MMA Winner Prediction with Machine Learning

Carlos Paredes

Abstract:

Our study uses a UFC dataset from Kaggle which covers MMA fights from the years 2010 through 2021 in order to predict fights using supervised machine learning models. The dataset consisted of 119 different features per match, which was a significant challenge for us considering the risk of overfitting, but through various different feature engineering and data cleaning, we were able to bring down the number of features to 31 prominent features that covered significant differences between fighters in order to enhance the accuracy of our models. The different models that we used were random forest, logistic regression, and gradient descent. Our analysis of the project shows that there were different model performances for title and non title fights, which suggests that the higher predictability in title fights is attributed to the normalization of fighter statistics along with their betting odds.

Introduction:

Mixed Martial Arts (MMA) is a dynamic sport with a history dating back to 1992. The fundamental concept of MMA is to provide a platform where martial artists from diverse backgrounds can challenge their skills in combat. The dataset we used was the UFC dataset available on Kaggle, encompassing fight data from mid 2010 to 2021. This dataset includes comprehensive information, such as takedown percentage, striking accuracy, age, and more, for both the victors and the defeated fighters. Because of the large amount of fights and data on each fight, and the fundamental question of predicting who would win, we thought predicting winners would be a good task for machine learning. To do this, we cleaned the data, engineered the features, and applied multiple models to test how their metrics compare to each other.

Task Description:

The task of this project is to predict the winners of UFC fights. The dataset that is loaded in has data on over ten years on MMA fights, from 2010 to 2021. It has many features on both fighters, such as takedown percentage, average significant strikes, etcetera. Fighters for each event are labeled ‘red,’ or ‘blue’, and there is a ‘winner’ column which shows whether the red or blue fighter won. The way our dataset is set up, the fighters in the ‘red’ column are primarily betting favorites, and the fighters in the ‘blue’ column are primarily betting underdogs. This fact will prove to be important, as seen later, since the betting features play a large part in the models’ evaluations. Because our model is trained with data which is already labeled, from the ‘winner’ feature of our dataset, it is a supervised model. Also, because each data sample is being classified into having the winner be either ‘red’ or ‘blue’, the task is to classify new data samples into one of two categories, meaning that this task is supervised discrete.

Major Challenges and Solutions:

One major challenge for this task was just how many features there were for each record, meaning how many columns there were for each fight. If a machine learning model has too many features, it can lead to overfitting, meaning that the model fits the training data so closely that it does not generalize well to new data and testing performance suffers.

```
df.head()
```

	R_fighter	B_fighter	R_odds	B_odds	R_ev	B_ev	date	location	country	Winner	...	finish_details
0	Thiago Santos	Johnny Walker	-150.0	130	66.666667	130.000000	2021-10-02	Las Vegas, Nevada, USA	USA	Red	...	NaN
1	Alex Oliveira	Niko Price	170.0	-200	170.000000	50.000000	2021-10-02	Las Vegas, Nevada, USA	USA	Blue	...	NaN
2	Misha Cirkunov	Krzysztof Jotko	110.0	-130	110.000000	76.923077	2021-10-02	Las Vegas, Nevada, USA	USA	Blue	...	NaN
3	Alexander Hernandez	Mike Breeden	-675.0	475	14.814815	475.000000	2021-10-02	Las Vegas, Nevada, USA	USA	Red	...	Punch
4	Joe Solecki	Jared Gordon	-135.0	115	74.074074	115.000000	2021-10-02	Las Vegas, Nevada, USA	USA	Blue	...	NaN

