UFC Upcoming Fights Statistics (December 2024)

Jared De La Serna, San Diego State University, San Diego, CA, USA

Abstract-UFC, the "Ultimate Fighting Championship," is a professional mixed martial arts (MMA) organization since 1993 designed to see the best athletes with different martial arts backgrounds, such as jiu-jitsu, boxing, and other combat sports. [1] UFC has changed many essential sports rules over the years, which shows us how combat sports have evolved. The project uses machine learning models to predict the outcomes of specific future UFC fights, such as the winner or the method of victory. We use the dataset "UFC Complete Dataset (All events 1996-2024)" from MaksBasher[3] on Kaggle. He obtained the dataset by web-scraping it from UFC's websites. We mainly focused on predicting PrizePicks categories. PrizePicks is a gambling platform where the user has to pick more or less player stats. The UFC PrizePicks' categories are the length of the fight, significant strikes, and takedowns. While predicting values, we had to address the missing values in our dataset. We handled those missing numeric and categorical values using Random Forest Regressors[6] and Classifiers[5]. We also did exploratory data analysis (EDA) to uncover insights into the variables we were interested in predicting. Additionally, we explored encoding [9] categorical variables, mainly fighter stances and methods of victory. This was necessary for linear regression for feature impact analysis. We also created pipelines to handle complex datasets, address missing values, and build strong, accurate predictive systems for new fight scenarios.

I. Introduction

The Kaggle repository had two datasets we could use: "large_dataset.csv" and "fighter_stats.csv." The large_dateset file is a dataset with statistics of every recorded UFC fight, and the fighter_stats has statistics of specific fighters. Both datasets provide necessary data to help predict the winner and statistics of certain fighters in a fight. We wanted to merge both datasets into one primary dataset to have all the information about each fighter and fight. We did two merges, first using the r_fighter column as the key to match the r_fighter to the name column in fighter_stats. Our left join kept all the rows from master_ufc, even if no match was found in fighter_stats. The second merge was the same, just with blue_fighter as the key. The merged dataset now contains red-corner fighter statistics and blue-corner fighter statistics. Now, we can work with our significant data set. Our main objective was to develop a machine-learning system for UFC analytics.

II. Objective

Our objective was to predict fight outcomes and relevant statistics, such as the winner, methods, duration, and the number of significant strikes and takedowns the fighter would have.

III. Motivation

Sports analytics has always fascinated us, especially as data science students. The field will grow exponentially, requiring people with strong sports analytics backgrounds. The University of New Haven states that sports analytics "is expected to grow at a CAGR of 40.1% to reach approximately $4 billion by 2022." [8] These staggering numbers give sports analytics immense potential for improvement and innovation.

IV. Challenges

Our main challenge was that the dataset was heterogeneous data, with numerical, categorical, and temporal datatypes in our dataset. Most of the data was numeric, but categorical variables such as “stance” and “method” could mess up our training models. Our only temporal variable was fight dates. For the simplicity of this project, these variables were not used. Many fighters' statistics were unavailable for certain fights or upcoming bouts. Certain fights do not have specific statistics because the UFC has only been around for 32 years. At the beginning of the UFC, no workers were counting and recording all the statistics. Model scalability refers to the model's ability to handle multiple prediction tasks efficiently without compromising performance. This project was a challenge due to the diversity of our dataset in predicting targets, including winner classification, fight duration, and significant strikes from both fighters. Each target requires the best model to adapt to the dependent variables.

