Openclassrooms.com

2022

**Project report**

OpenClassrooms, Project 8 (project libre)

Building predictive models for MMA sporting events

Mykola Mykhailytskyi

# ACKNOWLEDGEMENTS

I would like to thank everyone who supported me during this long journey. OpenClassrooms for creating the learning platform itself, my peers who were always there to help, and my mentors. Jean Savary, Valérie Tiedrez and especially Iselin Martial Naoussi Kuitche.

It was a valuable experience both in my new professional field of data analysis and in French. Despite studying in foreign language and oftentimes failing badly, everyone was supportive and helpful. With that in mind, I would like to apologize beforehand for any mistakes, misspellings and unnatural phrases that might occur in this report. Learning French is still an ongoing project for me. One day I hope to be fluent in both French and Data Analytics.

# TABLE OF CONTENTS

[**ACKNOWLEDGEMENTS**](#_nql5rhg3lvg) **2**

[**TABLE OF CONTENTS**](#_65nnlobwikkj) **3**

[**CONTEXT**](#_iy9lroca5mus) **4**

[**PROBLEM**](#_d5x13p9hgxjf) **5**

[**TASKS PERFORMED**](#_wsb2l5w5rgud) **6**

[Exploratory Data Analysis (EDA)](#_fjk29gir5hh9) 6

[Data cleaning](#_cmej4yvw7zm2) 9

[Building predictive models (with standardization)](#_3vl7d13ocz5q) 11

[Building predictive models (no standardization)](#_59syzlakhp2g) 13

[Weight of individual features](#_vg1qrrd0ldor) 15

[**CONCLUSIONS**](#_7royqa9y9dto) **17**

# CONTEXT

I am an ex-editor and customer support who decided to change careers. After analyzing my strengths, what I like to do and what I do best, my choice fell onto the domain of Data Analysis. Being one of the best customer support agents in Blizzard Entertainment, I always relied on the data that was accessible to me. I dug deep into statistics of my job - number of replies per category, per period of the day, types of problems, from which game, if our players were satisfied with the answer itself, if the issue was fixed with my advice or not, etc. Slowly but surely I was able to find answers to my own questions, such as how to give captivating, informative and valuable replies to the issues people were having with our games.

When it was time to move on, I took on the challenge of learning new things in order to become a Data Analyst. Part of the learning plan was to join OpenClassrooms platform and complete their Data Analyst course.

OpenClassrooms curriculum asks students to do a project in a professional field of their interest. Since my professional background is rooted into the gaming industry, I am generally a competitive person who likes sports and games. My projet libre would focus on predicting the outcome of Mixed Martial Arts fights (often shortened as MMA). The idea is to create a product (machine learning model) for a betting company or as a part of our own startup. We would like to build a model that will allow us to predict the outcome of MMA fights, based on historical data. Ideal outcome is a model that can both predict the winner and give high enough probability (>65-70%) for this outcome. This way we could also leverage risk factors when deciding to bet according to the model or against it.

Sports betting is a multibillion dollar industry. Its popularity on the level of regional and global markets makes it a very attractive business. Various reports predict two-three times growth for this industry in the span of 10 years. Fighting ranks among the top in the industry.

One of the most interesting and complex topics I have discovered in Data Analysis is machine learning and predictive algorithms. So while project goal is to build successful predictive model, my learning goals here are:

* Go deeper into the domain of machine learning
* Practice solving real-life problems with the help of DA and ML
* Experience various approaches to ML problems
* Understand what makes ML project work or not, what can be improved

Alternative purpose of this project could be for the MMA professionals themselves. For example coaching staff and fighters, who want to understand what is more or less important (key metrics) and steer the focus of their overall training. Or even build specialized training oriented towards defeating specific opponents.

# PROBLEM

Is it possible to build a reliable machine learning algorithm for predicting an outcome of a sporting event?

