Machine Learning Model Evaluation for UFC Fight Outcomes & Healthcare Diagnostics

By: Kanwarpartap Singh Brar

I. ABSTRACT

In this paper, we will explore the performance of 8 machine learning models on a binary classification and multiclass classification dataset. The binary classification dataset is UFC fight data, and the multiclass dataset is healthcare prediction data. The aim is to evaluate how different ML algorithms can predict the winners of UFC fights and correctly predict healthcare test results. The implemented algorithms are Decision Tree, Random Forest Classifier, Support Vector Machines (SVM), and Logistic Regression. Perceptron and K-Nearest Neighbors (KNN). Model performance was evaluated using accuracy scores, classification reports, confusion matrices, and hyperparameter tuning results to determine the most effective models per context. The findings from this evaluation will provide valuable insight into how ML models can be applied in both competitive sports forecasting and healthcare diagnostics, which demonstrates how algorithm selection can have an impact on predictive success in different real-world scenarios.

II. INTRODUCTION

In machine learning, there are two broad categories of classification problems: multiclass and binary classification. Binary classification predicts one of two possible classes (labels). In this report, our UFC dataset is binary classified, and we are predicting whether a fighter wins or loses.

The multiclass healthcare dataset predicts on three distinct classes (normal, inconclusive, or abnormal). In this report, we specifically implement the following machine learning algorithms on the respective binary or multiclass classification datasets.

Binary Classification Data (UFC Data):
- **Decision Tree Model**
- **Logistic Regression**
- **Support Vector Machines (SVM)**
- **Random Forest (Ensemble)**

Multiclass Classification Data (Healthcare):
- **K-Nearest Neighbors (KNN)**
- **Support Vector Machines (SVM)**
- **Random Forest (Ensemble)**
- **Perceptron**

Decision Tree classifier can be used for binary classification, multiclass classification, ranking, and regression. It uses information gain to select the best feature value to split the data. It is a form of supervised learning where each leaf node is a class label and each branch is a result of the split. The algorithm continuously splits recursively until the maximum depth value is met.
Logistic Regression is a binary classification model that uses the sigmoid function to map real values to probabilities. (ex. What is the probability that the point matches a specific class?) It finds a linear hyperplane that can best separate the classes, and it predicts

values between 0-1 to calculate the probability that an input belongs to a certain class.
Support Vector Machines, also known as SVMs, in a similar way to logistic regression, also find a hyperplane that separates different classes of data. Unlike logistic regression, however, SVMs also focus on staying as far away as possible from the closest points to the hyperplane on both sides (these are the support vectors). The goal of an SVM is to maximize the margin, which is the distance between the hyperplane and the nearest point from each class.
Random Forest is an ensemble learning method that uses multiple decision trees and combines their outputs. This is different from a singular decision tree because the combination of more trees increases the prediction accuracy and reduces the chances of overfitting.
K-Nearest Neighbors (KNN) is a classification algorithm that looks at the 'k' nearest points (neighbors) and assigns the majority class. It measures the distance to its neighbors using metrics such as Euclidean or Manhattan distance.
Perceptron using One-vs-Rest turns a binary classification problem into a multiclass classification problem. Each classifier learns to distinguish classes from one another, and the class with the highest confidence score is selected as the final prediction.
In the upcoming sections, we will go further into where these datasets come from and the results/performance of the mentioned machine learning algorithms on these datasets.

III. DATASETS

<u>**DATASET 1: UFC FIGHT DATA**</u>

This dataset is centered around UFC (Ultimate Fighting Championship) fight statistics. The UFC is an MMA (Mixed Martial Arts) fight promotion, which is one of the biggest fighting brands of all time. This dataset covers every UFC fight from July 2016 to November 2024, so approximately 8 years of fights are recorded in this dataset. The statistics are based on each round and have data on each fighter. The dataset was not split into testing and training data, so I used the train_test_split function from sklearn.model_selection to create a test dataset with a size of 20%. The rest of the EDA has been done on the training data. There are 3,333 examples and 194 features, with three different datatypes. There are 146 features with the datatype float64, 37 int64, and 11 objects. This dataset I will be using as my binary classification dataset, where the label will be 'Winner?' (tells us the fight winner). 1 = fighter 1 won and 0 = fighter 2 won.

