Prediction tool for mixed martial arts bout outcome

Danil Burov

Fontys University of Applied Sciences

January 9, 2025

Contents

1 Introduction

The purpose of this document is to report in a comprehensive way all the work that has been done throughout my fifth semester regarding the creation of an AI tool for predicting the outcome of MMA bouts. To do so, I will be using the simplified data science IBM methodology framework structure to convey all the necessary information regarding my project. Using this format I will be able to reflect on my work while explaining how the project was built and how it works.

2 Business understanding

The primary objective of this project is to develop a predictive tool that provides insights into the outcome of a bout between two fighters. Rather than offering a binary prediction (0 or 1, representing true or false outcomes), the tool generates probabilistic predictions on a scale between 0 and 1. This allows users to understand the likelihood of various outcomes, such as the probability of a fighter submitting their opponent, achieving a knockout, or winning by decision. By delivering granular insights, the tool empowers users to make data-driven assessments of match outcomes.

One notable risk associated with the project is its potential misuse for betting purposes. The tool could be exploited to gain unfair advantages in gambling scenarios. To mitigate this concern, the tool will be kept private and its access limited to legitimate and approved stakeholders.

Beyond the potential concerns, the tool holds immense value as an analytical resource for fighters and their coaching teams. Fighters can leverage the tool to analyze their strengths and weaknesses relative to their opponents. For instance, by understanding the likelihood of success in specific fighting techniques, fighters can tailor their training regimens to exploit their opponent's vulnerabilities and address their own shortcomings.

3 Data understanding

The dataset selected for this project is sourced from Kaggle and contains data on every UFC fight from mid-2010 to the present. Each row in the dataset represents an individual bout, making it well-suited for predictive modeling and analysis. This dataset was chosen for several reasons: it is regularly updated on a weekly basis , ensuring that the data remains current, and it already includes pre-processed statistics such as the difference in metrics between the two fighters in each bout. Additionally, it provides key variables such as betting odds, fighter rankings, and fight statistics, as well as the target variable for this project, the bout "Winner."

Despite its strengths, including over 6,000 entries and 118 features, there are still several limitations to consider regarding the accuracy and completeness of the model. Many critical factors that influence fight outcomes are not included in the dataset due to their subjective or difficult-to-quantify nature. For example:

Mental State: A fighter's psychological readiness, confidence, and focus leading up to the fight can significantly affect their performance.

Health Status: Factors such as injuries, illnesses, or the general health of the fighter during training camp or on fight day are not accounted for.

Weight Cutting and Rehydration: Many fighters undergo extreme weight cuts before weigh-ins, only to rehydrate and regain substantial weight before the fight. The extent of their weight recovery and its impact on their physical condition during the bout is a crucial factor that is not captured in the dataset.

Moreover, many day-to-day factors influencing a fighter's performance are inherently subjective and nearly impossible to gather systematically. Fighters themselves may not be fully aware of how these aspects—such as their recovery quality, nutrition, or emotional state—affect their performance in a bout. This lack of granular data presents a challenge in achieving high accuracy for predictive models.

While these limitations highlight gaps in the dataset, they also underscore the opportunity for future research and data collection in combat sports. If more detailed physiological and psychological data could be integrated into such datasets, it would pave the way for more accurate and robust predictive models in the domain of mixed martial arts.

4 Data preparation

In order to explain the data preparation done regarding the ufc dataset I will reference most of my explanations to the appendix. This was done to keep a neatly organized text. I will be using snippets from my Jupyter notebook. All these features are displayed here:
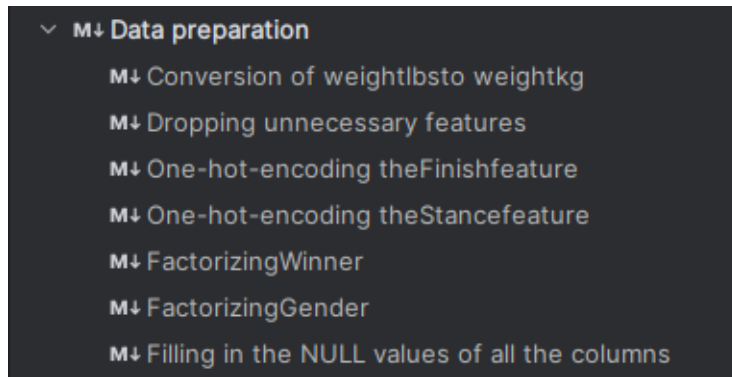
Figure 1: Overview of data preparation processing steps