# TASKS PERFORMED

## Exploratory Data Analysis (EDA)

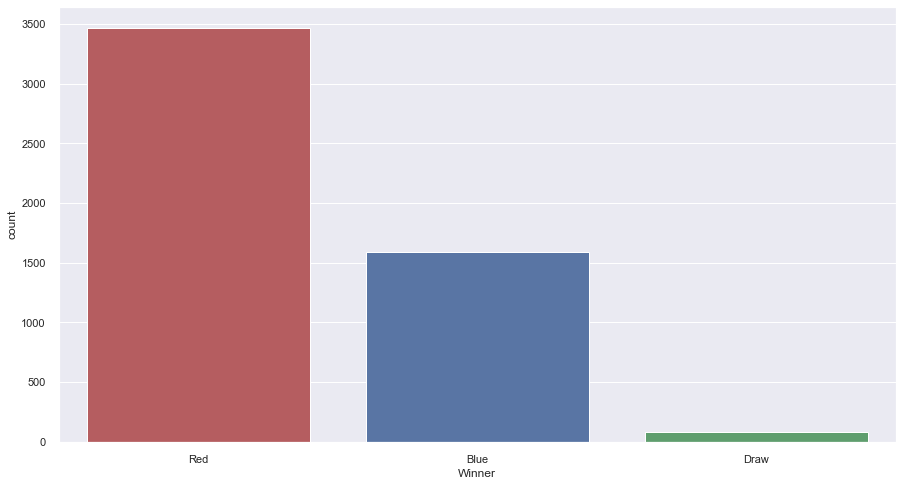
Since this type of project is new to me, I had to think about building a plan first, in order to have something to rely on. My initial plan turned out to be very similar to previous Data Analysis projects that were done for OpenClassrooms. Initial outline looked like this:

1. Grab MMA dataset from Kaggle (scraped from UFC website)
2. Load, explore, clean and prepare data for our models
3. Apply 3-5 various types of machine learning models and see which ones perform better
4. Try and adjust performance of these models
5. Give our best model(s) a completely new data and see how it will perform

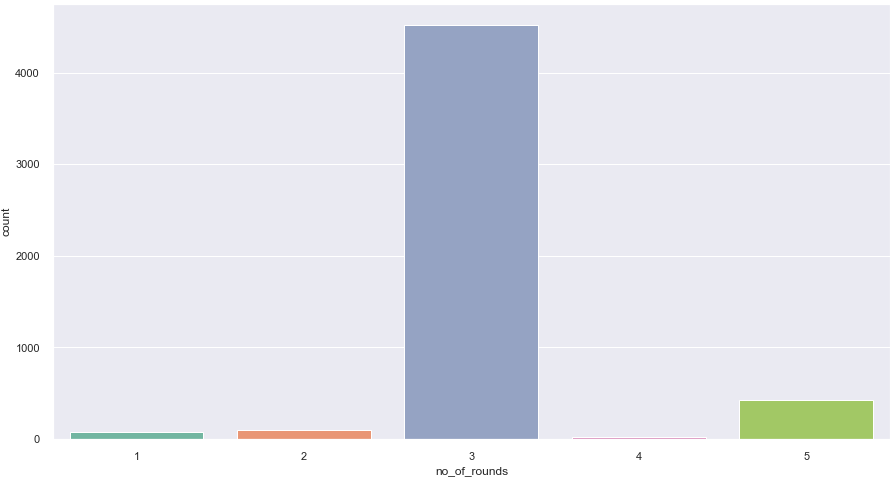
After loading raw data into Pandas dataframe I quickly realized that the amount of columns (features for ML models) are very high. During previous projects I had a chance to implement some simple ML models, and both theory and documentation was reminding me that very few or a lot of columns in your data could lead to problems for any model out there. Our raw data consists of 5144 rows, each row is one single fight. There are 145 columns that give various statistical or literal descriptions for those fights. Majority of data is numerical, for example individual fighter historical records (wins, losses, streaks of both types, how many times he fought for the title, height, weight, arm reach, etc) and each fight stats (how many times red fighter attempted to kick with his leg, how many times he landed leg kick successfully, how many rounds it took to decide a winner, etc). Some of the data is stored as strings and objects. This is mainly human-readable information like fighters names, referee, location, weight class, etc. With this in mind I was already planning changing string and object data into dummies or applying LabelEncoder, since ML models wouldn't be able to read original data types from these columns.

