CSI5155: Machine Learning

Scouting Best Playing Position For FIFA 2019 Players

Pahuldeep Singh Sidhu &
Saneer Gera

ABSTRACT

This project aims to apply machine learning concepts and models learned in class on a Fifa 19 dataset to perform player classification and do the statistical analysis of the performance of various algorithms under consideration. Feature extraction techniques were employed after visualizing data through Exploratory Data Analysis (EDA) to remove insignificant features, handle null values and convert categorical data to numeric. Dataset was used to train and build different machine learning models (Linear, Tree-Based, rule-based, distance-based, and ensembles) using the Train-Test split paradigm as well as employing the K-cross validations. Removing features using correlation and handling the outliers significantly improved the performance of all algorithms making ensembles and K-Nearest neighbours as best performing algorithms. Visualization suggested class forward as under-balanced, it was oversampled which resulted in improved performance. Statistical significance testing using Friedman's test showed that there is a significant difference in the performance of these algorithms with Ensembles being the best performers.

1. INTRODUCTION

FIFA, also known as FIFA Football or FIFA Soccer, is a series of football video games or football simulator, released every year by Electronic Arts under the label of EA Sports[1]. It holds the record of best selling video game franchise in Guinness Book World Records and had sold over 260 million copies [2][3]. The dataset for our project is derived from Fifa 19 which is the 26th instalment of a Fifa series developed by EA Vancouver. The problem domain is Sports analytics with the player position classification being the primary objective.

2. CASE STUDY

Fifa offers different modes to play, but the most popular is the career mode/manager mode whose simulation environment of acting as a football coach to train, purchase and sell world-class players to build a successful football team gives the audience a sense of real-time involvement in the game. All this simulation is practically impossible without the presence of the actual player data from the real world of football. Therefore, the role of accurate player data is a very crucial factor in the success of the Fifa series. Every year when the new version of the game is released, player data is updated on the basis of their performance in the real world and also new players are added.

To build a successful team in-game, as well as, in real-time correct classification of player's playing position is very essential, as it helps to determine the best position on the football pitch where we can get maximum output from the player. The position classification depends upon the number of significant factors pertaining to the physical attributes of the player. Classification of positions can also help when the new data arrives, as machine learning models trained can be used to predict the playing positions of the new players added to the dataset, with the least and most significant features in the dataset.

The concepts of feature engineering, data pre-processing from classroom learning were used to analyse and

evaluate the Fifa19 dataset to train and build five models: Tree model (Decision Tree), Linear Model (Naıve Bayes), Distance-based(KNN), Rule-Based(Dummy Classifier), Ensemble(Gradient Boosting). To draw a statistical comparison between the performance of models studied, they were trained using both Test-Train Split as well as using the K-Cross Folds, in addition to the oversampling of the minority class of Strikers.

3. PROBLEM DOMAIN AND DATASET DESCRIPTION:

The dataset used in this project is the Fifa 19 players data scraped from the sofifa.com [6][7]. The problem domain for this dataset can be broadly labelled as Sports Analytics, specifically, it can be associated with the football analytics with the data visualization and player position classification being the major focus of this project.
The dataset has 18207 rows with a vast number of 89 features, out of which 45 are categorical and 44 are numerical features.

4. Experimental Setup and Evaluation

4.1 Data Pre-processing

Introduction

Beginning with exploratory data analysis cleaning our data is very important. This cleaning involves importance steps such as understanding the size of data, separation of categorical features from numerical features, evaluating important features for analysing data visually and those important for training machine learning algorithms. Features that are redundant and are of no use are removed, although more features are removed as we proceed with feature engineering but at the initial phase features which are of no use can be removed according to our domain knowledge. We began by analysing the size of data which showed us we have 18207 rows and 89 columns. Amongst these 89 Columns, then we evaluate that we have 45 categorical features and 44 numerical features.

Handling Null values

Now amongst these features, we felt few were useless with respect to our project domain. Next, the important step is to cleaning null values from our data cause this step can be done by either removing rows, filling them with mean or putting the value of our own choice in that field. Looking at the null values we find that ‘Loaned From’ has 16943 null value which states that maximum entries are null in this particular column so we drop that column. Null values in the club field can be filled as no club as they are no so significant to our problem. Now when we observe ’Preferred Foot’ we can find that all the rows with preferred foot null i.e 48 rows other feature values for these particular rows are also null so we can remove all the rows with ’Preferred Foot’ null. Again looking at the list of features we concur that only 12 positions are null since the number is very small in comparison to the total number of rows we can also drop these rows. Next step is to check why player positional rating have null values we observe that there are 2025 rows where positional feature value is null and if we compare it with Position Goalkeeper we find that there are total 2025 goalkeeper so for goalkeeper there is no positional rating and that is appropriate because goalkeeper can’t have position rating so we can fill it with 0 .

Handling Categorical Variables

Categorical variables can not be directly submitted to a model if we want a good machine learning algorithm. So first we look at all the categorical variables we can find out that few of them can be converted to numerical values so we start with height and weight. We are given height in feet/inches which we convert into cms and weight in lbs is kept the same just the string lbs is removed from it using custom functions ‘height_conv’ and ‘weight_conv’.