5 rows × 119 columns

In the case of our dataset, there are 119 columns, many of which have data on each fighter's statistics, but also information on neither of the fighters, such as the location, whether or not the fight ended in a finish, etcetera. To reduce the dimensionality of our model, we dropped the columns that didn't have to do with each fighter's statistics, since they don't have much bearing on predicting future bouts. Additionally, we applied feature engineering where we could by replacing columns on individual stats with columns on the differentials between two fighter's stats. By doing this, we were able to reduce the number of features from 119 to 31.

```
df['loss_dif'] = df['B_losses'] - df['R_losses']
df['age_dif'] = df['B_age'] - df['R_age']
df['sig_str_land_dif'] = df['B_avg_SIG_STR_landed'] - df['R_avg_SIG_STR_landed']
df['lose_streak_dif'] = df['B_current_lose_streak'] - df['R_current_lose_streak']
df['win_streak_dif'] = df['B_current_win_streak'] - df['R_current_win_streak']
df['longest_win_streak_dif'] = df['B_longest_win_streak'] - df['R_longest_win_streak']
df['total_round_dif'] = df['B_total_rounds_fought'] - df['R_total_rounds_fought']
df['height_dif'] = df['B_Height_cms'] - df['R_Height_cms']
df['reach_dif'] = df['B_Reach_cms'] - df['R_Reach_cms']
df['avg_sub_att_dif'] = df['B_avg_SUB_ATT'] - df['R_avg_SUB_ATT']
df['avg_td_landed_dif'] = df['B_avg_TD_landed'] - df['R_avg_TD_landed']
df['Avg_TD_pct_dif'] = df['B_avg_TD_pct'] - df['R_avg_TD_pct']
df['avg_SIG_STR_pct_dif'] = df['B_avg_SIG_STR_pct'] - df['R_avg_SIG_STR_pct']
```

```
[453] df['draw_diff'] = (df['B_draw']-df['R_draw'])
      df['odds_diff'] = (df['B_odds']-df['R_odds'])
      df['ev_diff'] = (df['B_ev']-df['R_ev'])
      df['sig_str_landed_diff']=(df['B_avg_SIG_STR_landed']-df['R_avg_SIG_STR_landed'])
```

One other challenge for us in this project was creating a custom model for predicting the winners of fights. Because the task is supervised discrete, and we're predicting for each fight whether the winner would fall into one of two categories, we felt that building our own logistic regression model would be appropriate for this task. To create our own logistic regression model, we developed a training algorithm and a function for testing. The training algorithm involves computing a linear model and passing it through the sigmoid function, then computing gradients of weights and biases, and using them to update parameters. The way we structured our code, we were able to test different learning rates and numbers of iterations for the gradient descent. By testing different learning rates and numbers of iterations, we were able to fine-tune it to make our model closer to convergence.

The sigmoid function is fundamental for building binary classification problems as regression problems, mapping the linear combination of input features and their weights to a probability value. After applying the sigmoid function, we are able to calculate the gradients using the loss function, and our predicted and true output y. We then use our updated gradients, along with the learning rate, to update the weights and biases. After our model is trained for a set number of iterations, coming closer to convergence, we use a predict function to see how close our predicted outputs are to the true labels from our dataset. We use the weights and biases obtained from the fit function, and pass our linear model again through the sigmoid function, obtaining a measure of how closely our model fits to our validation data.

Experiments:

a) Dataset Description

The dataset we're working with is pulled from Kaggle, and contains data on UFC fights from 2010 to 2021. The site is pulled from two main sources, ufcstats.com and bestfughtodds.com, which have data statistics on bouts and fighters, as well as betting odds. The csv file we're working with has both data that is public knowledge before fights, and data that is learned after a fight. For example, a fighter's significant strike percentage can be learned before a fight starts, but whether or not a fight will end in a finish is something that can only be learned after a fight occurs. For this reason, data cleaning was important to us, to ensure that the model would be generalizable to new data, and therefore better at predicting the winners of new fights.

b) Evaluation Metrics

Entire Dataset evaluation:

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**MAE**
Random Forest	0.647	0.694	0.719	0.706	0.353
Logistic Regression	0.660	0.683	0.79	0.732	0.34
Gradient Descent Boosting	0.646	0.685	0.737	0.71	0.354
Logistic Regression(Custom)	0.626	0.759	0.535	0.628	0.374