V. Hypothesis

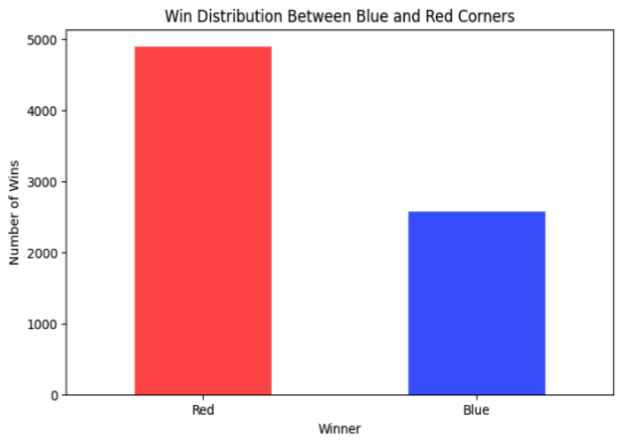
The outcome of a UFC fight can be influenced by a combination of fighter-specific statistics, such as their significant strikes landed and takedown accuracy, and situational factors, such as weight class and fight duration. Significant statistical differences between opponents are the strongest predictors of victory. Machine learning models can also uncover nonlinear interactions between variables, giving us better predictions when working with categorical variables.

VI. Task Description

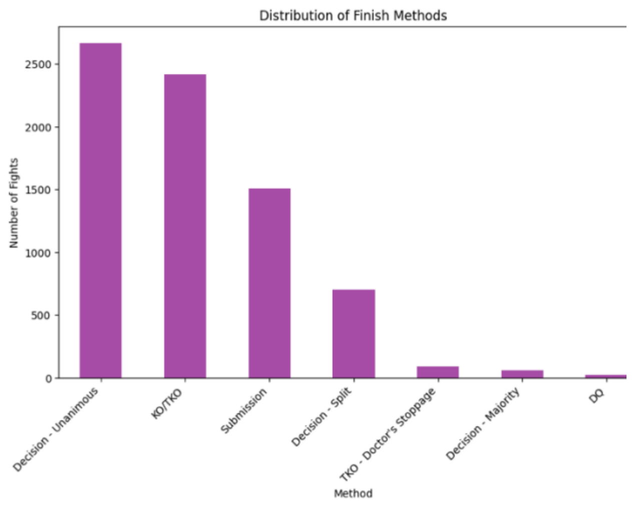
The four predicted variables were the fight's winner, the number of significant strikes landed by both fighters, the number of takedowns landed by both fighters, the time of the match, the round that the fight ended, and the method of victory for the winner. Our Winner Prediction used a binary classification to identify which corner the winner was from. Method Prediction used a multiclass classification to predict the victory method (KO, Submission, Decision, etc.). Fight Duration Prediction, Round Prediction, and predicting significant strikes and takedowns used individual regression tasks to predict the outcome

VII. Explanatory Analysis

Patterns from our dataset validated patterns in our predictions in a way that could explain specific patterns in our prediction values. We compared how many Red Fighters were winners versus how many were Blue Fighters.



The graph shows that the number of red fighters winners almost doubled. This implies that, according to the oddsmakers, the red fighter is usually the favorite to win. There should be a similar pattern when predicting the winners of upcoming fights: more red than blue fighters.



This bar chart visualizes the distribution of finish methods from the dataset with all the fight statistics. The unanimous decision was the most common way to fight, followed by KO/TKO and Submission. Our prediction is expected to follow this pattern.

A. Missing Values

The first challenge was several columns missing values, which could be attributed to those fight statistics requiring more thorough recording. The missing values were found in both numeric and categorical columns. To avoid biases in model predictions, we implemented two methods, Random Forest Regression for numeric columns and Random Forest Classifier[6] for categorical columns, to predict and impute the missing values. Missing numeric values were predicted using a Random Forest Regressor [7] . The model is well-suited for predicting continuous values because it can model complex relationship features. We split the data set into known and unknown data and added pd.get_dummies to apply to handle categorical variables. A random forest regression was trained on the known data, and the missing values were predicted in the target column. The categorical random forest classifiers were used to predict our categorical variables r_stance and b_stance. After our regression, the dataset will have complete values.

B. Upcoming Fight Prediction

We created a dataset with upcoming fights [4], with only two columns for each fight's blue and red fighters. Afterward, we concatenated this dataset with the cleaned data to make predictions for the upcoming fights. We filled the fighters' past statistics into the missing columns and handled missing fighter statistics using historical data. The variables we wanted to predict were kept empty.

