Machine Learning approach to predicting Mixed Martial Arts matches

Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Data Science & Society
Department of Cognitive Science & Artificial Intelligence
School of Humanities and Digital Sciences
Tilburg University

Supervisor: Dr. A.T. Hendrickson
Second reader: Dr. E.O.J. Vanmassenhove

M. Turgut - 2041954

January 15, 2021

Preface

Before you lies the thesis "Machine Learning approach to predicting Mixed Martial Arts matches". It is written to complete the graduation requirements of the Data Science & Society program at Tilburg University. The goal of this study was to apply the methods used in similar studies to the sport of MMA.

It was an intensive period especially when we consider the current situation that we as a society are going through. I would like to thank the following people who helped me in some way during this thesis period. First of all, my supervisor Dr. A.T. Hendrickson for his support and guidance through weekly video meetings. In addition, I would like to express my appreciation for my colleagues who were also part of the weekly meetings. Observing the issues they ran into, and their way of solving them helped me to think in a broader way with regards to issues in my own study. My second reader Dr. E.O.J. Vanmassenhove was also my lecturer for the Deep Learning course. This course really sparked my interest in Deep Learning and the way Artificial Neural Networks work, which eventually led me to choose this topic. Finally, I would like to thank my parents, brother, and my fiancee for supporting me during this period.

I hope you enjoy your reading.

Mehmetcan Turgut

Abstract

For this thesis, the predictive performances of Machine Learning and Deep Learning methods have been researched for predicting Mixed Martial Arts matches. The goal of this study was to answer the research question *What is the difference in the prediction performance that can be achieved by DL models compared to traditional ML models by predicting MMA matches?* During this study, a Random Forest and an Artificial Neural Network were trained on data from the past 22 years. The data is made available by the Ultimate Fighting Championship.

Two data sets were scraped from www.ufcstats.com. These were then combined into a single data set. During the feature engineering process, great emphasis was put on preventing information leakage from occurring. Using random search algorithms for hyperparameter tuning, the Random Forest and Neural Network were able to achieve test set accuracies of 58.98% and 59.11% respectively. These accuracies are in-line with the results of other similar studies focusing on sports prediction. So it is hard to say if the sport of Mixed Martial Arts is more or less predictable compared to other sports that have commonly been researched.

However, information leakage is not always taken into account while building models for sport prediction. An additional set of models were trained to find out what the effect would be if no precautions were taken to prevent information leakage. The Random Forest and Neural Network models with information leakage achieved test set accuracies of 65.11% and 68.59% respectively.

The results of this thesis highlight the predictive performance of Random Forests and Neural Networks with regards to Mixed Martial Arts predictions. In addition, the results also highlight the effects of information leakage, not just in sports prediction, but in all of Machine Learning. Insufficient measures to prevent information leakage can lead to too optimistic results, which are not realistically achievable in real-life settings once a model is deployed.

Contents

Chapter 1

Introduction

1.1 Context

The goal of this research is to study the predictive performance of ML and DL techniques on MMA matches. How well can these techniques predict the outcome of MMA matches using official UFC data? And how does the predictability of MMA compare to other sports that have often been used in other studies?

1.1.1 Relevance of prediction in MMA

MMA combines various martial arts into a single sport. Allowing athletes from different backgrounds to compete against each other. In the last decade, it went from being a small niche sport to one of the fastest growing sports in the world, quickly becoming mainstream (Stan, 2019). In 2019 the UFC even surpassed the NFL on Instagram in terms of followers (Forbes, 2019). In addition to betting, a large amount of money is involved in sponsor and broadcasting deals. In 2018 ESPN signed a five year broadcasting deal with the UFC for 1.5 billion USD (ESPN, 2018). The UFC is currently the largest organization in MMA. They hold worldwide events and have a large roster consisting of the highest level athletes in the world. Due to the tremendous growth of the sport, the UFC was bought for 4 billion USD by WME-IMG group in 2016 (Forbes, 2016).

Aside from deals that businesses and brands make with the UFC, many deals are also made with individual fighters. Since MMA is an individual sport, the athletes themselves are the main drivers behind their popularity. Athletes become popular by reaching the top of the rankings and/or having a personality that draws people's attention. To reach the top of the rankings athletes have to keep winning matches until they eventually are allowed to face the champion of their weight class (Chase, 2011). Once an athlete reaches a championship match

they usually reach their biggest audience to date, since these are the most-watched matches (Jabbar, 2020).

MMA is a high paced sport in the sense that the rankings change quickly. It is possible for a new UFC-athlete's career to take off quickly if he/she wins matches consecutively. The most recent example of this Swedish athlete Khamzat Chimaev. He made his UFC debut on July 16, 2020, which he won. Ten days later he competed for the second time which he also won. On September 6, 2020, he won his third match. This lead to him quickly becoming a popular name in the sport even though he still is not a high ranking athlete (RT.com, 2020). In addition to this example, the outcome of one match can be career defining for an athlete. A recent example of this is the New Zealand Nigerian athlete Israel Adesanya. In October 2019 he defeated the Australian athlete Robert Whittaker and became the world champion. Resulting in his popularity to rise tremendously. A few months later he was even named "Sportsman of the year" in New Zealand (1 NEWS, 2020).

Being able to accurately predict the outcome of these important matches could help brands to decide on which athlete to sponsor. It could also help the UFC itself. As mentioned earlier the popularity of the athletes influences how many people eventually watch the UFC event. It is in favor of the UFC that a popular athlete keeps his momentum by winning matches (Saunders, 2013). Predicting outcomes of certain matches might help the organization during the matchmaking process.

1.1.2 Scientific Relevance

The possibility of using ML and DL models to predict sports results has been researched extensively. Bunker & Thabtah (2019) describe how a high prediction accuracy can be of value in sports due to for example the large amounts of money involved in betting. In addition, club owners, trainers, and managers can benefit from being able to predict sports results.

The literature has focused mainly on team sports like football and cricket, while usually using several models to find out which provides the best results. The goal of this study is to extend this body of knowledge by applying some of these techniques to the sport of MMA. Since MMA is an individual sport it might provide different results compared to a team sport like football. Another important note is that the UFC has a dedicated website for hosting statistics about matches and athletes. These are official data dating back more than twenty years. This is contrary to some of the studies in this domain, that have used less data and/or less reliable data. An overview of related works is available in Chapter 2.

Although MMA predicting is not present in current literature, attempts were made to predict UFC results by some ML enthusiasts. These attempts have even delivered great results. However, during these attempts data has been used which was not available before the match took place. Which leads to information leakage.

Information or data leakage occurs when the data used to train a model, contains information about the data the model is trying to predict (Yildirim, 2020). There are different types of information leakage which are explained in Chapter 3. In the MMA prediction cases features which provide information about the outcome variable were used. These features were not available at the time the matches took place. If the model were to be deployed, these features would not be available to predict future matches. This study made sure to prevent information leakage through extensive preprocessing. These attempts are also highlighted in Chapter 2. At the same time the effects of information leakage are highlighted by creating an additional set of models with the same features. Information leakage has been explicitly implemented in these models to find out what the increase in predictive performance is for each model compared to the original non-leaking models.

1.2 Research Questions

This report will aim to achieve its goal by answering the research question:

- *What is the difference in the prediction performance that can be achieved by DL models compared to traditional ML models by predicting MMA matches?*

This is done by analyzing the following sub-questions:

- *While preventing information leakage, which features can be used for prediction?*
- *Which hyperparameters provide the best predictive results for each model?*
- *What happens if the same features are used to build a model, but with information leakage implemented in the model?*

Answering these sub-questions allows for discussion of the following question:

- *How does the accuracy obtained by these models compare to similar cases in other sports? Is MMA more predictable than for example football?*

1.3 Findings

Findings are that Randoms Forests (a traditional ML model) delivers almost identical results compared to an ANN (DL model). Both of these models achieved very similar results on the test set (around 58%). Not accounting for information leakage provides too optimistic results and increased accuracies by 6% to 10%.

Chapter 2

Related Work

This chapter outlines a selection of the relevant literature to provide context for this study. The first section covers studies regarding team sports. Followed by studies regarding individual sports in the second section. The third section provides an overview of cases in which similar UFC data is used to build ML and DL models. The last section provides the reasoning for choosing certain algorithms in this study, while keeping related works in mind.

2.1 Literature regarding team sports

Predicting sports results using ML and DL models has been researched extensively. Bunker & Thabtah (2019) provides a literature review of studies that have used ANN's to predict these results. The reason they focus on ANN's is because ANN's are most often applied for these sport prediction cases. The biases and adjustable weights of the network provide non-linearity to the model, which makes them flexible for many types of cases (Bunker & Thabtah, 2019).

One of the notable mentions in Bunker & Thabtah (2019) is the study of McCabe & Trevathan (2008). Instead of building several models on a single sport, they created a ANN to predict several different sports: Australian Football League (AFL), Australian National Rugby League (NRL), Super Rugby, and English Premier League Football (EPL). The accuracies they achieved on these sports ranged from 54.6% to 67.5%. The features for each team consisted of historical data for that team, avoiding any subjectivity. An example of such a feature is the average of points scored by a team in the last n games.

Tax & Joustra (2015) focused on the Dutch Football competition Eredivisie. They used publicly available data to create their own data set and build two models: one regular model, and another one with betting odds incorporated into the data. Their regular model achieved an accuracy of 54.7% and their hybrid model an accuracy of 56.05%. An important note that

is made in the study is that cross-validation is not appropriate for sports result prediction, because of the sequential nature of the data. Using data from future matches to predict an earlier match outcome should be avoided (Tax & Joustra, 2015).

Bosch (2018) used several Machine Learning and Deep Learning techniques to predict the outcomes of American Football matches. He used data from the National Football League (NFL) to train his models. Bosch (2018) was able to achieve accuracies between 61% and 63%. Interesting to note is that his Support Vector Machine (SVM), Random Forest (RF) and Logistic Regression models all performed slightly better compared to the ANN model. It is important to note that Bosch (2018) performed a grid search for hyperparameter tuning. In this study, the random search approach was used for tuning, which will be further explained in section 3.3.

Pettersson & Nyquist (2017) used Recurrent Neural Networks (RNN) to predict the outcome of football matches. The reason they chose an RNN structure was that they took a unique approach to predict matches. Before the game starts the model will produce a prediction, followed by a prediction every 15 minutes during the game. The follow-up predictions take into account the previous prediction and the events that took place up until that point into the match. This leads to producing eight predictions for a single match. The first prediction only uses data that is available before the match took place. The last prediction is based on previous match data and the data of the current match. Pettersson & Nyquist (2017) were able to achieve a accuracy of around 43.96% using only data known prior to the match, and 98.63% using full-time match data.

A similar approach to that of Pettersson & Nyquist (2017) could have been taken during this study. By creating multiple predictions for each match after each round. MMA matches usually consist of three or five rounds. However, the choice was made to keep it to one prediction per match, without using any data that has been generated during the match. Since this would be more in line with the rest of the studies in the field. It would also be more in line with the relevant machine learning cases in MMA mentioned in section 2.3, and make it easier to compare results across studies and sports.