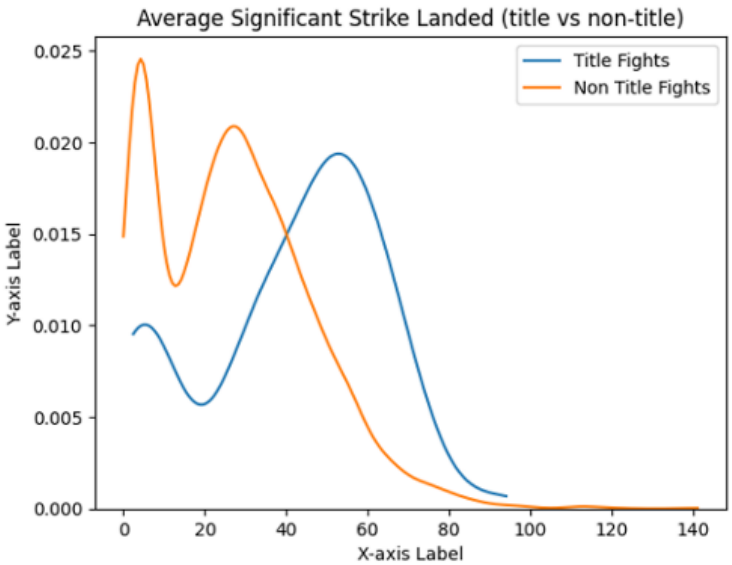
Title Fight evaluation:

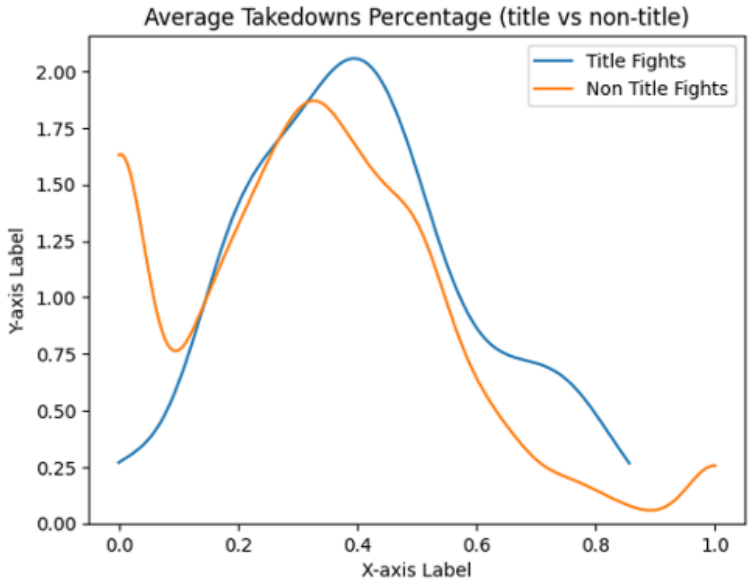
Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**MAE**
Random Forest	0.828	0.844	0.927	0.884	0.174
Logistic Regression	0.81	0.858	0.878	0.867	0.19
GDB	0.776	0.818	0.878	0.847	0.224
Logistic Regression(Custom)	0.793	0.796	0.951	0.866	0.207

c) Major results:

We applied different models to both the entire dataset, as well as the dataset with only title fights (fights where they are fighting for a championship belt). We were surprised to find that for all of the models, the metrics seemed to improve greatly for title fights as opposed to the general dataset. At first we thought this might be due to having a smaller dataset, as models trained on larger datasets may have high variance, since they may become overly complex due to having to fit so much data. However, when we tested the models with a slice of the dataset around the same size as the number of total title fights, our metrics were still close to the metrics of the whole dataset, leading us to understand that the title fights are actually easier for the model to predict.