Further exploration of the dataset showed that records started in November 1993 and currently the latest fight data we have is from June of 2019.

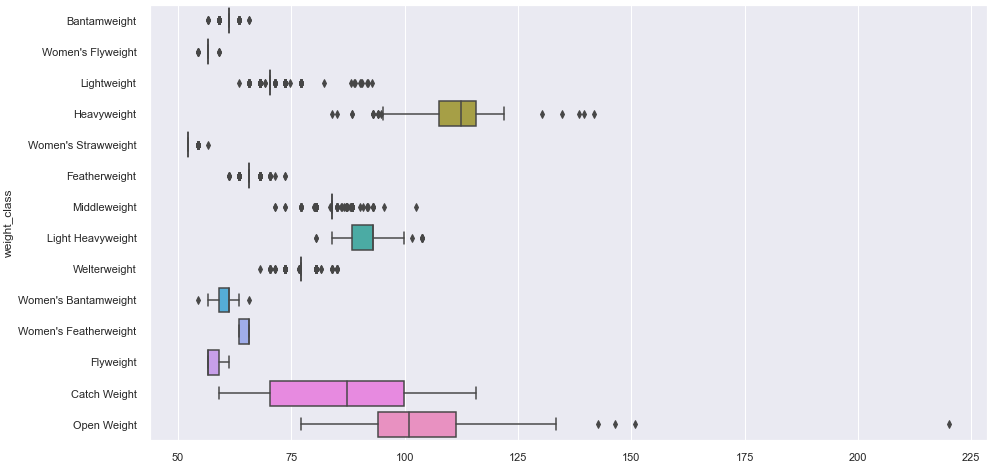
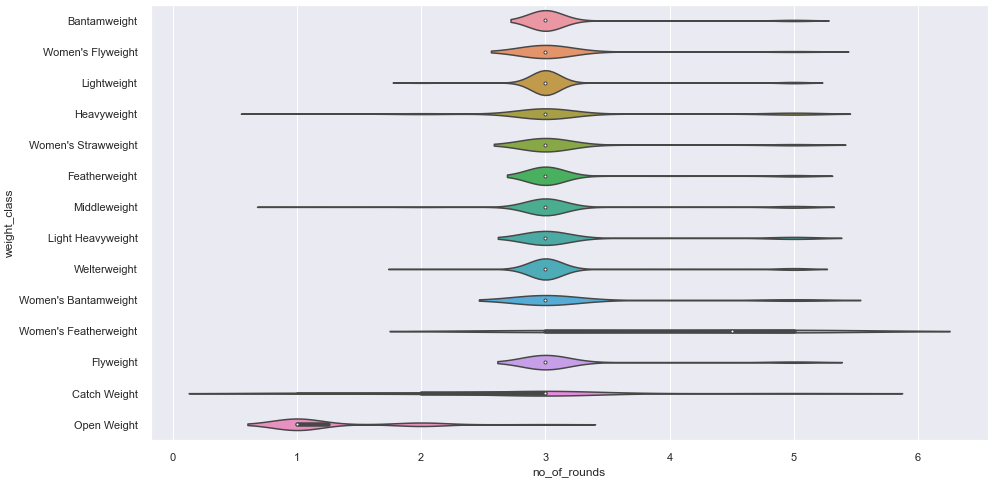
Since we want to predict who won the fight (Red or Blue), we know that our dependent variable will be the column “Winner”. We look at the distribution in this column and see that there is a big disproportion.

