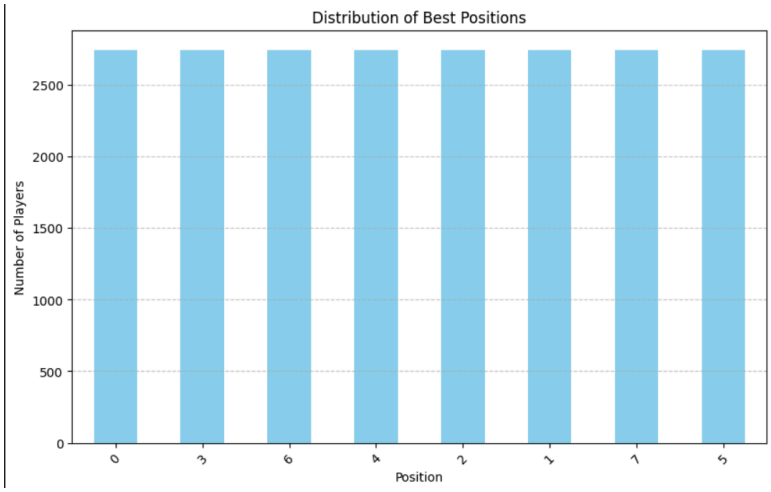


- Problem_11 - The "**Best_Position**" column has a total of 15 unique values. That was converted to 8.



- Problem_12 - Two types of encoding were done. Some columns were encoded with label encoder and some columns were encoded with ordinal encoder.

- Problem_13 - Resample using SMOTE to make all the label count same.



- Problem_14 - Split the dataset into train and test
    - Train - 80%
    - Test - 20%

Model Training and Visualization -

Decision Tree:
Decision Tree is a non-parametric supervised learning algorithm used for classification and regression tasks. It splits the dataset into smaller subsets based on features that best separate the target variable.

Concept:

- At each node of the tree, the algorithm selects the feature that best splits the data into the purest possible subsets (homogeneous with respect to the target variable).
- It continues this process recursively until it reaches a stopping criterion, such as a maximum tree depth or a minimum number of samples in a leaf node.

Math:

- The decision tree algorithm calculates the impurity of each split using a impurity measure such as Gini impurity or entropy.
- For each split, it calculates a splitting criterion (e.g., Gini gain or information gain) based on the impurity reduction achieved by the split.
- The split that maximizes the impurity reduction becomes the chosen split at each node.

Matrics:

```
Accuracy = 0.793943533697632
Precision = 0.795702656755994
Recall = 0.793943533697632
F1 Score = 0.7936577117068189
Confusion Matrix = [[350   0  14  73   7   0  17  75]
 [  2 463  41   4  56   0   1   1]
 [  3  20 441  68  20   0   0   2]
 [ 58   6  64 437   9   0   0   7]
 [  0  23  30  16 438   0   1  13]
 [  0   0   0   0   0 547   0   0]
 [ 28   0   1   4   1   0 474  31]
 [101   4   5  25  28   0  46 337]]
```

Random Forest:
Random Forest is an ensemble learning method based on decision trees. It builds multiple decision trees and combines their predictions to improve the accuracy and robustness of the model.

Concept:

- Random Forest constructs a forest of decision trees, where each tree is trained on a random subset of the data and a random subset of the features.

- During prediction, it aggregates the predictions of individual trees to produce a final prediction, such as taking the majority vote (for classification) or averaging (for regression).

Math

- Random Forest builds multiple decision trees using bootstrapped samples of the training data.
- At each node of each tree, it selects the best split among a random subset of features.
- During prediction, it aggregates the predictions of all trees to make the final prediction.

Matrics:

```
Accuracy = 0.8852459016393442
Precision = 0.8858390189311312
Recall = 0.8852459016393442
F1 Score = 0.8845146231124215
Confusion Matrix = [[405   0   3  44   3   0  13  68]
 [  2 488  34   4  40   0   0   0]
 [  0   9 502  33  10   0   0   0]
 [ 17   0  22 541   1   0   0   0]
 [  0  15  17   5 481   0   0   3]
 [  0   0   0   0   0 547   0   0]
 [  8   1   2   5   0   0 502  21]
 [ 62   0   2   8  24   0  28 422]]
```

XGBoost (Extreme Gradient Boosting):
XGBoost is an optimized implementation of gradient boosting, which is an ensemble learning technique that builds a strong predictive model by combining multiple weak models (typically decision trees).

