Applying Machine Learning and Convex Optimization to the UFC Betting Market

Eugene Han
Department of Statistics
Yale University
e.han@yale.edu

Matt Russo
Department of Computer Science
Yale University
matt.russo@yale.edu

Alex Yuan
Department of Computer Science
Yale University
alex.yuan@yale.edu

Aaron Yu
Department of Computer Science
Yale University
aaron.yu@yale.edu

1 Introduction

Mixed martial arts has seen a rapid increase in popularity in mainstream sports, with the Ultimate Fighting Championship (UFC) being one of the largest promotions in the world. Consequently, betting on UFC events has become increasingly prominent, intensified by aggressive marketing and easy-to-use platforms provided by bookmakers like DraftKings and FanDuel.

Betting markets are generally very efficient and the implied probabilities of sportsbooks' odds are often "accurate" in that they match up well with the relative frequency of events occurring. However, bookmakers can offer mispriced odds under/overestimating a fighter's chances of winning, yielding opportunities to generate profit if identified correctly.

This has prompted numerous attempts at modeling the UFC. These include the work of Holmes et al. (2022) with their predictive Markov Chain Model [2] and Cunha (2020) with their Deep Neural Network classifier for forecasting the results of mixed martial arts contests [1]. In most of these attempts, however, there are flaws in their methodologies with many performing backtesting – evaluating a model on historical data – incorrectly, using data features with leakage, or focusing on hard classifications. In addition, the authors used generic and easily accessible data from the official UFC Stats and ESPN websites.

We propose a more novel approach relative to public research on forecasting UFC fights. In addition to data from the official UFC Stats website, we analyze alternative data from separate disparate sources, including Sherdog, Fight Matrix, and FightOdds.io to create a rich feature set. Utilizing logistic regression, random forest, and gradient boosting models, we investigate the potential to gain an edge in the UFC betting market by producing well-calibrated probabilities for fight outcomes and formulating an optimal bet sizing algorithm to calculate optimal wager sizes using those predicted probabilities. Finally, we evaluate and compare our models' betting performance on unseen fights from 2022 through 2024 given average closing odds, which reflect the most information and are therefore the "sharpest" lines. In doing so, we achieved models comparable to the bookmakers, demonstrating that the UFC betting market can be modeled effectively.

Throughout this process, we faced numerous challenges. The most challenging and time-consuming part was creating a rich feature set from our scraped data, which was unusable without some transformation and aggregation. This required immense creativity, experimentation, and a deep understanding of the sport to produce the best features. Additionally, care had to be taken to prevent data leakage, ensuring that all features were constructed with information known strictly before a fight occurred.

2 Data

The data used for this project was scraped from the following sources:

- UFC Stats - Mainstream analytics website for the UFC that contains per-round fight statistics regarding aspects like striking and takedowns as well as fighter characteristics such as height and reach
- Sherdog - Third-party website not affiliated with the UFC that acts as a database for 300,000+ fights across multiple professional and amateur promotions; has information on UFC fighters' full professional careers including fights before signing with the UFC and extra data like nationalities
- Fight Matrix - Third-party website that uses Sherdog's data and proprietary algorithms to maintain a custom ranking system and Elo-like ratings
- FightOdds.io - Historical and upcoming betting odds across several bookmakers
- Open-Elevation API - Open-source API for getting elevation data rounded to the nearest meter for a given set of latitude and longitude coordinates

We utilize the Scrapy framework in Python to handle the scraping logic, cleaning, and general data normalization and store the data in SQLite databases. The source code can be found here. Since the data updates on an approximately weekly basis, we set a cutoff date of March 16, 2024, so that we could work with static databases.

We then augmented the databases with linkage tables to create one-to-one mappings between the fighter and event keys across the different sources so that all engineered features could be tied back to the UFC Stats data. Dedicated tables were necessary because fighter and event details such as name and date differed across the data sources. We accomplished this through clever table joins and unions, fuzzy string matching, and manual matching to handle leftover edge cases.

The full entity relationship diagram of our databases' tables can be found in Figure 4.

