Ultimate Fighting Championship Winner Predictions

Project report

Students:

Fejsal Perva

Harun Tucaković

Muhammed Musanović

Professor:

Kanita Karađuzović – Hadžiabdić, Assist. Prof, Dr.

Fall, 2021

Contents

Abstract

Ultimate Fighting Championship (UFC) is the biggest mixed martial arts fight promotion in the world. The application of machine learning algorithms in this field can produce reliable models capable of predicting UFC fight outcomes. Predicting fight outcomes is an important business and marketing tool, and reliable fight predictions can highly influence business and marketing strategies. Existing research on the topic has produced models that have relatively low prediction accuracy. This project focuses on one part of the possible fight outcomes to achieve a higher prediction accuracy. Using a decision tree, random forests, and KNN algorithms with different resampling methods (cross-validation, repeated cross-validation, and bootstrap sampling) to create and test machine learning models, higher prediction accuracy is achieved. The best performing machine learning algorithm, out of the three algorithms tested, was the random forests, even though the other two algorithms had just slightly worse performance.

Introduction

Mixed martial arts (MMA), also known as cage fighting, is a full-contact combat sport that emerged in the early 1990s in the United States. MMA fights or competitions are organized in the caged-in area, where competitors incorporate techniques from different combat sports and engage in striking, grappling, and ground fighting (Spencer 3). The Ultimate Fighting Championship is an MMA promotion company that organizes events, featuring multiple MMA fights at each event. UFC is the biggest MMA promotion company in the world (Hitkul et al. 67).

The goal of this project was to find the best machine learning (ML) algorithm that can be used to predict the winners of UFC fights in cases when the winner is decided by judges' decision, based on the historical data containing statistics and outcomes of every UFC fight since the birth of the company in 1993. Multiple research papers have been published on the topic of predicting the outcome of the MMA fights (Hitkul et al. 67-76; Johnson; Uttam and Gaurav 416-421). Previously published works sought to forecast the general outcome of MMA fights. Due to the nature of the sport, fight outcomes are quite unpredictable. Hence, the above-mentioned papers did not manage to achieve high prediction accuracy. Johnson achieved a prediction accuracy of 63.39% using logistic regression (Johnson 27), while Hitkul et al. achieved a prediction accuracy of 61.15% using support vector machines (Hitkul et al. 74). This paper will focus on a select set of fight outcomes to lower the amount of randomness in the dataset and improve prediction accuracy. The focus of this paper is fights in which the winner is decided by judges' decision.

Background

Outcomes of MMA fights are multiple, and some of them are more prone to randomness than others. Any MMA fight can have the following outcomes: win by KO/TKO, win by submission, win by unanimous decision, win by split decision, win by disqualification, win by doctors' stoppage, no contest (no winner). As mentioned before, this paper focuses on predicting a winner in fights that had one of two outcomes: win by unanimous decision or win by split decision. These outcomes mean that the winner of the fight is decided by the decision of the judges.

Judges in MMA fights are people tasked with scoring a fight according to the 10-point must system (Gray and Brian 53). This means that each round of the fight is scored separately, and the fighter that wins most rounds wins the match. For this reason, all fights have an odd number of rounds. Judges score strikes, knockdowns, takedowns, submission attempts, control time, and many other aspects of the fight. The unanimous decision is a situation where all judges scored the fight in favor of the same fighter. A split decision is a situation where the majority of judges scored a fight in favor of the same fighter, but at least one judge scored it in favor of the other fighter.

Materials and Methods

Dataset used in this research contains every fight that happened under UFC promotion since its birth in November 1993. up to December of 2021. In that period 6399 fights happened (6399 observations in the dataset). The dataset includes 302 variables. The main variables, that were the most important for our study are the winner of the fight, the number of significant strikes (attempted and landed as separate variables), number of takedowns, number of knockdowns, control time, number of significant strikes landed per body area, etc. All the variables that capture fight statistics information are recorded as fight total per fighter and every variable per round per fighter. Per round, data is used to create ML models because fights are scored per round, and we are interested in judges' scoring/decisions. A detailed explanation of each variable in the dataset is available to the reader in Appendix A.