2.2 Literature regarding individual sports

Maszczyk et al. (2014) compares the predictive power of a non-linear regression model to an ANN. They tried to predict the distance of javelin throws for the Polish national team. The goal of their research was to find out if an ANN model could be used as a recruitment tool to identify potential athletes. Their problem was formulated as a regression problem. As such,

they found that the ANN provided significantly better results compared to the regression model. The absolute errors from the ANN were 16.77 compared to 29.45 from the regression model. This study highlights the potential of ANN's compared to other models in sports result prediction. An interesting note is that Maszczyk et al. (2014) used a small data set of 70 throws during this study. They used a 40, 15, 15 split for training, validation, and testing. Despite the very small amount of data, they were able to achieve great results with their ANN model.

Edelmann-nusser et al. (2002) also used an ANN and a regression model to predict individual sports outcomes. Their study focused on predicting the performance of female Olympic swimmers during the 200 Olympic Games in Sydney. They used data from 19 previous competitions and 4 weeks of training data prior to the Olympics. Their regression model achieved a mean error of 34.19, while their ANN model achieved a mean error of 12.02. It is interesting that in this study as well as the study of Maszczyk et al. (2014) ANN's significantly outperformed regression models.

Davoodi & Khanteymoori (2010) applied ANN's to predict horse racing outcomes. Data of a hundred horse races were used to perform this research. The approach they took was to build a separate ANN for each horse in the race, with the output being the finish time of that particular horse. They found that the backpropagation algorithm provides the best results. They were able to achieve an accuracy of 77% for their classification task. This paper again highlights the potential of ANN's to predict individual sports results.

2.3 MMA prediction cases

Two results that show up when searching for "UFC Machine Learning" are Medium blog posts written by Tian (2018) and Pierce (2020). They both use data from www.ufcstats.com to build ANN's to predict matches. By combining athletes and matches data they are able to achieve accuracies of 69% and 86% respectively. However, it is important to note that both of their approaches do not account for information leakage. The concept information leakage is explained in section 3.4. Tian (2018) and Pierce (2020) both use athlete summary data, which are calculated using all historical matches of a certain athlete to predict those individual matches. This leads to information leakage which causes unrealistic performance of models during training and testing. A more extensive explanation of information leakage can be found in section 3.4. For this study measures have been taken to prevent information leakage from occurring. These steps will be explained further in Chapter 4. Even though the results may not be as reliable, Tian (2018) and Pierce (2020) do provide useful ways to structure the data to allow for model training. This will be explained in Chapter 4 as well.

A similar study to this one is that of McQuaide (2019) at Stanford University. McQuaide (2019) used the same source of data (UFCStats) to scrape data on athletes and matches. Her paper does not detail the features used, and preprocessing steps taken during the research. So it is hard to tell if information leakage has been taken into account. Four types of Machine Learning algorithms were applied to this prediction problem; Stochastic Gradient Descent (SGD), ANN, Decision Tree (DT), and Gradient Boosting (GB). There are a few interesting things to note from the results. First of all, the ANN performed the worst on the test set with a test accuracy of 57.7%. It also had the worst case of overfitting, with a difference of 31.8% between the training and testing accuracies. GB performed the best with a train accuracy of 88.5% and a test accuracy of 61.2%. DT came in as a close second with a train accuracy of 80.2% and a test accuracy of 60.3%. The DT algorithm had a less severe case of overfitting compared to the ANN and GB models. The high predictive performance and the smaller difference between the training accuracies highlight the potential of using a tree-based model for this prediction problem.

2.4 Algorithms used in the literature

This chapter contains the relevant literature for this study with regards to applying Machine Learning to predict sports outcomes. All the related works contain the application of ANN's, some even used RNN's. The reason for choosing to work with an ANN usually comes down to the assumption that Neural Networks can find complex non-linear patterns within the data as mentioned by (Bunker & Thabtah, 2019). The choice was made to also work with an ANN in this study to make the results more easily comparable with other studies. Another important point is that Tian (2018) and Pierce (2020) both used ANN's as well as mentioned in the previous section.

Bosch (2018) mentions in his study how the Random Forest model worked well when the dataset had a large number of features. The ability of tree-based models to handle large numbers of features can be seen as well in the study of McQuaide (2019). She used 134 features to predict UFC matches in a very similar study. These cases highlight the potential of tree-based models for answering the research questions of this study. The choice was made to work with a Random Forest which is an ensemble model of Decision Trees. Preference was made for an RF over a DT because RF's are less prone to overfitting compared to single DT's. In addition, RF's are very suitable to work with a mixture of categorical and numerical features compared to other algorithms like the SVM (Yiu, 2019). Finally, choosing the right kernel for non-linearly separable data can also be hard to accomplish (Kumar, 2019).

Chapter 3

Methods

This chapter provides an overview of the background theory used to perform this study. The theory is based on the selection of models used. The first section is related to the theory about tree-based algorithms and Random Forests. The second section covers the theory related to neural models.

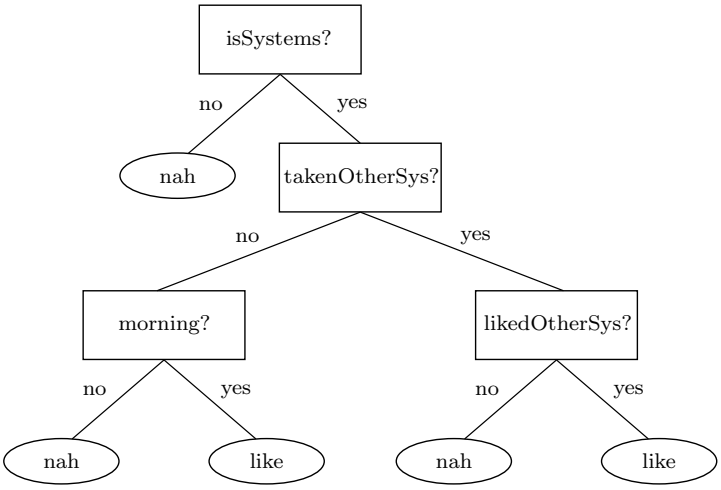
3.1 Decision Trees

Decision trees are a classic and natural model for learning, and make use of the computer science notion of "divide and conquer" (Daumé III, 2017). Decision trees work by a set of binary questions, which can be answered by either "yes" or "no". These are questions regarding the features of the dataset i.e., the X variables. According to the answer to each of these questions, also called the "feature values", the decision tree will make a guess regarding the to-be-predicted variable i.e., the y variable. These guesses are called "labels" in decision trees.

Daumé III (2017) provides a simple example of a decision tree, which is provided in figure 3.1 for illustrative purposes. The goal of this tree is to predict whether a student will enjoy a certain course. Both the student and the course are unknown. The tree will start at the top; the root node, depending on feature values it will choose a certain path through the different feature nodes until it reaches a leaf. If a leaf is reached the label of the leaf will be returned. This example contains the following nodes:

- Is the course part of "Systems"?
- Has the student taken other courses that are part of "Systems"?
- Do the lectures take place in the morning?
- Has the student liked other "Systems" courses?

Figure 3.1: Example of a decision tree (Daumé III, 2017)



The goal is to determine the optimal set of features to include in the tree, the order of these features, and which labels to return after checking features. An important advantage of decision trees is that numerical features do not need to be standardized or normalized before they can be fed into the model. In addition, categorical features do not need to be one-hot encoded.

3.1.1 Entropy & Gini Impurity

Entropy and Gini impurity are both measures of unorderedness (Hershy, 2019). As mentioned in the previous section, the goal while building decision trees is to find the optimal set of feature nodes and the order of these features nodes within the tree. Each feature node in a tree will split the samples into different subsets. If the homogeneity of labels is high in a subset the entropy and Gini impurity will be low. In that case, the tree will confidently be able to classify these examples as the dominant label in that subset. The formulas of entropy and Gini impurity are available below, where p_j is the probability of label j:

$$Entropy = -\sum_{j} p_j log_2 p_j \tag{3.1}$$

$$Gini = 1 - \sum_{j} p_j^2 \tag{3.2}$$

Algorithms select features based on the reduction of entropy or Gini impurity. This reduction is also referred to as the information gain (Hershy, 2019). Both entropy and Gini impurity can be used as a criterion while building decision trees to determine which features

to use and the order of the features, they also tend to generally produce similar results. However, because entropy contains log calculations it tends to take more time to compute. That is why generally Gini impurity is used as the default (Hershy, 2019).

3.1.2 Random Forest

A common approach to prevent overfitting while using decision trees is to build random forests (Yiu, 2019). Random forests are basically a collection of multiple decision trees, who are all trained on the training data. Instead of taking the prediction of one tree, multiple will trees will vote. The label with the most votes will be returned as a prediction.

Random forests overfit less compared to single decision trees because each tree in a forest can only choose feature nodes from a random subset of features. In addition to random features, each tree is trained on a random sample with replacement of the original training data. The randomness in each tree results in less overfitting by the random forest.

3.2 Artificial Neural Networks

ANN's are a collection of nodes positioned in a layered structure. The structure performs a series of calculations and returns an output. This structure is inspired by biological neural networks where neurons fire signals on certain conditions. The nodes in an ANN are mathematical representations of these neurons (Daumé III, 2017).

Figure 3.2: Example of a neural network with three input nodes, four hidden nodes, and a single output node.

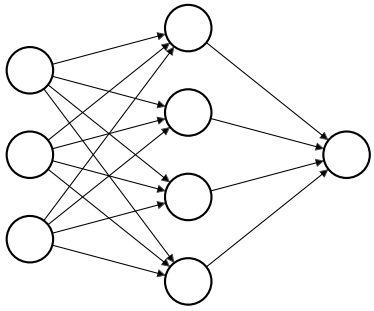


Figure 3.2 contains an example of an ANN with an input layer, a single hidden layer, and an output layer. The input layer represents the inputs that are fed into the network. Each node in the following layers receives inputs from the previous layer and performs the following

operation:

$$a = f(Wx + b) \tag{3.3}$$

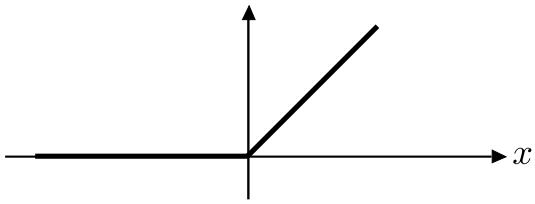
In this equation a stand for the activation of a neuron in the network. W is the set of weights that feeds into this particular neuron. x represents the activations from the previous layer. If the previous layer is the input layer, x will be equal to the input data fed into the network. b represents the bias of this particular neuron. f represents the activation function. The activation function receives the output of a neuron and transforms it into an activation which will be fed forward into the next layer (Nielsen, 2019).

3.2.1 Activation Functions

As mentioned in the previous section; activation functions take the output of a neuron and transform it into an activation. There are many types of activation functions, useful for different types of tasks e.g. classification and regression (Daumé III, 2017). This section covers two types of activation functions which are used in this study.