The next two categorical variables describe the financial parameters such as wage and value and we are given these values in K and M so we normalize them using our custom function ‘moneyChanger’. Position Rating our given like LS - ‘93+2’ which means that player current potential at a position left striker is 93 but can increase by 2 points in future so for domain simplicity, we sum

these up using a custom function ‘ positionScoreConverter’.

If we look at the Position column and compare it with our domain knowledge we can reduce these positions by combining them for example 'CF', 'LF','RF’ that defines centre forward, left forward and right forward are basically strikers similar is the case with 'CDM', 'LCM', 'CM' that defines Centre Defence Mid, Left Centre Mid and Centre Mid all are midfielders, so for simplicity of our project we combine the similar positions and simplify the multiclass evaluation by reducing the total number of classes to 4 i.e. is Striker, Midfielder, Defender, Goalkeeper .

The next problem that we find is in the body type whereby evaluating unique values in each column we found that the feature 'Body Type' has some faulty and inconsistent values 'Messi', 'C. Ronaldo', 'Neymar' for some players so we filled them with appropriate values according to their body types.

4.2 Data Visualization

Introduction

Data Visualization is the process of displaying information so that the end-user can easily get insights from data which is not possible from simply reading textual data. Data visualization itself is a broad area to explore, many libraries are used for data visualization some of the most famous ones being matplotlib and Seaborn, etc.

Country-wise Player Participation

It is important to understand which country has the maximum number of players in FIFA. Such type of data can be used by selection teams to select more players from low participating countries, these statistics can also encourage low participating countries to send more players from their country put more funding in training players and providing them with appropriate facilities. We used bar plot for visualizing country vs player participation. This bar plot suggests that England has the maximum number of players in FIFA while countries like Berlin, Gabon, Honduras, Iran, and many others have very few players

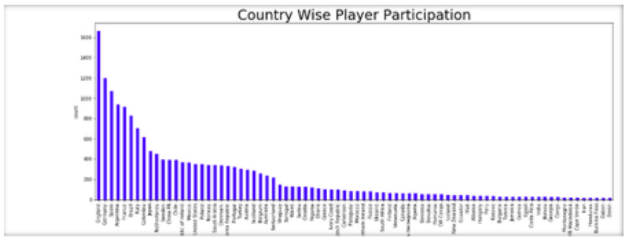

Figure 1– Country wise player participation

Most Preferred Foot Of The Player

While positioning players foot preference plays a crucial role because it helps the coach to assign a position to the player. Usually, right foot players are placed on the left side of the field because of the ease with which they can take the ball to another side of the field. Such type of graph can also help selectors in the selection of the players if they feel according to stats they have more right foot player than left they can choose more right foot players. In-game Right and Left foot players are not balanced meaning that there is always more right foot player than left foot players. This fact is clearly depicted my our data set we can see that 77% of our players are Right footed and others are Left footed

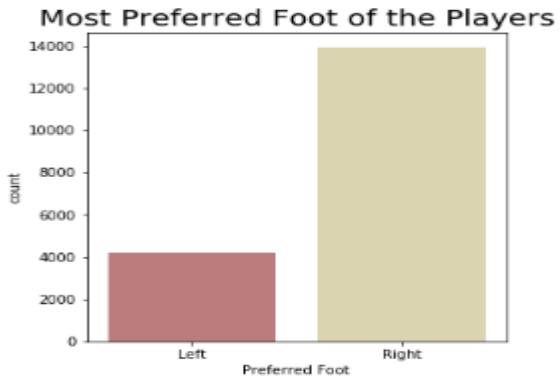

Figure2 – Most Preferred Foot of the Player Using counterplot

Age Distribution

A Player who is aged might have a lot of experience and can handle a lot of pressure but his physical fitness will be less in comparison to a young player so such age is a very important factor in the establishment of a good team. A good team has few young players and senior players while the maximum middle-aged player maintains a proper balance of players in the team plus

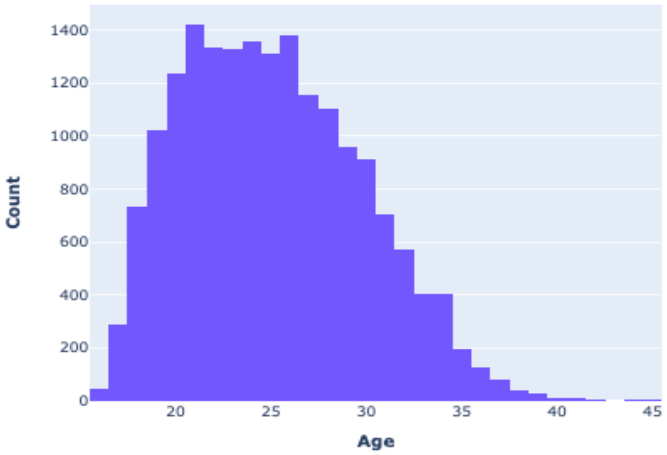

Figure 3 – Player age distribution using Histograph

when selecting new players these charts can be used by selectors to know which age group of people should they select more according to team requirement. We used Histogram to depict Age distribution of players. Our data depicts that the maximum player age 22 and more than 50% of players are between the age of 20 to 30.

