Machine learning applied to a UFC match database

Preprint · January 2021

DOI: 10.6084/m9.figshare.13567184.v1

1 author:

Erick Axel Martinez Rios
Tecnológico de Monterrey
2 PUBLICATIONS **1** CITATION

SEE PROFILE

Instituto Tecnológico y de Estudios Superiores de Monterrey

Author:

Erick Axel Martinez Ríos A01331212

Project

“Machine learning applied to a UFC match database”

Index

1. Introduction

Machine Learning (ML) is a subset of artificial intelligence that focuses on the development of techniques or algorithms that can best represent a set of data. Unlike the classical programming approach in which an algorithm can be explicitly coded using a set of known characteristics, ML uses a set of data to generate the algorithm. The comparison of both paradigms is illustrated in Figure 1. Machine learning techniques are divided based on the types of learning that they can perform on a given dataset. The four types of learning that are commonly studied in the literature are supervised, unsupervised, semi-supervised, and reinforcement learning, with the most common ones being supervised and unsupervised learning. The supervised learning algorithm is mainly used to solve classification problems or to solve regression problems to make forecasting tasks. This is done by feeding the ML technique with input data that has a particular output response. In the case of a classification problem, the output variables are categorical or discrete while in regression problems the output is a continuous variable. On the other hand, the unsupervised techniques do not have a particular output response, but it is intended to make clustering of unlabeled information to find patterns on the given data [1].

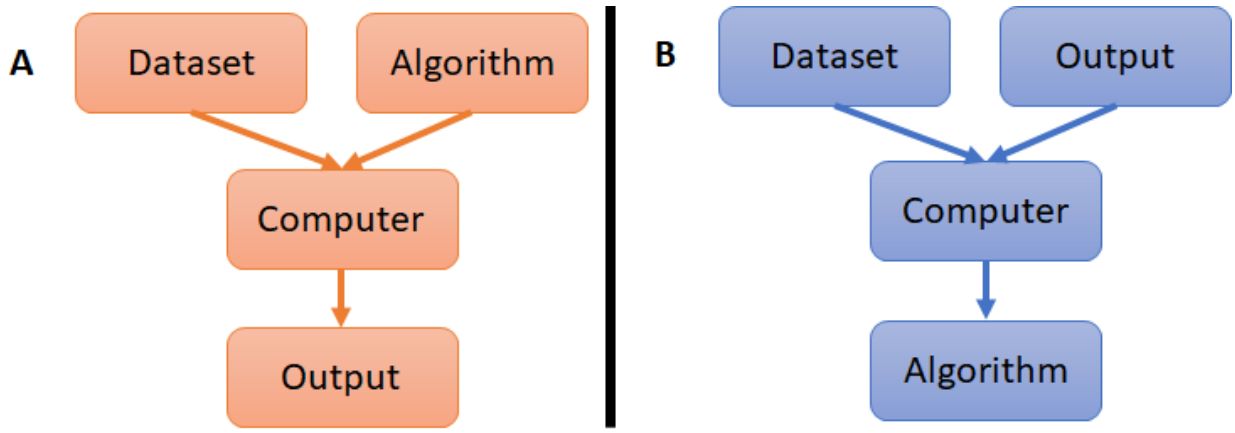

Figure 1 - Classical programming (A) versus machine learning paradigm (B) [1].

One important step in machine learning is the algorithm validation or performance evaluation. To evaluate the algorithm, the data used to create the algorithm is split into two types of subsets: the training set and the validation set. The training set is used to generate the algorithm using any kind of machine learning technique and the validation set is used to measure the performance of the given model. To measure the performance of different types of metrics must be used. In the case of the classification algorithm, the confusion matrix is computed to extract metrics such as accuracy, precision, and recall. On the other hand, in regression problems, the mean square error (MSE) is often computed during the validation stage. The characteristics and development of machine learning techniques have promoted its use in different types of fields generating a great variety of applications. For instance, machine learning has promoted the development of

algorithms for image recognition, medical diagnosis, text classification, speech recognition, product recommendations in entertainment web pages like Amazon or Netflix, and self-driving cars [2].

Taking into consideration the characteristics of machine learning techniques and the great diversity of applications that have been promoted by it, the aim of the present work is the implementation of machine learning techniques to generate a model that can predict the winner in a UFC (Ultimate Fighting Championship) based on the statistics of the fighters by employing the available database presented in [3]. The UFC is the largest mixed martial arts (MMA) promotion company in the world, and it has some of the highest-level fighters. The goal of UFC is to find the ultimate fighter by hosting a one-night tournament between the top fighters in disciplines such as karate, jiu-jitsu, boxing, kickboxing, grappling, wrestling, and sumo. The fights have an average duration of 5 minutes and the better-known fighter is label as the red fighter. There are several forms to win the fight like a knockout, submission, or by decision [4].

This work is organized as follows, Section II presents the problem and the proposed methodology to solve it, Section III present de description and the analysis of the database, Section IV presents a brief description of the machine learning techniques used to solve the classification task, Section V shows the results and analysis of each classification technique that were used on the database, and Section VI presents the conclusions of this study.

2. Problem Statement and Methodology

The challenge consists of developing a prediction model that could determine the outcome of a UFC match based on a database with information related to the fighter statistics [3]. Since the target variable is categorical and has three possible outcomes Red, and Blue which refers to the color of the fighter that won the match, and a third-class label as Draw. Therefore, this challenge can be labeled as a classification problem in which three discrete outcomes could be predicted based on a particular set of inputs or in this case fight statistics. Nevertheless, the database is of high dimensionality due to the number of variables that it has, it has a great number of missing values and the output classes are not equally distributed or in other words the database is unbalanced, therefore it is important to establish a methodology that can overcome those issues and provide an algorithm that it is easy to train, interpret, and provides good performance when evaluating it with new samples.

In machine learning and statistics, the models are based on assumptions, therefore it is complicated to find a single model that can represent all the reality. The No Free Lunch Theorem [5], states that there is no single model that works best for all possible problems. That is why it is common to try or test different models and find the one that works best for a problem (validation). Therefore, there will be an iterative process between the “Model Generation” and “Model Validation” in which different types of techniques will be tested to determine the proper technique to classify the winner in a UFC match. The general methodology used in machine learning problems is presented in Figure 2.