Data Processing
To be able to accurately create a model where I could input any two fighters and predict who would win, I needed to adjust this dataset and take out each fighter's data from all the fights that were recorded in the UFC fight data. To prepare this dataset for modeling and obtain the information I needed, I engineered a new dataset called fighter_statistics.
I extracted fight data for every feature, leaving out irrelevant features such as

location, and placed that data into the new dataset. Each row in this dataset corresponds to a fighter, and the values represent their career averages for the feature categories across all their past fights. This is ideal because it will allow the machine learning algorithms to learn comparative performance between the fighters, allowing for a more accurate prediction. An issue I ran into with this dataset was that most UFC fights are 3 rounds, however, championship fights are 5 rounds. Alongside that, just because a fight is 3 or 5 rounds, it does not mean it is going to last 3 or 5 rounds. This results in there being many null values for rounds that never occurred. The most appropriate approach was to turn all the null values into 0. Technically the fighter has never fought in those rounds, so their career stats are 0. Filling the missing values with the column average would have introduced misleading data, inflating their statistics and potentially biasing the model to incorrectly favor them as winners.

Feature Vector

To be able to use this dataset, once fighter names are entered, a feature vector is created that subtracts the 2 fighters' statistics for each feature. This results in a row of numeric values showing how Fighter1 compares to Fighter2 on various feature metrics. This feature vector is X, and the 'Winner?' column is the label y.

DATASET 2: HEALTHCARE DATA

This dataset is my multiclass classification dataset centered around healthcare data. The dataset was not split into training and testing data, so similar to what I did for the UFC dataset, I did the same thing for this and conducted the EDA on the training data. This dataset contains 44,400 examples and 15 features. The label for this dataset will be 'Test Results'. 'Test Results' has three different classifications: abnormal, normal, and inconclusive. The features have 4 different data types. Datetime64 (2), float64(1), int64(2), and object(10). None of the features have any null values.

Data Processing

The first step I took in the preprocessing for this dataset was to remove any irrelevant features. 'Doctor', 'Hospital', 'Insurance Provider', and 'Room Number' were irrelevant features that do not help in predicting the test results. The second step was to feature engineer and create a new feature called LengthOfStay. 'LengthOfStay' = 'Dishcharge Date' - 'Date of Admission'. This new feature captures how long a patient was hospitalized, which can help correlate with the severity of their condition, in turn helping aid with the prediction of their test result.
The third step was to apply One-Hot Encoding to all the categorical features. This transformation allows for compatibility with the machine learning algorithms mentioned above because they require a numerical input. Alongside One-Hot Encoding, Label Encoding was also used to treat the label as a multiclass classification task.

Lastly, I used StandardScaler to scale the features to have a mean of 0 and a standard deviation of 1, which helps the models converge more efficiently and calculate distances for models like KNN.

IV. HYPOTHESES ON ALGORITHM PERFORMANCES

Below are my hypotheses on how the different algorithms will perform on their respective datasets. This section also explains why each model was chosen.

UFC FIGHT DATA:

Decision Tree: This model was chosen because decision trees are well-suited to datasets with both numerical and categorical data, which this dataset has. Decision Tree models are also known for handling non-linear decision boundaries well. I expect this model to perform relatively well due to their ability to handle complex feature interactions effectively.
Logistic Regression: This model was chosen because it is a well-known binary classification model that can use linear decision boundaries with probabilities. I expect this model to perform moderately because it may be prone to overfitting and not be able to handle the complex features as well as the decision tree model.
SVM: SVMs work well in high dimensions and on datasets with a large number of features due to the use of kernels. I expect this model to perform the best due to its ability to capture complex patterns effectively.
Random Forest: This model was chosen due to its ability to combine multiple decision trees, increasing robustness and decreasing overfitting. I expect this model to perform better than a single decision tree.