4.1 Deleting columns

When the dataset was chosen it was known that many of the features will not be used for the purpose of the tool. All these features were dropped for a reason and in the upcoming subsection it will be explained why. Initially the number of features was 118. After dropping the unnecessary features, the number was reduced to 49.

These features were rankings, streak orientated, or keeping track of the fighters' decisions and win/lose streak, and other miscelanous features that are irrelevant.

The code for the feature removal can be found **here**

4.2 One-Hot-Encoding

One-hot-encoding is used to trnasform categorical values into numberical values. In this case the 'Finish' feature had 5 different types of finished. When I one-hot-encode the feature I get 5 different columns with all the possible finishes and every bout (each row in the dataset) has a true or false to the according finish. It is not a good idea to factorize with that many different options because the model could develop bias towards the bigger numbers.

One-hot-encoding was used not only to encode the 'Finish' feature but also the 'Stance' feature. 'Stance' had three possible values 'Southpaw', 'Ortohodox', 'Switch'.

The code for one-hot-encoding the 'Finish' feature can be seen **here**
The code for one-hot-encoding the 'Stance' feature can be seen **here**

4.3   Scalling

In general all the features in the dataset need to be scalled to standard deviation. That is the reason why all the features that had percentage were scalled to a decimal to the first zero.

Additionally I converted the 'Weight' feature from LBS to KG, because it is more convenient for me.

The code can be found **here**

4.4   Factorzing

For some features one-hot-encoding is not exactly needed. In this case it was decided to factorize values that can have only two possible values. However in some cases it could be better to even use one-hot-encoding with these features as well because some models can deal with more features more efficiently. The reason these features were factorized is because in the beginning the project was supposed to train a classification model where the output is either 0 or 1, however the idea changed but factorizing is still good enoigh for the model that is currently being used.

Since the dataset contains both genders the feature that was factorized is the 'Gender' where the 0 is <u>man</u> and the 1 is a <u>woman</u>.

Another feature that has only two values is the 'Winner' feature, which has either red or blue, depending on who won the fight (Red and Blue are universal colours used to indicate the corner that fighter fought from.). When factorized, 0 corresponds to <u>Red</u> and 1 corresponds to <u>Blue</u>.

'Gender' feature code for factorizing

'Winner' feature code for factorizing

4.5   Filling NaN values

The dataset has **415296** total entries from all features from which **19601** are NULL values. There were three options how to deal with all the different values. First was to fill in the missing values with 0 or specific string text wherever is needed. Second was to delete the rows that had these NULL values. However, the condition for this is the rows that are missing values to be less than 10 percent. And the last possible action was to develop a ML model that feels in these values. This option was not developed.

Most of the features did not have a significant impact on the accuracy of the model that is why they were filled with 0. Most of the rows that had NULL features were both from the red and blue corner (e.g. The 'RedOdds' feature and 'BlueOdds' feature in the row were both NULL so filling them with 0 just fixes the empty value)

Code for filling the NaN values

5 Modelling

Choosing a model for the data was difficult given the reason that the project idea was to not predict if a fighter is going to win (0 or 1), but rather see the likelihood of a fighter winning in three different outcomes against the opponent. Knowing this, logistic regression seemed like a good fit for the project.

Logistic regression under the hood is a classification model that uses the Sigmoid function. Although, given its classification nature, it provides the probability output between 0 and 1.

In this project, four different models were trained with the data. This is due to the fact that the target variable needed to be existent in the dataset. In the beginning, an imaginary target variable was created, which was a combination of all the targets: 'Winner', 'Finish_KO/TKO', 'Finish_SUB', 'Finish_U-DEC'. For this project, other finishes were not taken into account for simplicity reasons. Although further development could be the prediction of the rest finish outcomes of a match, e.g., 'Doctor stoppage'.