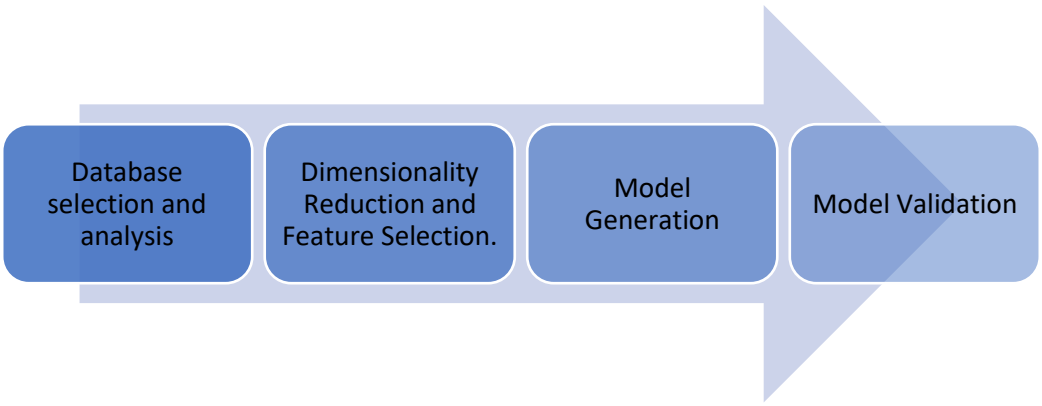

Figure 2. Methodology to implement machine learning techniques.

3. Dataset Description and Analysis

The database is a list of all the UFC fights until the year 2019 and it was obtained from the UFC stats website. Each row in the database has information related to the fight details of fighter Red and fighter Blue. The stats include damage done by the red or blue fighter on the opponent and the damage done by the opponent on the fighter (represented by 'opp' in the columns). The target or response variable is one, and it is named Winner, it is a categorical variable with three labels Blue, Red, and Draw. Therefore, a supervised learning technique used for classification tasks should be applied to the data. The database has a total of 5144 samples and 144 variables which description is presented in Table 1 [3].

Table 1. UFC Database Variables Description.

Variable	Description
R_ and B_	Prefix signifies red and blue corner fighter stats respectively
opp	Containing columns is the average of damage done by the opponent on the fighter
KD	It is the number of knockdowns
SIG_STR	It is no. of significant strikes 'landed of attempted'
SIG_STR_pct	It is the significant strikes percentage
TOTAL_STR	It is total strikes 'landed of attempted'
TD	It is the no. of takedowns
TD_pct	It is the takedown percentages
SUB_ATT	It is the no. of submission attempts
PASS	It is no. times the guard was passed?
REV	It is the no. of Reversals landed
HEAD	It is the no. of significant strikes to the head 'landed of attempted'
BODY	It is the no. of significant strikes to the body 'landed of attempted'
CLINCH	It is no. of significant strikes in the clinch 'landed of attempted'
GROUND	It is the no. of significant strikes on the ground 'landed of attempted'
win_by	It is the method of win
last_round	It is the last round of the fight (ex. if it was a KO in 1st, then this will be 1)

last_round_time	It is when the fight ended in the last round
Format	It is the format of the fight (3 rounds, 5 rounds, etc.)
Referee	It is the name of the Ref
Date	It is the date of the fight
location	It is the location in which the event took place
Fight_type	It is which weight class and whether it's a title bout or not
Winner	It is the winner of the fight
Stance	It is the stance of the fighter (orthodox, southpaw, etc.)
Height_cms	It is the height in centimeter
Reach_cms	It is the reach of the fighter (arm span) in centimeter
Weight_lbs	It is the weight of the fighter in pounds (lbs)
age	It is the age of the fighter
title_bout	A boolean value of whether it is a title fight or not
weight_class	It is which weight class the fight is in (Bantamweight, heavyweight, Women's flyweight, etc.)
no_of_rounds	It is the number of rounds the fight was scheduled for
current_lose_streak	It is the count of current concurrent losses of the fighter
current_win_streak	It is the count of current concurrent wins of the fighter
draw	It is the number of draws in the fighter's UFC career
wins	It is the number of wins in the fighter's UFC career
losses	It is the number of losses in the fighter's UFC career
total_rounds_fought	It is the average of total rounds fought by the fighter
total_time_fought(seconds)	It is the count of total time spent fighting in seconds
total_title_bouts	It is the total number of title bouts taken part in by the fighter
win_by_Decision_Majority	It is the number of wins by majority judge's decision in the fighter's UFC career
win_by_Decision_Split	It is the number of wins by split judge's decision in the fighter's UFC career
win_by_Decision_Unanimous	It is the number of wins by unanimous judge's decision in the fighter's UFC career
win_by_KO/TKO	It is the number of wins by knockout in the fighter's UFC career
win_by_Submission	It is the number of wins by submission in the fighter's UFC career
win_by_TKO_Doctor_Stoppage	It is the number of wins by doctor stoppage in the fighter's UFC career

Exploratory Data Analysis

Originally the dataset had a total of 5144 samples with 144 features and 1 target variable. Nonetheless, some features or columns had missing values. In this case, the rows that had missing values were removed from the dataset which leads to a total of 3202 samples. On the other hand, the target variable (Winner) had three unbalanced classes. The classes and the percentage of them in the final dataset can be appreciated in Table 2 and Figure 3 below.

Table 2. Classes of the target variable Winner.

Target Variable (Winner)	Number of Samples	Percentage
Blue	1135	35.4466%
Draw	51	1.5928%
Red	2016	62.9606%

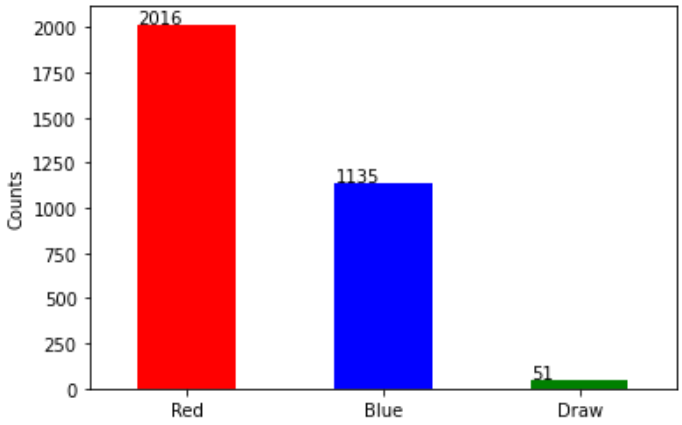

Figure 3. Bar plot representation of the number of samples for each class.

Since the dataset is unbalanced, it is important to split the data before fitting any machine learning model. For this purpose, 80% of the whole data set was used for training the models while the other 20% was used for testing. The data set had 9 categorical variables that were related to the location, name of the fighters, referee's name, date of the fight, fighter stance, weight class, and title bout. Table 3 shows the names of the categorical variables and the number of categories that each of the variables had.

Table 3. Categorical Variables in the UFC database.