Dataset is scraped from the website ufcstats.com. This is the official website for collecting and publishing UFC fight statistics. The scraping process is explained in the subsection below.

Data acquisition

Script for scraping the data from the web is written in R using the "rvest" package. For string and data manipulation/formatting following external R packages were used: stringr, sjmisc, plyr. This subsection goes through the general idea of the web-scraping process step by step.

The first step in the web-scraping process is to get an HTML markdown of the ufcstats.com webpage. This is achieved with help of the xml2 package and 'read_html' function. When HTML markdown is loaded, URLs to every single UFC event are extracted from it. With the list of URL links to every UFC event prepared, the HTML markdown of each UFC event web page is extracted. The web page of each UFC event contains a list of fights that happened at that event. URLs to each fight are extracted. With URL links of each fight extracted, the next step is to load the HTML markdown for each web page containing information about a single fight. From the HTML markdown of a single fight, fight outcome and fight statistics are extracted and saved into the data frame.

Data preprocessing

To simplify the data scraping process, all available data from the website is scraped and saved into the data frame. Naturally, the next step is the cleaning and formatting the data to prepare it for ML algorithms. During the preprocessing process, several important tasks are performed: dropping the columns that are not needed, dropping the rows that are not needed, handling missing values, the transformation of data values, and preparing correct data types.

Columns that hold information about the event and the personal information about the fighters are excluded from the dataset because they are not necessary for data analysis and ML algorithms. Examples of these columns are event location, event name, fighter names, fighter nicknames, etc. The original dataset contains information about all fights that happened in the UFC since 1993. As mentioned before, this paper focuses only on fights in which the winner was decided by judges' decision. Hence, in preprocessing phase all rows that had fight outcome of a win by KO/TKO, win by submission, win by disqualification, win by doctors' stoppage, or no contest is dropped. These fights (rows) are not relevant to our research. In the exploratory analysis of the dataset, one important conclusion is made: fights that were not in a 3-round time format had a lot of missing values. For this reason, all rows containing information about fights in other time formats than the 3-round format are dropped. Also, for the same reason, all columns that held information about the 4^{th} and 5^{th} rounds are dropped. All data regarding control time in the fight is transformed from string holding the number of minutes and seconds separated with a colon, to the number representing a total number of seconds. Finally, all character columns are converted to factors, and all other columns are converted to numeric data types. After preprocessing was done, training data had 37.29% blue and 62.71% red values in the target variable, while in test data those values were 40% for blue and 60% for red which is acceptable.

Features selection

After the preprocessing, the dataset contains 127 columns, 126 features. In our feature selection process we used the random forests model and *varImp()* function to determine the most important features in our dataset. Besides that, to lower the number of features in the dataset we decided to merge features from the dataset with all (126) features together. As mentioned before, this dataset contains fight statistics per fighter, meaning that for one single statistic we have two columns, one for each fighter (e.g. *red_control_time, blue_control_time*). These columns are merged into one column (e.g. *control_time*) by subtracting statistic of the second fighter from the statistic from the first fighter (e.g. *control_time = red_control_time – blue_control_time*). By merging columns this way, we preserve all information from original columns since we still have information about whether the first or second fighter had the advantage in specific statistic and how much of an advantage it was.

Due to the nature of the data, we were not certain that the feature selection process would produce a better-performing model. Hence, we decided to run all ML algorithms on three different datasets, the first one being a dataset including all features, the second one including only features selected through the feature selection process, and the third one containing merged features from the first dataset.

Methods

Machine learning algorithms used in this project are the C5.0 decision tree, random forests, and k-nearest neighbors (KNN) algorithm, while resampling methods used in the project are cross-validation (CV), bootstrap sampling, and repeated cross-validation.