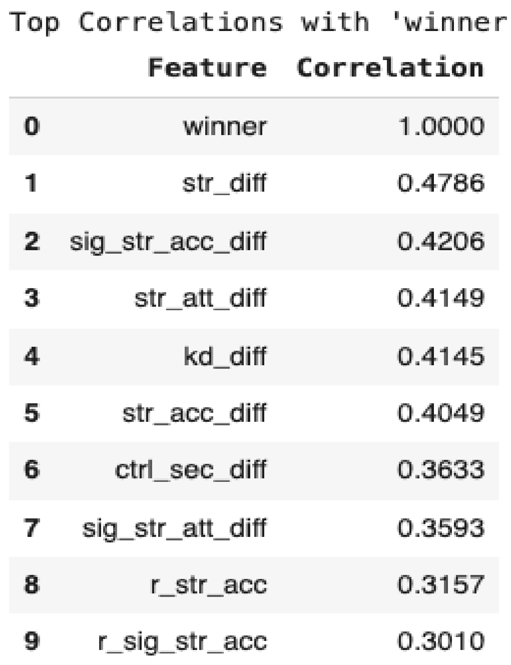
Experiments

I. Dataset description

Our dataset, "UFC Complete Dataset: All Events (1996-2024)," was created by MaksBasher on Kaggle using web-scraping from the UFC websites. This dataset comprehensively collects Ultimate Fighting Championship (UFC) events, fighters, and fight outcomes for the past 28 years. The dataset contains 95 columns and over 7000 rows for most of our data analysis. Besides some of the basic statistics of each fight, such as the event name, fighter names, and winner of each battle, the dataset also incorporates more identifying information about the fight, namely the weight class, gender, and title bout status of each battle. Similarly, the dataset has the basic demographics of each fighter, including each fighter's physical attributes, record, and stance, as well as advanced statistics, including control time, significant strike accuracy, and submission and takedown attempt averages. Most importantly, the dataset also includes data on the ending of each fight, like the finish round, method of victory, and time of fight. Our group's primary goal is to predict the statistics of the ending of a battle using the data provided in our dataset.

II. Evaluation Metrics
We first used a Random Forest Classifier and Regressor to apply regression techniques to eight target features: winner, method, finish round, time, red fighter’s significant strikes, red fighter’s takedowns, and blue fighter’s significant strikes and takedowns.

A. Correlation Of The Features
For our regression techniques to only use correctly associated variables with our target features, we had to use the corr() feature on each target feature to identify which features to apply Random Forest Classification and Regression on. Here is an example of the results:

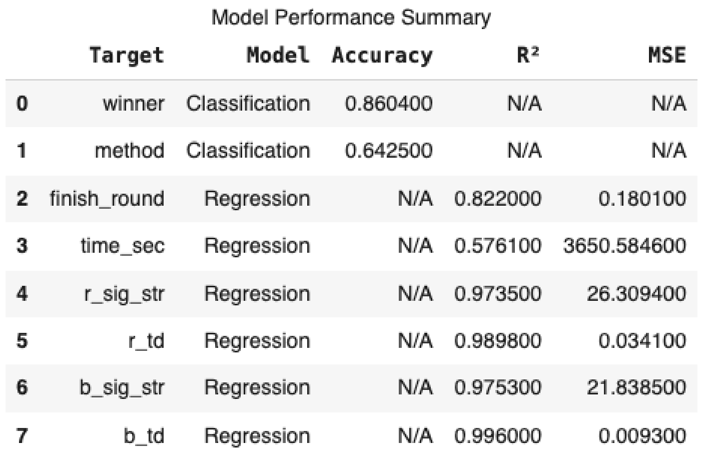

Top Correlations with 'winner':

	Feature	Correlation
0	winner	1.0000
1	str_diff	0.4786
2	sig_str_acc_diff	0.4206
3	str_att_diff	0.4149
4	kd_diff	0.4145
5	str_acc_diff	0.4049
6	ctrl_sec_diff	0.3633
7	sig_str_att_diff	0.3593
8	r_str_acc	0.3157
9	r_sig_str_acc	0.3010

We can train the model once the correct variables to use with the winner feature have been determined. We have individual evaluation metrics for each target feature based on the nine associated variables discovered through the corr() function.