The first results I received by training the data with the combined and cleaned dataset was **0.63**% score accuracy. However, looking into the precision, recall, and f1-score, it was quickly established that the data has some sort of bias towards the red corner. This could be due to the fact that in the history of the UFC, more red corner fighters actually won than blue corner fighters. Or the model is simply predicting red fighter winning better than blue fighter.

Here will be displayed the scores of training without any hyperparameter tuning or anything to improve the performance of the model.

```
Logistic Regression model accuracy (in %): 0.6385542168674698
Classification Report:
              precision    recall  f1-score   support

           0       0.69      0.64      0.67       562
           1       0.58      0.63      0.60       434

    accuracy                           0.64       996
   macro avg       0.64      0.64      0.64       996
weighted avg       0.64      0.64      0.64       996
```
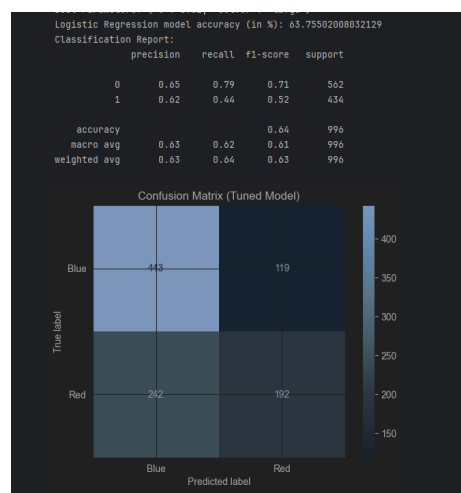
5.1 Optimization

Logistic regression was used to train four models with four different target variable. However on all of them this bias towards the red corner was persistent. In order to mitigate this bias, balancing the set was needed. Hyperparameter tuning, oversampling and undersampling were performed on the datasets and then the data was trained again using Logistic regression. Using undersampling, the accuracy got better and the bias was mitigated better than the rest of them. Unfortunately, the other techniques didn't really improve the accuracy nor mitigated the red corner winning prediction.
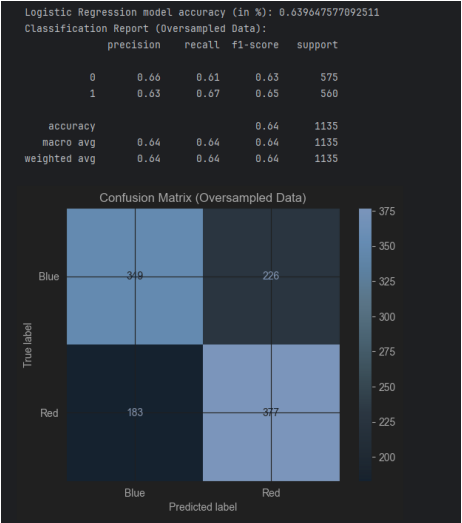
The different scores of the optimization of the dataset will be displayed below.
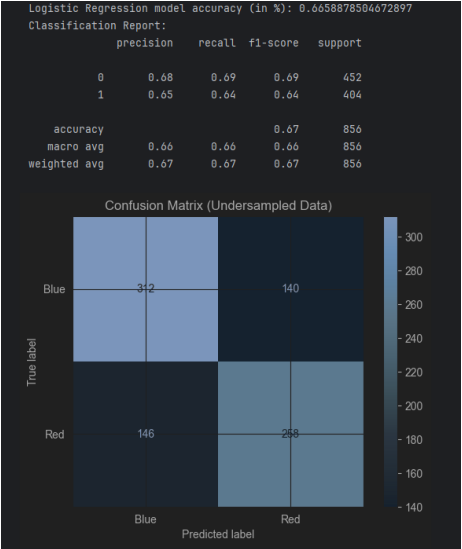
Hyperparameter Tuning Score