Decision trees are used to build the classification models by dividing the dataset into smaller subsets, where each node in the decision tree is a feature, and each branch represents a value (Pandya and Jayati 18). C5.0 classification algorithm is a descendant of the classic C4.5 algorithm which creates classification models by discovering and analyzing patterns found in the dataset (Quinlan 2). C5.0 comes with improvements such as improved speed, less memory usage, and smaller decision trees (Pandya and Jayati 18). Random forests is an ensemble-based method that combines random feature selection with bagging (Lantz 344). When used in classification tasks, random forest outputs a class that was selected by most decision trees in the random forest. K-nearest neighbors (KNN) is one of the most basic and simple classification algorithms that attempt to classify unknown samples based on the known classification of their neighbors (Mucherino et al. 83).

Cross-validation is a resampling method that is used to evaluate machine learning algorithms by using different portions of data to test and train the model on different iterations (Refaeilzadeh et al. 532). Data is firstly divided into *k* equal folds, which is followed by *k* iterations of training and testing with different folds used for testing in each iteration. Bootstrap sampling is a resampling method that uses random sampling with replacement. Finally, repeated cross-validation is a method of repeating cross-validation. Repeating *k*-fold cross-validation can increase estimates precision but maintain low bias (Kuhn and Kjell 70).

Results and Discussion

As mentioned before, we used three different datasets to run described algorithms. In this section, we will present and discuss obtained results. Models for all machine learning algorithms were created using default hyperparameters, with cross-validation, repeated cross-validation, and with bootstrap sampling. Furthermore, for the decision tree and random forests algorithm, we also created models with the best-performing hyperparameters.

Decision Tree (Default), Random Forest (Default), and KNN (Default) were created using the rule of thumbs hyperparameters for C5.0 (trials = 1), randomForest (mtry = square root of training data length, ntree = 500), knn (k = square root of training data length), respectively. Regarding Decision Tree (Final), we created 20 different C5.0 models based on a different number of boosting iterations (trials ranging from 1 to 20). Similarly, for Random Forest (Final) we created 40 models with different combinations of the number of trees (ntree ranging from 100 to 1000, with step 100) and the number of features per tree (mtry being the square root of training data length, 10, 15 and 20). Afterward, the R script automatically selected the best performing hyperparameters which we then used to create Decision Tree (Final) and Random Forest (Final). Hyperparameter trials for Decision Tree (Final) were 20, 20, and 7 for datasets with all, selected, and combined features, while hyperparameters ntree and mtry for Random Forest (Final) were (11, 100), (15, 400), (8, 700) for those datasets.

When it comes to resampling methods, cross-validation models (CV) were trained using 10-fold cross-validation (number = 10). Models trained with repeated cross-validation (Repeated CV) used 3-fold cross-validation (number = 3) that repeated four times (repeats = 4). For bootstrap sampling, the number of sampling iterations was 20 (number = 20). Regarding hyperparameters for decision tree and random forests models using resampling methods, they were trials = 1 for decision tree models, and mtry = 10 for random forest models. When it comes to KNN models, hyperparameter *k* for KNN (CV), KNN (Repeated CV), and KNN (Bootstrap) ranged from 3 to 47 with step 2. The functions then determined the one that led to the highest AUC. Hyperparameter k for KNN (CV), KNN (Repeated CV), and KNN (Bootstrap) on a dataset with all features were 45, 39, and 47, respectively. On the dataset with selected features, hyperparameter k was 47 for all three models. Finally, using a dataset with combined features we got that hyperparameter k was 43, 47, and 47 for KNN (CV), KNN (Repeated CV), and KNN (Bootstrap), respectively.

Dataset including all features

The best accuracy out of all models on this dataset was achieved by Random Forest (Bootstrap) which is 0.829907, while Random Forest (Default) achieved the best AUC, 0.924513. Concerning sensitivity and specificity, the best models were KNN (Default) – sensitivity of 0.925234, and

Decision Tree (CV), Decision Tree (Repeated CV), and Decision Tree (Bootstrap) (tied) – specificity of 0.742991. We believe that the most important performance metric is AUC, hence, the best performing algorithm on this dataset was Random Forest (Default) with an accuracy of 0.826168, sensitivity of 0.919003, specificity of 0.686916, and AUC of 0.924513. The worst performing model-based this criteria was KNN (Default) with an AUC of 0.690453. Overall, all decision tree and random forests models produced excellent results, while KNN only had similar results with different resampling methods. Besides that, for all three algorithms, all three resampling methods produced similar results.