Height Distribution

Height is an important characteristic in the game, a player with good height has a better heading accuracy but due to their height, their agility can decrease a bit. Running backs tend to be good for shorter it helps them to hide behind the line. Line backers and linemen are benefited by being taller for 2 reasons: it's easier to pack more muscle and weight onto a larger frame and it helps them to see and get a wider array of leverage on blockers. Pocket passing QBs are most successful when they're over 6′0 because it gives them better vision over the linemen down the field.

We used a counterplot to plot the height with player count. Maximum players are with a height of 6'0. Maximum players fall in the range of height 5'9 to

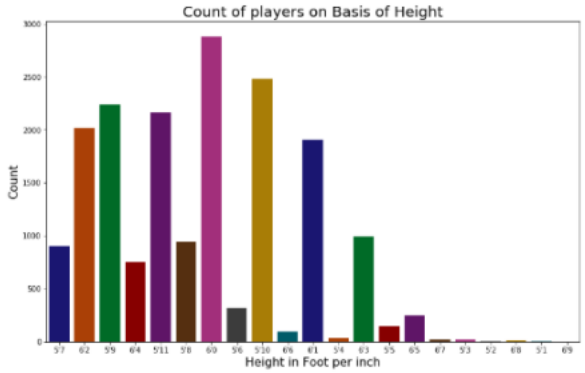

Figure 4 – Count of player on basis of height

Player Distribution According To Position

Our aim is to classify players on the basis of their position for this we will use machine learning models. Models are not capable of predicting values on their they need to be trained for that. To train data really well our data must be balanced. What does balanced Data mean? A model is trained where we get 90% accuracy. The result seems fantastic but when we closely observe data we find that the number of rows with positive records is 10 times more than the number of rows with negative records that is the reason it was classified so well so we analysing data before pre-processing is really important.

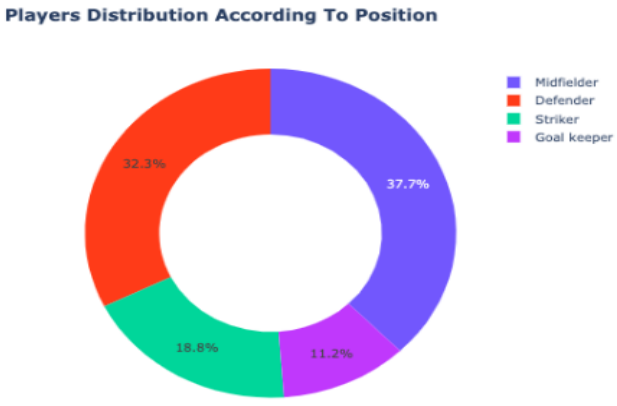

Figure 5 – Player Distribution According to Position

To analyse the data before passing it to the machine learning model we use pyplot graph object. In the figure,

we observe that there are 37% midfielders, 32.3% defender, 18.8% striker, and 11.2% goalkeeper. We can clearly see that goalkeepers and strikers are less in comparison to midfielders and defenders. We can leave goalkeepers as it because their features are quite distinct and they won't make our model biased. We will have to take care of strikers this can be done in 3 ways

Oversampling - the minority class is over-sampled by taking each minority class sample and introducing synthetic examples

Under-sampling - the majority class is reduced by taking out rows it

Balanced sampling - This involves the under sampling of the majority class and oversampling of minority class to make them approximately the same.
In our case, we will oversample the minority class but we will oversample the data at end to very how much oversampling can affect our model accuracies

4.3 Training Models

To train our models we used both the approaches the Train-Test split as well as evaluating our models using k-fold cross-validations.

Using Train Test Split

Train Test approach split the data into training and test set, the model is then trained on the training set, while the test set is used to check how well a model performs on the unseen data. A train test split is a good approach when the dataset is very large as it takes less computational power and time to run. [8].In our project, we used 77% data for training and 33% to do testing. The evaluation measures were calculated and evaluated on both the training as well as the test data.

K-Fold Cross Validations

Cross-Validations involve a resampling process to evaluate the machine learning models if data is limited. It reshuffles the dataset randomly splitting it into k-groups. For each unique group, each holds out-group as a test data and take the remaining groups as training data. Fit a model on the training data and evaluates it on test data. Then, it retains the evaluate score and discard the model and summarizes the evaluation scores of all models.
The value of k must be chosen carefully, The value for k as 10, has been found through experimentation to generally result in a model with low bias and a modest variance[9]. Thus, we have used k as 10 for evaluating our models

a) Tree-based classification

Tree-based algorithms are one of the most widely used supervised learning methods. Tree-based methods result in predictive models that have high stability, accuracy, and greater ease of interpretation[10]. Unlike, linear models, they also perform efficiently on mapping non-linear relationships as well. They work for both classification and regression problems. For this project, we will be using a decision tree classifier for training and building our model.

Decision Tree Classifier

Decision Tree classifier is a tree-based popular classification algorithm to understand and interpret. It can be used for classification as well as regression. It constructs a flow chart like structure where a node represents a feature, branch corresponds to a decision rule, and the outcome is represented by the leaf node [11].
Decision trees for classification work by deciding the Best Split, with an aim to remove any impurity. Any split is clean with no mix of positive and negative instances. Measures of impurity include calculating Gini Index and Entropy. The decision tree yielded the best result among the classifiers being studied without the significant feature engineering. However, it got the accuracy of 100 percent implying it perfectly fits on the training data, which is visible that it doesn't work perfectly on the test data, where its accuracy reduces to 81 percent.

b) Linear classification

Linear classifier makes classification decisions through the linear combination of the characteristics[12]. The object's characteristics known as feature values are represented in the machine in a vector known as the feature vector.