Oversampling Score


```
Logistic Regression model accuracy (in %): 0.639647577092511
Classification Report (Oversampled Data):
              precision    recall  f1-score   support

           0       0.66      0.61      0.63       575
           1       0.63      0.67      0.65       560

    accuracy                           0.64      1135
   macro avg       0.64      0.64      0.64      1135
weighted avg       0.64      0.64      0.64      1135
```

Confusion Matrix (Oversampled Data)

Undersampling Score


```
Logistic Regression model accuracy (in %): 0.6658878504672897
Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.69      0.69       452
           1       0.65      0.64      0.64       404

    accuracy                           0.67       856
   macro avg       0.66      0.66      0.66       856
weighted avg       0.67      0.67      0.67       856
```

Confusion Matrix (Undersampled Data)

6   Conclusion

Overall, a good accuracy was achieved of **0.66%** an increase from the raw modeling of logistic regression 0.63%. It is very difficult to predict an outcome of a match or which fighter will win. There are many attributes that need to be taken into account. There is a possible further continuation of the project, looking into betting odds. I personally think that this could be a good direction of the project.

7   Appendix


```python
features_to_drop = [

    # Blue features
    "BlueFighter","BlueExpectedValue", "BMatchWCRank", "BWFlyweightRank", "BWFeatherweightRank",
    "BWStrawweightRank", "BWBantamweightRank", "BHeavyweightRank",
    "BLightHeavyweightRank", "BMiddleweightRank", "BWelterweightRank",
    "BLightweightRank", "BFeatherweightRank", "BBantamweightRank",
    "BFlyweightRank", "BPFPRank","BlueCurrentLoseStreak","BlueCurrentWinStreak",
    "BlueDraws", "BlueLongestWinStreak","BlueLosses","BlueTotalRoundsFought","BlueTotalTitleBouts",
    "BlueWinsByDecisionMajority","BlueWinsByDecisionSplit","BlueWinsByDecisionUnanimous",
    "BlueWinsByKO","BlueWinsBySubmission","BlueWinsByTKODoctorStoppage",
    "BlueWins",

    # Red features
    "RedFighter","RedExpectedValue", "RMatchWCRank", "RWFlyweightRank", "RWFeatherweightRank",
    "RWStrawweightRank", "RWBantamweightRank", "RHeavyweightRank",
    "RLightHeavyweightRank", "RMiddleweightRank", "RWelterweightRank",
    "RLightweightRank", "RFeatherweightRank", "RBantamweightRank",
    "RFlyweightRank", "RPFPRank","RedCurrentLoseStreak","RedCurrentWinStreak","RedDraws","RedLongestWinStreak","RedLosses",
    "RedTotalRoundsFought","RedTotalTitleBouts","RedWinsByDecisionMajority",
    "RedWinsByDecisionSplit","RedWinsByDecisionUnanimous","RedWinsByKO",
    "RedWinsBySubmission","RedWinsByTKODoctorStoppage", "RedWins",

    # Other
    "Date", "Location", "Country", "TitleBout","WeightClass", "EmptyArena","BetterRank", "FinishDetails","FinishRoundTime",
]

ufc_data_master = ufc_data_master.drop(columns = features_to_drop)

ufc_data_master.columns
[7]
```

Figure 2: Unnecessary features dropped

One-hot-encoding the 'Finish' feature

One-hot-encoding is used to transform categorical values into numerical values. In my case the 'Finish' feature had 5 different types of finishes. When I one-hot-encode the 'Finish' feature I get 5 different columns with all the possible finishes and a match has a true or false to the according finish. It is not a good idea to use factorizing with that many different options because the model could develop bias towards a bigger number.

Issue -> The issue it solves is the models I am using need to have a numerical value instead of categorical.