The Dependent variable contains 3470 Red winners, 1591 Blue winners and 83 draws. Remembering the advice we got from the “detect fake banknotes” project, we make a note here that during data split into test and train, we need to make sure that distributions are equal. This allows for better models, and as a result, better predictions.

Lets see how many rounds it takes to decide a winner of one particular match:



Surprisingly, the absolute majority of fights tend to last for 3 rounds. Very few fights last 1, 2 or 4 rounds. Second most popular length is 5 rounds.

MMA, same as many fighting sports, has multiple weight classes. In order to find a visual approach to these classes, we convert american Lbs in the dataset into Kgs and plot the following:Now, it would be interesting to see if weight class has any effect on the duration of the fight. To get this information we use violin plot. This type of plot works great here:

Given our previous findings about fight lengths, it is not surprising that the majority of weight classes on average take 3 rounds to decide a winner. But thanks to this visualization we can see some exceptions. Those are open weight (lower than average) and women's featherweight (higher than average).

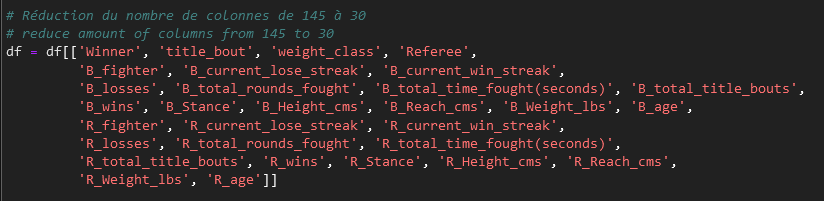
## Data cleaning

Next we start preparing our dataset for the project objective. Starting with missing values, we see that a significant amount of data is missing. Notably ~25% of data in columns that correspond to averages of Blue fighters and ~12.64% of data that corresponds to averages of Red fighters. Other columns had very few missing values, which do not warrant any concerns at this point.

Our goal calls for ability to determine between 2 outcomes. As we saw previously, the dependent variable contains 3 outcomes, but the 3rd one has very few data points. Based on this we decide to remove Draw outcome from the dataset. At the same time we remove the date column and all fights (rows) that contain any NaN values. This reduces the amount of rows in our dataset from 5144 to 3151.

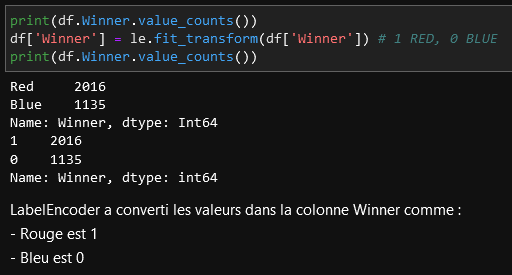
Now we need to deal with 145 columns (features). After going through Scikit-learn documentation and online tutorials, we decide that the best course of action is to drop columns that we will find irrelevant.

We could go many ways here, but the optimal solution at this point is to leave only critical columns (title\_bout, weight\_class, etc) and those that contain data that should always exist before an actual fight happens. Such columns as height, weight, wins and losses, streaks, reach, total rounds fought, title fights fought, etc. This should be a good starting point for our objective. If need be, we can alter our approach or reintroduce some columns in the future.

As a result, we reduced the number of columns (features) from 145 to 30.