Table 1 Metrics for the whole dataset

	Accuracy	Sensitivity	Specificity	AUC
Decision Tree (Default)	0.766355	0.847352	0.644860	0.807560
Decision Tree (Final)	0.818692	0.900312	0.696262	0.897327
Decision Tree (CV)	0.822430	0.875389	**0.742991**	0.902451
Decision Tree (Repeated CV)	0.822430	0.875389	**0.742991**	0.902451
Decision Tree (Bootstrap)	0.822430	0.875389	**0.742991**	0.902451
Random Forest (Default)	0.826168	0.919003	0.686916	**0.924513**
Random Forest (Final)	0.828037	0.922118	0.686916	0.917904
Random Forest (CV)	0.828037	0.912773	0.700935	0.922075
Random Forest (Repeated CV)	0.822430	0.909657	0.691589	0.923429
Random Forest (Bootstrap)	**0.829907**	0.922118	0.691589	0.923829
KNN (Default)	0.803738	**0.925234**	0.621495	0.690453
KNN (CV)	0.798131	0.912773	0.626168	0.895282
KNN (Repeated CV)	0.800000	0.909657	0.635514	0.898128
KNN (Bootstrap)	0.801869	0.912773	0.635514	0.895995

Concerning the ROC curve for this dataset, random forests models overall have the best AUC, with all decision tree models and KNN models with resampling methods not being left too far behind. Additionally, we can see that KNN (Default) model had the worst performance by far.