Categorical Variable	Number of categories
R_fighter	875
B_fighter	1048
Referee	166
Date	445
Location	146
Weight_class	13
Title_bout	2
B_Stance	4
R_Stance	4

By looking at Table 3, it is possible to notice that some categorical variables have a great number of categories within them. To manage these variables and use them as input variables in the machine learning classification techniques, Label Encoding was performed in the Title_bout variable since it only has two categories (True or False). On the other hand, for the Weight_class, B_Stance, and R_Stance variables One Hot Encoding was performed on them which generated a total of four dummy variables for the R_Stance and B_Stance variables and 13 dummy variables for the Weight_class. In the case of the variables that contained information related to the names of the Blue or Red Fighters, the referee's name, the location, and the date of the match it was determined to remove them from the dataset due to the number of categories that they had to perform either Label Encoding or One Hot Encoding. After performing the Encoding of the categorical variables, the total size of the dataset was 3202 samples and 158 variables. Another

aspect that is important to consider, is the number of variables or features that are employed to generate the models. In this case, the dataset had a great number of variables which some of them could provide redundancy while classifying the output variable. To determine which variables could be used a Feature Selection process was performed on the database.

Feature Selection and Reduction

To avoid a high dimensional model that uses the 158 input variables, Principal Component Analysis (PCA) was used as a feature selection technique to take the variables that are going to be used to train the proposed models and discard the rest. The idea behind PCA is to generate a new coordinate system that transforms a set of features that are possibly correlated into a new set of features that has no linear correlation, this new set of features are known as principal components. The number of components that can be generated with PCA is equal to the number of input variables. Nonetheless, the first component of PCA is the direction with maximum variance. The end goal of PCA is to summarize the underlying variance structure of a large set of variables with a few linear combinations of those variables, in other words, find the best subspace in which it is possible to visualize the data.

PCA was implanted using the sklearn library of Python. It is important to mention that PCA requires that the data must be standardized before applying the technique, for that purpose, the Standard Scaler of the same library was used. Since the goal of using machine learning is to solve a classification problem, only the first two components of the PCA are analyzed. The variance percentage of the first two components is shown in Table 4. On the other hand, Figure 4 shows the percent of variance explained for all the principal components.

Table 4 - Variance Percentage of the first two principal components.

PCA Component	Variance Percentage
PC0	**14.1075%**
PC1	**8.8113%**
Total	**22.9189%**

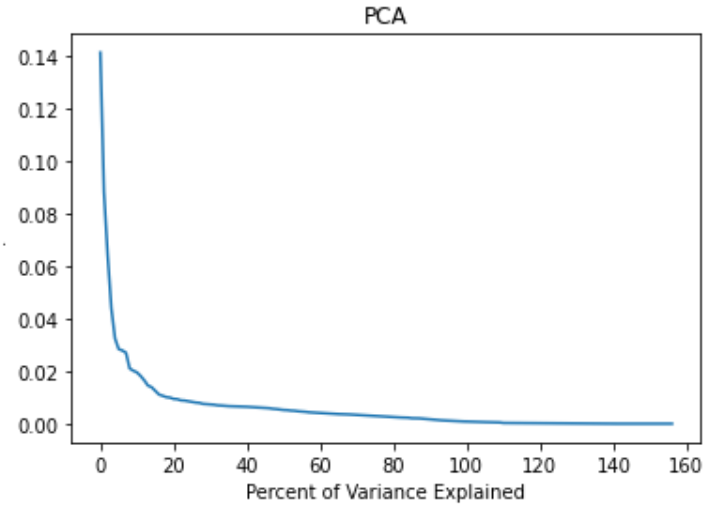

Figure 4. Percent of Variance explained

Since the first component of PCA has a greater variance, the loading scores of each of the variables for the first PCA were computed using the *sklearn* package. The higher the absolute value of the loading score of each variable the greater its impact on the first component. The computed values were sorted from the highest value to the lowest. By making this procedure, the 15 variables with the highest loading score are shown in Figure 5. These 15 variables were filtered from the original dataset to generate reduced models and apply the machine learning classification techniques.

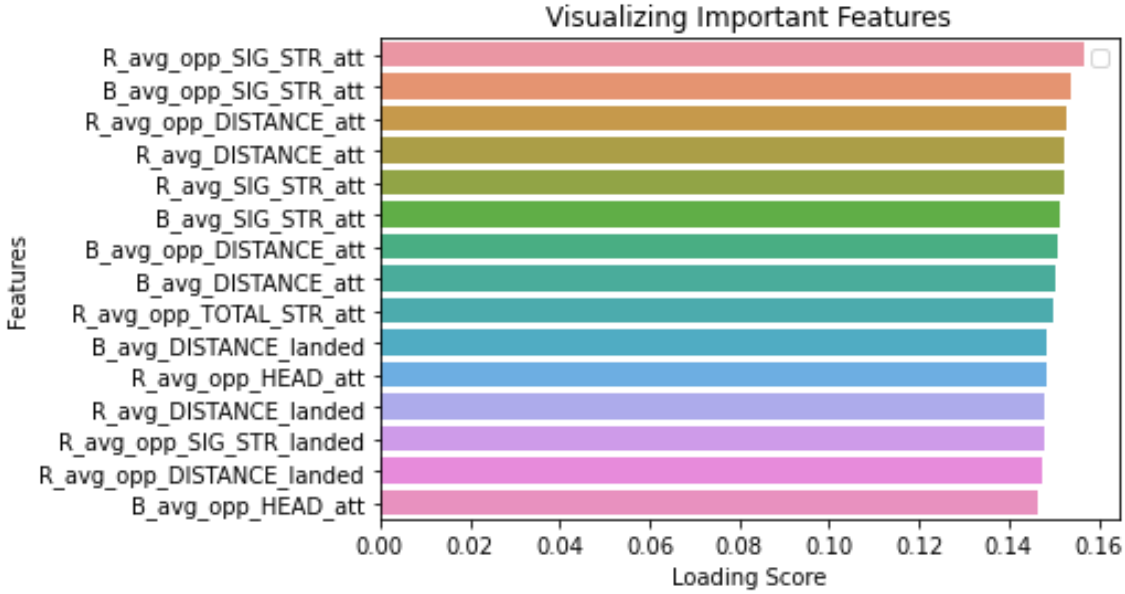

Figure 5. Variables with the highest loading score in the first principal component

A second approach that was tested to reduce the variables of the database was through the execution of the Random Forest Classifier and selecting the most important features according to the score given by the algorithm. This score is calculated based on the Gini importance also known as the Gini Impurity Index of each of the variables. The most important variables resultant from this procedure are shown in Figure 6 by using the sklearn package from Python, for this package the highest the score the highest the importance of the feature. Like in the PCA procedure, only the 15 variables with the highest score were selected to train the classification techniques.