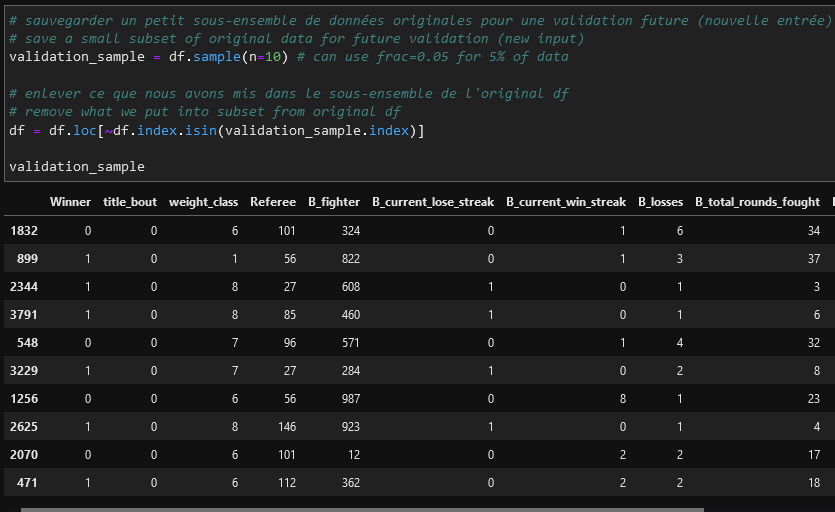
Next we call on for .info() method of pandas and note down columns that need their data type changed to fit ML models. Columns 'Referee', 'R\_fighter', 'B\_fighter', 'R\_Stance', 'B\_Stance', 'Winner' and 'weight\_class' are changed into proper string format. Right after we apply LabelEncoder fit\_transform in order to transform strings into numerical values, readable by ML. Essentially we give each Referee, Fighter, Stance etc a unique ID number.

After the first pass of LabelEncoder over our dependent variable, it was hard to understand which numerical value corresponds to previous Red\Blue. We were able to bypass this confusion using the following approach:



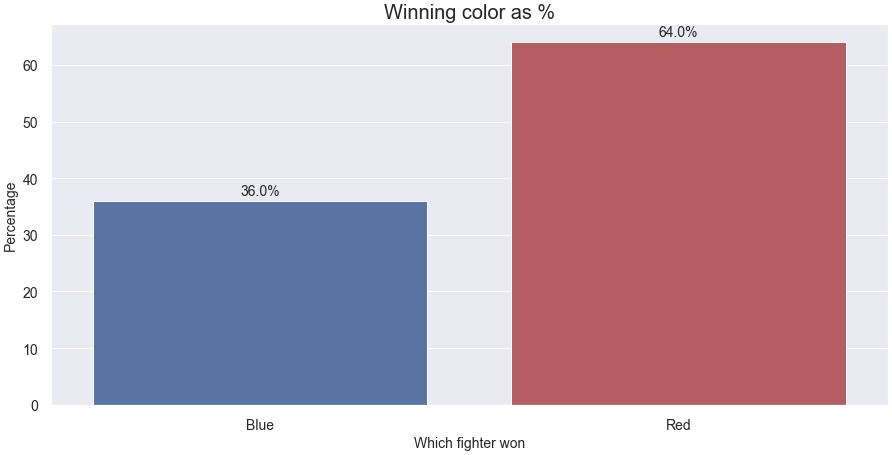
Now we have visual confirmation and a reminder that LabelEncoder changed all Red winners to 1, and all Blue winners to 0. This will be important in the future.

At this point it was obvious that we would need some way of testing our model(s) on completely new, unseen data. In other words, we had a need of simulating real-world application of our model(s). This step can be done via APIs and web-scraping, but given time constraints, our mentor Martial advised using existing dataset. Thanks Martial for the idea!



What we did here is randomly selected 10 fights (rows), and then deleted those exact fights from the dataset. This way validation\_sample retains the same format as our dataset, but will never be seen by the models until we try to test them.

Since we made a lot of manipulations with our dataset, we would like to see the current distribution in our dependent variable:



The situation improved, but we can say with confidence that it is still far from perfect. Balance of 36 to 64 means data points are still unequal and we will need to do stratification or weight balance.

## Building predictive models (with standardization)

Now we need to select models that can suit our objective. Having very little experience, we rely heavily on documentation and personal experience of others. We applied an algorithm for finding best models for a specific task from scikit-learn <https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html> as well as taking notes from our mentor.