ReLU Activation

Figure 3.3: Graph of ReLU activation (Wikimedia Commons, 2018)

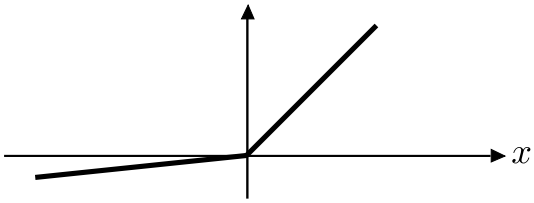


The Rectified Linear Unit (ReLU) function transforms any input it receives into 0 if it is negative. Otherwise, it returns the input (Goodfellow et al., 2016). The formula for ReLU is defined as follows:

$$ReLU(x) = max(x, 0) \tag{3.4}$$

Leaky ReLU

Figure 3.4: Graph of Leaky ReLU activation (Wikimedia Commons, 2018)

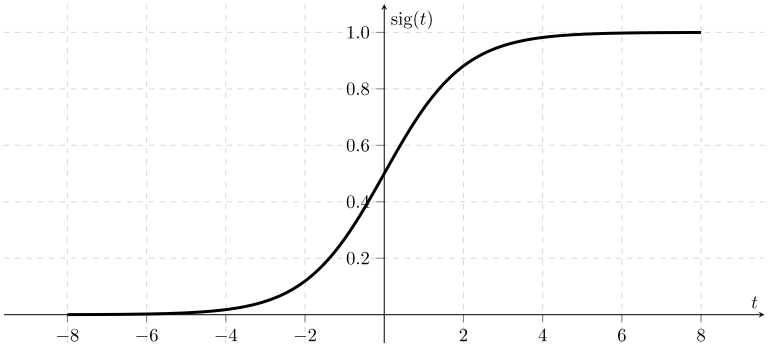


A limitation of the ReLU activation function is when weight updates cause the summed inputs for a neuron to be negative. This means that even if the inputs change the neuron is "stuck" at zero. This is also referred to as dying neurons or dying ReLU. The Leaky ReLU activation provides a solution to this problem. Instead of setting negative values to zero, they are set to a very small value close to zero (Versloot, 2019). This is done by multiplying negative inputs with a small a value. This a is usually set to 0.01. Leaky ReLU is defined below:

$$Leaky\,ReLU(x) = \begin{cases} ax & \text{if} \quad x < 0 \\ x & \text{otherwise} \end{cases} \tag{3.5}$$

Sigmoid Activation

Figure 3.5: Graph of Sigmoid function (Wikimedia Commons, 2014)



The Sigmoid function squishes any input it receives between 0 and 1. This ensures that all activations have the same range (Goodfellow et al., 2016). It is one of the most widely used activation functions. Figure 3.5 depicts a visualization of the Sigmoid function. It is

defined as follows:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.6}$$

3.2.2 Loss Function

After all the inputs have been fed in the network and an output has been produced it is time to calculate the loss. As is the case with activation functions, there are many types of loss functions, each suitable for different purposes (Daumé III, 2017). Classification tasks generally use some form of cross-entropy like categorical cross-entropy, which incorporates entropy, which is mentioned in the previous section. This study focuses on binary classification, for that reason binary cross-entropy is used while building ANN's. Binary cross-entropy is defined below. Where y is either 0 or 1, representing each label and *p(y)* is the predicted probability of that instance being 1 or positive (Godoy, 2018).

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i)) \tag{3.7}$$

3.2.3 Optimizers

After defining the loss function, the goal will be to minimize this loss function to get optimal results from the model. This is achieved by using the gradient of the loss. According to this gradient, the weights and biases in the model can be adjusted through back-propagation (Daumé III, 2017). There are algorithms available that perform this optimization process. AdaGrad (Adaptive Gradient) is an optimizer that allows for a variable learning rate to be used for each training parameter. In addition, it does not need any manual tuning of the learning rate. RMSprop is a modification of AdaGrad which limits vertical oscillations during descent. In addition, it fixes the issue of continuously decreasing the learning rate and not converging. Adaptive Moment Estimation (Adam) uses momentum to converge more quickly. However, it is computationally expensive to use and may need manual tweaking of the learning rate (Goodfellow et al., 2016) (Gandhi, 2018).

3.2.4 Regularization

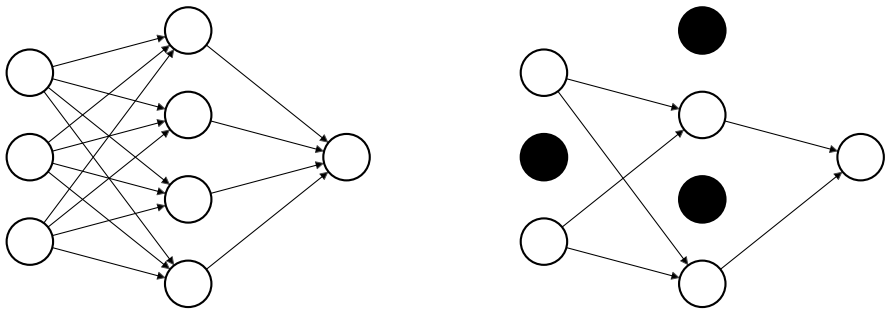
Goodfellow et al. (2016) describe regularization as modifications to a learning algorithm, with the intention to reduce the generalization error, but not the training error. This section covers regularization techniques that have been used in this study.

Batch normalization

Training deep neural networks can be hard because the input distribution for each layer changes during the training process. Batch normalization aims to coordinate the updates made of the multiple layers in the network by standardizing the activations of each input variable. (Ioffe & Szegedy, 2015).

Dropout

Figure 3.6: Example of a neural network before and after applying dropout



Dropout is a computationally inexpensive way to perform regularization (Goodfellow et al., 2016). With unlimited computation power, the best way to prevent overfitting would be to train models with all possible hyperparameters. However, since ANN's can take a long time to train, this is not feasible. Dropout randomly drops nodes from the network, along with all the in-going and outgoing connections (Brownlee, 2018). Nodes in a network can adapt to complex noise patterns in the training data, which leads to overfitting. Removing nodes randomly breaks-up these situations where the network might fit on noise (Srivastava et al., 2014).

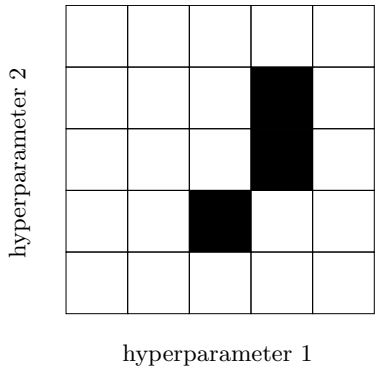
Early stopping

Before training a model, the number of iterations on the training data (epochs) can be declared. A challenge that this brings is choosing the right amount of epochs to train the data. Too little training will cause underfitting, too much training will cause overfitting and takes more time. Early stopping is a way of stopping the training when the model performance does not seem to improve any further or even starts overfitting (Prechelt, 2002). It monitors the performance of the model with each epoch. If the model starts overfitting the early stopping mechanism will be triggered and will stop the training process.

3.3 Random Search

Building models requires tweaking many hyperparameters. E.g. number of trees in a random forest, number of nodes and layers in a neural network, activation functions, etc. As mentioned in the previous subsection, in an ideal world with unlimited computational resources and time, the best solution would be to try every combination of hyperparameters possible. However, most of the time this is not possible, because the number of possible hyperparameter combinations grows quickly with each added option.

Figure 3.7: Example of grid search of two hyperparameters with black squares indicating good performing combinations



An option to overcome this issue is to perform a grid search. With a grid search, a selection of values is chosen for each hyperparameter. This selection provides a number of combinations to try. For each combination, a model will be trained. As illustrated in figure 3.7 this can be inefficient because a lot of time can be spent in "bad areas" with bad performing models. Bergstra & Bengio (2012) highlight the efficiency of another approach called "random search". Random search, like grid search, receives a set of values to try for each hyperparameter. Instead of trying all combinations, random search will try n combinations. Random search is more efficient compared to grid search because not all hyperparameters are important for the performance of the model. In addition due to the randomness, the probability of "missing" areas with good combinations is low (Bergstra & Bengio, 2012).

3.4 Information leakage

Information leakage, also known as data leakage occurs when the training data contains information about the outcome variable that the model is trying to predict (Yildirim, 2020). It is important to note that information leakage occurs without anyone being aware of it. It is also not anyone's intention for information leakage to happen. That is why it is called

leakage instead of cheating. Information leakage allows the model to make unrealistically good predictions. This is because information is revealed to the model, that would not have happened in a realistic scenario (Brownlee, 2020).

Yildirim (2020) provides some examples of information leakage. A difference is made between obvious mistakes that are easy to spot or prevent and less obvious instances of information leakage. Examples of obvious mistakes are including the outcome variable in the training data set, or including test samples in the training set. Less obvious examples are using giveaway features. Which is the case in the MMA models mentioned in section 2.3. Giveaway features are features that contain information about the outcome variable, which is not possible to get if the model would be deployed. Tian (2018) and Pierce (2020) both used summary data of all matches of athletes to predict each individual match. Other less obvious examples happen during preprocessing. Removing outliers, scaling, or normalizing are examples of things that should be done solely on the training set. Information leakage may occur if the whole data set is used during these steps.

Data processing needs to be carefully performed to prevent information leakage. Otherwise, the models might provide unrealistically good results during training and testing, setting too optimistic expectations for deployment in the real world.

Chapter 4

Experimental Setup

This chapter provides a detailed description of the experimental setup of this study. The first section is about the data. It provides a description of the data, data collection procedure, preprocessing procedure, and feature engineering procedure. The second section focuses on modeling. It covers both the random forest and ANN setup. Everything in this study has been done in Python (Van Rossum & Drake Jr, 1995).

4.1 Data

4.1.1 Dataset Descriptions

For this study official data from the UFC was used. UFCstats.com is a specific UFC website which has data from every athlete who is or has competed in the UFC. In addition, it has data available on all matches since the event *UFC 2*, which took place in 1994. Two datasets were created for this study which are described below:

Athletes

The athletes dataset contains all fighters that are available on UFCstats.com. The dataset contains 3559 athletes sorted by last name and has 20 features. The features could be divided into two groups: "personal" features and "career" features. Personal features are names, date of birth, height, weight, reach, and stance. The reason these features can be grouped together is because they only tell something about the athlete itself, nothing about his achievements or performances. The career features contain information about the performance of an athlete during matches. It contains their records and career summary statistics. E.g., "Strikes landed per minute" and "Takedown accuracy". A full overview of these features is available in table 9.1.

Matches

The matches dataset contains 5615 matches with 62 features. It contains all matches from October 1998 up until September 2020 in descending order by date. Each row again contains two types of features; features describing the standard details of a match. Like the date, location, competing athletes, whether it is a championship match, etc. In addition, it also contains features which contain data about what happened during the match and the outcome of the match e.g., the outcome of the match, the way the match ended, which round, etc. It also contains features describing the output of each athlete. An example of these are "takedowns landed by fighter1" or "significant strikes landed by fighter1". A full overview of all features is available in table 9.2.

It is important to note that each match only contains data about that particular match. The rows do not contain any information about previous performances of each athlete. In addition, each match is structured in such a way that "fighter1" is always the winning athlete. Except for matches that ended in a draw or no contest of course.