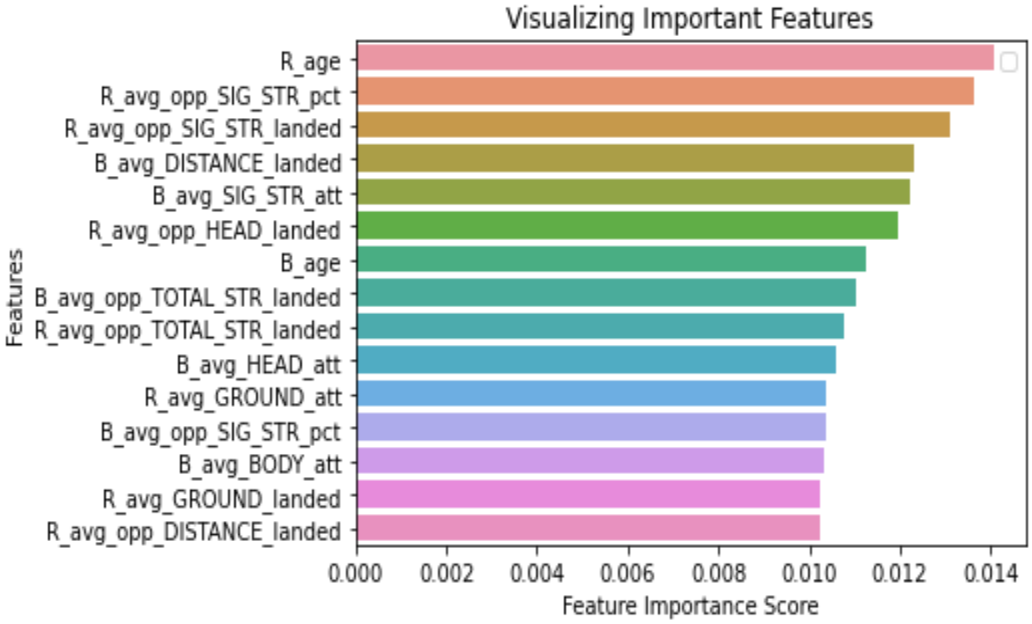

Figure 6. Relative feature importance score according to the Gini importance executing the Random Forest.

4. Proposed Techniques

This section makes a brief description and characteristics of the machine learning techniques that were tested on the UFC database.

Decision Trees

Decision trees are binary trees that are composed of nodes and leaves also known as terminal nodes. Each terminal node has two branches that represent the decision rule. A sample enters the tree at the root node place at the top. In each node of the tree, a decision takes place based on the value of the input feature. If the input variable is continuous, the decision is taken whether its value is greater or smaller than a threshold value. Based on the conditions that are in each leaf or node, the sample travels the tree until it reaches the end of the tree and it is assigned the label that the final leaf has. The time complexity of the decision tree is in the function of the number of samples and features of the data set. The decision tree is a nonparametric method, this implies that it does not depend on probability distribution assumptions [6]. An illustration of the decision tree can be appreciated in Figure 7.

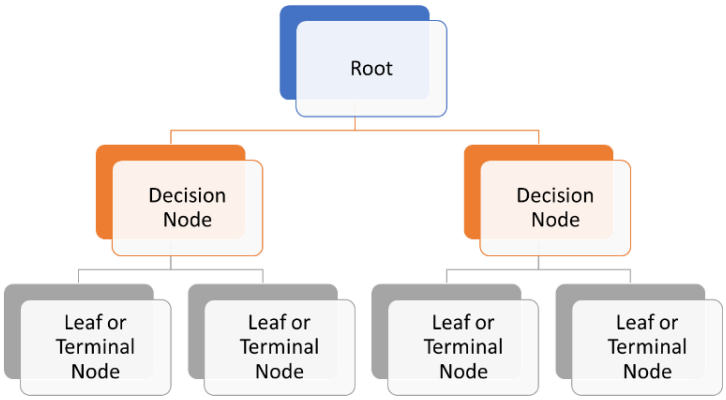

Figure 7. Schematic representation of a decision tree [7].

How decision trees are trained is by first selecting the best attributes of the data set; this procedure is called Attribute Selection Measures (ASM). Each of the attributes that were selected is converted into decision nodes that break the dataset into smaller subsets. The process is repeated for each child until there are no more remaining attributes. Some of the most common methods for attribute selection are the Information Gain, Gain Ratio, and Gini Index.

K - Nearest Neighbor

The basic idea behind the K-Nearest Neighbor (KNN) algorithm is that it assumes that similar things exist in proximity or similar things are near each other. In the KNN algorithm, the K stands for the number of the nearest neighbors that are going to be evaluated. The number of neighbors is the core behind the algorithm since that number is the one that helps in the decision of how to classify a new sample. The value of K is generally odd when facing binary classifications problems. When a new sample is going to classify the algorithm computes the distance between the new sample and the K closest points and finally, the class with the most votes is taken as the prediction. To calculate the distance between the new sample and the nearest neighbors, there exist different types of distance measures like the Euclidean Distance, Hamming Distance, Manhattan Distance, and Minkowski distance. Figure 8 describes graphically the process of the KNN algorithm [8].

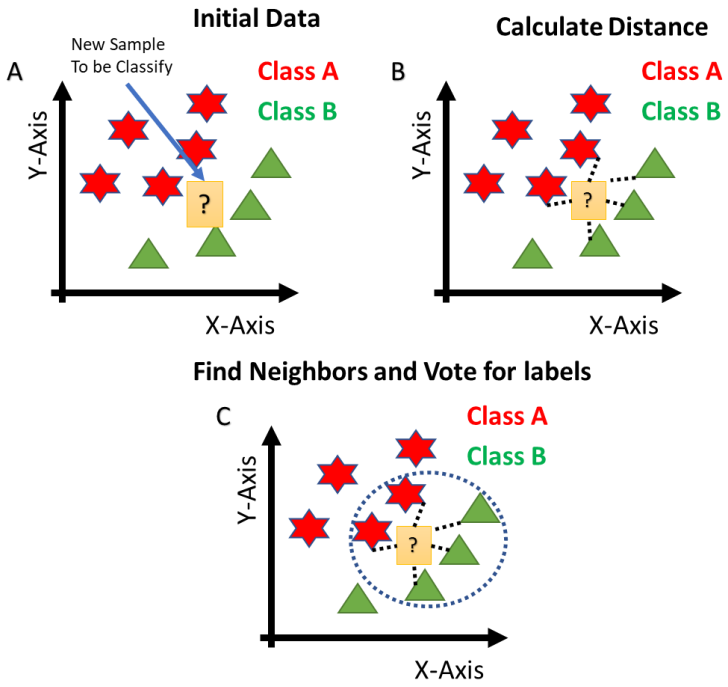

Figure 8. New sample to classify (A), Calculate the distance between the new data point and the K-nearest neighbor (B), find the K neighbors, and vote for labels (C) [8].