Following the steps above, we declare that our sample has more than 50 samples, we want to predict a category (0 or 1), our data is labeled and the amount of data is less than 100k. As a result, we end up with classification-type models that will suit us best. This means that we will try using the following models:

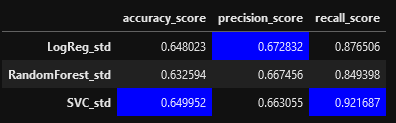
* Gaussian Naive Bayes
* KNN (K Nearest Neighbors classifier)
* SVC
* Random Forest classifier
* Decision Trees classifier

We can also try and apply other types of models, for the sake of experimentation.

Our first approach to building ML models is to standardize our data after train\_test\_split, in order to avoid any noise from the data itself. After writing 6 different types of models (5 noted before plus Logistic Regression), we note their accuracy scores:

* Logistic Regression Average Score 0.6549
* Gaussian Naive Bayes Average Score 0.56034
* Decision Tree Average Score 0.5760
* KNN Average Score 0.6112
* Random Forest Average Score 0.6454
* Support Vector Classification Average Score 0.6682

Judging by average cross validation score, top performing models are logistic regression, random forest and svc. We move onto testing those 3 while storing their various scores in a separate table. Our final results are:



Accuracy score is how many labels models predict exactly right. Top performing model by this score is SVC.

Precision score is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative. The best value is 1 and the worst value is 0. Top performing model by this score is Logistic Regression.

Recall score is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples. The best value is 1 and the worst value is 0. Top performing model by this score is SVC.

Overall SVC shows best results, however Logistic Regression only slightly behind.

At this point we try and “feed” top models absolutely new data, the one we stored in validation\_sample. Results are below

**Logistic Regression** predicts the following winners: [0 0 1 1 1 0 1 1 0 0]

Actual fight results: [0 1 1 1 0 1 0 1 0 1]

Accuracy score for Logistic Regression on new data is 0.5

Precision score for Logistic Regression on new data is 0.6

Recall score for Logistic Regression on new data is 0.5

**RandomForest** predicts the following winners: [0 0 1 1 0 1 0 1 0 1]

Actual fight results: [0 1 1 1 0 1 0 1 0 1]

Accuracy score for RandomForest on new data is 0.9

Precision score for RandomForest on new data is 1.0

Recall score for RandomForest on new data is 0.8333333333333334

**SVC** predicts the following winners: [1 1 1 1 1 1 1 1 1 1]

Actual fight results: [0 1 1 1 0 1 0 1 0 1]

Accuracy score for SVC on new data is 0.6

Precision score for SVC on new data is 0.6

Recall score for SVC on new data is 1.0

Conclusions about models with standardized data:

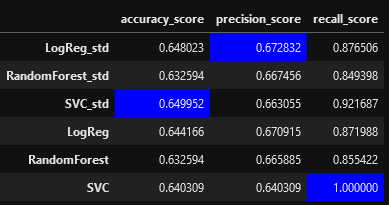
* Neither of the 3 top performing models look good (based on multiple runs)
* all 3 are inconsistent, from 0.0 to 0.9 accuracy. Averaging at 0.5 (based on multiple runs)
* current models do not allow us to use them in predicting fights. We should try training the same models on original, non-standardized data.

## Building predictive models (no standardization)

Second approach is training our models on original, non-standardized data. After writing same types of models, we note their accuracy scores:

* Logistic Regression Average Score 0.6582 (slightly increased from 0.6549)
* Gaussian Naive Bayes Average Score 0.56034 (remained the same, 0.56034)
* Decision Tree Average Score 0.5798 (slightly increased from 0.5760)
* KNN Average Score 0.5541 (decreased from 0.6112)
* Random Forest Average Score 0.6473 (slightly increased from 0.6454)
* Support Vector Classification Average Score 0.6397 (decreased from 0.6682)

Once again, top performing models are logistic regression, random forest and svc. We move onto testing those 3 while storing their various scores in a separate table. Our final results are:



There is no clear winner, however our experiments proved that results here are inconsistent. Sometimes models trained on standardized data perform slightly better, sometimes models trained on original data perform slightly better. Most consistent observation at this stage is that the top 2 performing models are Logistic Regression and Support Vector Classifier.