3 Methodology

Our approach consisted of the following components:

1. *Feature Engineering* - Leveraging domain knowledge, we constructed a rich set of features that we believed would strongly influence the results of a fight. See Section 4.1 for more details.
2. *Modeling* - By framing each fight as a binary classification, we trained three models to produce well-calibrated win probabilities, based on our engineered features. Our first method was Logistic Regression, which we selected because it provides probabilities rather than hard classifications and is generally well-calibrated and easy to compute. We included L2 penalization as a means to prevent overfitting. Our second model was Random Forest, which we chose to implement as a means to capture any nonlinear relationships between two features – logistic regression fails to do this unless those nonlinear relationships are engineered as a feature in advance. Our third model involved Gradient Boosting selected to capture potentially complex relationships between features and outcomes. Each of these models was fit to minimize log loss.
3. *Bet Sizing* - We extended the Kelly criterion for optimal bet sizing to a general number of simultaneous bets. Given there are n fights in an upcoming event and restricting to moneyline straight bets (no parlays, spreads, or over/under bets), there exist $2n + 1$ potential positions–two outcomes for each fight and a risk-free asset (no bet). Maximizing for expected log wealth, $\mathbb{E}\left[\log\left(Rb\right)\right] = \pi^\intercal \log\left(Rb\right)$, we formulated this as the following optimization problem:

$$\begin{aligned} \max_{b} \quad & \pi^\intercal \log\left(Rb\right) \\ \text{s.t.} \quad & 1^\intercal b = 1 \\ & b \succeq 0 \end{aligned}$$

where $\pi \in \mathbb{R}^{2^n}$ is a vector of probabilities corresponding to each of the 2^n different event outcomes, $R \in \mathbb{R}^{2^n \times (2n+1)}$ is the returns matrix encoding the returns for each of the $2n+1$ positions for each of the 2^n outcomes, and $b \in \mathbb{R}^{2n+1}$ is a vector of wager proportions for each bet. This problem is always feasible: in the worst case, take $b^* = e_{2n+1}$, the $(2n+1)$-th basis vector in $\mathbb{R}^{2n+1}$, which is equivalent to not betting at all and guarantees a profit/loss of 0. Since π is intractable, we computed an estimate $\hat{\pi}$ using the predicted probabilities from our classifier and assuming independence of fights within an event.

4. *Backtesting* - Finally, we evaluated and compared our models' betting performance on unseen fights from 2022 through 2024 given average closing odds, which reflect the most information and are therefore the "sharpest" lines. We consciously chose to not engineer our models with features whose values were not measurable at the time of the fight. Additionally, to simulate realism, we did not drop fights that were canceled or ended in a draw, as a real bettor using a predictive model could bet on fights that don't end with a winner.

4 Implementation Details

4.1 Feature Engineering

The feature engineering process relied heavily on intuition and domain knowledge of the sport and the data itself. Much of the idea generation came from reflecting on what characteristics of a fighter would influence the outcome of a fighter. For instance, it makes sense to hypothesize that younger fighters tend to beat older fighters who are well past their prime. Alternatively, fighters who have received a lot of damage from accumulating head strikes or being knocked out frequently will on average be more susceptible to getting knocked out again or fighting more anxiously.

The general strategy was to create the same features for each of the fighters in the red and blue corners of the arena and then derive a final set of features by taking differences (red fighter's feature - blue fighter's feature) and ratios (red fighter's feature / blue fighter's feature), discarding the individual fighters' features at the end. The motivation was to avoid having too many features; we could have half the number of features while effectively retaining most of the information in the original individual fighters' features. From there, we removed fights that ended in draws and no contests from the training set, which consisted of events from 2010 to the end of 2021, transforming the problem into a binary classification problem by letting the red fighter winning be the target column.

Our final feature set consisted of 51 features, with implementation done exclusively in SQL to exploit the relational structure of our data. Ten such features can be found in Table 1 below.

4.2 Modeling

4.2.1 Logistic Regression

We expected logistic regression to perform relatively well on the dataset because of the approximately linear relationships between some features and the target classes. First, we constructed a pipeline that rescaled the input to minimize the effect of differences in scale of different features (for example, `SHERDOG_TIME_SECONDS_FOUGHT_DIFF` could have absolute values of over 10000, while `SHERDOG_LOSING_STREAK_DIFF` had a maximum absolute value of 6). Next, the pipeline used Scikit-learn's logistic regression model to classify each observation.

Then, we performed a grid search over the regularization constant `C`. We initially performed a grid search over both the value of `C` and the penalty used (L1 or L2), but we chose to restrict the model to L2 regularization for two primary reasons. First, we used L2 regularization to avoid losing information in the data, since our testing showed that the models produced using L1 regularization had worse calibration than those using L2. Second, L1 regularization does not support the `lbfgs` solver in Scikit-learn, which gives faster convergence and increased stability compared to alternatives, so we chose to restrict the grid search to different values of `C` for L2 regularization. The optimal value of C returned by the grid search was approximately 0.006 (but not exactly equal to 0.006 because we used a log scale for the grid search).