4.1.2 Data Collection

Since the data is stored on www.ufcstats.com and was not readily available as a dataset, it had to be collected specifically for this study. This was done by using a web scraping library for Python called Beautiful Soup 4 (Richardson, 2007). The script was written using Python version 3.7.3 and Visual Studio Code as editor (Microsoft, 2015). The athletes dataset was scraped by opening the "fighters" section of the website and accessing each individual fighter page from there. Matches were scraped by opening the "Events & Fights" section. From there the pages for each event were accessed. These pages contain all the matches that took place during that event. Which are usually around 10 or 12 per event. From this page, each individual match page was opened and scraped. The choice was made to not include matches from before October 1998. Because of frequently missing data in older matches. The athletes and matches data were stored in CSV files (Comma-separated values).

4.1.3 Data Preprocessing & EDA

After data collection, everything in this study was done by using Python 3.6.9. in the online environment of Google Colab. The main libraries used for preprocessing were Pandas (Wes McKinney, 2010), NumPy (Harris et al., 2020), and Matplotlib (Hunter, 2007).

Athletes Cleaning

After loading the athletes dataset, the first task was to convert all missing values into proper "NaN" values. Since many features used double dashes ("--") for missing data.

Some features were stored in the wrong formats. Many of these were numerical or DateTime data stored as strings. The date of birth column contained dates stored as "13-Jul-78" in string format. These were converted to proper Pandas DateTime format. The height, weight and reach columns were stored as follows: 5' 11", 155 lbs. and 76.0". All in string format. Height was converted to inches before storing as numerical data. Weight and reach were stored as numerical data as well. In addition, there were some career summary statistics presented as percentages like "38%". These were converted to floats between 0 and 1.

Since athletes in the matches dataset are indicated by their full name. The "first_name" and "last_name" columns were concatenated into one column. This was done to make it possible to look up athletes in both datasets.

Table 4.1: Male weightclasses in the UFC (UFC, 2020)

Weightclass	Weight limit
Flyweight	125 lbs.
Bantamweight	135 lbs.
Featherweight	145 lbs.
Lightweight	155 lbs.
Welterweight	170 lbs.
Middleweight	185 lbs.
Light Heavyweight	205 lbs.
Heavyweight	265 lbs.

Table 4.1 provides an overview of all the current male weight classes in the UFC. Back in the early days of the UFC, there was an additional weight class called "Super Heavyweight" which had no upper weight limit. However, due to the lack of sufficient number of athletes this weight class was removed. So all athletes who are heavier than 265 lbs. are removed from the dataset. In addition, all athletes with missing weights are also removed from the dataset. This results in the removal of 121 entries of which 75 were entries with missing weight values.

The athletes dataset also contained fighters who have only competed in other non-UFC events. The matches of these fighters are not included in the matches dataset, since this dataset contains only UFC events. So all fighters who have never participated in an official UFC event are removed from the dataset as well (1477 athletes).

There were three athletes with duplicate names left in the dataset. The weight of each duplicate athlete was appended to their name to make them unique. For example, the dataset at this point contained athletes named "Joey Gomez" and "Joey Gomez 155".

Table 4.2: Handling of duplicate names in the athletes dataset

Before	After
Joey Gomez	Joey Gomez
Joey Gomez	Joey Gomez 155
Michael McDonald	Michael McDonald
Michael McDonald	Michael McDonald 135
Bruno Silva	Bruno Silva
Bruno Silva	Bruno Silva 185

Athletes Missing Data

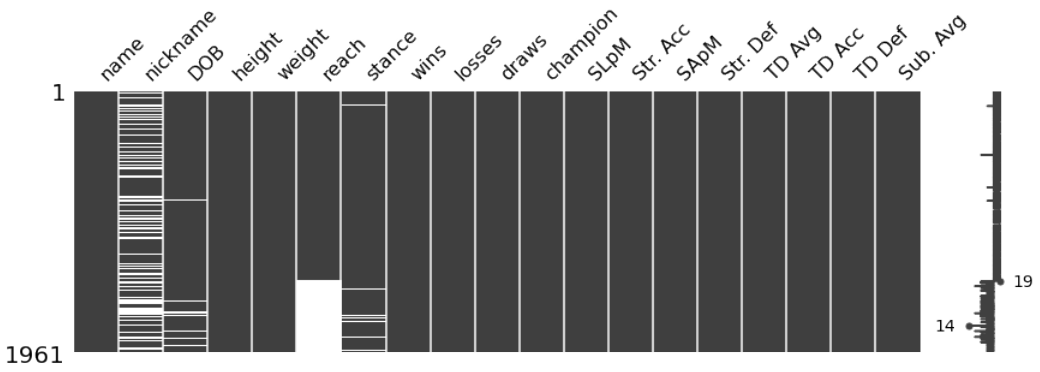
At this point, 1961 athletes of the original 3559 are still left in the dataset. After cleaning the dataset, missing data were analyzed. Table 4.3 provides an overview of missing data.

Table 4.3: Missing data in the athletes dataset

column	missing entries	ratio missing
nickname	668	0.34
DOB	57	0.03
height	3	0.00
reach	538	0.27
stance	51	0.03

Not every athlete uses a nickname so it is not surprising to see that this is the most missing feature. Since it is unlikely that the nickname of an athlete is a predictive feature in match outcomes, it can be ignored. The most important missing feature is the reach column. Since it is missing in around 27% of the entries and is an important physical feature of a athlete.

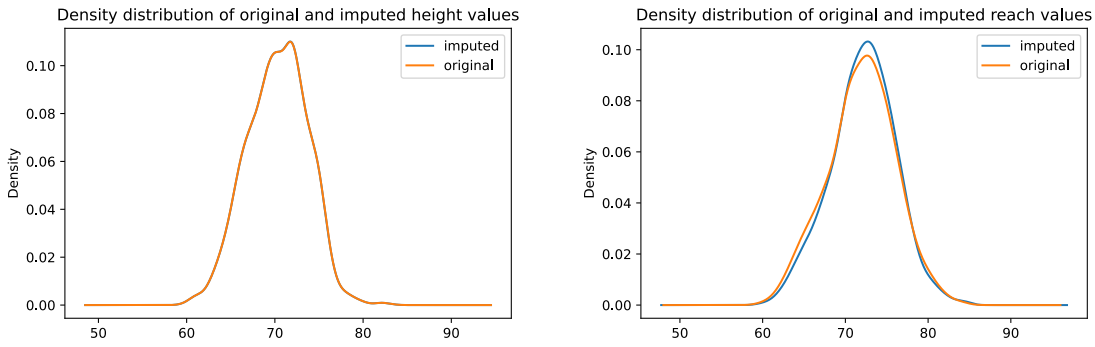
Figure 4.1: Visualization of missing data in athletes dataset



To find any missing data patterns the missing data were visualized using the missingno library (Bilogur, 2016). Figure 4.1 shows a visualization of missing data, while the dataset is sorted by reach. It immediately becomes clear that most of the data in DOB and stance columns are missing when reach is missing.

To avoid throwing out useful information, missing data in the height and reach columns were imputed using the Multiple Imputation technique also known as MICE (Buuren, 2012). Figure 4.2 shows the distributions before and after imputing missing data. These visualizations are useful to check whether the imputed data follows the same distributions as the original data. There does not seem to be a difference in the height distributions since there were only three missing entries imputed.

Figure 4.2: Density distributions of the height and reach columns, before and after imputations



Matches Cleaning

Just like in the athletes dataset, the dates were stored as strings in the matches dataset. These were converted to Pandas DateTime formats. The match statistics indicating percentages like "td%_fighter1" contained missing values when no action of that type was attempted by the athlete. This is probably due to a "division by zero" error. These missing values were replaced by zeros. In addition, all missing data in the dataset that were stored as double dashes (--) or empty values were replaced by NaN values.

The columns "ctrl_fighter1" and "ctrl_fighter2" contain the time an athlete was in control during the grappling and wrestling exchanges. The times are stored as minutes: seconds in string formats. These have been converted to seconds and stored as integers.

The athletes dataset contained three duplicate names. After cleaning the athletes dataset only one athlete is still present in the matches dataset; Bantamweight Micheal McDonald. As shown in Table 4.2, his name was changed to "Michael McDonald 135" in the athletes dataset. Instead of changing it in the matches dataset as well, the choice was made to remove the other Michael McDonald from the athletes dataset, and return this athlete's name back to its original state.

At this point, there were 58 matches, in which at least one of the athletes was not present in the athletes dataset. These were marked within a new column. So that they could be removed after the feature engineering process.

Table 4.4: All unique values in the "weight_class" column

Women's Strawweight	Women's Flyweight	Women's Bantamweight
Women's Featherweight	Flyweight	Bantamweight
Featherweight	Lightweight	Welterweight
Middleweight	Light Heavyweight	Heavyweight
Super Heavyweight	Open Weight	Catch Weight

As shown in Table 4.4, there are still "Super Heavyweight" matches present in this dataset. In addition, some matches been noted as "Open Weight". Open Weight and Super Heavyweight basically mean the same; there is no upper weight limit. So these will be removed from the dataset (4 matches). Catch Weight matches are matches with weight limits between existing weightclasses. For example the Lightweight and Welterweight limits are 155 lbs. and 170 lbs. respectively. A Catchweight match between Lightweight and Welterweight athletes could take place at for example 160 or 165 lbs. limit. These only take places very rarely though.

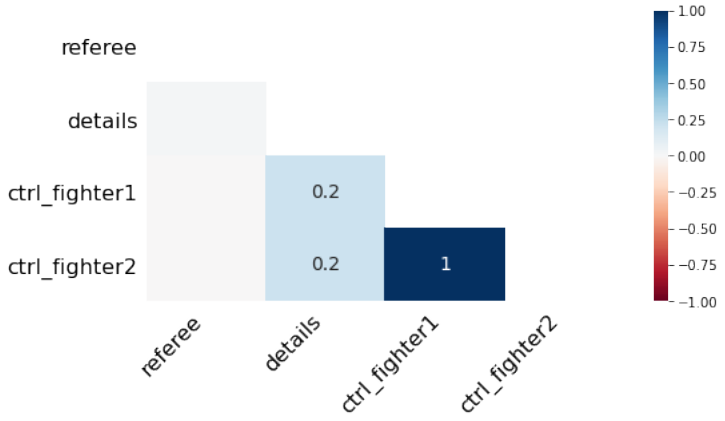
Matches Missing Data

Table 4.5: Missing data in the matches dataset

column	missing entries	ratio missing
referee	29	0.005
details	46	0.008
ctrl_fighter1	29	0.005
ctrl_fighter2	29	0.005

After removing four matches there are still 5611 entries left in the dataset. As shown in Table 4.5, there are very few cases of missing data in this dataset, especially considering the large number of features. However, it is interesting to see that three of the four features have 29 missing entries. Using the missingno library a heatmap was created to analyze these correlations (Bilogur, 2016) as shown in Figure 4.3.

Figure 4.3: Heatmap of missing data correlations in the matches dataset



If the correlation is 1 it means that every time feature1 is present, feature2 is present as well. If it is -1 it means that every time feature1 is present, feature2 is not present. Grey areas indicate that there is no correlation. Referee and details columns are removed from the dataset since they are not relevant for this study. They also do not heavily correlate with any other missing feature.