Solution -> One-hot-encoding is used instead of factorizing to reduce the model bias.

```python
ufc_data_master = pd.get_dummies(ufc_data_master, columns = ['Finish'], prefix='Finish')

ufc_data_master.head()
[10]
```
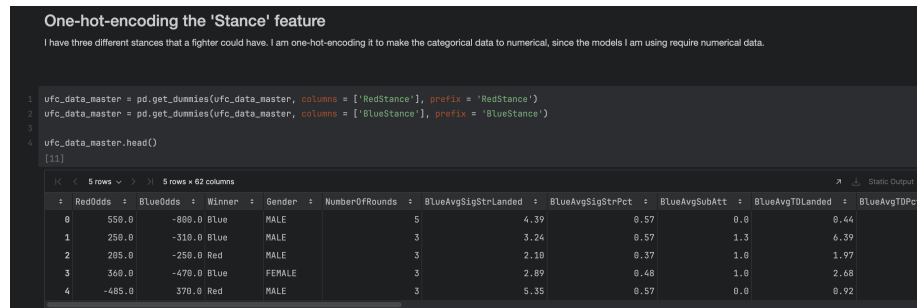
Figure 3: One-Hot-Encoding 'Finish' feature

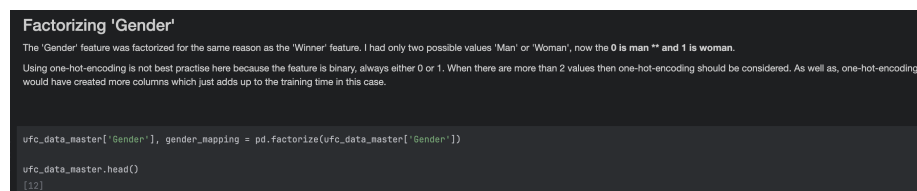Figure 4: One-Hot-Encoding 'Stance' feature
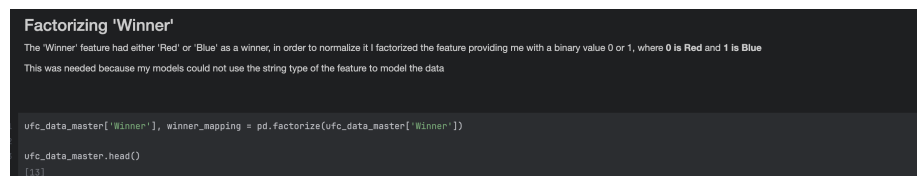

Figure 5: Factorizing the 'Gender' feature

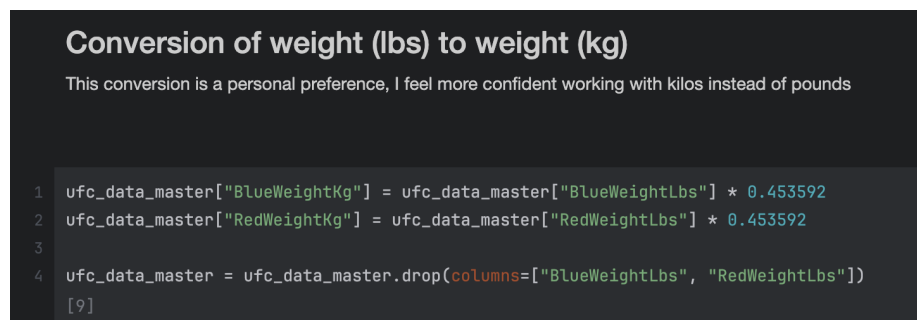Figure 6: Factorizing the 'Winner' feature

Figure 7: Converting the 'Weight' feature from lbs to kg

```
The rows that had NaN values were dropped because they are <10% of the whole dataset.

1  ufc_data_master = ufc_data_master.dropna(subset=['FinishRound'])
2  ufc_data_master = ufc_data_master.dropna(subset=['TotalFightTimeSecs'])
   [14]

1  null_values_finish_rounds = ufc_data_master['FinishRound'].isnull().sum()
2  null_valies_finish_round_time = ufc_data_master['FinishRoundTime'].isnull().sum()
3  null_values_total_fight_time = ufc_data_master['TotalFightTimeSecs'].isnull().sum()
4
5  print(f"Number of null values in 'FinishRound': {null_values_finish_rounds}")
6  print(f"Number of null values in 'FinishRoundTimeSecs': {null_valies_finish_round_time}")
7  print(f"Number of null values in 'TotalFightTimeSecs': {null_values_total_fight_time}")
   [15]
```

Figure 8: filling the NaN values