Feature Name	Description
SHERDOG_AGE_DAYS_DIFF	Difference in fighters' ages in days at the time of the event based on dates of birth sourced from Sherdog; a proxy for wear and tear
FIGHTMATRIX_RANKING_POINTS_RATIO	Ratio between fighters' most recent point values from monthly Fight Matrix rankings before the fight
UFCSTATS_OPPONENT_SIGNIFICANT_STRIKES_LANDED_PER_MIN_DIFF	Difference in the number of significant strikes absorbed per minute based on cumulative pre-fight counts as recorded on UFC Stats; a comparison of the relative effectiveness at defending strikes
FIGHTMATRIX_ELO_DIFF	Difference in pre-fight Elo scores based on a modified Elo rating system maintained by Fight Matrix
FIGHTODDSIO_ODDS_AVERAGE_DIFF	Difference in the running average of closing decimal odds, averaged across multiple bookmakers listed on FightOdds.io, over the fighters' past fights; note that this does **not** include the closing odds for that particular fight, so no leakage is present
SHERDOG_DAYS_SINCE_DEBUT_DIFF	Difference in the number of days since fighters' professional debuts at the time of the fight based on full professional fight history data from Sherdog
UFCSTATS_SIGNIFICANT_STRIKES_EXPECTED_ACCURACY_DIFF	Difference in the geometric mean of a fighter's significant strikes accuracy and the proportion of significant strikes their opponent has absorbed based on UFC Stats data
FIGHTMATRIX_GLICKO1_DIFF	Difference in pre-fight Glicko-1 scores based on a Glicko-1 system maintained by Fight Matrix; similar to Elo but uses a different formulation
FIGHTMATRIX_ELO_RATIO	Ratio between pre-fight Elo scores based on a modified Elo system maintained by Fight Matrix
FIGHTODDSIO_IMPLIED_PROB_AVERAGE_DIFF	Difference in the running average of the implied probabilities derived from averaged closing odds from FightOdds.io over the fighters' past fights; note that this does **not** include the closing odds for that particular fight, so no leakage is present

Table 1: Top 10 most important features by the absolute coefficient values in the logistic regression model. Differences and ratios are in the form of red corner fighter vs. blue corner fighter.

4.2.2 Random Forest

Given its ability to easily evaluate nonlinear feature importance and effectively take advantage of having many trees to infer probabilities, we expected the Random Forest model to perform relatively well on the dataset. To implement it, we first initialized Scikit-learn's `RandomForestClassifier` with `n_estimators` $= 200$, `criterion` $=$ `"gini"`, `random_state` $= 0$, and `n_jobs` $= -1$.

To determine the optimal values for `max_depth` and `max_leaf_nodes`, we ran a grid search, an exhaustive search over specified parameter values for an estimator, using Scikit-learn's `GridSearchCV` with a stratified 5-fold cross-validation scheme and scoring based on negative log loss. After experimentation with various parameter values, we found that `max_depth` $= 12$ and `max_leaf_nodes` $=$ 70 were the most optimal.

4.2.3 Gradient Boosting

Although libraries such as XGBoost, LightGBM, and Catboost offer high-performing implementations of gradient boosting, we found from experimentation that they were prone to overfitting on our dataset. This is not surprising given their large number of parameters and the relatively small training set with less than 4000 rows. As a result, we opted for Scikit-learn's implementation which has significantly fewer parameters, making it easier to tune.

We initialized the `GradientBoostingClassifier` with `n_estimators` $= 100$, `max_features` $=$ `"sqrt"`, and `max_leaf_nodes` $=$ `None`. We then ran a grid search over `learning_rate` $\in \{0.01, 0.1\}$ and `max_depth` $\in \{2, 3, 5\}$ using a stratified 5-fold cross-validation scheme with scoring based on negative log loss, resulting in optimal values `learning_rate` $= 0.1$ and `max_depth` $= 2$.

4.3 Simultaneous Kelly Bet Sizing

Given our optimization formulation

$$\begin{aligned} \max_{b} \quad & \hat{\pi}^\intercal \log(Rb) \\ \text{s.t.} \quad & 1^\intercal b = 1 \\ & b \succeq 0 \end{aligned}$$

we construct R and $\hat{\pi}$ as follows.