4.1.4 Feature Engineering

As mentioned in Chapter 2, there have been attempts made to build models that are able to predict UFC matches, examples are the attempts made by Tian (2018) and Pierce (2020). However, an important flaw in their approach is that they both used the "summary career" statistics from the athletes dataset as predictor variables. The problem is that these statistics change after each match, it contains the averages over all the fights of an athlete. So these statistics would not have been available at the time that these matches happened. It would only be fair to use them, to predict upcoming matches in the future. However, it is important to note that both Tian (2018) and Pierce (2020) showed some useful approaches when it comes to feature engineering and preparing the data for training. Some of these have been implemented in this study as well.

Currently, the athletes dataset contains present data and the matches dataset contains all matches that took place. With each match containing information about things that happened during that match. The goal of feature engineering was to combine information from both the athletes and matches datasets while preventing information leakage. In other words, every match in the matches dataset should only contain information that was available before the match took place. So that a model is able to be trained to predict the winner of each match while only using historical data.

Age Feature

The first feature that was added was the age of each athlete during the matches. This was done by iterating over the matches dataset. The ages were calculated by subtracting the date of birth of each athlete from the date of the match. This is possible because these features were transformed into DateTime formats during preprocessing. The reason to add this feature was because the assumption was made that age might affect the performance of an athlete.

Records Feature

Another feature that might be of predictive value is the records (wins, losses, draws) of athletes at the time the match took place. The athletes dataset contains the current records of each fighter. First "wins", "losses", and "draws" columns were added for each athlete. Then a for loop was initiated to iterate over all the matches. If an athlete appeared for the first time during this for loop, their records for their most recent match would be copied from the athletes dataset.

Table 4.6: Unique values in the outcome column

Outcome value	Description
win	fighter1 has won the match
draw	match is declared a draw
nc	match is declared a "No Contest"

The "outcome" column contains values indicating how the match has ended, see Table 4.6. As mentioned in the previous section, the dataset is structured in such a way that unless the outcome is a "draw" or "NC", fighter1 is always the winning athlete. After an athlete has already appeared for the first time in the loop, the script will get the records from previous matches. By looking at the position of each fighter (1 or 2) and the value in the "outcome" column, the script will calculate what the record was at the time of that current match. Table 4.7 summarizes this process.

Table 4.7: Example of record calculation of an athlete with three matches

Match	Record Calculation
3^{rd} match	copied from athletes dataset, since this is most recent match
2^{nd} match	record from 3^{rd} match, outcome (2^{nd} match) and position of fighter (2^{nd} match)
1^{st} match	record from 2^{nd} match, outcome (1^{st} match) and position of fighter (1^{st} match)

Win Rate Feature

After creating the records feature, it is possible to calculate the win rate of each athlete at the time of each match. The reason win rate is added because it is a simple metric that contains information about how successful an athlete has been performing, regardless of how far they are in their career. This feature was implemented in the approach of Tian (2018) as well. The win rate feature is calculated as follows:

$$win\_rate = \frac{wins}{wins + losses + draws} \tag{4.1}$$

Winning Streak Feature

Winning streaks are often mentioned by commentators during MMA matches since they convey information about the momentum of an athlete. Winning streaks might provide predictive information since they are often used by fans and professionals to make a distinction between

"good" and "great" athletes. As with the records, a column was added for each athlete's winning streak. This time however, iteration through the dataset happened backward, so in chronological order. All athletes start with a zero winning streak. Depending on the outcome of each match, the following matches are calculated accordingly.

Table 4.8: Example of winning streak calculation of an athlete with three matches

Match	Winning Steak Calculation
1^{st} match	winning streak is set to 0
2^{nd} match	calculated according to result of 1^{st} match
3^{rd} match	calculated according to result of 2^{nd} match

An example of this process is shown in Table 4.8. The winning streaks are incremented if the previous match was a win for that fighter. If the previous match was a draw or a loss, the winning streaks are reset to zero. If it was a No Contest the winning streaks remain the same. It is important to note that this feature represents winning streaks within the UFC. So all athlete's start with a zero winning streak, regardless of their performance in prior organizations. The UFC makes this distinction between overall winning streaks and UFC winning streaks as well.

Average Fight Statistics

As mentioned before, the fight statistics in each match row contain information about what happened during that match. The choice was made to replace these statistics with the averages of all previous matches for each athlete. This approach is inspired by the work of McCabe & Trevathan (2008). In their paper, they mention how they used features like "performance in previous n games", and "points-for in previous n games" to predict matches in various team sports.

To calculate these averages the dataset was again iterated through backward If an athlete appeared for the first time, the averages were all set to zero. For the following matches the averages were calculated over all previous matches for each athlete. Table 4.9 provides a summary of this process.

Using loops to create features that use data from previous or future matches requires some caution. Since there needs to be a step during each iteration to check if fighter1 in match A, is fighter1 or 2 in match B.

Using the averages can mess up the distributions for these statistics because the assumption is made that each match takes the same amount of time. This is usually true for the

Table 4.9: Example of mean calculation of an athlete with three matches

Match	Averages Calculation
1^{st} match	averages are set to 0
2^{nd} match	averages are stats from 1^{st} match
3^{rd} match	averages calculated over stats of 1^{st} and 2^{nd} match

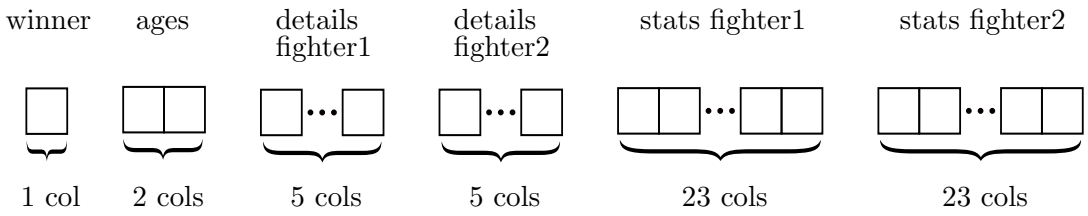
many team sports used in the study of McCabe & Trevathan (2008). However this is not the case in combat sports. "Going the distance" means that a match takes the full time it was scheduled for e.g., three rounds of five minutes. However, matches end early if a knockout or a submission takes place. So an athlete who manages to finish his/her fights early will have less output compared to an athlete who "goes the distance" in every match. To normalize this data, the averages are calculated per minute. The "round" and "time" columns are used to calculate the amount of time each match took.

4.1.5 Further Processing

After the features were created, some final cleaning/processing took place. During missing data analysis some matches were marked because they contained at least one athlete who was not present in the athletes dataset. These matches were removed from the dataset.

At this point there are 5553 matches left in the dataset. 118 (around 2%) are missing data in at least one column. These matches were removed from the dataset. In addition, matches which ended in a draw of No Contest are removed from the dataset as well to allow for binary classification. Finally, all matches in which either athlete is competing for the first time are removed from the dataset as well. Since these athletes do not have any previous data available their average fight statistics were set to zero. The columns that were created during feature engineering are converted to proper formats as well (integers and floats).

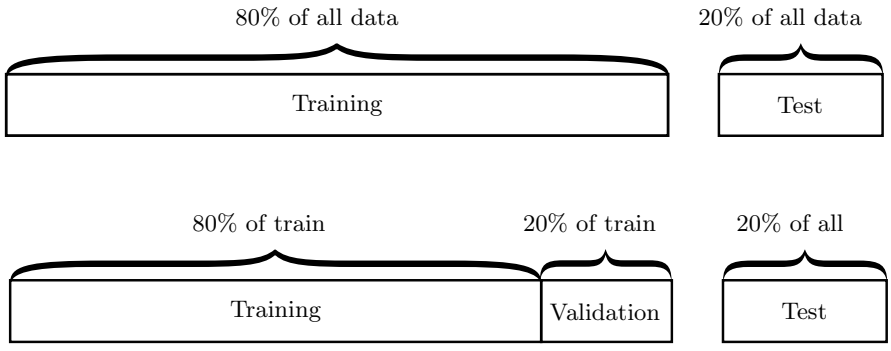
Figure 4.4: Order of features in the matches dataset



The methods used in this paragraph are inspired by the approach of Tian (2018). The matches dataset is structured in such a way that fighter1 is always the winning athlete. To prevent an imbalance in the dataset, fighter1 and fighter2 will be swapped in 50% of the rows. These rows are indexed randomly. Finally, a winner column is added as a binary value. If fighter1 is the winner the value will be 0. If fighter2 is the winner it will be 1. So in a way it provides an answer to the question "is fighter2 the winner?". Figure 4.4 shows the structure of the dataset after removing the names of the athletes. To shorten training times the number of features was reduced by replacing the "stats fighter1" and "stats fighter2" columns by subtraction of "stats fighter1" from "stats fighter2". This reduces the number of features by 23. These statistics describe some "output" of each fighter. The higher these values are the "better" the fighter has performed. But now it means that fighter2 has performed "better" compared to fighter1, because of the subtraction. An overview of all features in the final dataset is available in Table 9.3. The final dataset contains 3925 rows.

4.2 Modeling

Figure 4.5: Split between train, validation, and test data



The goal of this study is to compare the predictive performance of a traditional ML model (Random Forest in this case) to a DL model (ANN). The first section covers the setup used

to model a Random Forest. The second section covers the ANN. To keep setups consistent both models will both use the same train, validation, and test data as shown in Figure 4.5. As in the study of Tax & Joustra (2015), the matches are not randomly shuffled. This is done to avoid training on matches that happened later compared to matches in the validation and test sets. So the test set contains the most recent matches. The Random Forest and ANN models were both trained on all features. Random Search was used to find hyperparameters and tune both models (Bergstra & Bengio, 2012).

4.2.1 Random Forest

Scikit-learn (Pedregosa et al., 2011) was used to build the tree-based models in this section. A single Decision Tree was built with *DecisionTreeClassifier()* (Pedregosa et al., 2011) to serve as a baseline model. Without any tuning, the Decision Tree was able to achieve an accuracy of 51.6% on the validation set. A similar approach was taken to build a baseline Random Forest using *RandomForestClassifier()* (Pedregosa et al., 2011). Without any tuning the Random Forest already showed significant improvement. It achieved an accuracy of 57.3% on the validation set.

After building the baseline models a hyperparameter grid was set up to initiate the Random Search algorithm. To limit the training time five hyperparameters were chosen to include in the grid. Which are listed in Table 4.10. These features were chosen because they provide a great range of model architectures.

Table 4.10: Hyperparameters and values used in the search grid. Descriptions are derived from (Pedregosa et al., 2011)

Hyperparameter	Description	Values
n_estimators	Number of trees in the forest	100 to 500, 20 steps
max_depth	Limit to how tall a tree can become	5 to 50, 5 steps
min_samples_split	Number of samples needed in a node to split it	1, 3, 5, 10, 20
min_samples_leaf	Number of samples needed in a each leaf	1, 3, 5, 10, 20
bootstrap	Indicates if bootstrap samples can be used	True or False

Note that the values set for each hyperparameter in the grid were set after multiple iterations of Random Searches. If the best performing model contained a value that is near the beginning or end of a range, the range is increased to allows the Random Search to try more possibilities for that hyperparameter.

4.2.2 Random Forest Results

RandomizedSearchCV() (Pedregosa et al., 2011) was used to perform the Random Search. The parameters of the best model are listed in table 4.11.