Logistic Regression

Logistic regression is a special case of linear regression and utilizes a statistical methodology for predicting classes. It predicts the probability of an event using a logit function by using a log of odds as the dependent variable. The dependent variable follows the Bernoulli Distribution and estimation is done through the maximum likelihood[13]

Naive Bayes

Naive Bayes belongs to the family of probabilistic classifiers and is based on Bayes theorem assuming the strong independence between the features. Naive Bayes classifiers are quite scalable, with a requirement of linear parameters in the number of variables in a learning problem. The maximum likelihood training takes linear time rather than iterative approximation[14].

For the project, we have used the Gaussian Naive Bayes, which assumes the Gaussian distribution for the real-valued attributes. It is the simples distribution to work with since only mean and standard deviation needs to be estimated from the training data[15]

C) Distance-Based

Distance-based classifiers make the decision on the basis of distance calculated from the target instances to the training instances, that is, measuring the similarity between data instances. Measures of the distance can be Euclidian, Manhattan, etc. In our project, we have used the K-Nearest Neighbour (KNN)

K-Nearest Neighbour

The KNN algorithm assumes that similar things are near to each other. It clubs the idea of proximity with the mathematics to calculate the point between points on a graph[16]. The straight-line Euclidean distance is the most popular choice for the distance measure.

Value of k is crucial to the outcome and decision is based on the vote, that is, k-NN returns the class label of the k nearest neighbours. The increasing value of k results in increased bias and decreased variance. Thus, 1NN perfectly separates the training data, but increased error rate on test data. We have used the value of N as 5 for training our KNN model as it gives a balanced performance on training as well as test data

d) Rule-Based

Rule-based classification scheme makes use of If-Then rules for the class prediction. Rule-based classification typically consists of the following components[17].

1) Rule Induction algorithm: It refers to the process of extracting relevant if-then rules using sequential covering algorithms[18, 19, 20]

2) Rule ranking Measures: This is used to prune off unnecessary rules and improve efficiency in the rule induction algorithm

3) Class Prediction Algorithm: It predicts the class of new algorithm using the If-Then rules given by the rule induction algorithm

For our project, we have used two Rule-Based classifiers, Dummy and Ripper.

Dummy Classifier

Dummy classifier uses simple rules to make predictions. It performs random guessing and we used it as a baseline for comparison with the other classifiers[21]. It is not a recommended classifier for real problems.

RIPPER

Another rule-based algorithm we used was the Ripper(repeated incremental pruning to produce error reduction), which is a proportional rule learner and an optimized version of IREP[22].

We used the Wittgenstein package for python to build rules on the subset of our dataset since applying it on the whole dataset was not computationally feasible owing to large number of features in the dataset.

e) Ensembles

Ensembles use multiple algorithms to obtain better performance as compared to constituent algorithms run alone[23]. It is based on the idea of creating a committee. Diversity in learning style, data provided can lead to overall improved performance. 'No Free Lunch', that is, no single algorithm is always the most accurate in all situations. Therefore, create a set of algorithms that combined have higher accuracy.

Types of ensembles:

Bootstrap aggregating (bagging)

Bagging involves sampling with replacement with each model in the ensemble vote with equal rights[24]. The majority voting used to predict the class. For instance, the Random forest combines different unpruned decision trees against a subset of the features utilizing the diversity in features to obtain a high classification accuracy.
Bagging is a variance-reduction technique and is used with high-variance models such as trees.

Boosting

Boosting turns a weak learner into a strong learner by focussing on the difficult to learn examples, by incrementing the weights and learning sequentially. Boosting a bias-reduction technique and is used with high-bias models such as linear classifiers or univariate decision trees (also called decision stumps). On average, Boosting provides a larger increase in accuracy than Bagging but is subject to overfitting if training data is noisy.

Gradient Boosting

Gradient boosting is a machine learning technique for classification and regression problems, which builds a model in the form of an ensemble of weak prediction models[25]. Gradient boosting typically trains model in a sequential and additive manner[26]. Algorithmically, it minimizes the loss function, so that test loss reaches minima. It does so by using gradients in the loss function[27]

4.4 Performance Evaluation

We evaluated the models using both Train-test split and using K-cross validations. In the first round, we trained our model on the data obtained after the initial feature selection and pre-processing done, discussed in section 3.1

	Decision Tree	Naive Bayes	KNN	Rule-Based	Gradient boosting
Training Data	100	45	76	29	86
Test Data	81	44	63	28	86
K-Fold	78	45	56	29	84

Table 1: Accuracy scores for algorithms in percent

	0 (Goal Keepers)	1 (Defenders)	2 (Midfielders)	3 (Strikers)
Decision Tree	100	84	73	68
Naive Bayes	100	38	51	32
KNN	74	57	54	49
Rule-Based	11	31	37	19
Gradient boosting	100	89	78	78

Table 2: Precision scores for four classes in percent

From table 1, it's clear that models are not underfitting or overfitting since there is no significant difference between there accuracies on training and test data. The decision tree, at first, seems to be massively overfitting looking at its 100% accuracy on the training data, however, on being evaluated on the unseen test data it gave a reasonable accuracy of 81%.