Let $o_r = (o_{r,1}, o_{r,2}, \ldots, o_{r,n})^\intercal$ and $o_b = (o_{b,1}, o_{b,2}, \ldots, o_{b,n})^\intercal$ represent decimal odds for the red and blue fighters respectively such that $o_{r,i}$ and $o_{b,i}$ are the red and blue fighters' odds in fight i in an event. Moreover, let $\hat{p}_r$ and $\hat{p}_b$ be defined similarly so that $\hat{p}_{r,i}$ and $\hat{p}_{b,i}$ are the red and blue fighters' predicted win probabilities in fight i as taken from a classifier.

Each row of the matrix R represents the returns for one of the 2^n possible sequences of fight results. Since decimal odds are equal to the return for a unit wager, the entries of R can be directly filled using o_r and o_b. To illustrate, for $n = 2$ fights in an event,

$$R = \begin{pmatrix} o_{r,1} & 0 & o_{r,2} & 0 & 1 \\ o_{r,1} & 0 & 0 & o_{b,2} & 1 \\ 0 & o_{b,1} & o_{r,2} & 0 & 1 \\ 0 & o_{b,1} & 0 & o_{b,2} & 1 \end{pmatrix}$$

where the first row encodes returns for when the red corner fighter wins in both fights. Since we want the entries of $\hat{\pi}$ to match up with this logic, we then have

$$\hat{\pi} = \begin{pmatrix} \hat{p}_{r,1} \cdot \hat{p}_{r,2} \\ \hat{p}_{r,1} \cdot \hat{p}_{b,2} \\ \hat{p}_{b,1} \cdot \hat{p}_{r,2} \\ \hat{p}_{b,1} \cdot \hat{p}_{b,2} \end{pmatrix}$$

After solving for the optimal values for b^*, we multiply b^* by the current bankroll to get the actual wagers in dollars and clip them to stay above a minimum bet size of \$0.10. To compute R and $\hat{\pi}$ efficiently, we used `numpy` and `itertools`. The problem was specified using `cvxpy` and solved using the Clarabel solver. The entire logic is implemented in the `SimultaneousKelly` class.

4.4 Backtesting

To evaluate the profitability of our approach, we backtested our models and bet sizing strategy over fights from 2022 to the present day, which were not part of the training data. The implementation of this method involved two classes, namely `SimultaneousKelly` (as discussed in Section 4.3) and `BacktestingFramework`. We iterated over a dataframe that merged a model's outputted probabilities and the betting odds, and sliced by event, which was done to simulate placing wagers on all bouts in a given event before actually receiving profits from the result of those bouts. This data was fed to the `SimultaneousKelly` class, which calculated the optimal wagers in dollars for each fight given the pre-event bankroll. Finally, net profit on these wagers was calculated based on the result of the fight and added to the bankroll, such that the next set of wagers was calculated with the new profit or loss included in the bankroll. In the event that a draw occurred or a bout was canceled, the initial wager was returned with no net profit or loss, just as it would in real life.

In effect, the `BacktestingFramework` simulates a real bettor that uses one of our models to estimate probabilities of each fighter winning, uses the Kelly criterion to place wagers as a percentage of his/her current bankroll, and starts with a 100 dollar allowance. This framework was run on each of our models, as well as a dummy strategy, which simulated a naive bettor that simply bet 1% of his/her pre-event allowance on the betting favorite. Refer to Section 5.3 for the results.

5 Results

5.1 Performance Metrics

To evaluate our approach, we ran our tuned models on our test set, which consisted of fights from the start of 2022 to our cutoff date of March 16, 2024. We will discuss both of the primary criteria

Model	Log Loss	Brier Score
Logistic Regression	**0.620396**	**0.216587**
Random Forest	0.627897	0.219215
Gradient Boosting	0.626428	0.218898
Bookmakers	*0.610120*	*0.210988*

Table 2: Comparison of bookmaker's and models' log losses and Brier scores on test set

outlined in our proposal to measure success: log loss, which approximates how close our model came to the bookmakers' implied probabilities, and profitability, which measures how our model performs on data it hasn't seen yet. We computed log losses and Brier scores for each model and the bookmakers' average closing odds, omitting fights that ended in draws or no contests from the calculation. The results are shown in Table 2.

Of the three models we constructed, none of the models outperformed the log loss and Brier score of the bookmakers' closing odds. This result is unsurprising because we would expect the bookmakers to have access to models with more information, including information about fighters or previous fights that are not publicly available, larger datasets that may include more features and more data points, and more compute. Bookmakers also have the ability to adjust prices based on betting behavior, which means they may change their odds based on bets placed by "sharp" bettors, or bettors who have a history of winning well over 50% of their bets. Since our models were unable to take the behavior of other bettors into account, it may be impossible to outperform the bookmakers using only data from previous fights.