HEALTHCARE DATA:

KNN: This model was chosen because of its ability to be effective when classes are separated. I expect this model to perform the weakest due to it being affected by noisy features, which this dataset has plenty of.
SVM: This model was chosen for its ability to work well in high dimensions. Once again, similar to the UFC dataset, I expect this model to perform very well due to its ability to handle complex features.
Random Forest: This model was chosen because of its ability to handle both categorical and numerical features. I expect this model to perform consistently alongside the SVM model due to its tree ensemble averaging property.
Perceptron: This model was chosen as a linear model that can be turned into a multiclass classification model using One-Vs-Rest. I expect this model to perform the worst, but it will be interesting to see how its linear nature can handle the non-linear data.

Overall Hypothesis
Overall, I expected the SVM and Random Forest models to perform best in both datasets. I expect Logistic Regression, Perceptron, & KNN to struggle with the complexity of the datasets.

V. ANALYSIS & RESULTS

DATASET 1: UFC FIGHT DATA

Figure 1 presents the optimal test accuracy, precision, recall, and F1-score for each machine learning model applied to the UFC fight prediction dataset.

Model	Accuracy	Precision	Recall	F-1 Score
Decision Tree	0.634	0.63	0.61	0.6
Logistic Regression	0.673	0.67	0.65	0.65
SVM	0.668	0.66	0.65	0.65
RF Ensemble	0.673	0.67	0.65	0.65

Figure 1: UFC Dataset - Optimal Test Accuracy, Precision, Recall, & F-1 Score

Figure 2 illustrates the hyperparameter tuning results for the four machine learning models evaluated on the UFC dataset.

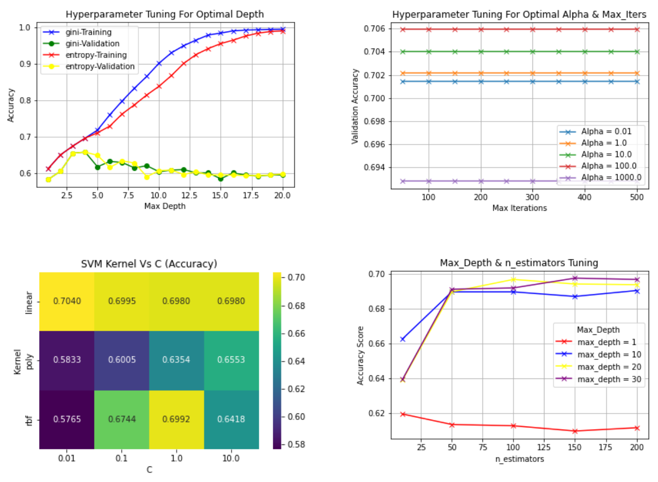

Figure 2: UFC Fight Data Hyperparameter Tuning Plots

Figure 3 displays the optimal hyperparameter values identified for the four machine learning models applied to the UFC dataset.

Model	Optimal Hyperparam 1	Optimal Hyperparam 2
Decision Tree	'Max Depth' : 4	'Criterion': Gini
Logistic Regression	'C': .01	'Max Iter': 50
SVM	'C': .01	'Kernel': Linear
RF Ensemble	'Max Depth' : 30	'n_estimators': 150

Figure 3: UFC Dataset - Optimal Hyperparameters

Figure 4 illustrates the confusion matrices generated by the four machine learning models evaluated on the UFC dataset.