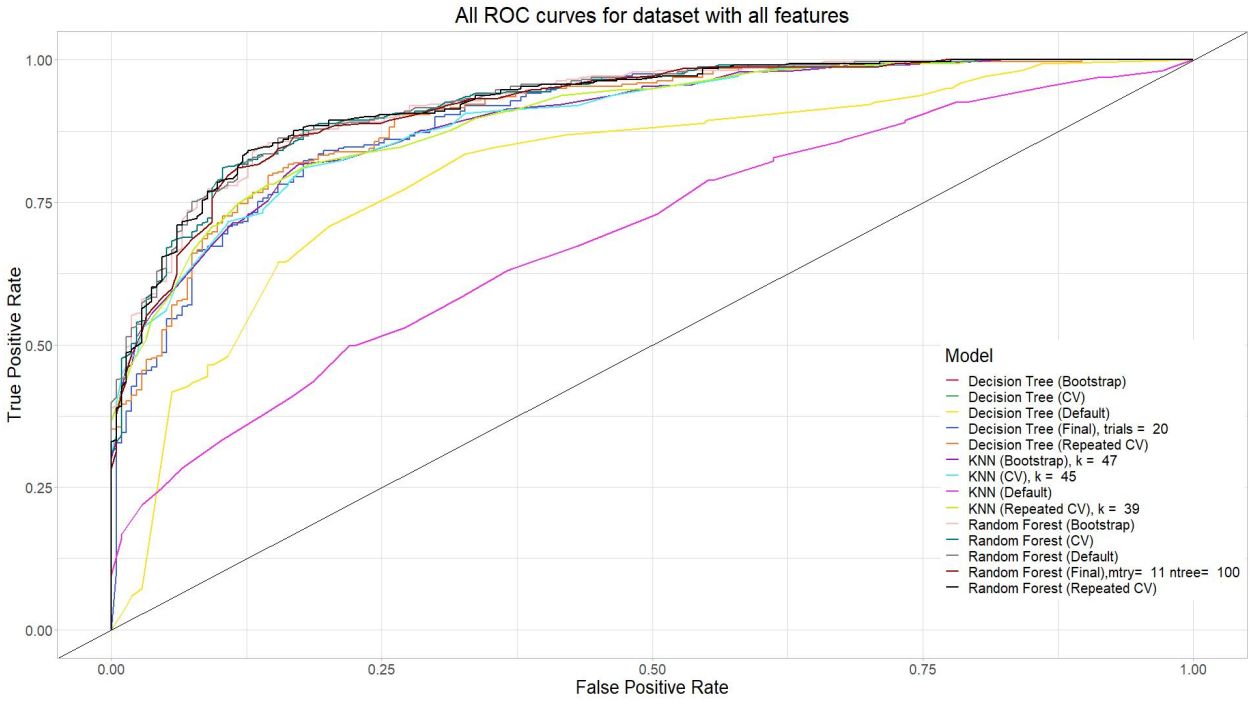

Figure 1 ROC curves for the dataset with all features

Dataset with selected features

Comparing the performances of algorithms we got similar results as with the previous dataset, with the decision tree having the highest accuracy and random forest having the highest AUC. Once again, KNN (Default) had the worst performance, while KNN models with resampling methods performed almost as equal, if not better in some cases, compared to the decision tree and random forests models. The best performing model was again Random Forest (Default) with an accuracy of 0.835514, sensitivity of 0.919003, specificity of 0.71028, and AUC of 0.926238. Generally, besides taking significantly less time to train models, the performance for all four metrics for the models on this dataset was somewhat better than on the dataset with all features.

Table 2 Metrics for selected features using RF

	Accuracy	Sensitivity	Specificity	AUC
Decision Tree (Default)	0,781308	0,866044	0,654206	0,820057
Decision Tree (Final)	0,824299	0,890966	0,724299	0,909381
Decision Tree (CV)	**0,846729**	0,894081	**0,775701**	0,918101
Decision Tree (Repeated CV)	**0,846729**	0,894081	**0,775701**	0,918101
Decision Tree (Bootstrap)	**0,846729**	0,894081	**0,775701**	0,918101
Random Forest (Default)	0,835514	**0,919003**	0,71028	**0,926238**
Random Forest (Final)	0,839252	0,909657	0,733645	0,922322
Random Forest (CV)	0,835514	0,915888	0,714953	0,921543
Random Forest (Repeated CV)	0,833645	0,912773	0,714953	0,922759
Random Forest (Bootstrap)	0,833645	**0,919003**	0,705607	0,922752
KNN (Default)	0,820561	0,915888	0,67757	0,685999
KNN (CV)	0,813084	0,894081	0,691589	0,912685
KNN (Repeated CV)	0,814953	0,894081	0,696262	0,912685
KNN (Bootstrap)	0,814953	0,894081	0,696262	0,912685