Table 4.11: Hyperparameters of best model returned by the Random Search

Hyperparameter	Value
n_estimators	310
max_depth	27
min_samples_split	3
min_samples_leaf	10
bootstrap	True

The tuned Random Forest achieved a validation set accuracy of 59.24%. Which is 1.9% higher compared to the baseline Random Forest. Results are shown in Table 4.12. Only the tuned Random Forest was evaluated on the test set. It achieved a test accuracy of 58.98%.

Table 4.12: Validation accuracies of all tree based models

Model	Validation Accuracy
Baseline Decision Tree	51.59%
Baseline Random Forest	57.32%
Tuned Random Forest	59.24%

4.2.3 Neural Network

The Keras library (Chollet et al., 2015) was used to build ANN's. As with the Random Forest, Random Search was performed to find the best hyperparameters for the ANN. The Keras Tuner library (O'Malley et al., 2019) was used to perform the Random Search with the *Sequential()* model of Keras. The values for each hyperparameter are shown in Table 4.13. Scikit-learn's *StandardScaler()* function was used to scale the feature values before feeding them to the models. ReLU and Leaky ReLU activation was used for input and hidden layers. Sigmoid was used for the output layer.

Table 4.13: Hyperparameters and values used in Random Search

Hyperparameters	Values
Optimizer	adagrad, adam, rmsprop
Activation input and hidden layer	ReLU, Leaky ReLU
Alpha value for Leaky ReLU	0.1, 0.2, 0.3
Dropout rate	0.0, 0.1, 0.2
Batch Normalization	True, False
Number of nodes per hidden layer	4 to 16, step size of 2
Number of hidden layers	1 to 10

4.2.4 Neural Network Results

Many Random Searches were performed to find the optimal hyperparameter values. With regards to nodes and layers, many options were tried. Even models with thousands of nodes and dozens of layers were built. However, these results were comparable to smaller networks. Since most of the studies mentioned in Chapter 2 use small networks in terms of nodes and layers, the choice was made to perform a Random Search with similar small values for nodes and layers as is shown in table 4.13. In addition, during experimentation, the choice was made not to implement Batch Normalization and Drop Out since the search is limited to smaller networks. Early Stopping provided sufficient regularization results. The results of the Random Search are available in Table 4.14. This model achieved a validation accuracy of 61.62% and a test accuracy of 59.11%.

Table 4.14: Hyperparameters of best model ANN model

Hyperparameters	Values
Optimizer	RMSprop
Activation input and hidden layers	Leaky ReLU
Alpha value Leaky ReLU	0.3
Dropout rate	0.0
Batch Normalization	False
Number of hidden layers	6
Number of nodes per hidden layer	6, 4, 10, 6, 12, 14

4.3 Adding Information Leakage

During this study, great emphasis is put on training a model while preventing any information leakage from occurring. As mentioned in section 2.3, Tian (2018) and Pierce (2020) both built models that achieved high predictive performances in MMA. However, these results may be overly optimistic due to them using giveaway features as explained in section 3.4. To determine the effect of using giveaway features, a new data set was created with information leakage purposely built into the features. This will provide an answer to the third research question of this study. Two additional models (ANN & RF) were built to be trained on this data set.

4.3.1 Feature Engineering

For this data set, the same features are used. However, this time the records, winning rate, and the fight statistics of each athlete were not calculated backward in time. So for example, the feature regarding "average kicks thrown" is the same for all matches for each athlete. This produces similar giveaway features compared to those used in the cases of Tian (2018) and Pierce (2020).

4.3.2 Random Forest Information Leakage Results

The Random Forest and ANN models with information leakage have been trained in the same way as the regular models. The same hyperparameter values were used for the Random Search algorithms. The hyperparameter values found by the Random Search for the Random Forest model are listed in table 4.15.

Table 4.15: Hyperparameters of best model returned by the Random Search

Hyperparameter	Value
n_estimators	436
max_depth	16
min_samples_split	10
min_samples_leaf	1
bootstrap	True

The Random Forest with information leakage achieved a validation accuracy of 62.70% and a test accuracy of 65.11%. Which is a large improvement compared to the Random Forest without information leakage. A comparison of the accuracies between these models is available in table 4.16 below. It is interesting to see that the model with information leakage performs better on the test set compared to the validation set. The regular model produced similar results on the validation and test set.

Table 4.16: Accuracies of both Random Forest models

Model	Validation Accuracy	Test Accuracy
Regular RF	59.24%	58.98%
RF with Information Leakage	62.70%	65.11%

4.3.3 ANN Information Leakage Results

Table 4.17 provides an overview of the hyperparameter values found by the Random Search algorithm. It is interesting to see that both the regular ANN and the ANN with information leakage have many similarities in hyperparameters. Both use the RMSprop optimizer and Leaky ReLU activation with an alpha value of 0.3. In addition, the number of layers is close as well; 6 and 7 layers respectively.

Table 4.17: Hyperparameters of best model ANN model with IL

Hyperparameters	Values
Optimizer	RMSprop
Activation input and hidden layers	Leaky ReLU
Alpha value Leaky ReLU	0.3
Dropout rate	0.0
Batch Normalization	False
Number of hidden layers	7
Number of nodes per hidden layer	10, 16, 10, 14, 12, 4, 16

The ANN with information leakage achieved an accuracy of 67.48% on the validation set and an accuracy of 69.43% on the test set. This is an improvement of 9.48% compared to the regular ANN. A comparison of the accuracies achieved by both ANN's is available in Table 4.18 below. Another interesting note is that the accuracies of the ANN with information leakage are very similar, whereas the original ANN had a little reduction performance on the test set compared to the validation set.

Table 4.18: Accuracies of both ANN models

Model	Validation Accuracy	Test Accuracy
Regular ANN	61.62%	59.11%
ANN with Information Leakage	68.65%	68.59%

Chapter 5

Results

5.1 Features

Preprocessing and Feature Engineering resulted in a new dataset, which combines information from the athletes and matches datasets. Feature engineering allowed for the creation of 35 features while preventing any information leakage. It contains historical information about ages, records, and fight output per athlete. An overview of the dataset and its features is available in Table 9.3

5.2 Models

The results of the Random Forest and ANN models are summarized in Table 5.1 on the next page. The Random Search resulted in a Random Forest with 310 trees and a maximum depth of 27 splits. In addition, an ANN with a (35, 6, 4, 10, 6, 12, 14, 1) structure provided the best results. The two models produced very similar results in test sets while preventing information leakage. As mentioned in the previous section regularization methods were limited to Early Stopping (*patience = 3*).

An additional Random Forest and ANN were built, which used giveaway features for information leakage to occur. Both models showed a large improvement while using giveaway features. Adding information leakage to the Random Forest model increased accuracy on the test set by 6.13%. Adding information leakage to the ANN model caused an increase of 9.48% on the test set. It is interesting to see that without information leakage both models performed very similarly on the test set. However, adding information leakage resulted in a difference of 3.48% between the ANN and Random Forest.

Table 5.1: Hyperparameters and results of the Random Forest and ANN models

Model		Validation Accuracy	Test Accuracy	Difference
Random Forest		59.24%	58.98%	-0.26%
n_estimators	310			
max_depth	27			
min_samples_split	3			
min_samples_leaf	10			
bootstrap	True			
Random Forest with information leakage		62.70%	65.11%	2.41%
n_estimators	436			
max_depth	16			
min_samples_split	10			
min_samples_leaf	1			
bootstrap	True			
Neural Network		61.62%	59.11%	-2.51%
optimizer	RMSprop			
activation	Leaky ReLU			
alpha value Leaky ReLU	0.3			
dropout rate	0.0			
batch normalization	False			
hidden layers	6			
nodes per hidden layer	6, 4, 10, 6, 12, 14			
Neural Network with information leakage		68.65%	68.59%	-0.06%
optimizer	RMSprop			
activation	Leaky ReLU			
alpha value Leaky ReLU	0.3			
dropout rate	0.0			
batch normalization	False			
hidden layers	7			
nodes per hidden layer	10, 16, 10, 14, 12, 4, 16			

Table 5.2: Five most important features for each Random Forest

Regular RF	RF with information leakage
win_rate_fighter2	win_rate_fighter_2
diff_avg_td_att_total	win_rate_fighter_1
diff_avg_td_landed_total	diff_avg_ground_land
diff_avg_subatt	wins_fighter1
diff_avg_ground_att	diff_avg_head_land

Table 5.2 shows an overview of the five most important features for each Random Forest model. It is interesting to see the same feature was the most important in both models. Comparing both models it becomes clear that the model with information leakage get predictive information out of features related to the records of the athletes (3 out of 5). While the regular model mainly looks at fight output features (4 out of 5). A full overview of feature importance for both models is available in Figure 9.1 and 9.2 in the appendix.

Another interesting thing to note is that the top five most important features of the regular model only consists of wrestling and grappling related metrics. This means that if there is a large difference between athletes based on wrestling/grappling based output, the model uses that as predictive information.

Chapter 6

Discussion

The goal of this study was to compare the predictive performance of a DL model to a traditional ML model. The answer to the first sub-question *"While preventing information leakage, which features can be used for prediction?"* is presented as a final data set in Table 9.3. This set of 35 features captures information from both the athletes and matches data set, while preventing any information leakage. The answer to the second sub-question *"Which hyperparameters provide the best predictive results for each model?"* is provided in Table 5.1. The third research question *"What happens if the same features are used to build a model, but with information leakage implemented in the model?"* is answered in section 4.3.

After many iterations of Random Searches these hyperparameter settings provided the best results. With regards to the main question *"What is the difference in the prediction performance that can be achieved by DL models compared to traditional ML models by predicting MMA matches?"* the results are surprising. The models provided almost identical results on the test sets. The difference of accuracy on the validation set was 2.38%. On the test set, the difference was very small: 0.13%.

How does the predictability of MMA compare to other sports in comparable studies? The accuracies achieved in this study are lower compared to the results of Tian (2018) and Pierce (2020), who both built models for MMA, and achieved accuracies of 69% and 86% respectively. However, because this study put emphasis on only using data that is available prior to a match, the results are much more in line with those of other studies in the field. McCabe & Trevathan (2008) tested several team sports in their study. They achieved test set accuracies ranging from 54.6% to 67.5%. In addition, Tax & Joustra (2015) analyzed the Dutch Football competition Eredivisie. They achieved accuracies 54.7% and 56.5% with two different models. Looking at these results, the conclusion can be made that predictability of MMA is similar to other sports. To find out if the Random Forest and ANN would be able

to achieve similar results as Tian (2018) and Pierce (2020) two new models were built with giveaway features. The RF and ANN achieved results of 65.11% and 68.59%, which is a large improvement compared to the regular models. Adding information leakage to the models does indeed improve prediction performance. The ANN in this study achieved a very similar accuracy to that of Tian (2018).

The workflow in Data Science is often described as spending 80% of the time gathering, cleaning, and preprocessing the data (Press, 2016). The other 20% of the time is spent on actual modeling. I would say that it was the case in this study as well. However, contrary to people describing it as the least enjoyable task, I found it quite interesting and enjoyable to do.