Concept:

- XGBoost builds trees sequentially, where each tree corrects the errors made by the previous trees.
- It uses a gradient descent algorithm to minimize a loss function (e.g., mean squared error for regression or cross-entropy loss for classification) by adding new trees to the model.

Math:

- XGBoost minimizes the objective function by iteratively adding trees to the model.
- At each iteration, it calculates the gradient and Hessian of the loss function with respect to the model's predictions.
- It then fits a new tree to the negative gradient of the loss function using the residuals as the target values.
- Finally, it updates the model's predictions by adding the predictions of the new tree, weighted by a learning rate.

Matrics:

```
Accuracy = 0.8515482695810564
Precision = 0.8556772910104631
Recall = 0.8515482695810564
F1 Score = 0.851634919937884
Confusion Matrix = [[375   0   6  74   3   0  10  68]
 [  0 484  40   3  41   0   0   0]
 [  1   6 455  78  14   0   0   0]
 [ 16   0  65 490   8   0   1   1]
 [  0   9  22   6 481   0   0   3]
 [  0   0   0   0   0 547   0   0]
 [ 10   0   1   6   1   0 499  22]
 [ 66   0   3  14  27   0  27 409]]
```

Logistic Regression:
Logistic Regression is a linear model used for binary classification tasks. Despite its name, it's a classification algorithm rather than a regression algorithm.

Concept:

- Logistic Regression models the probability that a given input belongs to a particular class using a logistic (sigmoid) function.
- It estimates the coefficients of the linear combination of features using maximum likelihood estimation.

Math:

- Logistic Regression models the log-odds of the probability of the positive class using a linear combination of features:

  $$: \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n.$$

- The logistic function (sigmoid) transforms the log-odds to a probability:

  $$\frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_n x_n)}}.$$

- Logistic Regression estimates the coefficients (parameters) \(\beta \) that maximize the likelihood of the observed labels given the features.

Matrics:

```
Accuracy = 0.8743169398907104
Precision = 0.8758964635783923
Recall = 0.8743169398907104
F1 Score = 0.8740562210461565
Confusion Matrix = [[399   0   2  59   0   0  14  62]
 [  1 497  35   2  33   0   0   0]
 [  0  11 478  53  12   0   0   0]
 [ 15   1  56 497   6   0   0   6]
 [  1  10  15   4 489   0   0   2]
 [  0   0   0   0   0 547   0   0]
 [  8   1   1   5   0   0 506  18]
 [ 58   0   0  15  23   0  23 427]]
```

Naive Bayes Classifier:
Naive Bayes is a probabilistic classifier based on Bayes' theorem with the "naive" assumption of independence between features.

Concept:

- Naive Bayes calculates the posterior probability of each class given the features using Bayes' theorem.
- It assumes that the features are conditionally independent given the class label, which simplifies the calculation of the posterior probability.

Math:

- Naive Bayes calculates the posterior probability of each class given the features using Bayes' theorem:

$$P(y|x_1, x_2, \ldots, x_n) = \frac{P(x_1,x_2,\ldots,x_n|y) \cdot P(y)}{P(x_1,x_2,\ldots,x_n)}$$

- The "naive" assumption of feature independence allows us to factorize the likelihood as