High Accuracy scores obtained for our models cannot be treated as the true measure for the performance of our classifiers because Goalkeepers is a very distinct class and is predicted by very high precision and recall by the majority of the classifiers. Therefore, in order to evaluate the performance of our classifiers, we need to look at the precision and recall scores for the other three classes namely: 1(Defenders), 2(Midfielders), 3(Strikers).

	0 (Goal Keepers)	1 (Defenders)	2 (Midfielders)	3 (Strikers)
Decision Tree	100	87	75	73
Naive Bayes	95	91	11	17
KNN	62	69	67	44
Rule-Based	11	33	36	16
Gradient boosting	100	88	80	75

Table 3: Recall scores for four classes in percent

The Recall and Precision scores from Table 2 and Table 3 clearly indicate that all classes except goalkeepers are misclassified by all the algorithms. The misclassification is more evident in class 2(Midfielders) and class 3(Strikers). The worst performing Algorithms are rule-based and Naive Bayes. This may be attributed to the presence of a large number of weakly correlated features and features with a similar correlation.

Feature Engineering (Using Correlation)

We evaluated the correlation between Position, which is our target variable and other features and discovered that features such as RF, LF, RS, LS, CF and so on have quite similar correlations, and therefore we combined them as they represented the similar information.

In addition to this, we removed features such as Value, Wage, Jersey Number, Overall, Preferred Foot, etc. which has a very low correlation with Position. The number of features was reduced to 45 after analysing their correlation with the Position.

	Decision Tree	Naive Bayes	KNN	Rule-Based	Gradient boosting
Training Data	100	72	90	30	86
Test Data	81	71	86	28	85
K-Fold	79	71	85	29	82

Table 4: Accuracy scores post correlation analysis in percent

There is a significant improvement in the accuracy scores of two algorithms Naive Bayes and KNN. Naive Bayes assumes independence among the features, merging of dependent features can be attributed to the improvement in the performance of Naive Bayes. For KNN, the improvement is because of the removal of wage and value feature, who had very low correlation and were of higher magnitude, thereby, inversely impacting the performance of KNN which is a distance-based model.

	0 (Goal Keepers)	1 (Defenders)	2 (Midfielders)	3 (Strikers)
Decision Tree	100	85	73	70
Naive Bayes	100	78	68	58
KNN	100	90	79	81
Rule-Based	11	32	38	20
Gradient boosting	100	87	77	78

Table 5: Precision scores post correlation analysis

Also, there is a considerable improvement in the precision and recall scores for the KNN and Naive Bayes, as reflected in Table 5 and Table 6, especially for the Defenders, Midfielders and Strikers. KNN now is now able to predict Goalkeeper class with 100 percent precision.

	0 (Goal Keepers)	1 (Defenders)	2 (Midfielders)	3 (Strikers)
Decision Tree	100	84	73	71
Naive Bayes	100	77	47	93
KNN	100	89	83	75
Rule-Based	11	32	38	20
Gradient boosting	100	86	79	75

Table 6: Recall scores post correlation analysis in percent

The improvement in the performance of algorithms signifies the importance of feature selection and feature engineering.

Visualizing and removing outliers

Introduction

Observation in statistics that are far removed from the normalized distribution observation in any data set. Outliers occur because the data collection had a mistake in the collection of the data, the data was corrupt and it's

not always that the data is wrong, it's just that it can be away from the normalized data in that particular category. It is very important that we have a close understanding of our dataset to identify an outlier. Models can have a drastic effect if outliers are overlooked due to errors of omission or being far from normal statistical distribution [28] There are several methods used by a data scientist to find outliers, for the our problem we have used scatter plot

Scatter Plot

Scatter plots are used to plot data points on a horizontal and a vertical axis in the attempt to show how much one variable is affected by another. Every row from the data table is shown through a marker whose position depends on values in the columns set on the X and Y axes.[**].In our case, we are plotting Finishing on X-axis whereas Positioning on Y-axis because of our domain knowledge plus the area with the maximum crowd on the graph helps us to find the right range finishing range for players at each position. The scatter plot we get for all positions is as follows the plot

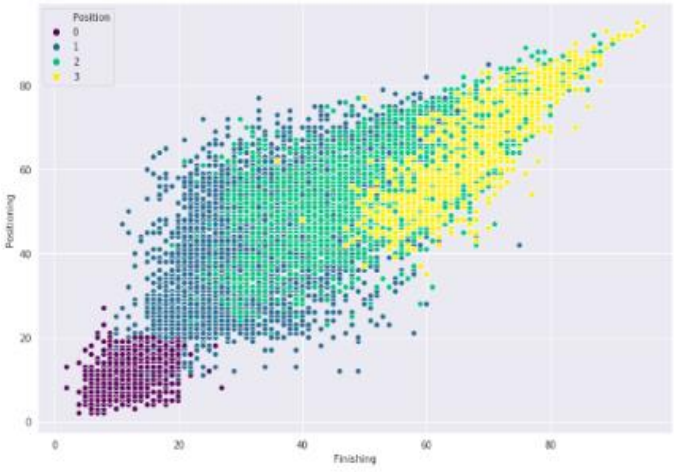

Figure 6 – Scatterplot for Positioning and Finishing before removing outliers

from the figure, we can analyse that for defender Positioning less than 61 and finishing greater than 31 is an outlier, for midfielder finishing greater than 61 /positioning less than 81 and finishing greater than 66 /positioning greater than 71, for striker finishing less than 46 and finishing less than 61 /positioning greater than 71 are outliers . After removing the outliers we get the following scatterplot