B. Major Results
We could predict the statistics for 196 upcoming fights in a new dataset. Each feature used a Random Forest. For our winner feature, we used the variables 'sig_str_diff,' 'str_diff,' 'sig_str_acc_diff,' 'kd_diff,' 'str_acc_diff,' 'str_att_diff,' 'td_diff,' 'r_td,' and 'ctrl_sec_diff' for our regression model. The model achieved 84.80% accuracy, which indicates good performance in predicting the winner of a fight based on strike differences, significant strike differences, and knockdown differences. For our method of victory feature, we used the variables 'winner,' 'r_sub_avg,' 'sig_str_diff,' 'r_sub_att,' 'td_diff,' 'b_td,' 'r_sig_str_acc,' 'sub_att_diff,' and 'r_td' for our regression model. The model achieved 64.57% accuracy, which indicates moderate performance in predicting the method of victory based on the winner, submission averages, and significant strike differences. For our finish round feature, we used the variables 'r_str_att,' 'b_str_att,' 'b_str,' 'b_sig_str_att,' 'r_str,' 'r_sig_str_att,' 'b_td,' 'r_td,' and 'time_sec' for our regression model. The model performed well with an R^2 of 0.8118 and MSE of 0.1948, which indicates good performance and low variance in predicting the finish round based on the strike attempts, significant strike attempts, and takedown statistics. For our time feature, we used the variables 'finish_round,' 'b_td,' 'b_str_att,' 'b_str,' 'r_str_att,' 'b_sig_str_att,' 'r_td,' 'r_sig_str_att,' and 'r_str' for our regression model. The model performed poorly with an R^2 of 0.5734 and MSE of 3673.0353, which indicates moderate performance but extremely high variance in predicting the time of a fight based on the strike attempts, significant strike attempts, and takedown statistics. Seeking an additional model suited us for this feature. For our red fighter significant strikes feature, we used the variables 'finish_round,' 'r_str_att,' 'r_sig_str_att,' 'b_sig_str,' 'r_str,' 'b_sig_str_att,' 'b_str_att', 'r_td', and 'b_td' for our regression model. The model

performed very well with an R² of 0.9754 and MSE of 24.6589, which indicates excellent performance but moderate variance in predicting the red fighter's significant strikes based on the finish round, strike attempts, and significant strike attempts. For our red fighter takedowns feature, we used the variables 'td_diff,' 'r_td_att,' 'r_ctrl_sec,' 'ctrl_sec_diff,' 'r_td_acc,' 'td_att_diff,' 'finish_round,' 'r_td_avg,' and 'td_acc_diff' for our regression model. The model performed very well with an R² of 0.9905 and MSE of 0.0314, which indicates outstanding performance with very low variance in predicting the red fighter takedowns based on the takedown difference, control time, and finish round. For our blue fighter significant strikes feature, we used the variables 'finish_round,' 'b_str_att,' 'b_sig_str_att,' 'b_str,' 'b_td,' 'r_sig_str_att,' 'r_str_att,' 'r_sig_str,' and 'b_SLpM_total' for our regression model. The model performed slightly worse than the red fighter equivalent with an R² of 0.9715 and MSE of 26.7956, which indicates excellent performance but moderate variance in predicting the blue fighter's significant strikes based on the finish round, strike attempts, and significant strike attempts. For our blue fighter takedowns feature, we used the variables 'b_ctrl_sec,' 'b_td_att,' 'finish_round,' 'b_td_acc,' 'b_td_avg,' 'b_str,' 'time_sec,' 'b_sig_str,' and 'b_str_att' for our regression model. The model performed slightly better than the red fighter equivalent with an R² of 0.9960 and MSE of 0.0088, which indicates an excellent performance with very low variance in predicting the blue fighter takedowns based on the takedown difference, control time, and finish round.