All three models achieved losses and Brier scores that were close to those of the bookmakers (within 0.02 for each model's log loss). The log loss for each model was well within the difference of 0.05 outlined in our proposal, meaning each model can be considered a successful approach to using machine learning techniques to model win probabilities for UFC fights. However, it is important to note that the bookmakers' log loss and Brier score were calculated using closing odds, which are time-dependent and calculated close to the start of the fight. Our features were extracted only from features that were not time dependent, meaning they were not able to update their predictions leading up to each fight, unlike the bookmakers they were being compared to. Therefore, it is possible that our models would have achieved better log losses and Brier scores than the bookmakers at earlier times, when, the bookmakers' odds would have been more uncertain and speculative.

5.2 Probability Calibration

Next, the calibration plots for each model are shown in Figure 1, along with the bookmakers for reference.

We initially believed that we could create a model that took advantage of the fact that the bookmakers systematically underestimated high win probabilities and overestimated low win probabilities. As can be seen from the calibration plots, our models experienced the same problems to a certain extent, with each model predicting higher probabilities than the true proportions for low probabilities, and predicting lower probabilities than the true proportions for high probabilities. Since the bookmakers' probabilities also exhibited the same behavior, it is reasonable to conclude that the imperfect calibration emerges from the underlying data that was used to train both the bookmakers' model and our model. Because of the nature of UFC fight data, both our models and the bookmakers' odds rarely predict very high or very low probabilities of winning, making achieving perfect calibration in those regions very difficult. Therefore, the calibration of our models can best be compared to the bookmakers' calibration when the predicted probability is close to 0.5.

Although obtaining a perfectly calibrated model is therefore impossible or very difficult using the available data, it can be observed from the plots above that the logistic regression model exhibited better calibration than the bookmakers when the predicted probability was between approximately 0.25 and 0.75. Most of the predicted probabilities for our logistic regression model in this range are very close to the actual probabilities, compared to the bookmakers' predicted probabilities, which appear to be consistently lower than the actual probabilities in the same range. Therefore, we were successful in using logistic regression to produce better-calibrated probabilities than the bookmakers