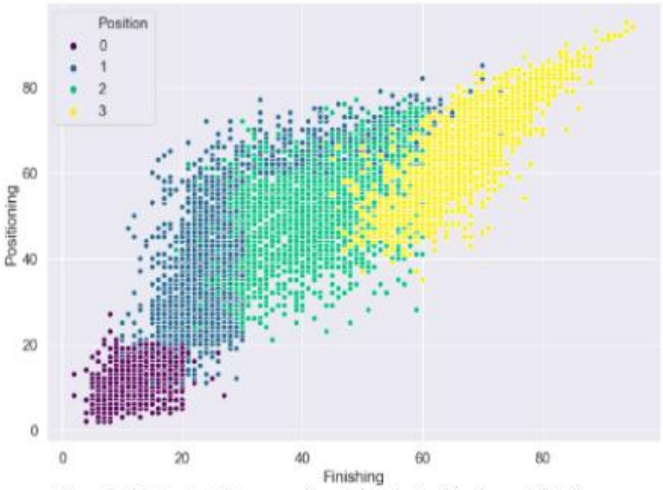

Figure 7 – Scatterplot after removing outliers for Positioning and Finishing

Performance Evaluation after removing outliers and performing a balancing

Removing outliers must improve the performance of our algorithms to make these algorithms, even more, we will perform oversampling for the about which we talked in section 3.2 we observe how the imbalanced data is balanced using sampling

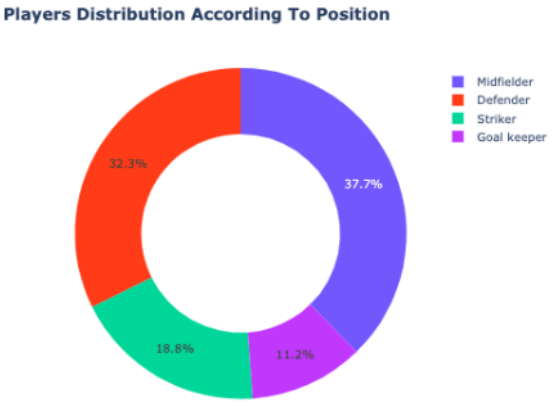

Figure 8 – Player Distribution According to Position Prior To Balancing

as discussed in section 3.2 we choose to oversample over other sampling methods because we would like to keep a maximum of original data and minimum synthetically created data so we have the new distribution as shown in figure 9.

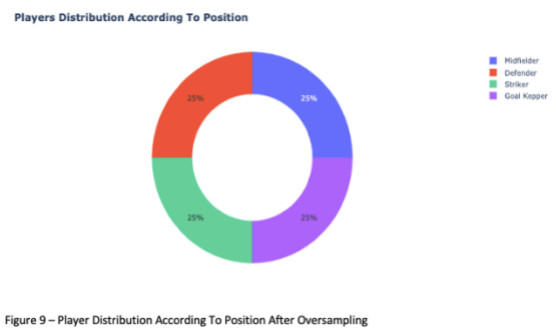

Figure 9 – Player Distribution According To Position After Oversampling

After these final steps of improving the data that we are feeding to our classifiers, we note down the final observations related to our model accuracies, precisions and recalls

	Decision Tree	Naive Bayes	KNN	Rule-Based	Gradient boosting
Training Data	100	79	91	27	93
Test Data	90	80	93	27	92
K-Fold	92	82	92	23	93

Table 7: Accuracy scores post outliers removal and sampling

We can clearly say that there has been an increase in accuracy for all the classifiers except rule-based. Now let us evaluate if classifiers evaluated each class properly this can be done by evaluating precision and recall

	0 (Goal Keepers)	1 (Defenders)	2 (Midfielders)	3 (Strikers)
Decision Tree	100	91	86	90
Naive Bayes	100	81	77	79
KNN	100	90	88	87
Rule-Based	23	24	23	23
Gradient boosting	100	92	85	93

Table 8: Precision scores for four classes in percent

We can concur that our precision for each class have improved and the same results can be found in a recall

	0 (Goal Keepers)	1 (Defenders)	2 (Midfielders)	3 (Strikers)
Decision Tree	100	93	83	92
Naive Bayes	100	76	58	96
KNN	100	96	76	93
Rule-Based	23	23	23	23
Gradient boosting	100	94	87	91

Table 9: Recall scores for four classes in percent

4.5 Statistical Difference between models

To test whether there is a significant difference between the performances of algorithms being studied, we chose Freidman test, Wilcoxon test could also have been used but we had only one dataset [29].