Now again we “feed” top models absolutely new data, the one we stored in validation\_sample. Results are below:

Logistic Regression predicts the following winners: [1 1 1 0 0 1 1 1 1 1]

Actual fight results: [0 1 1 1 0 1 0 1 0 1]

Accuracy score for Logistic Regression on new data is 0.6

Precision score for Logistic Regression on new data is 0.625

Recall score for Logistic Regression on new data is 0.8333333333333334

RandomForest predicts the following winners: [1 1 0 1 1 1 1 1 1 1]

Actual fight results: [0 1 1 1 0 1 0 1 0 1]

Accuracy score for RandomForest on new data is 0.5

Precision score for RandomForest on new data is 0.5555555555555556

Recall score for RandomForest on new data is 0.8333333333333334

SVC predicts the following winners: [1 1 1 1 1 1 1 1 1 1]

Actual fight results: [0 1 1 1 0 1 0 1 0 1]

Accuracy score for SVC on new data is 0.6

Precision score for SVC on new data is 0.6

Recall score for SVC on new data is 1.0

Conclusions about models with non-standardized data:

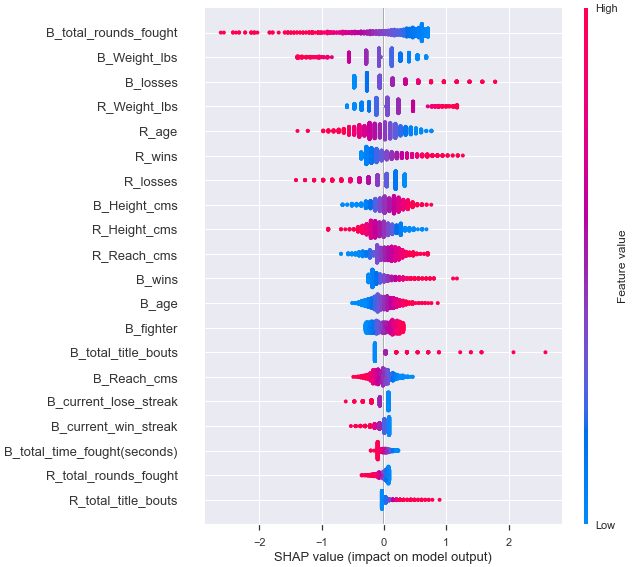
* the difference in scores is very small, but also inconsistent. Sometimes scores from original data models are slightly higher, sometimes slightly lower (based on multiple runs)
* it seems that training on original data makes for more consistent predictions (based on multiple runs)
* we might need to evaluate ability to see which feature (column) has more value for the model, compared to others.

## Weight of individual features

In order to find important features (ones that our models think has the most value when trying to predict the outcome of the fight), we use SHAP library for Python: <https://shap-lrjball.readthedocs.io/en/latest/index.html>

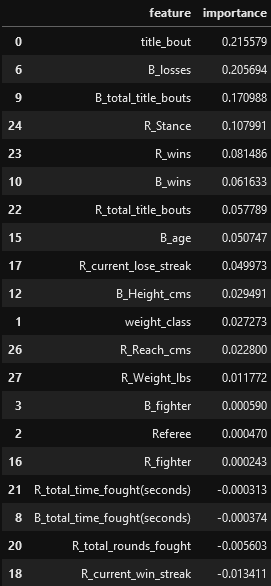
SHAP (SHapley Additive exPlanations) is a game theoretic approach to explain the output of any machine learning model. It connects optimal credit allocation with local explanations using the classic Shapley values from game theory and their related extensions.