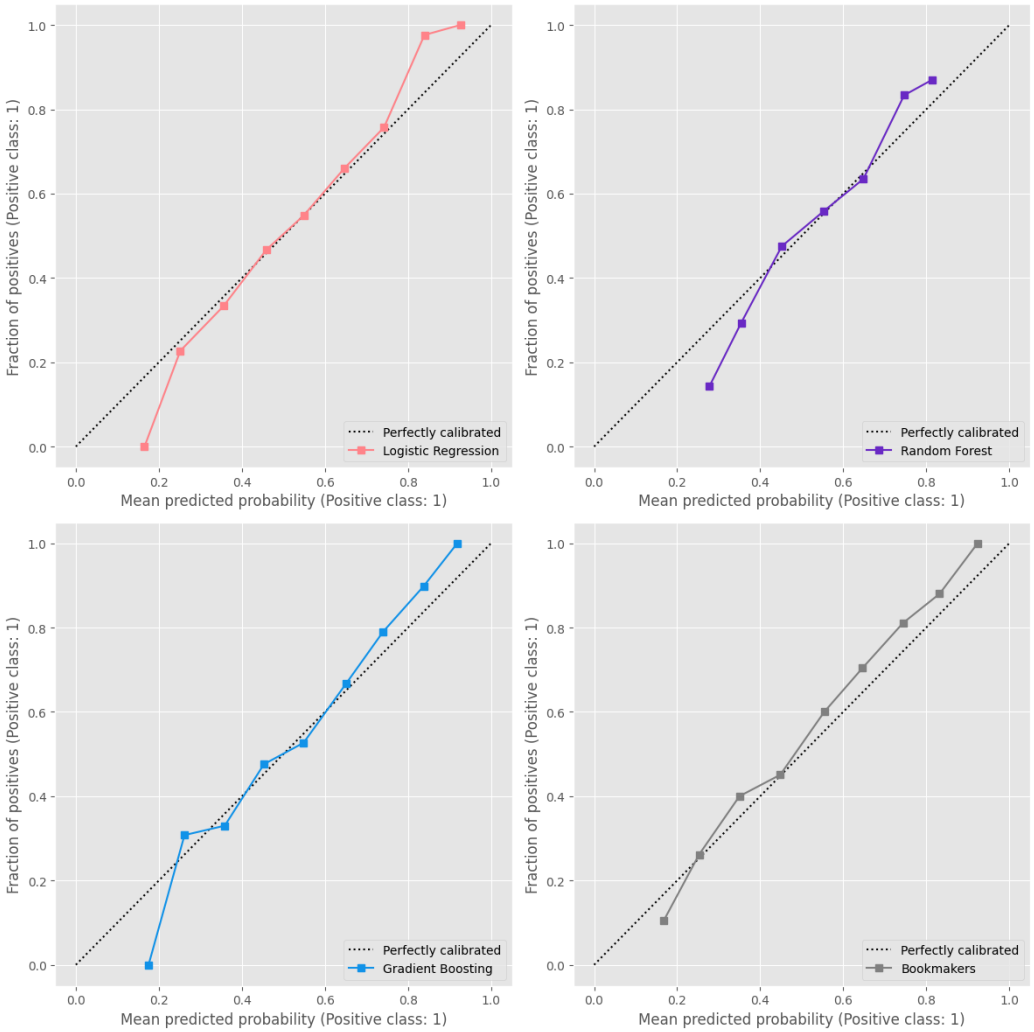

Figure 1: Calibration plots for the bookmaker's and models' probabilities

Strategy	Final Bankroll ($)	ROI (%)
Logistic Regression + Kelly	**216.67**	**11.39**
Random Forest + Kelly	158.01	7.55
Gradient Boosting + Kelly	148.70	6.15
1% Favorites Only ("Dummy")	90.55	-1.07

Table 3: Backtest results by strategy given an initial bankroll of $100

in the most common range of probabilities. With a larger dataset, it may be possible to outperform the bookmakers over all predicted probabilities, but because fighters are unlikely to have a very high or low true probability of winning a given fight, we can conclude that the logistic regression model outperformed the bookmakers in probability calibration.

5.3 Backtesting

Lastly, the results of backtesting are shown in Table 3 and Figures 2 and 3. The three models we constructed are shown, along with a dummy strategy for comparison, which bets 1% of the bankroll on every betting favorite (the fighter with lower decimal odds).