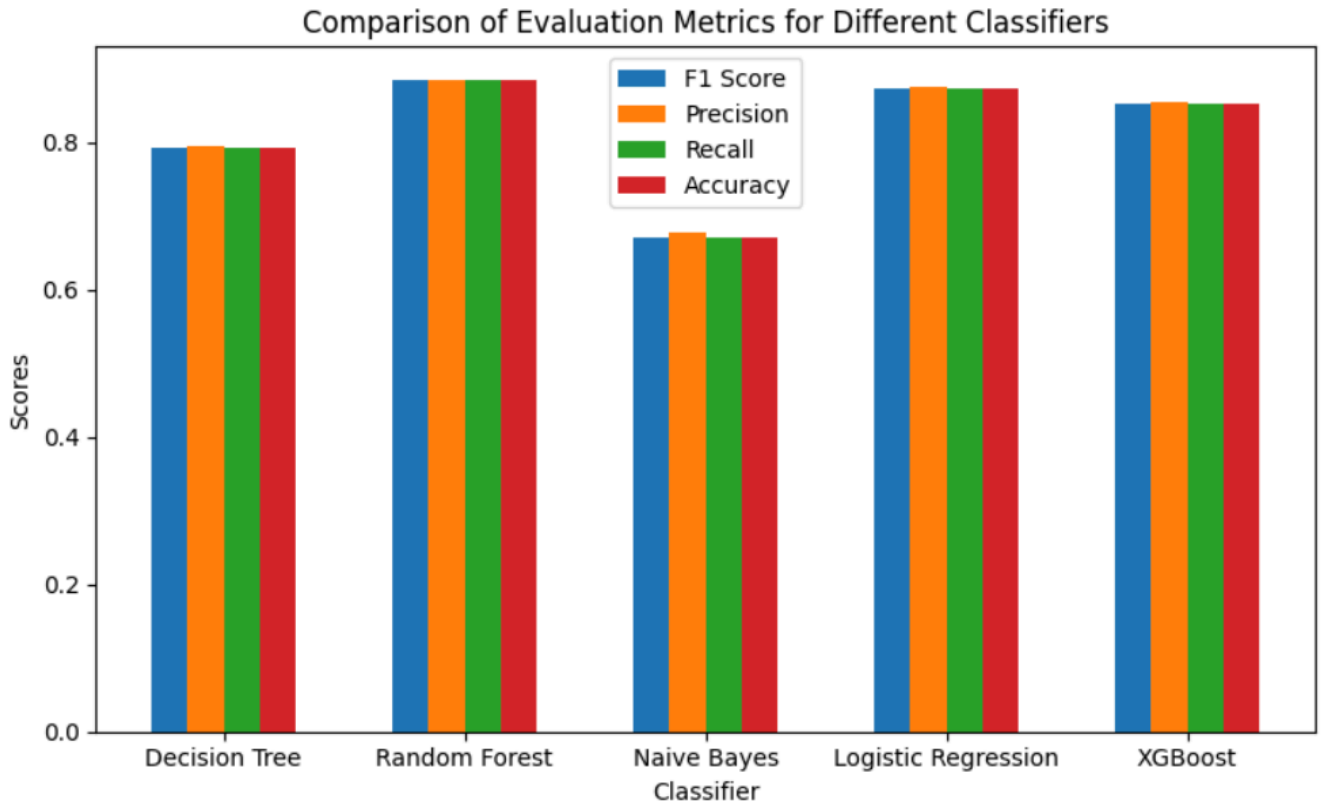
$$P(x_1, x_2, \ldots, x_n|y) = P(x_1|y) \cdot P(x_2|y) \cdot \ldots \cdot P(x_n|y)$$

- Naive Bayes estimates the class prior \(P(y) \) and the conditional probabilities P(xi | y) from the training data.
- During prediction, it selects the class with the highest posterior probability as the predicted class for a given input.

Matrics:

```
Accuracy = 0.6707650273224044
Precision = 0.678473838516849
Recall = 0.6707650273224044
F1 Score = 0.6714191507487143
Confusion Matrix = [[285   0  22  51  23   0  22 133]
 [  1 420  79  11  57   0   0   0]
 [  2  39 337 120  56   0   0   0]
 [ 33   1 186 285  67   0   0   9]
 [  2  42  77  43 357   0   0   0]
 [  0   0   0   0   0 547   0   0]
 [ 15   1   9   7   1   0 463  43]
 [144   1  19  21  52   0  57 252]]
```

Comparison-



Though random forest and logistic regression get the highest accuracy, the best model is XGBoost. Cause random forest was overfitted and logistic regression was underfitted. But the perfect and accurate was XGBoost.