Support Vector Machines

Support vector machines are a type of machine learning technique in which a hyperplane is constructed on a multidimensional space to separate multiple classes. To explain this point better, it is possible to look at Figure 6. The resultant hyperplane is the blue line between the red stars and green triangles. On the other hand, the support vectors are placed in the closest data point near to the other class. This is done to find a maximum marginal hyperplane that best divides the dataset into classes. Nonetheless, the problem of support vector machines is that it only works with linearly separable data as shown in Figure 9 in which it is possible to separate the two classes with a line. In the case in which the data is presented as shown in Figure 10, the support vector machine uses what it is called a kernel to project the information into a higher dimensional space in which the two classes can be separated with a hyperplane [9].

The components of the support vector machines are:

- **Support vectors:** Referred to the data points that are closest to the hyperplane used to separate the given classes. These points are also used to calculate the hyperplane better by computing margins.
- **Hyperplane:** It is the decision surface that separates the given classes in the multidimensional space.
- **Margin:** It is the distance between the two lines on the closest class data points.
- **Kernel:** A kernel is a function that transforms the input data to transform it from a low dimensional input space to a higher-dimensional space. The use of kernel is useful for the

non-linear separable classes and it generates a classifier with better accuracy. There are different types of kernel some of them are linear, polynomial, and radial basis function kernels.

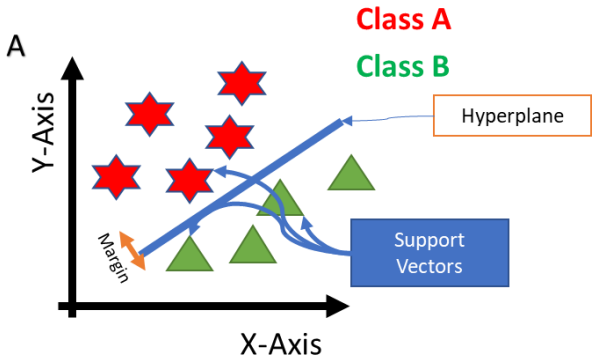

Figure 9. Support vector machine graphical analysis [9].

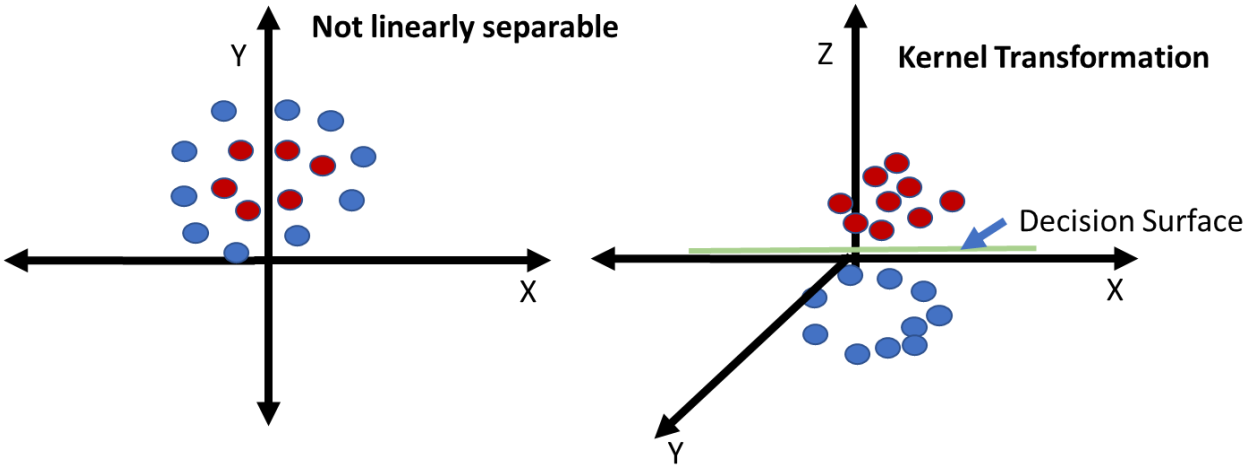

Figure 10. Kernel trick to separate two classes that are not linearly separable [10].

Random Forest

The random forest algorithm creates a set of decision trees on randomly selected data samples, for each of the decision trees the algorithm makes a prediction and selects the best solution by a voting mechanism. This procedure also provides a good indicator of which features, or variables are more important for the classification task. The steps of this algorithm are described in Figure 11. On the other hand, Figure 12 shows a schematic representation of the Random Forest algorithm.

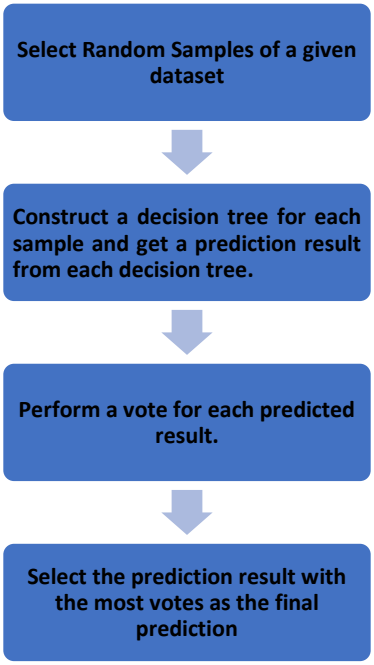

Figure 11. Random forest algorithm steps.

Random forest is considered a highly accurate and robust technique due to the number of decision trees that are made to execute the algorithm. It is not affected by over-fitting since it takes the average of all the predictions which cancel the biases. Moreover, it can be used to determine the relative feature importance during the classification task. The relevance score is computed using the Gini importance or total decrease in node impurity of each feature. The larger the decrease, the more significant the variable is for the prediction task. Nonetheless, this method has some drawbacks, the first one is that it is slow in generating predictions because it must run multiple decision trees for the same input and then perform voting according to the output of each decision tree. The execution of this process is time-consuming, and this type of algorithm is more difficult to interpret than a decision tree [11].

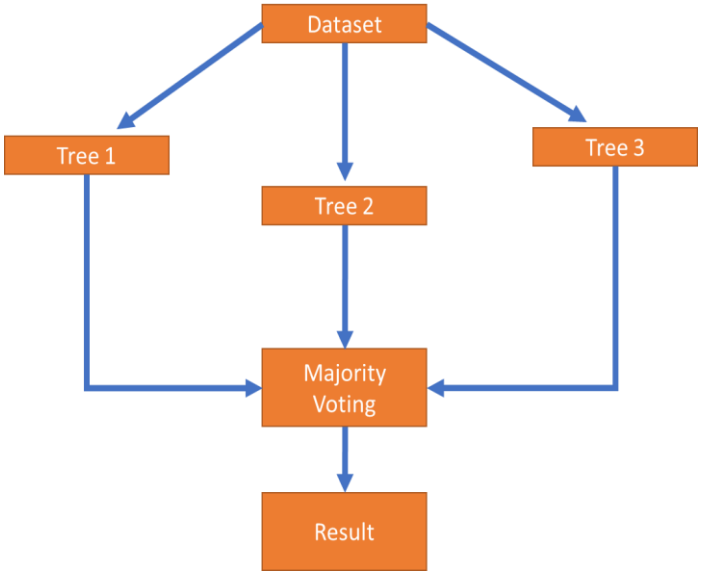

Figure 12. Schematic Representation of the Random Forest Algorithm with three trees.