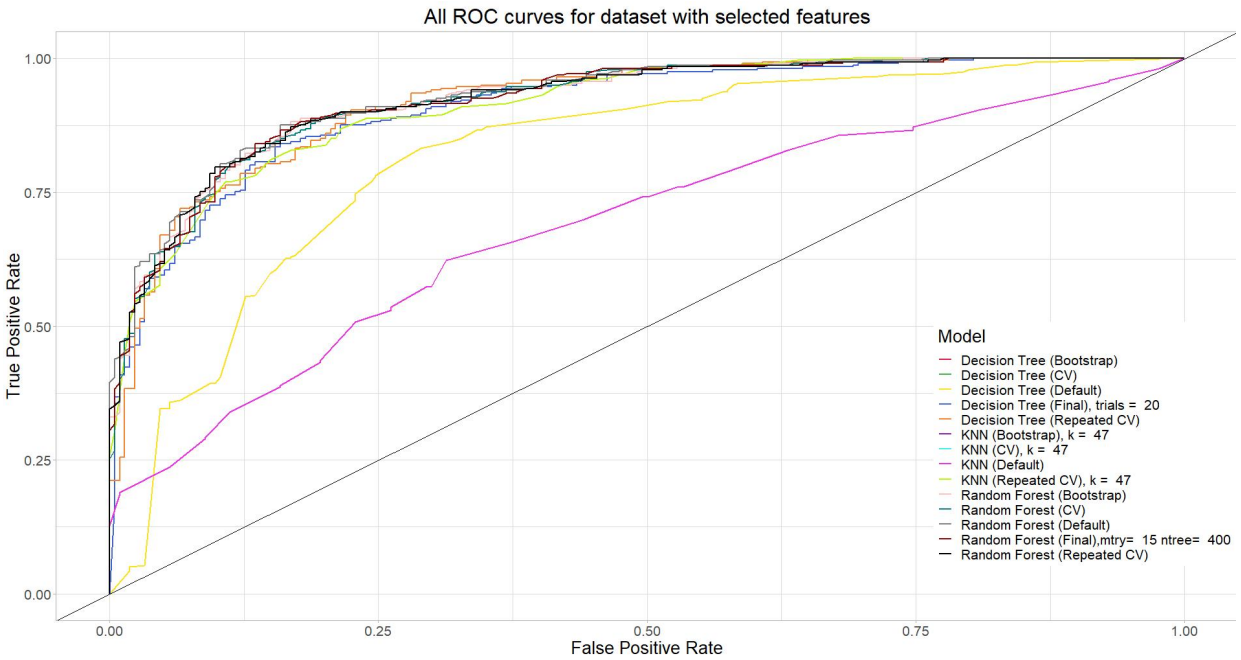

Figure 2 ROC curves for the dataset with selected features

Dataset with combined features

Regarding this dataset, the best performing algorithm on this dataset was random forests. Random Forest (Default) had the highest accuracy of 0.844840, followed by Random Forest (Repeated CV) and Random Forest (Final) with an accuracy of 0.841121. Random Forest (Repeated CV) had the highest AUC of 0.924455. Surprisingly, KNN models with resampling methods had the best sensitivity of 0.919003. As with the other two datasets, AUC was deciding metric, meaning that for the best performing model on this dataset, Random Forest (Repeated CV) was chosen. As with the selected features dataset, this dataset produced better results compared to the dataset with all features for all four metrics for all three algorithms. Compared to the selected features dataset, it produced more or less the same results; for some metrics was better (e.g. better specificity for random forest models), but for some, it was worse (AUC for decision tree models).

Table 3 Metrics for the dataset with combined features

	Accuracy	Sensitivity	Specificity	AUC
Decision Tree (Default)	0,816822	0,847352	**0,771028**	0,845387
Decision Tree (Final)	0,828037	0,878505	0,752336	0,918173
Decision Tree (CV)	0,831776	0,900312	0,728972	0,907976
Decision Tree (Repeated CV)	0,828037	0,875389	0,757009	0,921041
Decision Tree (Bootstrap)	0,828037	0,875389	0,757009	0,921041
Random Forest (Default)	**0,844860**	0,900312	0,761682	0,923283
Random Forest (Final)	0,841121	0,894081	0,761682	0,921289
Random Forest (CV)	0,837383	0,890966	0,757009	0,923822
Random Forest (Repeated CV)	0,841121	0,890966	0,766355	**0,924455**
Random Forest (Bootstrap)	0,837383	0,890966	0,757009	0,920699
KNN (Default)	0,803738	0,897196	0,663551	0,716191
KNN (CV)	0,826168	**0,919003**	0,686916	0,906746
KNN (Repeated CV)	0,826168	**0,919003**	0,686916	0,908544
KNN (Bootstrap)	0,826168	**0,919003**	0,686916	0,908544

As with the other two datasets, KNN (Default) and Decision Tree (Default) for this dataset performed worse than other models while random forests had the overall best performance. Similar to the selected features dataset, curves for models (excluding KNN (Default) and Decision Tree (Default)) were closer to each other i.e. performed more similar to each other, compared to the dataset containing all features.