It would be interesting to see if performance can be improved by adding more features to the models. However, the problem is that it would take more effort to gather reliable data from other sources than UFCStats.com. Tax & Joustra (2015) for example included betting odds in their second model. This seemed to slightly improve model performance. Features that capture information about things outside of the actual match might be useful to improve model results.

For example, athletes are weighed in one day prior to each match to check if they weigh appropriately for their weight class. Athletes usually do not compete in their "natural" weight class, but often try to lose as many pounds as possible to compete in a lower weight class. After the weigh-in they try to gain most of that weight back to achieve a size advantage against their opponent. The problem is that because nowadays everyone cuts weight, it would be disadvantageous to not do it. Losing large amounts of weight in a short time involves draining as much water from your body as possible. The intensity of these weight cuts might influence the performance of the athlete in their match. Another example might be injuries prior to a match. These might affect the performance of an athlete as well. However, the problem with these examples is that often it is not possible to find reliable data about these things.

Another idea for future research is to build specific models for each weight class. Because athletes who are competing in a 135 lbs. weight class have very different styles compared to athletes who compete in a 265 lbs. weight class. Matches in the lighter weight classes are more high paced. So the output of each athlete would be more compared to Heavyweight athletes who move much slower. In addition, knockouts happen more often in heavier weight classes. This may result in models with better predictive performance.

Chapter 7

Conclusion

The predictive power of Random Forests and Artificial Neural Networks in MMA were compared in this study. A new data set was created through data collection, preprocessing, and feature engineering. The features in this dataset listed in Table 9.3 provide the answer to the first sub-question of this study: *"While preventing information leakage, which features can be used for prediction?"*. A Random Search algorithm was used to find which hyperparameters provide the best results in each model. These are available in Table 5.1 and provide the answer for the second research question in this study: *"Which hyperparameters provide the best predictive results for each model?"*. The effect of information leakage are highlighted in section 4.3 and table 5.1. This provides an answer to the third sub question *"What happens if the same features are used to build a model, but with information leakage implemented in the model?"*. The main question of this study: *"What is the difference in the prediction performance that can be achieved by DL models compared to traditional ML models by predicting MMA matches?"* can be answered by comparing the results of each model. The Random Forest and ANN provided similar results on the validation and test sets. Random Forest and ANN achieved accuracies of 58.98% and 59.11% respectively. These results are similar to those of other studies. Future studies could look into the possibility of adding more features or testing different ML / DL algorithms. More data will probably make a significant difference in model performance since the data set used in this study is relatively small. However, this would be hard to achieve since this study already covers data since the beginning of the UFC. In other words, there is no more data to collect. Perhaps data augmentation techniques might provide a solution for this problem, or limiting the number of features used to build models.

Chapter 8

References

1 NEWS. (2020). *MMA fighter Israel Adesanya awarded Sportsman of the Year at Halberg Awards.* Retrieved from https://www.tvnz.co.nz/one-news/sport/other/mma-fighter-israel-adesanya-awarded-sportsman-year-halberg-awards#:~:text=MMA%20fighter%20Israel%20Adesanya%20awarded%20Sportsman%20of%20the%20Year%20at%20Halberg%20Awards,-Thu%2C%20Feb%2013&text=MMA%20fighter%20Israel%20Adesanya%20has,win%20the%20award%20since%201953.

Bergstra, J., & Bengio, Y. (2012, February). Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, *13*(null), 281–305.

Bilogur, A. (2016). missingno. Retrieved from https://github.com/ResidentMario/missingno

Bosch, P. (2018). Predicting the winner of NFL-games using Machine and Deep Learning. Retrieved from https://beta.vu.nl/nl/Images/werkstuk-bosch{_}tcm235-888637.pdf

Brownlee, J. (2018). *A gentle introduction to dropout for regularizing deep neural networks.* Retrieved from https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

Brownlee, J. (2020). *How to Avoid Data Leakage When Performing Data Preparation.* Retrieved from https://machinelearningmastery.com/data-preparation-without-data-leakage/

Bunker, R. P., & Thabtah, F. (2019). A machine learning framework for sport result prediction. *Applied Computing and Informatics*, *15*(1), 27–33. Retrieved from https://doi.org/10.1016/j.aci.2017.09.005 doi: 10.1016/j.aci.2017.09.005

Buuren, S. (2012). *Flexible imputation of missing data.* doi: 10.1201/b11826

Chase, G. (2011). *MMA: What Do Fighters Have To Do for a Title Shot?* Retrieved from https://bleacherreport.com/articles/956977-what-do-fighters-have-to-do-for-a-title-shot

Chollet, F., et al. (2015). *Keras.* https://keras.io.

Daumé III, H. (2017). *A Course in Machine Learning.* Retrieved from http://ciml.info/

Davoodi, E., & Khanteymoori, A. R. (2010). Horse racing prediction using Artificial Neural Networks. *Proc. of the 11th WSEAS Int. Conf. on Neural Networks, NN '10, Proceedings of the 11th WSEAS Int. Conf. on Evolutionary Computing, EC '10, Proc. of the 11th WSEAS Int. Conf. on Fuzzy Systems, FS '10*(August), 155–160.

Edelmann-nusser, J., Hohmann, A., & Henneberg, B. (2002). Modeling and prediction of competitive performance in swimming upon neural networks. *European Journal of Sport Science*, *2*(2), 1-10. Retrieved from https://doi.org/10.1080/17461390200072201 doi: 10.1080/17461390200072201

ESPN. (2018). *ESPN to broadcast 30 UFC events per year during 5-year deal.* Retrieved from https://www.espn.com/mma/story/_/id/23581729/espn-ufc-reach-five-year-television-rights-deal

Forbes. (2016). *UFC Sale Officially Closes For $4 Billion, Fertitta Brothers Earn Huge Payday.* Retrieved from https://www.forbes.com/sites/noahkirsch/2016/08/22/ufc-sale-officially-closes-for-4-billion-fertitta-brothers-earn-huge-payday/?sh=7f736d354eda

Forbes. (2019). *The UFC Has Surpassed The NFL On Instagram, And There Are 2 Big Reasons.* Retrieved from https://www.forbes.com/sites/brianmazique/2019/08/25/the-ufc-has-surpassed-the-nfl-on-instagram-and-there-are-2-big-reasons/?sh=5639eb0a2417

Gandhi, R. (2018). *A look at gradient descent and rmsprop optimizers.* Retrieved from https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b

Godoy, D. (2018). *Understanding binary cross-entropy / log loss: a visual explanation.* Retrieved from https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* MIT Press. (http://www.deeplearningbook.org)

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... Oliphant, T. E. (2020, September). Array programming with NumPy. *Nature*, *585*(7825), 357–362. Retrieved from https://doi.org/10.1038/s41586-020-2649-2 doi: 10.1038/s41586-020-2649-2

Hershy, A. (2019). *Gini Index vs Information Entropy.* Retrieved from https://towardsdatascience.com/gini-index-vs-information-entropy-7a7e4fed3fcb

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. doi: 10.1109/MCSE.2007.55

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, *abs/1502.03167*. Retrieved from http://arxiv.org/abs/1502.03167

Jabbar, N. (2020). *The 16 Biggest Pay-Per-View Buys In UFC History.* Retrieved from https://www.sportbible.com/ufc/mma-news-the-16-biggest-pay-per-view-buys-in-ufc-history-20200326

Kumar, N. (2019). *Advantages and Disadvantages of SVM (Support Vector Machine) in Machine Learning.* Retrieved from http://theprofessionalspoint.blogspot.com/2019/03/advantages-and-disadvantages-of-svm.html

Maszczyk, A., Gołaś, A., Pietraszewski, P., Roczniok, R., Zając, A., & Stanula, A. (2014). Application of Neural and Regression Models in Sports Results Prediction. *Procedia - Social and Behavioral Sciences*, *117*, 482–487. doi: 10.1016/j.sbspro.2014.02.249

McCabe, A., & Trevathan, J. (2008). Artificial intelligence in sports prediction. *Proceedings - International Conference on Information Technology: New Generations, ITNG 2008*, 1194–1197. doi: 10.1109/ITNG.2008.203

McQuaide, M. (2019). *Applying Machine Learning Algorithms to Predict UFC Fight Outcomes.* Retrieved from http://cs229.stanford.edu/proj2019aut/data/assignment_308832_raw/26647731.pdf

Microsoft. (2015). *Visual studio code.*

Nielsen, M. (2019). *Neural Networks and Deep Learning.* Retrieved from http://neuralnetworksanddeeplearning.com

O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L., et al. (2019). *Keras Tuner.* https://github.com/keras-team/keras-tuner.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pettersson, D., & Nyquist, R. (2017). Football Match Prediction using Deep Learning Recurrent Neural Network Applications. Retrieved from `http://publications.lib.chalmers.se/records/fulltext/250411/250411.pdf`

Pierce, C. (2020). *Predicting UFC Fights With Machine Learning.* Retrieved from `https://towardsdatascience.com/predicting-ufc-fights-with-machine-learning-5d66b58e2e3a`

Prechelt, L. (2002). Early Stopping - But When? *Orr G.B.*, *1524*. doi: https://doi-org.tilburguniversity.idm.oclc.org/10.1007/3-540-49430-8_3

Press, G. (2016). *Cleaning big data: Most time-consuming, least enjoyable data science task, survey says.*

Richardson, L. (2007). Beautiful soup documentation. *April*.

RT.com. (2020). *'They've got to keep that market': UFC star Chimaev's rise is part of plot linked to Russian champ Khabib's retirement, says rival.* Retrieved from `https://www.rt.com/sport/510187-khabib-ufc-retirement-chimaev/`

Saunders, A. (2013). *Pros and Cons of the UFC Ignoring Their Own Rankings When Matchmaking.* Retrieved from `https://bleacherreport.com/articles/1608869-pros-and-cons-of-the-ufc-ignoring-their-own-rankings-when-matchmaking`

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929-1958. Retrieved from `http://jmlr.org/papers/v15/srivastava14a.html`

Stan, S. V. (2019). Strategic Management in Sports. The Rise of MMA Around the World – The Evolution of the UFC. *Ovidius University Annals, Economic Sciences Series*, *XIX*(1), 540–545.

Tax, N., & Joustra, Y. (2015). Predicting The Dutch Football Competition Using Public Data: A Machine Learning Approach. *Transactions on knowledge and data engineering*, *10*(SEPTEMBER), 1–13. Retrieved from `https://www.researchgate.net/profile/Niek_Tax/publication/282026611_Predicting`

_The_Dutch_Football_Competition_Using_Public_Data_A_Machine_Learning_Approach/links/5601a25108aeb30ba7355371/Predicting-The-Dutch-Football-Competition-Using-Public-Data-A-Machine-Learning-Approach.pdf doi: 10.13140/RG.2.1.1383.4729

Tian, Y. (2018). *Predict UFC Fights with Deep Learning II — Data collection and implementation in PyTorch.* Retrieved from https://medium.com/@yuan_tian/predict-ufc-fights-with-deep-learning-ii-data-collection-and-implementation-in-pytorch-ff7a95062554

UFC. (2020). *Ufc stats.* Retrieved from http://www.ufcstats.com

Van Rossum, G., & Drake Jr, F. L. (1995). *Python tutorial.* Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands.