5. Results and Analysis

For each of the machine learning techniques presented in the previous section the confusion matrix of the algorithm was computed to extract the accuracy and F1-score of each machine learning technique that was tested. All the training and testing of the models were generated with the use of the sklearn package of Python [12]. As it was previously established since the classes of the dataset are unbalanced, the methodology to implement each technique and testing it was by splitting the data and using 80% for the training phase and 20% for the testing phase. Besides, each of the machine learning technique was trained in three different scenarios, the first one was by using all the features of the dataset, the second was by using only the 15 features extracted from the PCA analysis presented in Section 2 (Figure 5), and in the third scenario, the features were extracted according to the values of Gini importance (Figure 6). Taking the above into account, three different types of models were tested as listed below.

- Model 1: Machine Learning Techniques were trained with all the variables.
- Model 2: Machine Learning Techniques were trained with the variables that have the highest loading score based on the PCA analysis.
- Model 3: Machine Learning Techniques were trained with the variables selected through the Gini importance score by executing a random forest algorithm.

Decision Tree Classifier

The decision trees were trained by using the Gini importance criteria and by establishing a max depth of 25 for the complete model and a max depth of 6 for the reduced models. In Table 5 it is

possible to observe the accuracy comparison between the model generated using all the variables in the database and the proposed reductions. For Model 1, the accuracy value for the case of the training test suffers from over-fitting since a value of 1 was obtained. This behavior is not appreciated by the reduced models. This helps to rule out Model 1 besides, Model 1 has a greater complexity due to the number of input variables it has.

In the case of Model 2, the accuracy obtained with the training set is slightly higher compared to the accuracy obtained with the test set. Besides, the decision tree trained with the characteristics obtained from the Gini importance scores (Model 3) obtained greater precision both in the training and in the test phase compared to the results of Model 2. Additionally, to know the classification performance of each of the classes that each model has, the F1 score was calculated, this can be seen in Table 6. By looking at the F1-scores of the three models it is possible to notice that the completed model is better in predicting when the Blue Fighter will win compared to the reduced models. However, the F1 score of the Blue class of Model 1 and Model 3 does not differ much.

Table 5. Decision Tree Classifier Accuracy comparison between the complete and simplified models.

Accuracy Decision Trees	**Complete Model (Model 1)**	**Reduced Model from PCA Loading scores (Model 2)**	**Reduced model from Gini importance scores (Model 3)**
Train Set	1.0000	0.6790	0.6790
Test	0.5663	0.6256	0.6349

Table 6. Classification report of the Decision Tree classifiers.

Complete Model (Model 1)	**F1-score**
Blue	0.37
Draw	0.11
Red	0.68
Reduced Model with PCA (Model 2)	**F1-score**
Blue	0.26
Draw	0
Red	0.75
Reduced Model with Gini importance (Model 3)	**F1-score**
Blue	0.35
Draw	0
Red	0.75

K-Nearest Neighbor

For the KNN algorithm, the variables were standardized using the Standard Scaler function of the sklearn package before feeding the data into the algorithm. For the complete and reduced models, 7 nearest neighbors were used to perform the classification task. Table 7 shows the accuracy results of the complete and reduced models using the KNN algorithm. The lowest performance was from the reduced Model 2 in both the training and test sets. On the other hand, the complete model presents a lower accuracy with the test set than Model 3. Table 8 shows the F1-score of the three models. There is not much difference between the F1 scores values for the Blue and Red class by comparing the three models, nevertheless, the highest score was obtained by Model 3. In the three models, the Blue class was not classified correctly during the test process compared to the Red class. Finally, none of the three models could classify the Draw class.

Table 7. Classification report of the K-NN classifiers.

Accuracy ***K- Nearest Neighbor***	**Complete Model** **(Model 1)**	**Reduced Model from PCA Loading scores (Model 2)**	**Reduced model from Gini importance scores (Model 3)**
Train Set	0.7107	0.6947	0.7224
Test Set	0.6209	0.5897	0.6490

Table 8. Classification report of the K-NN classifiers.

Complete Model (Model 1)	**F1-score**
Blue	0.39
Draw	0
Red	0.73
Reduced Model with PCA (Model 2)	**F1-score**
Blue	0.37
Draw	0
Red	0.70
Reduced Model with Gini Importance (Model 3)	**F1-score**
Blue	0.45
Draw	0
Red	0.75

Support Vector Machine

The third machine learning technique that was tested was a Support Vector Machine Classifier. Before feeding the data into the algorithm, the features were standardized using the Standard Scaler function of the sklearn package. This technique was tested with different types of kernels to find the one that provides better accuracy. In this case, a radial basis function (RBF) kernel was the one that provides better accuracy for the complete and reduced models. It is worth mentioning that for the linear and polynomial kernel the training process was slower compared to the RBF. The table below shows the accuracies of the complete models and the reduced model for each of the proposed methodologies for feature selection. It is possible to notice that the complete model provides the best accuracy with both the train set and the test set. In the case of the reduced models, Mode 2 produced the lowest accuracy of the three. Moreover, the reduced models provide a lower F1-score for the blue class than the one achieved with the complete model; this implies that the support vector machine classifier has problems in classifying when the Blue fighter wins the match.

Table 9. Accuracy comparison between the complete and simplified models using a Support Vector Machine Classifier.

Accuracy Support Vector Machine	**Complete Model (Model 1)**	**Reduced Model from PCA Loading scores (Model 2)**	**Reduced model from Gini importance scores (Model 3)**
Train Set	**0.7930**	**0.6474**	**0.6974**
Test	**0.6771**	**0.6271**	**0.6349**

Table 10. Classification report of the Support Vector classifiers.

Complete Model (Model 1)	**F1-score**
Blue	0.38
Draw	0
Red	0.79
Reduced Model with PCA (Model 2)	**F1-score**
Blue	0.07
Draw	0
Red	0.77
Reduced Model with Gini Importance (Model 3)	**F1-score**
Blue	0.22
Draw	0
Red	0.76

Random Forest

Finally, the Random Forest Classifier was used to train the UFC data. 120 estimators or trees were used to train the random forest based on the Gini importance criteria and with a max depth of 25 for the complete model and a max depth of 6 for the reduced models. The table below shows the accuracy of each model. For the complete model, it is possible to notice an overfitting problem since the accuracy is equal to 1 when using the training data. In the reduced models, the accuracy for the training data is low which implies that there is no overfitting and the accuracy with the test data provides similar results to the ones obtained with the complete model. Finally, Table 12 shows the F1 scores of each model for the Red and Blue class. Model 3 has the highest F1-score value for the red class which implies that it classifies better when the Red fighter will win compared to the complete model and model 2, and model 3 is worst at classifying when the Blue fighter wins compared to Model 1.