Model Performance Summary

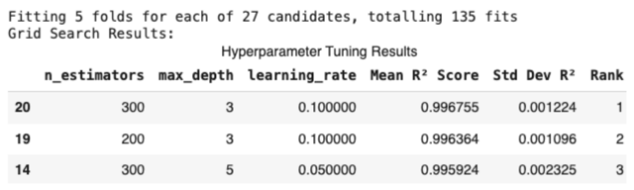
	Target	Model	Accuracy	R²	MSE
0	winner	Classification	0.860400	N/A	N/A
1	method	Classification	0.642500	N/A	N/A
2	finish_round	Regression	N/A	0.822000	0.180100
3	time_sec	Regression	N/A	0.576100	3650.584600
4	r_sig_str	Regression	N/A	0.973500	26.309400
5	r_td	Regression	N/A	0.989800	0.034100
6	b_sig_str	Regression	N/A	0.975300	21.838500
7	b_td	Regression	N/A	0.996000	0.009300

C. Inefficient Model

We eventually discovered that the time feature had poor performance on Random Forest, so we shifted to using an XGB Regressor for that feature. The XGBRegressor is a regression model from the XGBoost library that handles regression tasks using the gradient boosting framework.[5] Gradient Boosting is a build that ensembles decision trees iteratively to minimize prediction errors. It has customizable hyperparameters such as learning rate, n estimators, and max depth. These hyperparameters aim to control how the model learns and prevents overfitting. Since time_sec had poor performance in prediction, we will use XGB Regressor to fix the performance. First, we have to analyze the correct parameters.

D. XBG Regressor Parameters

The output table shows the performance of various hyperparameter combinations evaluated during the tuning process for the XGBoost model. Our three key hyperparameters used are **n_estimators** (number of boosting rounds: [100, 200, 300]), **max_depth** (maximum tree depth: [3, 5, 7]), and **learning_rate** (step size: [0.01, 0.05, 0.1]). These hyperparameters control the number of trees, model complexity, and learning pace to optimize performance and generalization.

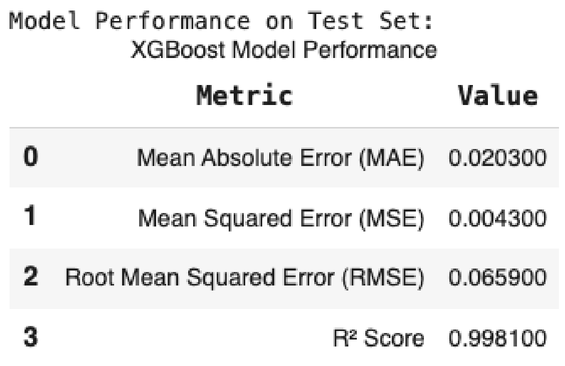


```
Fitting 5 folds for each of 27 candidates, totalling 135 fits
Grid Search Results:
```

Hyperparameter Tuning Results

	n_estimators	max_depth	learning_rate	Mean R² Score	Std Dev R²	Rank
20	300	3	0.100000	0.996755	0.001224	1
19	200	3	0.100000	0.996364	0.001096	2
14	300	5	0.050000	0.995924	0.002325	3

Our best hyperparameters were n_estimators of 300, max depth of 3, and learning rate of 0.1, which calculated a mean R-squared of 0.9953 and a standard deviation R-squared of 0.0045. Now, use those parameters in the actual XGB Regressor.