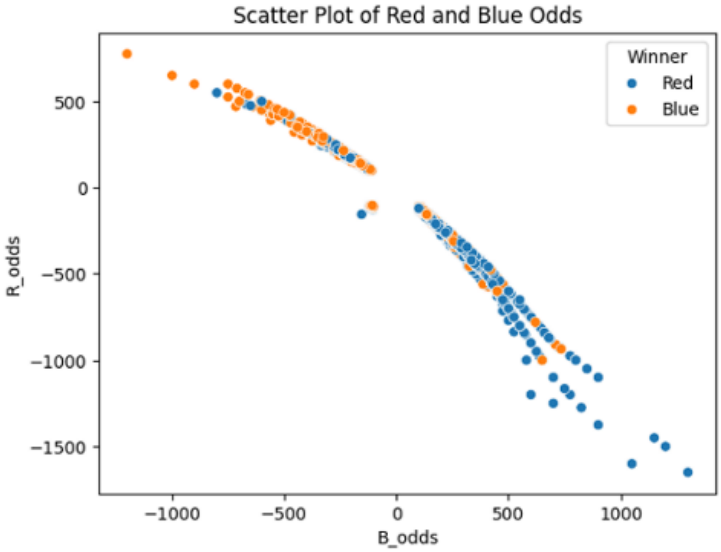
This is an interesting development, and made us wonder why the models find title fights easier to predict. One thing we do know is that title fights usually represent the pinnacle of skills in the sport, with there usually being a large skill disparity between championship-level fighters and the rest of the division. For this reason, we performed some additional analysis on the stats of fighters in title vs non-title fights.

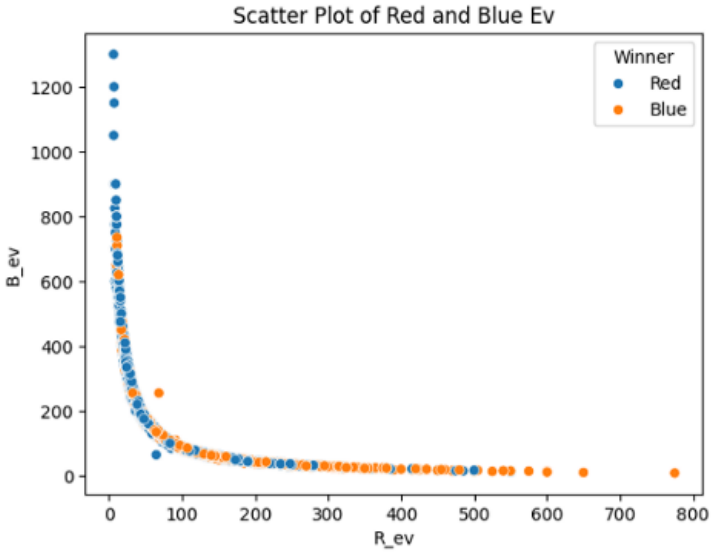




As seen from the above Kernal Density graphs, these fighter statistics follow more of a normal distribution in the title fights, while in the non-title fights their distribution is more chaotic and bimodal. Models trained on data with features following normal distributions are often easier to interpret, since they exhibit clear patterns and have a well-defined central tendency and dispersion. It may be because title holders and contenders have well defined skills and they are able to capitalize on those skills and more reliably beat competitors with lower skills, that their stats are more normalized than non-title fighters.

It is important to note as well that when we calculated the Pearson Correlation Coefficient for each of the features, the ones most strongly correlated with the 'Winner' column were the odds differential and EV differential, the features most strongly associated with betting odds. For this reason, it's also possible that part of the reason the models' predictions are more accurate for title fights might be that the outcomes more closely follow betting odds' predictions.





d) Analysis

Gathering all the information found in above, we are able to see that for the entire dataset evaluation, the best model for Accuracy, Prediction, Recall, F1 Score and MAE, are Random Forest, custom Logistic Regression, Logistic Regression, Logistic Regression and Logistic Regression respectively. While not by a large amount, it does seem that the Random Forest model and Logistic Regression model are the best. We can analyze the reason some models performed better or worse on certain metrics by analyzing how they're defined and trained. Random Forest models are known for being robust to overfitting, due to the way they are made up of multiple decision trees, trained on different subsets of our dataset. This robust approach to building a model is likely the reason why this model performs the best for our dataset.

When we started, there were 119 features in our dataset, and through feature engineering, we reduced this to 31. While this can greatly improve our results, it can result in the GDB model taking a hit in its predictions. GDM models are great with large amounts of variables, as they are able to modify the importance of a variable, resulting in a strong model against outliers (considering the large size). 31 variables is a respectable amount, but perhaps the process of feature engineering and dimensionality reduction may have had enough impact to make the GDB model output lower results in comparison to the other models. This is not to say that it was a wrong choice, but for a model like GDB that does well with said circumstances, it makes sense that the other models performed better. If we were to only use the GDB model, there's a chance that the model would have performed better if we had used all 119 variables and had specified the importance of each.