Table 11. Random Forest Accuracy comparison between the complete and simplified models.

Accuracy Random Forest	**Complete Model**	**Reduced Model from PCA Loading scores**	**Reduced model from Gini importance scores**
	Model 1	**Model 2**	**Model 3**
Train Set	1.0000	0.6818	0.7146
Test	0.6583	0.6537	0.6775

Table 12. Classification report of the Random Forest classifiers.

Complete Model (Model 1)	**F1-score**
Blue	0.32
Draw	0
Red	0.78
Reduced Model with PCA (Model 2)	**F1-score**
Blue	0.16
Draw	0
Red	0.78
Reduced Model with Gini Importance (Model 3)	**F1-score**
Blue	0.30
Draw	0
Red	0.79

Model selection and testing

Since analyzing high-dimensional models can be complicated and computationally expensive, it is better to study the proposed reduced models. In the case of the Random Forest reduced models (Table 12), the use of a high number of estimators or trees makes difficult its interpretation despite the reduction of variables that was proposed using both PCA and the Gini importance criteria. Also, the F1-scores show greater difficulties to classify the Blue class. A similar problem can be observed for the Support Vector Machine reduced models (Table 10) whose F1-scores for the Blue class are lower compared to the ones obtained for the Red class. Another disadvantage is the need of standardizing the input data before training the algorithm and making a prediction. This disadvantage is also shared with the K-NN algorithm that requires the standardization of the data, nevertheless, the F1-scores of the K-NN for the Blue class (Table 8) are higher compared to the Support Vector Machine values. The problem of using the K-NN is that it requires the use of the whole dataset to evaluate a new input since it computes distance with the nearest data points.

Finally, the Model 3 based on a decision tree provided a higher F1-score for the Blue class (Table 6), it only requires a single decision tree to estimate the output, it is more interpretable than the other techniques that were tested, and it does not require the input data to be preprocessed for the training of the model or predict with it. This leads to determining that the decision tree using as inputs the most important variables according to the Gini criterion (Figure 6) provides a better classification of the blue and red classes and it also generates a much simpler model.

Model 3 based on a decision tree was tested with the matches that can be appreciated in Table 13. The results of testing the decision tree by using as inputs the variables presented in Figure 6 were expressed in probability. It is important to remark that the variable name was not considered for this model, therefore, it was necessary to search in the original database the statistics of each fighter and take the average of them and introduce this average as input to make the prediction. Some fighters only had one fight until the day the database was created, therefore some of them do not have the complete information due to the missing values that the database originally had. In these cases, the prediction was made with the information from another fighter that had similar age, weight class, gender, or height.

Some of the matches that are presented in Table 13, had taken place by the day this study was made, and the results can be consulted on the UFC stats website [13]. The fighters labeled as red were the ones that commonly win the match, so it is expected that this model provides a greater probability to the red fighter compared to the blue fighter. Nonetheless, some predictions failed to classify the actual winner of the fight, this was expected since the accuracy of the model was not so high and some of the features that could provide a better classification were discarded to provide a simpler model. Finally, the Rank Probability Score (RPS) was used to measure the error between the prediction and the actual result. This rank was computed for each of the matches according to equation 1.

$$RPS = \frac{1}{r-1}\sum_{i=1}^{r-1}\left(\sum_{j=1}^{i}(p_j - a_j)\right)^2 \quad (1)$$

- Where r are the scores to evaluate (r=3) p_j is the prediction at position j, such that $p_j \in [0,1]$ for j=1,2,3 and $p_1+p_2+p_3=1$.
- a_j is the real result at the position j, such that $a_j \in \{0,1\}$ for j=1,2,3 and $a_1+a_2+a_3=1$ this is (1,0,0) for a Red Fighter victory, (0,1,0) for a Draw and (0,0,1) for a Blue Fighter victory.
- An RSP = 0 result indicates a perfect prediction, and an RSP = 1 result means a completely bad prediction.

Table 13. Prediction matches tested in the model based on a decision tree and employing as inputs that stats presented in Figure 6.

Matches			Probability						Actual Results		
R_fighter		B_fighter	Red	Draw	Blue	SUM	Prediction	RPS	R_fighter	Draw	B_fighter
Khabib Nurmagomedov	vs	Justin Gaethje	0.836538	0.00962	0.1538462	1	Red	0.025194	1	0	0
Robert Whittaker	vs	Jared Cannonier	0.597403	0	0.4025974	1	Red	0.162085	1	0	0
Alexander Volkov	vs	Walt Harris	0.597403	0	0.4025974	1	Red	0.162085	1	0	0
Magomed Ankalaev	vs	Ion Cutelaba	0.580645	0.03226	0.3870968	1	Red	0.162851	1	0	0
Tai Tuivasa	vs	Stefan Struve	0.869565	0	0.1304348	1	Red	0.017013	1	0	0
Joel Alvarez	vs	Alexander Yakovlev	0.597403	0	0.4025974	1	Red	0.162085	1	0	0
Israel Adesanya	vs	Paulo Costa	0.498333	0.01667	0.485	1	Red	0.243447	1	0	0
Jake Matthews	vs	Diego Sanchez	0.793103	0	0.2068966	1	Red	0.042806	1	0	0
Islam Makhachev	vs	Rafael dos Anjos	0.773684	0.01053	0.2157895	1	Red	0.048892	1	0	0
Jose Quinonez	vs	Louis Smolka	0.498333	0.01667	0.485	1	Red	0.243447	1	0	0
Stipe Miocic	vs	Daniel Cormier	0.655172	0.01149	0.3333333	1	Red	0.115009	1	0	0
Marlon Vera	vs	Sean O'Malley	0.498333	0.01667	0.485	1	Red	0.243447	1	0	0
Merab Dvalishvili	vs	John Dodson	0.793103	0	0.2068966	1	Red	0.042806	1	0	0
Vinc Pichel	vs	Jim Miller	0.173913	0.02899	0.7971014	1	Blue	0.658895	1	0	0
Brian Ortega	vs	Chan Sung Jung	0.498333	0.01667	0.485	1	Red	0.243447	1	0	0
Jessica Andrade	vs	Katlyn Chookagian	0.498333	0.01667	0.485	1	Red	0.243447	1	0	0
James Krause	vs	Claudio Silva	0.477679	0.02679	0.4955357	1	Blue	0.259188	1	0	0
Gillian Robertson	vs	Poliana Botelho	0	0	1	1	Blue	1	1	0	0
Neil Magny	vs	Robbie Lawler	0.655172	0.01149	0.3333333	1	Red	0.115009	1	0	0
Alexa Grasso	vs	Ji Yeon Kim	0.498333	0.01667	0.485	1	Red	0.243447	1	0	0
Zak Cummings	vs	Alessio Di Chirico	0.477679	0.02679	0.4955357	1	Blue	0.259188	1	0	0
Colby Covington	vs	Tyron Woodley	0.75	0	0.25	1	Red	0.0625	1	0	0
Donald Cerrone	vs	Niko Price	0.545455	0	0.4545455	1	Red	0.252066	0	1	0
Mackenzie Dern	vs	Randa Markos	0.793103	0	0.2068966	1	Red	0.042806	1	0	0
Kevin Holland	vs	Darren Stewart	0.935484	0.01613	0.0483871	1	Red	0.003252	1	0	0
Damon Jackson	vs	Mirsad Bektic	0.498333	0.01667	0.485	1	Red	0.243447	1	0	0
Mayra Bueno Silva	vs	Mara Romero Borella	0.7	0.025	0.275	1	Red	0.082813	1	0	0
Michelle Waterson	vs	Angela Hill	0.477679	0.02679	0.4955357	1	Blue	0.259188	1	0	0
Bobby Green	vs	Alan Patrick	0.655172	0.01149	0.3333333	1	Red	0.115009	1	0	0
Sijara Eubanks	vs	Julia Avila	0.935484	0.01613	0.0483871	1	Red	0.003252	1	0	0
Sabina Mazo	vs	Justine Kish	0.498333	0.01667	0.485	1	Red	0.243447	1	0	0