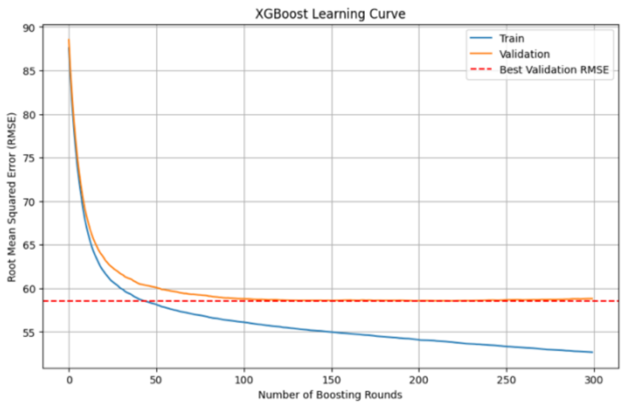
```
Model Performance on Test Set:
```

XGBoost Model Performance

	Metric	Value
0	Mean Absolute Error (MAE)	0.020300
1	Mean Squared Error (MSE)	0.004300
2	Root Mean Squared Error (RMSE)	0.065900
3	R² Score	0.998100

The results of the XGBoost Model performance were the mean absolute error(MAE) of 0.0179, indicating that the model's predictions are very close to the actual values. The mean-squared error(MSE) was 0.0037, which implies that the model suggests that it doesn't make significant prediction errors. The root mean squared error(RSME) was 0.0606, which is a small RSME that explains that the model has high prediction accuracy and low variance in model errors. R-squared was 0.9983, which indicates that the model explains most of the variability. This model is a better model for predicting a better time_sec. Additionally, we can plot the learning curve to prove these findings, representing how the model's error changes throughout the increased number of iterations.

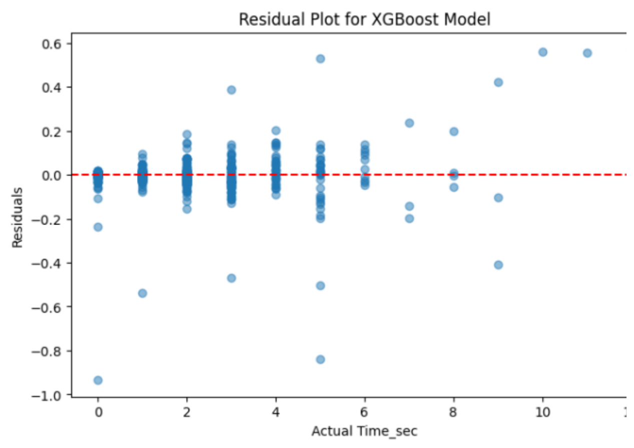
E. Learning Curve for the XGB Regressor



The XGBoost Learning Curve (visualized above) visualizes the model's performance during training and validation over 300 boosting rounds. The training curve (blue) consistently decreases Root Mean Squared Error (RMSE), indicating that the model is learning effectively. Meanwhile the validation curve (orange) initially decreases but flattens around boosting around 100, suggesting diminishing returns and a balance between underfitting and overfitting. The red dashed line marks the lowest RMSE on the validation set, representing the model's best performance. Our learning curve highlights the importance of carefully tuning hyperparameters such as n_estimators and stopping training early to avoid overfitting while maintaining generalization on unseen data.

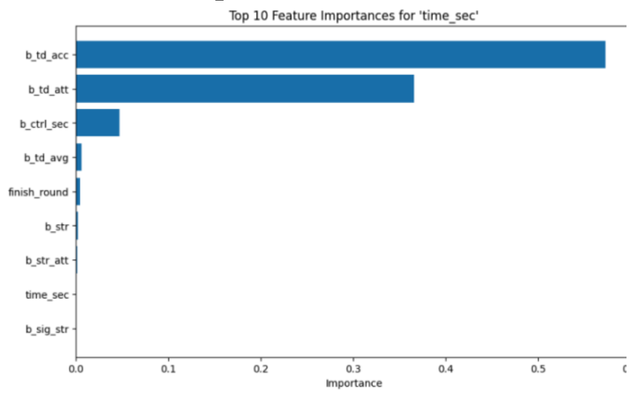
F. XGB Residual Plot

Using our residual plot, we can visually represent the difference between the predicted and actual values for the XGB model.