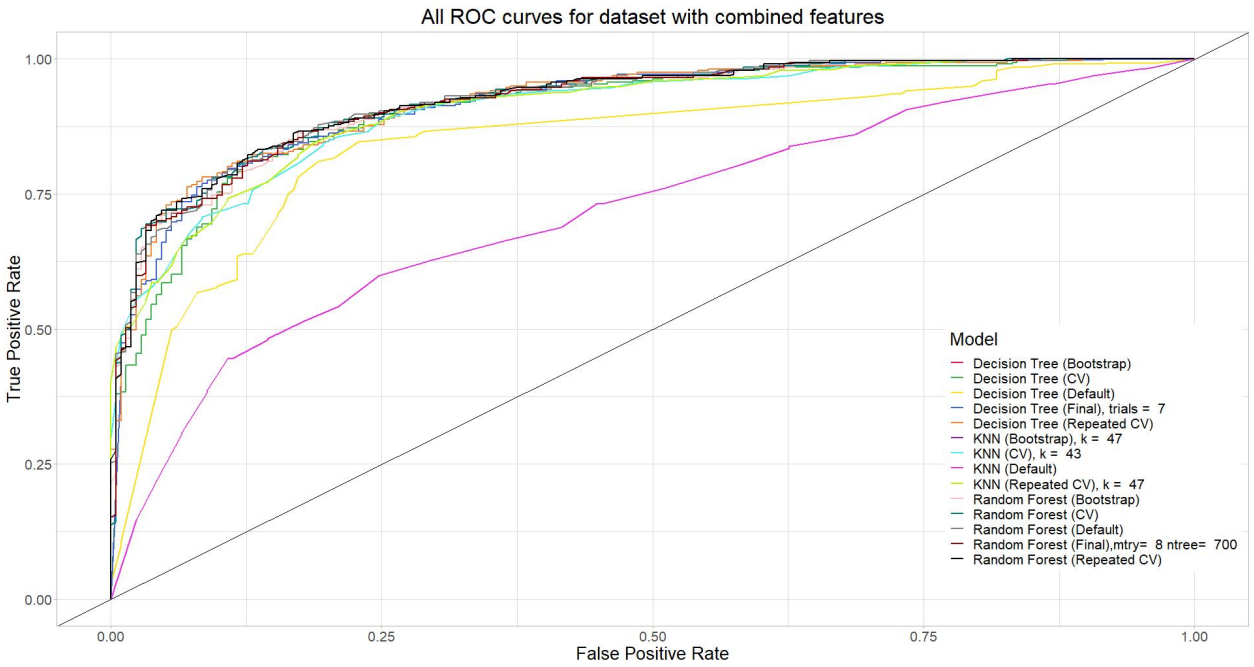

Figure 3 ROC curves for the dataset with combined features

Conclusion

In conclusion, random forests was the best performing algorithm out of three tested machine learning algorithms. Considering three different datasets used to train and test machine learning models, it can be concluded that all three datasets produced similar results. Furthermore, tuning the hyperparameters made a big difference in models performances. While tuning the hyperparameters made a somewhat noticeable difference in random forests, the difference was greater for the decision tree model on all three datasets. By choosing a more appropriate trials hyperparameter, an increase of around 5-10% in all four metrics was achieved on all three datasets.

Besides tuning the hyperparameters, using different resampling methods produced significantly better results, especially in the decision tree and KNN models, while random forests did more or less the same. Decision tree models with resampling methods achieved the same results as the Decision Tree (Final) i.e., the decision tree with optimized trials hyperparameter. When it comes to KNN, resampling methods improved all four metrics. However, the improvement was most noticeable in AUC which increased by around 30% on all three datasets. We can also say that all three resampling methods achieved the same results, on all three datasets for all three algorithms i.e., choosing whether to use cross-validation, repeated cross-validation or bootstrap sampling did not have almost any impact.

Finally, the best performance was achieved by Random Forests (Default) on the dataset with selected (most important) features with an accuracy of 0.835514, the sensitivity of 0.919003, specificity of 0.71028, and AUC of 0.926238. Based on these metrics, it is possible to confidently say that we can accurately predict the outcome of UFC fights.

References

Gray, John Scott, and Brian R. Russ. "The Ethics of Knowing the Score: Recommendations for Improving Boxing's 10-Point Must System." *Details Journal of Ethical Urban Living (ISSN: 2470-2641). March, 2019* 2.1 (2019): 51-62.

Hitkul, Karmanya Aggarwal, Neha Yadav, and Maheshwar Dwivedy. "A Comparative Study of Machine Learning Algorithms for Prior Prediction of UFC Fights." *Harmony Search and Nature Inspired Optimization Algorithms Advances in Intelligent Systems and Computing* (2018): 67-76.