Friedman's test

Friedman test requires multiple datasets being trained and modelled using different machine learning algorithms. In our case, we used the accuracy scores from the train-test split, K-Cross validations and Oversampling to have three different instances of the same algorithm[30] , ranks are mentioned for each dataset and at the end an average rank is taken to perform Friedman test . In case of tie for a position an average is taken between the two ranking positions that need to be occupied

	Decision Tree	Naive Bayes	KNN	Rule-Based	Gradient boosting
Split	90(3)	80(4)	93(1)	27(5)	92(2)
K-fold	89(3)	79(4)	90(2)	26(5)	91(5)
Oversampling	92(2.5)	82(4)	92(2.5)	23(5)	93(1)
Avg Rank	2.8	4	2.2	5	1

According to our values, Friedman statistics s 2.6. The critical value for $k = 5$ and $n = 3$ at the $\alpha = 0.05$ level is 8.53, so we can reject the null hypothesis and say that there is a significant difference in the performance of five algorithms under consideration.

Since there is a significant difference, therefore we can calculate the critical difference and construct the Nemenyi test to graphically depict the difference between the performance of the algorithms. Figure 10, represents the Nemenyi Diagram with Ensemble-based gradient boosting as the best performer followed by KNN and Rule-based Dummy classifier being the worst performer.

```
Critical Difference: 3.5215410914058634
```

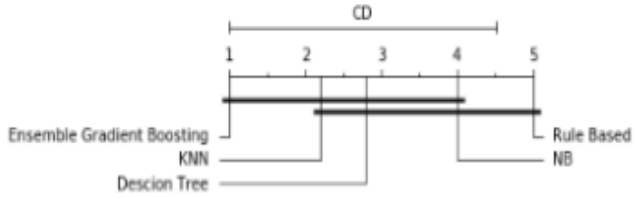

Figure 10 – Critical Difference and Nemenyi Diagram

ROC (Receiver Operating Characteristics) Curves and Area under Curve (AUC).

ROC curves also helps in the performance evaluation for the classification problems at the different threshold values. It tells how well model is able to differentiate between the two classes. Higher the AUC (Area Under Curve) better the model. ROC curves are plotted with TPR on y-axis and FPR on the x-axis.

ROC curve is constructed for differentiating between the binary classes. Therefore, to construct ROC curves for our Multiclass problem, we had to use oneVsRest classifier to separate out one class and evaluate it with the remaining classes. Since, we had four classes there would be 4 ROC curve for each model evaluated resulting in 20 ROC curves. Therefore, in the report we have only included the ROC curves for our best performing ensemble based gradient boosting classifier.