The RPS results can be appreciated in Table 13 in the column marked with orange, the real results were obtained from the UFC stats website [13]. The average RPS of all the test matches was 0.19356, which indicates the average performance of the predictions. However, using the RPS does not assure the overall performance of the model since it only measures the quality of the predictions that are been tested. Also, it assigned greater penalties when the opposite class is predicted (Blue class) compared to the Draw class as shown in the fight between Gillian Robertson and Polian Botelho.

6. Conclusions

In this work, a decision tree, the K-Nearest Neighbor algorithm, a Support Vector Machine, and a Random Forest classifier were tested to predict the outcome in a UFC match, in this study the target was the color of the winner. Three models were proposed using the complete database and a reduction of the predictors based on a PCA analysis and the Gini importance impurity criteria. It was concluded that the proposed reduction has similar performance results in terms of accuracy during the testing compared to the complete models by employing the same machine learning techniques and in some cases, the accuracy was higher. Nevertheless, none of the reduced models could predict when a Draw will occur during a match, and the prediction of the Blue class was lower based on the F1 scores for most of the models, this behavior is expected since the database is unbalanced and there are fewer samples for the Blue and Draw class compared to the Red class. Moreover, based on the analysis and results presented in Table 5 and 6, it was determined that the reduced model using a decision tree and the features selected through the Gini importance provide a model that is quick to train, it is interpretable, compact, and provide better results in comparison with the complete model and the reduced models that used PCA for feature selection. This is also supported by the low RPS values that were obtained during the predictions phase and that gave an average value of 0.19356.

Also, it is determined that the database can be improved to provide a better classification for the two classes. This improvement can be made by taking a sample that focuses on capturing the data related to the most important variables according to the Gini impurity criterion that was presented. In the same way, having a balanced database could help to have a model that allows having a better classification to determine the winner based on color, since all the models classify better the victories of the red fighters than blue fighters victories. Although Python provides certain tools to facilitate data management as well as the training of the machine learning algorithms, having a large amount of data and samples was a challenge since the fairly extensive processing of the data that had to be done during the training and testing of the models. However, this also helped to have some experience to handle real data that does not always have all the information or presents too much to be interpreted by a single individual. Therefore, the studied machine learning techniques allowed us to generate a proposal that can solve the problem and proposed objective described at the beginning of this work.

References

- [1] Choi, R., Coyner, A., Kalpathy-Cramer, J., Chiang, M., & Campbell, J. (2020, January 28). Introduction to Machine Learning, Neural Networks, and Deep Learning. Retrieved November 08, 2020, from https://tvst.arvojournals.org/article.aspx?articleid=2762344

- [2] Shinde, P. P., & Shah, S. (2018). A Review of Machine Learning and Deep Learning Applications. 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA). doi:10.1109/iccubea.2018.8697857

- [3] Warrier, R. (2019, July 05). UFC-Fight historical data from 1993 to 2019. Retrieved November 08, 2020, from https://www.kaggle.com/rajeevw/ufcdata

- [4] Hitkul, Aggarwal, K., Yadav, N., & Dwivedy, M. (2018). A Comparative Study of Machine Learning Algorithms for Prior Prediction of UFC Fights. Advances in Intelligent Systems and Computing, 67–76. doi:10.1007/978-981-13-0761-4_7

- [5] D.H. Wolpert, The Supervised Learning No-Free-Lunch Theorems, in: Soft Computing and Industry, Springer London, 2002: pp. 25–42. doi:10.1007/978-1-4471-0123-9_3.

- [6] A.C. Faul, A Concise Introduction to Machine Learning, Chapman and Hall/CRC, 2019. doi:10.1201/9781351204750.

- [7] G. Rebala, A. Ravi, S. Churiwala, An Introduction to Machine Learning, Springer International Publishing, 2019. doi:10.1007/978-3-030-15729-6.

- [8] Navlani, A. (2018, August 2). KNN Classification using Scikit-learn. Retrieved November 08, 2020, from https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn.

- [9] Navlani, A. (2019, December). (Tutorial) Support Vector Machines (SVM) in Scikit-learn. Retrieved November 08, 2020, from https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python

- [10] Sharma, S. (2019, December 18). How to Classify Non-linear Data to Linear Data? Retrieved November 08, 2020, from https://medium.com/analytics-vidhya/how-to-classify-non-linear-data-to-linear-data-bb2df1a6b781

- [11] Louppe, Gilles. (2014). Understanding Random Forests: From Theory to Practice. 10.13140/2.1.1570.5928.

- [12] Swamynathan, M. (2019). Mastering machine learning with Python in six steps: A practical implementation guide to predictive data analytics using Python. Berkeley: Apress. doi:10.1007/978-1-4842-2866-1.

- [13] Ultimate Fighting Championship, U. (2020). UFC stats. Retrieved November 18, 2020, from http://ufcstats.com/statistics/events/completed

Appendix

The code used to predict the results of the matches presented in Table 13, can be appreciated in the attached Python notebook names as Final_Model_Testing_Matches_DT_A01331212. ipynb. On the other hand, the code to see the whole analysis of the data and testing of the proposed models described along this document can be consulted in the attached file name as Final_Project_A01331212_ver_3. Finally, the RPS is calculated in the CSV file named Prediction_Matches_UFC_Team_1.