As for the Logistic Regression model and the Random Forest model, each has reasons as to why they performed the way they did. For Logistic Regression, we can prevent overfitting even further with L1/L2 regularization. Random Forest, like the GDB model, can help with the importance of variables, and also does it automatically.

One surprising outcome to these experiments was that the custom logistic regression model only performed best in one category, however, our custom LR model performed better than Gradient Descent Boosing in several categories. Seeing as how the logistic regression model had the overall best performance, being best in 3 of the 5, it would make sense that a modified version would do around as good, or hopefully better. Our custom Logistic Regression model has higher recall, F1 score, and

accuracy than the GDB, but a lower precision than the GDB model. This means that the overall correctness of our model is higher than GDB, and the higher recall means that the proportion of true positive predictions to actual positives is higher. This indicates that our custom LR model correctly identifies more actual positive instances than the GDB model. The lower recall, however, indicates that there are more false positives picked up by our model than by the GDB model. In the context of our dataset, 'positives' are represented as 'red' fighters, and 'negatives' are represented as blue fighters. Because of this, a large ratio of false positives indicates that the custom model predicts 'red' fighters are the winners oftentimes, even when they are not. Given that the dataset was organized in a way that 'red' fighters are almost often the betting favorites, this may indicate that the model was relying to heavily on the betting features as opposed to the other features.

Looking at the second subset of the dataset we tested, the Title fights, the best models for Accuracy, Prediction, Recall, F1 Score and MAE are Random Forest, Logistic Regression, Custom Logistic Regression, Random Forest and Random Forest. As mentioned above, all models performed noticeably better for each category, around 10-15% to be more specific. However, we do see a bit of a difference as to which is best for each category, mainly in the F1 score, and MAE (the Prediction category and Recall rate do have different 'best models' but seeing as how they are both logistic regression, we can group them together lightly). The reasoning for this is a bit hard to determine, but one possibility for this can be because of a detail mentioned above.

Due to the way the distributions are more normalized for title fights than non-title fights, this may result in the models being able to more accurately make predictions. This is likely due to the distributions' exhibiting clearer patterns, and having a well-defined central tendency and dispersion. This resulted in much better accuracy, as well as better performance all other metrics.

Conclusion/Future works:

By analyzing 10 years of UFC data, applying feature engineering multiple different machine learning models, we were able to see how different models compare for the task of predicting the winners of UFC fights. Random Forest Classifiers, with their robustness to overfitting, resulted in the overall best results for us. One major result for us is that when we ran all of the models again, but with a subset of the dataset that represented only title fights, we were able to achieve much better results on all metrics.

With a Random Forest model on the title fights, we were able to achieve 82.8% accuracy, 84.4% precision, 92.7% recall, an F1 score of 88.4% and an MAE of 17.4%. While these are not perfect metrics, they do seem impressive compared to the metrics on the general dataset, and given the chaotic nature and unpredictability of UFC fights. Also, given the 10-15% jump in all of the metrics from the general dataset to the dataset involving title fights, we were able to use this project to demonstrate how normalization affects model performance. Specifically, most of the fighter statistics follow a more normal distribution in the title fights compared to the non-title fights, which seems to be a strong factor for the models' ability to predict winners, given the large improvement in metrics.

As far as future works, it would be imperative that we tried more reguarlization methods, along with the feature engineering, to see how that would affect performance. We did try regularization methods, but they affected model performance negatively, so it may have been due to the way we did the dimensionality reduction and feature engineering in tangent with the regularization. Due to the way the

Pearson Correlation revealed the betting features to have the strongest correlation to the winner column, regularization would be imperative to ensure that the models aren't fitting too closely to the betting features. We would also like to design a web scraper and web application for this project. Since the dataset we worked with only encompasses 10 years of data, a web scraper would be useful for pulling all of the data needed to predict a particular fight, if that fighter's current statistics aren't in our dataset. Also, a web application would we a great addition, as it would allow a user to input two particular fighters, to see how models will predict the outcome.