Johnson, Jeremiah Douglas. *Predicting outcomes of mixed martial arts fights with novel fight variables*. Diss. University of Georgia, 2012.

Kuhn, Max, and Kjell Johnson. *Applied predictive modeling*. Vol. 26. New York: Springer, 2013.

Lantz, Brett. Machine learning with R: expert techniques for predictive modeling. Packt publishing ltd, 2019.

Mucherino, Antonio, Petraq J. Papajorgji, and Panos M. Pardalos. "K-nearest neighbor classification." *Data mining in agriculture*. Springer, New York, NY, 2009. 83-106.

Pandya, Rutvija, and Jayati Pandya. "C5. 0 algorithm to improved decision tree with feature selection and reduced error pruning." *International Journal of Computer Applications* 117.16 (2015): 18-21.

Quinlan, J. Ross. *C4. 5: programs for machine learning*. Elsevier, 2014.

Refaeilzadeh, Payam, Lei Tang, and Huan Liu. "Cross-validation." *Encyclopedia of database systems* 5 (2009): 532-538.

Spencer, Dale C. *Ultimate fighting and embodiment: Violence, gender and mixed martial arts*. Routledge, 2013.

Uttam, Atul Kumar, and Gaurav Sharma. "Application of artificial neural network based supervised machine learning based model for forecast of winner in mixed martial arts." *Recent Trends in Communication and Electronics*. CRC Press, 2021. 416-421.

Appendix A: Data set variable definitions

Suffixes

_tot – total / in the entire fight (all rounds summed up)
_rd1 – in round 1
_rd2 – in round 2
_rd3 – in round 3
_rd4 – in round 4
_rd5 – in round 5

Prefixes

red_ – the name of the fighter fighting from the red corner
blue_ – the name of the fighter fighting from the blue corner

Variables

event_title – the name of the event
event_date – date on which event took place
event_location – location of the venue where the event took place
nickname – fighters nickname
winner – winner of the fight
division – weight division of the fighters
win_method – how did the winner win
win_round – in which round did the winner win
win_time – what was the time on the clock when the fight finished
time_format – what was the time format of the fight (how many rounds, and how long are they)
referee – the name of the referee
kd – number of knockdowns
sig_str – number of significant strikes landed
sig_str_attempt - number of significant strikes thrown
sig_str_prec – accuracy of thrown significant strikes
tot_str – number of total strikes landed
tot_str_attempt – number of total strikes attempted
td – number of successful takedowns
td_attempt – number of takedowns attempted

td_acc – accuracy of takedown attempts
sub_attempt – number of submission attempts
rev - /
ctrl_time – control time (how much time did fighter control body and position of another fighter)
sig_str_head – number of strikes landed to the head
sig_str_head_attempt – number of strikes to the head attempted
sig_str_body – number of strikes landed on the body
sig_str_body_attempt – number of strikes to the body attempted
sig_str_leg – number of strikes landed on the leg
sig_str_leg_attempt – number of strikes to the leg attempted
sig_str_dist – number of strikes landed from the distance
sig_str_dist_attempt – number of strikes from the distance attempted
sig_str_clinch – number of strikes landed in the clinch
sig_str_clinch_attempt – number of strikes from in the clinch
sig_str_gnd – number of strikes landed on the ground
sig_str_gnd_attempt – number of strikes on the ground attempted

Appendix B: Project Contribution

Project Name: Ultimate Fighting Championship Winner Predictions

All project tasks were done by all project members together using Discord call and screen sharing functionalities. This means that all code was coded using a pair-programming paradigm, with all three project members coding together. The project report was written in the same way that code was coded, by all three members working together over Discord call.

Appendix C: Modifications

Regarding the modifications, we have added proportion of each output class (in the Data preprocessing section), as well as hyperparameters in text and in figures that we got for Decision Tree (Final), Random Forest (Final), KNN (CV), KNN (Repeated CV), and KNN (Bootstrap) (in the Results and Discussion section). Besides that, we added a few sentences regarding results and conclusions (in the Results and Discussion, and Conclusion sections).