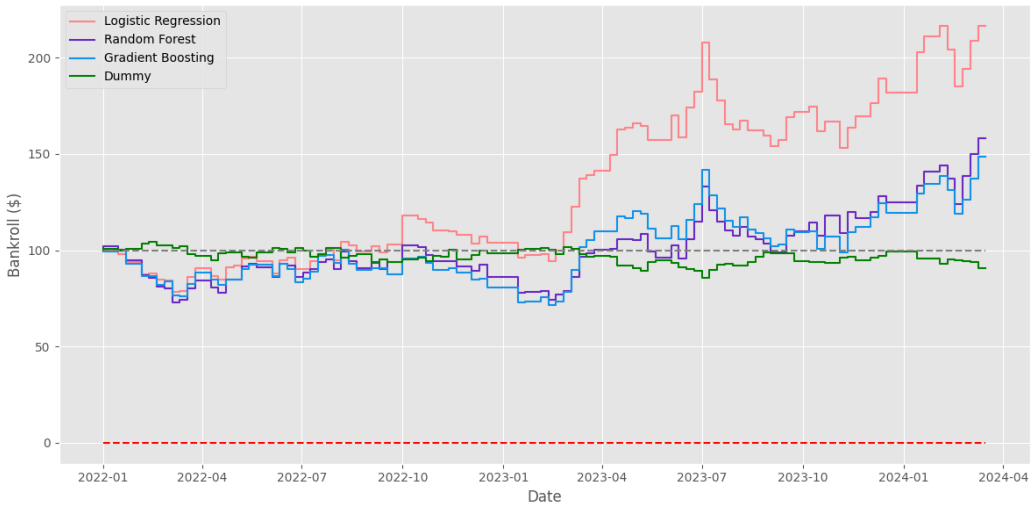

Figure 2: Bankroll by strategy over backtest period

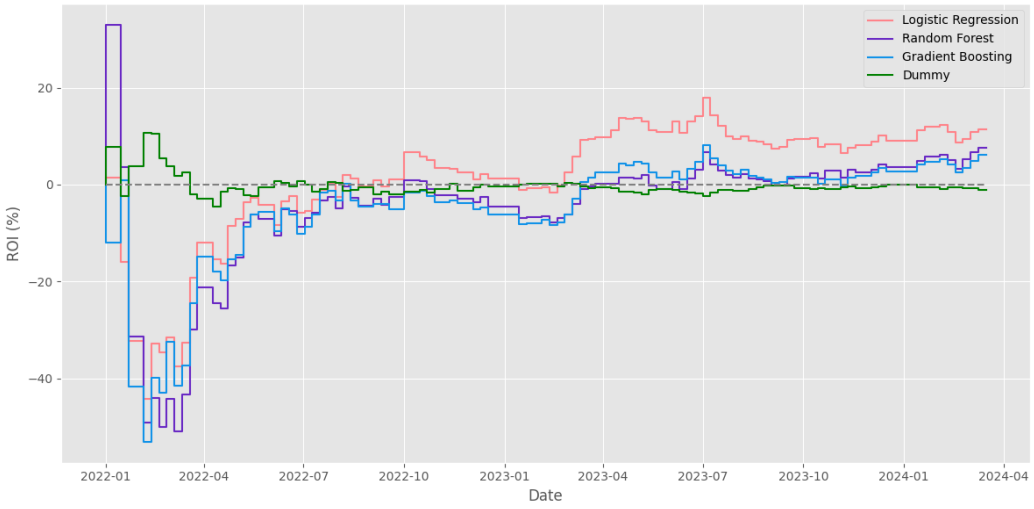

Figure 3: Return on investment by strategy over backtest period

In our proposal, we outlined positive returns as the primary success metric for backtesting of models. All three models made substantial gains when considering their final bankrolls, with logistic regression ending with more than double the amount it started with. It is also clear from the performance of the dummy strategy that a positive return is not guaranteed. Therefore, in this project, we have successfully shown that it is possible to use machine learning to generate long-term profits from the UFC betting market.

Although the random forest model and gradient boosting models had similar performance, the logistic regression model finished with a much higher bankroll than the other two models. This difference in performance could be attributed to the fact that the data may have an approximately linear relationship between the features and the target. If the data were approximately linear, we would expect logistic regression to perform better, while tree-based methods, including both the random forest model and the gradient boosting model, would perform better on data with nonlinear relationships. It is also likely that the relatively small sample size impacted the performance of the random forest and gradient boosting models more than the logistic regression model, because of a combination of potential overfitting, difficulties with calibration on a small dataset, or high variance in the training data.

6 Conclusion

In this paper, we explored whether or not one can obtain an edge in the UFC betting market. To this end, in addition to using easily accessible data from the official UFC Stats website, we analyzed alternative data from several disparate sources. We then constructed logistic regression, random forest, and gradient boosting models to produce well-calibrated probabilities for fight outcomes and formulated an optimization problem based on the Kelly Criterion for optimal bet sizing to calculate optimal wager sizes using those predicted probabilities. Finally, we evaluated our model through backtesting, comparing our models' betting performance on unseen fights from 2022 through 2024 given average closing odds.

In our proposal, we defined success as having achieved two things. The first is in terms of how our models compare to closing odds as well as the profitability of each pipeline. We wanted to achieve a log loss as close as possible to or even better than the bookmakers' implied probabilities–no worse than a 0.05 difference. Although our models did not outperform the bookmakers' log loss and Brier score, this was expected given that the bookmakers have an information advantage over retail bettors. Despite this, all three of our models achieved losses and Brier scores close to those of the bookmakers (within 0.02 for each model's log loss). This was well within the target difference of 0.05 outlined in our proposal, meaning that each of our constructed models can be considered a successful approach to using machine learning techniques to model win probabilities for UFC fights.

The second is in terms of profitability where success would be a positive return over the backtesting period. All three of our models made substantial gains when considering their final bankrolls, with logistic regression ending with more than double the amount it started with. This was especially significant given the performance of a naive strategy (favorites only) which showed that a positive return is not guaranteed. Therefore, we successfully demonstrated that it is possible to use machine learning to generate long-term profits from the UFC betting market.

Although our current project focuses on a select set of features, investigating additional features and various combinations could enhance predictive power. It would also be natural to experiment with other machine learning models. We hope to pursue these extensions in future works.

7 Statement of Contributions

Eugene was responsible for data sourcing and acquisition, database design, feature engineering, implementing simultaneous Kelly betting, training the gradient boosting model, implementing the performance metric and probability calibration evaluation framework, and overall project management and idea generation.

Matt was responsible for assisting in data acquisition and reviewing queries in the feature engineering process. Matt also took responsibility for designing and implementing core functionality and logic for the backtesting framework.

Alex was responsible for assisting in data acquisition, maintaining the general code infrastructure, implementing and training the random forest model, and creating the 1% Favorites Only Dummy strategy for backtesting.

Aaron was responsible for reviewing the queries in the feature engineering process, writing, training, and testing the logistic regression model and preprocessing pipeline, and analyzing and reporting on the final results of all models and experiments.