The residual plot appears scattered around the red line, which is a good sign. There's no pattern, which implies that the model is unbiased. Although there are a few outliers, they can be explained by fights that ended early, which most of the audience didn't expect.

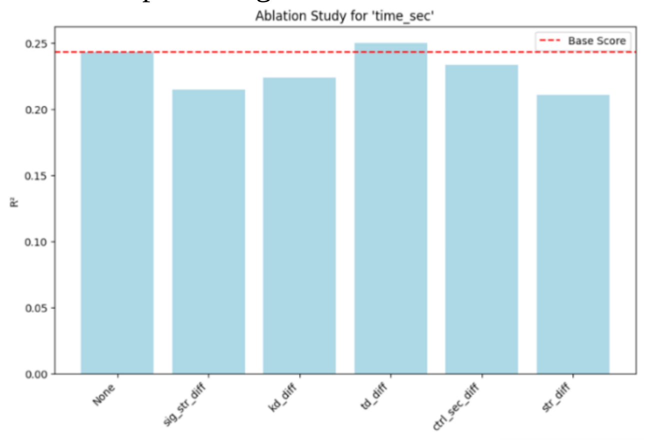
G. XGB Important Features



The main features for predicting time mostly rely on three features, less than the original model.

I. Ablation studies

The alation study examined how removing specific features correlates to the model's performance. We focused on the features that represented the difference in comparing certain aspects of fighting. All features that ended in _diff were td_diff, ctrl_sec_diff, kd_diff, sig_str_diff and str_diff. The base model performance was the baseline R-Squared of 0.2431. Removing those features had an insignificant impact, while removing td_diff resulted in the lowest performance drip with an R-squared of 0.2500. A slightly higher R-squared value means that td_diff may not contribute much to the model's predictions and could potentially worsen it. Removing ctrl_sec_diff and kd_diff, their removal reduced our R-squared to 0.2336 and 0.2236, showing a moderate decrease in performance. The removal of sig_str_diff and str_diff had the most significant impact on the model, dropping the R-squared to 0.2135 and 0.2104. These two features are crucial for predicting the time of a match.



II. Conclusion

After using all these models and predicting the values for the significant statistics for each upcoming fight, using machine learning algorithms such as Random Forest and XGBoost, and properly selecting models, we have saved them as a final_combined_prediction_csv. Overall, this project showed us the power of machine learning in sports analytics. This project had major success in predicting almost all features that were targeted in our model.

REFERENCES

[1] "History of MMA and the UFC." *Verdictmma.com*, verdictmma.com/guides/history-of-mixed-martial-arts-and-the-ufc.

[2] "What Is PrizePicks?" *Www.prizepicks.com*, www.prizepicks.com/resources/what-is-prizepicks-how-to-play-promo-code.

[3] "UFC Complete Dataset (All Events 1996-2024)." *Www.kaggle.com*, www.kaggle.com/datasets/maksbasher/ufc-complete-dataset-all-events-1996-2024.

[4] Carmona, Jared. "Upcomingdatasetproject." *Kaggle.com*, 2024, www.kaggle.com/datasets/jaredcarmona/upcomingdatasetproject. Accessed 18 Dec. 2024.

[5] "Python API Reference — Xgboost 1.3.0-SNAPSHOT Documentation." *Xgboost.readthedocs.io*, xgboost.readthedocs.io/en/latest/python/python_api.html.

[6] "RandomForestClassifier." *Scikit-Learn*, 2024, scikit-learn.org/dev/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[7] scikit-learn. "3.2.4.3.2. Sklearn.ensemble.RandomForestRegressor — Scikit-Learn 0.20.3 Documentation." *Scikit-Learn.org*, 2018, scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html.

[8] "Sport Analytics - Employment Guide for the Sports Industry." *University of New Haven*, www.newhaven.edu/business/sports-industry-employment-guide/analytics.php.

[9] "LabelEncoder." *Scikit-Learn*, 2024, scikit-learn.org/dev/modules/generated/sklearn.preprocessing.LabelEncoder.html.