With the help of this library we can calculate the importance values of each feature and plot it on the graph. Our results for Logistic Regression model is below:



Features are sorted by their impact, from highest (top) to lowest (bottom). This is axis Y. Axis X is SHAP values. Each dot is a separate observation. Color-coding helps understand the meaning of the corresponding feature. Blue is low, Red is high (there is no connection between this color-coding and our dependable variable of Red\Blue winner). For example:

* If the B\_total\_rounds\_fought feature gets higher, most likely blue fighter will win.
* If R\_total\_title\_bouts gets higher, most likely the red fighter will win.
* if B\_reach\_cms gets high, the blue fighter is more likely to win; if it gets low, the red fighter is more likely to win.

In addition to SHAP library weights we can use coefficients of LogisticRegression itself and compare the two.

Comparing the two we can see both drastic differences and similar weights. For example SHAP quantifies B\_total\_rounds\_fought feature as the most important, but coefficients from the model itself have an opposite value, placing it lower than 20th. Meanwhile B\_losses feature is top 3 for both methods.

Based on the documentation provided for SHAP and scikit-learn, we see that their calculations are different. It means that we should not blindly trust one single method of feature weight evaluation. Might be a good idea to experiment with feature engineering based on coefficients from both sources (or even find more ways to quantify weights).

# CONCLUSIONS

One of the first important lessons learned during the project was that a large number of columns (features) does not necessarily translate into better machine learning models. It might introduce “data noise”, overcomplicate model training (the more data we have, the more processing time it will take for a PC to run all the calculations) and become very time consuming.

When we were planning on which models to use, we relied heavily on existing documentation and experience of our colleagues. But surprisingly, the best performing model by score turned out to be Logistic Regression, which was not proposed by scikit-learn algorithm. Only slightly behind it is Random Forest and SVC. Key takeaway here is that not every problem has a “perfect” solution and “best” model. Sometimes we need to experiment and find what works best for our particular case, dataset and problem.

Additionally, it was perplexing to see that the results of all 3 top performing models are inconsistent. While we did see sometimes an accuracy of ~65-67%, this score was not consistent enough. Meaning even if Logistic Regression or SVC produced an accuracy of 0.9 on one particular run, the next one could produce accuracy of 0.3 or 0.5. Sadly, this was not looking good for our project goal of making reliable predictions. Simply put, it would be too risky predicting both according to our models and against them.

Possible improvements for this project include, but not limited to:

1. Give more weight to recent data

Our idea here is that even though the dataset starts describing fights from 1993, those old statistics might have lower quality or importance compared to nowadays. Afterall, everything evolves and improves over time. If we don’t specify to our models this, they might treat old data equally important, thus skewing results.

1. Try using all of the existing columns/features.

One of the possible improvements for our project is to increase the number of features. We deleted stats from each individual fight (how many punches or kicks were launched by each fighter, how many landed successfully, etc). We might keep them and deduce various historical parameters for each fighter that might help with predictions, e.g. % of successful attacks for each fighter. This might help with predictions.

1. Equalize validation samples to have 50% Red and 50% Blue winners.

Our validation sample was based on the randomized sample of 10 fights. While we were able to balance distributions in the training dataset, we did not perform this for validation samples.

1. Try other types of models (XGBoost, etc).
2. Build a better representation of predictions.

We could improve user experience by color coding the outcome, giving winners names, etc.

1. Build a better way of showing how confident the model is in one specific prediction.

Along with predicted winners, we could offer users a confidence level of this prediction.

1. Presenting the user with a set of most valuable factors for prediction.

This circles back to our attempt of weighting individual features. For example, if a model gives a prediction with low confidence level, displaying valuable factors that played into this decision might help users decide - bet with or against the prediction.

1. Create a user-friendly interface (web-deployment).

Cherry on top for any project like this would be a fully developed web app, that will allow anyone to use it with ease from any device.