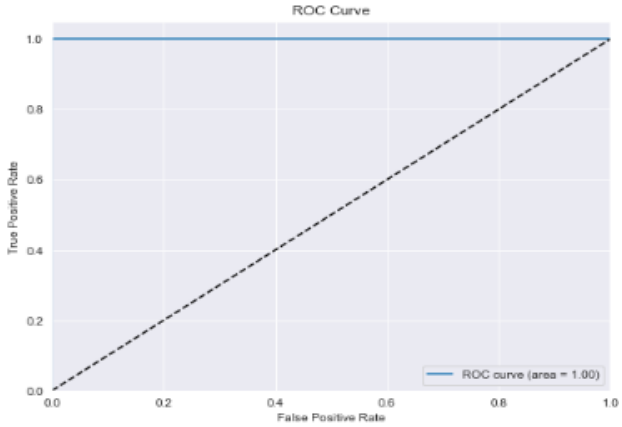

Figure 11: ROC curve for Goalkeepers

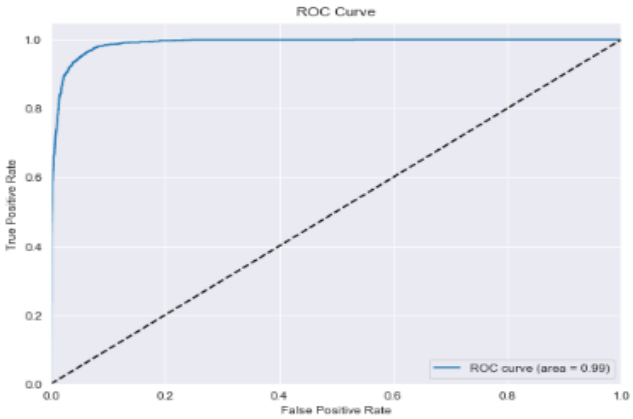

Figure 12: ROC curve for Defenders

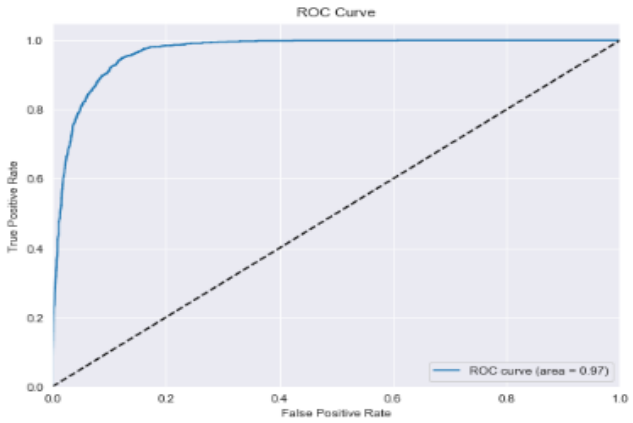

Figure 13: ROC curve for Midfielders

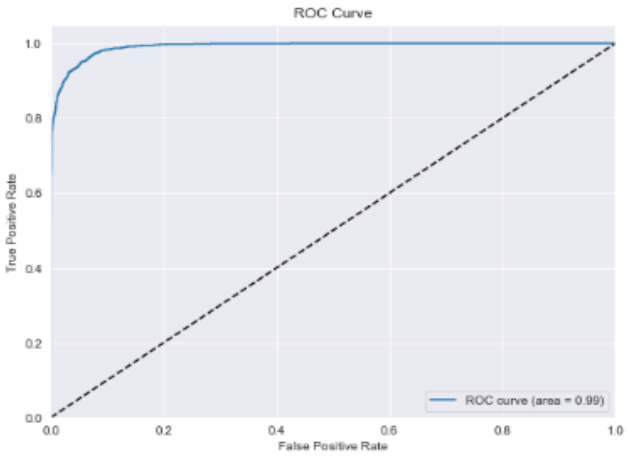

Figure 14: ROC curve for Midfielders

Figures 11, 12, 13 and 14 represents ROC curves for Goalkeepers (class 0), Defenders (class 1), Midfielders (class 2), and Strikers (class 3) respectively.

5. CONCLUSION AND FUTURE WORK

The dataset from the popular football game Fifa 19 was taken for the football analytics and classifying player positions based on their playing attributes. After doing Data Analytics, handling categorical features, using feature engineering, removing outliers, performing oversampling and doing statistical testing bring us to the following conclusion -:

1. Eliminating redundant features and outliers significantly improved the performance of algorithms in terms of accuracy, precision, and recall.
2. There was a significant difference between the performance of algorithms under consideration as stated by the Friedman Test
3. Ensemble-based algorithm Gradient Boosting performed best for our classification task
4. Balanced data is important for training an accurate model, as oversampling the minority class of Strikers yielded better performance from the algorithms.
5. However, there was no significant difference between performance metrics for the models constructed using Train-Test split and K-Cross fold validations.

Player position classification can act as a base and can open up the numerous opportunities for future tasks in the domain of sports analytics, as models trained using Fifa 19 dataset can be extended to the future versions of Fifa to correctly classify the player positions just by using the significant features.

Additionally, once the player positions are correctly classified, linear regression can be implied to predict the overall rating of the players.

There is also plenty of scopes extending these models to generalize which team has the best overall squad and what can be the best playing 11 in the world. The approach used in this project can also be extended to other popular games like NBA, NHL, and Cricket.

REFERENCES

[1]. https://en.m.wikipedia.org/wiki/FIFA_(video_game_series)

[2]. http://www.montrealgazette.com/business/SPORTS+FIFA+Soccer+Stores+Throughout+North+America+Available+Europe+Asia+Beginning+September/3590367/story.html

[3]. https://www.gamesindustry.biz/articles/2018-09-05-fifa-18-sells-over-24-million-copies

[4]. https://www.businesswire.com/news/home/20101104006782/en

[5]. https://en.wikipedia.org/wiki/FIFA_19

[6]. https://www.kaggle.com/karangadiya/fifa19

[7]. https://sofifa.com/

[8]. https://medium.com/@eijaz/holdout-vs-cross-validation-in-machine-learning-7637112d3f8f

[9]. https://machinelearningmastery.com/k-fold-cross-validation/

[10]. https://www.datacamp.com/community/tutorials/decision-tree-classification-python

[11]. https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/

[12]. https://en.wikipedia.org/wiki/Linear_classifier

[13]. https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python

[14]. https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[15]. https://machinelearningmastery.com/naive-bayes-for-machine-learning/

[16]. https://www.quora.com/What-is-the-difference-between-distance-based-classifiers-and-SVM

[17]. https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-39940-9_559
[18]. Clark P. and Niblett T. The CN2 induction algorithm. Mach. Learn., 3(4):261–283, 1989.'
[19]. Cohen W. Fast effective rule induction. In Proc. 12th Int. Conf. on Machine Learning, 1995, pp. 115–123.
[20]. Furnkranz J. and Widmer G. Incremental reduced error pruning. In Proc. 11th Int. Conf. on Machine Learning, 1994, pp. 70–77
[21]. https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html
[22]. Agah, Arvin (2013). Medical Applications of Artificial Intelligence. CRC Press. ISBN 9781439884331. Retrieved 13 August 2017
[23]. Opitz, D.; Maclin, R. (1999). "Popular ensemble methods: An empirical study". Journal of Artificial Intelligence Research. 11: 169–198.
[24]. https://en.wikipedia.org/wiki/Ensemble_learning
[25]. https://en.wikipedia.org › wiki › Gradient_boosting
[26]. https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d
[27]. https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab
[28]. https://towardsdatascience.com/a-brief-overview-of-outlier-detection-techniques-1e0b2c19e561
[29]. https://en.wikipedia.org/wiki/Statistical_hypothesis_testing
[30]. https://en.wikipedia.org/wiki/Friedman_test