Versloot, C. (2019). *Leaky ReLU: improving traditional ReLU.* Retrieved from https://www.machinecurve.com/index.php/2019/10/15/leaky-relu-improving-traditional-relu/

Wes McKinney. (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt & Jarrod Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (p. 56 - 61). doi: 10.25080/Majora-92bf1922-00a

Wikimedia Commons. (2014). *Sigmoid-function-2.* Retrieved from https://commons.wikimedia.org/wiki/File:Sigmoid-function-2.svg

Wikimedia Commons. (2018). *Relu and nonnegative soft thresholding functions.* Retrieved from https://commons.wikimedia.org/wiki/File:ReLU_and_Nonnegative_Soft_Thresholding_Functions.svg

Yildirim, S. (2020). *Data Leakage in Machine Learning.* Retrieved from https://towardsdatascience.com/data-leakage-in-machine-learning-6161c167e8ba

Yiu, T. (2019). *Understanding Random Forest.* Retrieved from https://towardsdatascience.com/understanding-random-forest-58381e0602d2

Chapter 9

Appendices and Supplementary Materials

Table 9.1: Athletes dataset features (UFC, 2020)

Feature	Data type	Description
first_name	string	First name of athlete.
last_name	string	Last name of athlete.
nickname	string	Nickname of athlete, if he/she has any.
DOB	string	Date of birth in 01-Jan-99 format.
height	string	Height in feet & inches e.g. 5'11".
weight	string	Weight in pounds e.g. 155 lbs.
reach	string	Distance in inches between fingertips, aka wingspan e.g. 71.0".
stance	string	Indicates which foot is placed on front. Orthodox: left, Southpaw: right.
wins	integer	Amount of matches the athlete has won.
losses	integer	Amount of matches the athlete has lost.
draw	integer	Amount of matches that ended in a draw.
champion	string	Indicates whether the athlete is currently the champion of a weight class.
SLpM	float	Significant strikes landed per minute.
Str.Acc	string	Significant striking accuracy in percentages.
SApM	float	Significant strikes absorbed per minute.
Str.Def	string	Significant strike defence. The % of opponents strikes that did not land.
TD_Avg	float	Average takedowns landed per 15 minutes.
TD_Acc	string	Takedown accuracy in percentages.
TD_Def	string	Takedown defense. The % of opponent's takedowns that did not land.
Sub.Avg	float	Average submissions attempted per 15 minutes.

Table 9.2: Matches dataset features, note that for each feature which ends in "fighter1" there is an equivalent feature available for fighter2 (UFC, 2020)

Feature	Data type	Description
date	string	Date of the match
location	string	City and country match took place in
attendance	integer	Amount of people that attended the event
fighter1	string	First and last name of the winning athlete
fighter2	string	First and last name of losing athlete
weight_class	string	Name of the weight class
championship	boolean	Indicates whether it was a championship match
bonus	string	Whether a bonus was awarded for performance and if so, which one
method	string	Indicates how the match ended Eg knock out or jury decision
round	integer	Indicates in which round of the match the fight ended
time	string	Amount of time passed in the last round of the match
time_format	string	Number of rounds and minutes per round
referee	string	Name of the referee that supervised the match
details	string	Additional description of the match
kd_fighter1	integer	Number of knockdowns scored by fighter 1
sigstr_land_total_fighter1	integer	Number of significant strikes by fighter 1
sigstr_att_total_fighter1	integer	Number of significant strikes attempted by fighter 1
sigstr%_fighter1	integer	Percentage of strikes landed by fighter 1
total_str_land_total_fighter1	integer	Total strikes landed by fighter 1
total_str_att_fighter1	integer	Total strikes attempted by fighter 1
td_land_total_fighter1	integer	Total take downs landed by fighter 1
td_att_total_fighter1	integer	Total take downs attempted by fighter 1
td%_fighter1	integer	Percentage of take downs landed by fighter 1
subatt_fighter1	integer	Total submissions attempted by fighter 1
rev_fighter1	integer	Total reversals while grappling by fighter 1
ctrl_fighter1	string	Total time that fighter 1 was in control during grappling exchanges
head_land_fighter1	integer	Total head strikes landed by fighter 1
head_att_fighter1	integer	Total head strikes attempted by fighter 1
body_land_fighter1	integer	Total body strikes landed by fighter 1
body_att_fighter1	integer	Total body strikes attempted by fighter 1
body_land_fighter1	integer	Total body strikes landed by fighter 1
leg_land_fighter1	integer	Total leg strikes landed by fighter 1
leg_att_fighter1	integer	Total leg strikes attempted by fighter 1
dis_land_fighter1	integer	Total distant strikes landed by fighter 1
dis_att_fighter1	integer	Total distant strikes attempted by fighter 1
clinch_land_fighter1	integer	Total successful clinch initiations by fighter 1
clinch_att_fighter1	integer	Total clinch initiations attempted by fighter 1
ground_land_fighter1	integer	Total strikes landed by fighter 1 while opponent is grounded
ground_att_fighter1	integer	Total strikes attempted by fighter 1 while opponent is grounded

Table 9.3: Features of matches dataset after preprocessing and feature engineering, original data from (UFC, 2020)

Feature	Data Type	Description
winner	integer	Binary value indicates whether fighter2 is winner
age_fighter1	integer	Age of fighter1
age_fighter2	integer	Age of fighter2
wins_fighter1	integer	Wins of fighter1
losses_fighter1	integer	Losses of fighter1
draws_fighter1	integer	Draws of fighter1
winstreak_fighter1	integer	Winning streak of fighter1
win_rate_fighter1	float	Win rate of fighter1
wins_fighter2	integer	Wins of fighter2
losses_fighter2	integer	Losses of fighter2
draws_fighter2	integer	Draws of fighter2
winstreak_fighter2	integer	Winning streak of fighter2
win_rate_fighter2	float	Win rate of fighter2
diff_avg_kd	float	Difference between average knockdowns per minute
diff_avg_sigstr_land_total	float	Difference between average significant strikes landed per minute
diff_avg_sigstr_att_total	float	Difference between average significant strikes attempted per minute
diff_avg_sigstr%	float	Difference between average % of significant strikes landed
diff_avg_total_str_land_total	float	Difference between average strikes landed per minute
diff_avg_total_str_att	float	Difference between average strikes attempted per minute
diff_avg_td_land_total	float	Difference between average takedowns landed per minute
diff_avg_td_att_total	float	Difference between average takedowns attempted per minute
diff_avg_td%	float	Difference between average takedown % per minute
diff_avg_subatt	float	Difference between average submission attempts per minute
diff_avg_rev	float	Difference between average reversals per minute
diff_avg_head_land	float	Difference between average head strikes landed per minute
diff_avg_head_att	float	Difference between average head strikes attempted per minute
diff_avg_body_land	float	Difference between average body strikes landed per minute
diff_avg_body_att	float	Difference between average body strikes attempted per minute
diff_avg_leg_land	float	Difference between average leg strikes landed per minute
diff_avg_leg_att	float	Difference between average leg strikes attempted per minute
diff_avg_dis_land	float	Difference between average distance strikes landed per minute
diff_avg_dis_att	float	Difference between average distance strikes attempted per minute
diff_avg_clinch_land	float	Difference between average clinches landed per minute
diff_avg_clinch_att	float	Difference between average clinches attempted per minute
diff_avg_ground_land	float	Difference between average ground strikes landed per minute
diff_avg_ground_att	float	Difference between average ground strikes attempted per minute

Figure 9.1: Feature importance of regular Random Forest model

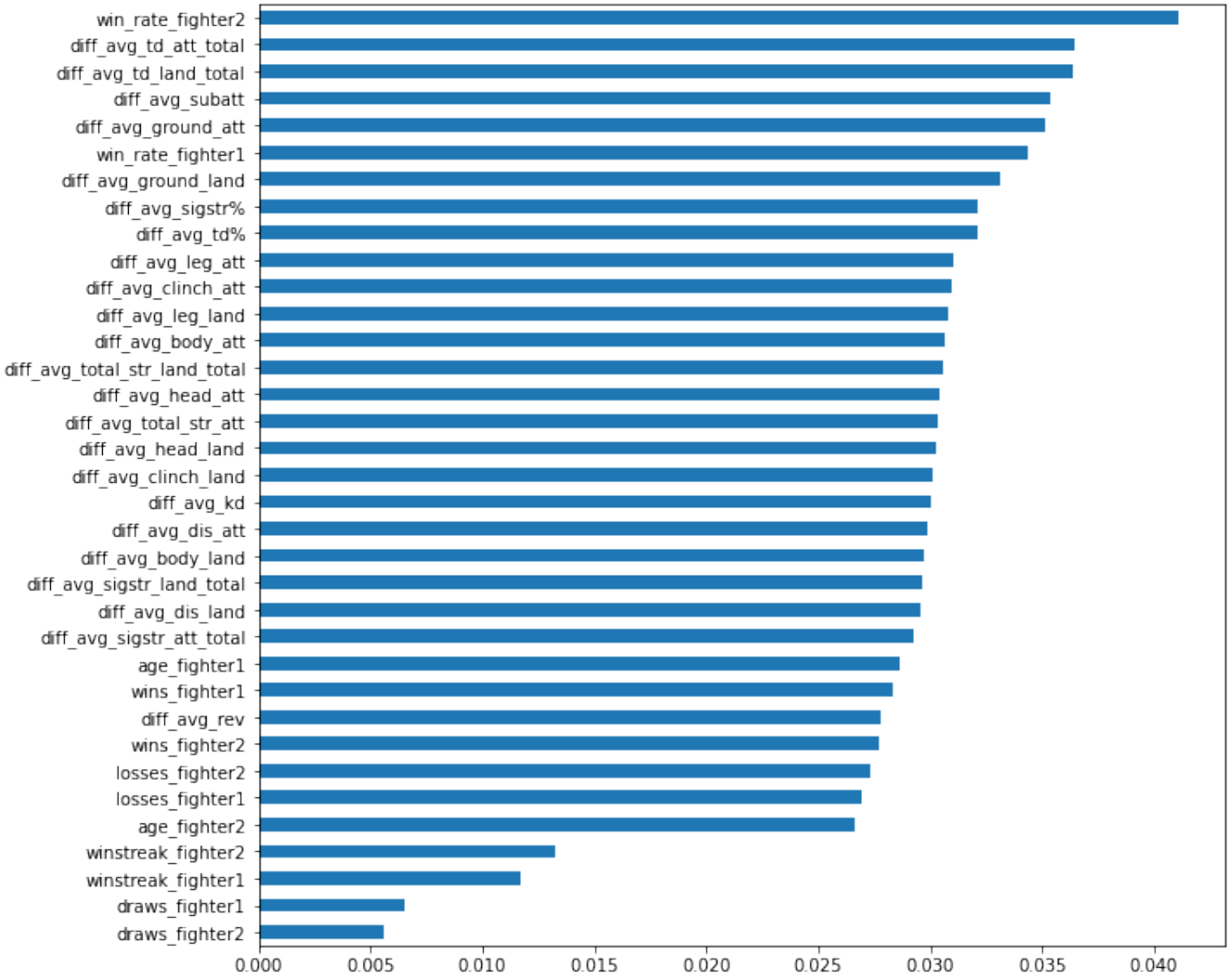


Figure 9.2: Feature importance of Random Forest model with information leakage