References

[1] Cunha, Felipe (2020). Predicting UFC bouts with a DNN classifier, *Towards Data Science*. https://towardsdatascience.com/predicting-ufc-bouts-with-dnn-classifier-f955e9abe6c6.

[2] Holmes, Benjamin, et al (2023). A Markov Chain Model for Forecasting Results of Mixed Martial Arts Contests. *International Journal of Forecasting*, vol. 39, no. 2, 1 Apr. 2023, pp. 623–640, https://doi.org/10.1016/j.ijforecast.2022.01.007.

Appendix

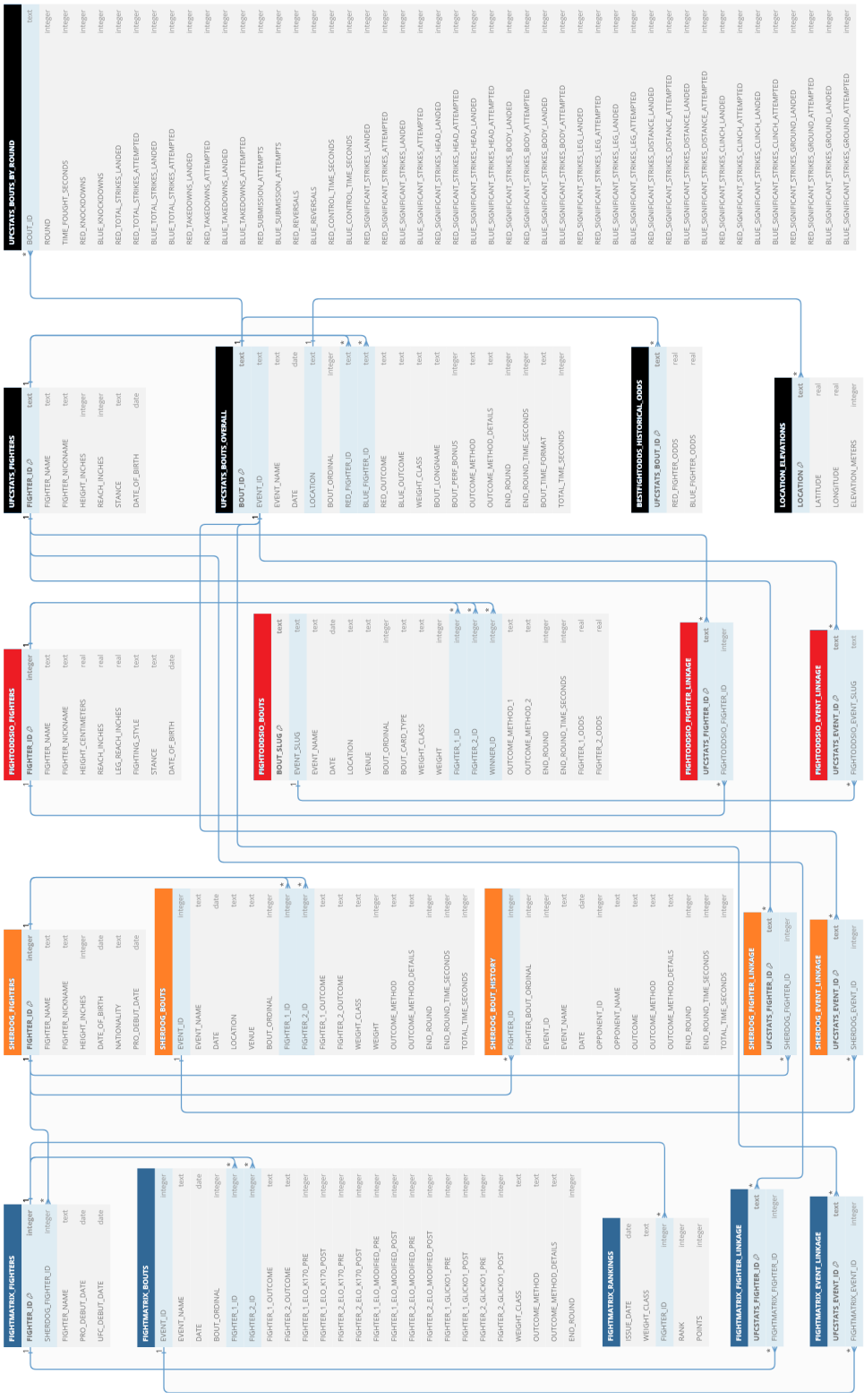

Figure 4: Entity relationship diagram of the data