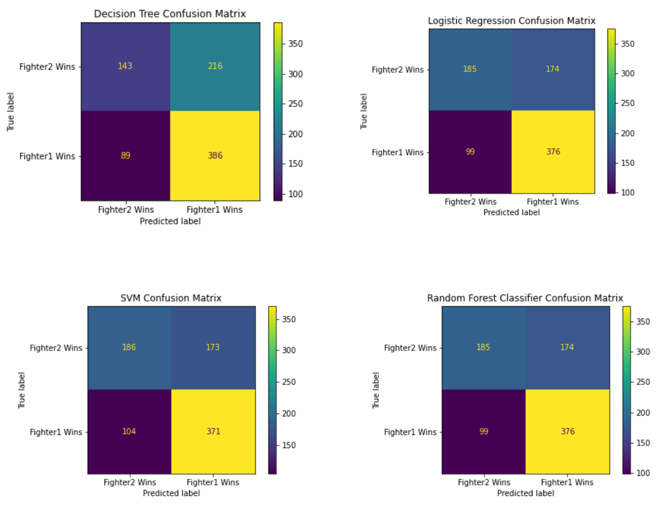

FIgure 4: UFC Fight Data Confustion Matricies

Decision Tree Analysis

From the given results in Figures 1-4, we see that the decision tree model has a moderate performance with an optimal test accuracy of 63.4%. This tells us that the model predicts the correct fight winner approximately 63% of the time on average, which is better than a 50-50 guess when it comes to a binary classification task. Based on the hyperparameter tuning plots, the optimal max_depth was 4 and the optimal criterion was Gini. When looking at the tuning plots, we see that as max_depth increased, this led to overfitting, with the training accuracy increasing to nearly 100% while the validation accuracy declined. The Gini criterion increases and outperforms the entropy criterion across majority depths, hence why it is the optimal criterion. The precision (**.63**), recall **(.61**), and F1-score **(.60)** tells us that the model struggles with the imbalance of all the actual wins by a fighter and how many the model correctly identifies. The confusion matrix backs this claim and shows that the performance for the Fighter1 class is dominating. Over 60% of Fighter2 wins are misclassified, showing that it is more skewed towards predicting Fighter1 as the winner. This skew likely comes from the fact the dataset has more examples of Fighter1 wins, which influences the trees splits. This imbalance is understandable given that Fighter1 typically is represented as the 'favorite' in a fight and Fighter2 is represented as the 'underdog'. As a result, the dataset naturally contains more examples of Fighter1 wins, which contributes to the model's bias towards predicting Fighter1 as the winner.

Logistic Regression Analysis

From the given results in Figures 1-4, we see that the logistic regression model is tied for the best performance, achieving an optimal test accuracy score of 67.3%, which, for a binary classification task, is much better than a 50-50 guess. The model was trained and validated using 5-fold cross-validation, which tells us that the model's performance did not depend on only the training data, and it provides a more reliable generalization to unseen data. Based on the macro-averaged metrics in Figure 1 and the confusion matrix in Figure 4, we see that even though the model is slightly biased towards Fighter1 again, it is an improvement from the decision tree model when it comes to handling the class imbalance. True positives and false positives for Fighter2 were 185 and 174, respectively. True positives and false positives for Fighter1 were 376 and 99, respectively. This confusion matrix also indicates that the model is more confident at predicting Fighter1 as the winner, which is not surprising because Fighter1 is the favorite in the way the feature was set up.

Looking at Figures 2 and 3, the optimal hyperparameters selected were C = .01 and max iterations = 50. The small C value suggests the model prefers a higher regularization, leading to smaller weight values because larger weight values in the model could lead to overfitting. This is good because it shows this model is less likely to memorize and fit onto noise in the training data, and instead generalizes better to unseen patterns. When it comes to UFC data, there can be a lot of irrelevant features impacting the label and generating noise, and a stronger regularization with a smaller C value can limit the impact of the noise and make the model more robust. After 50 iterations, we see that the model doesn't improve much on validation performance and that it converges fairly quickly.

SVM Analysis
From the given results in Figures 1-4, this model shows consistent performance with an optimal test accuracy score of 66.8%. As shown in Figure 1, the macro-averaged precision, recall, and F1-scores are .66, .65, and .65, respectively. This shows there is a somewhat balanced performance across the classes, which aligns with our hypothesis that SVMs are good at generalizing in high dimensional datasets.
The confusion matrix in Figure 4 shows that Fighter2 was correctly predicted 186 times and misclassified 173 times, whereas Fighter1 was correctly predicted 371 times and mislcassified 104 times. Once again showings the class imbalance and the favorement towards Fighter1.
In Figure 2, instead of a hyperparameter tuning plot, the hyperparameter heat map shows that the best hyperparameters are C = .01 and the kernel = linear. The smaller C is similar to the logistic regression model, where a stronger regularization is preferred due to there being noisy features because of the unpredictability of a UFC fight. The heatmap also shows the linear kernel outperforming both the RBF and polynomial kernels. Similar to the logistic regression model, the hyperparameter tuning was performed using 5-fold cross-validation to increase model robustness and reduce the risk of overfitting to a single training/test split.

Random Forest Analysis
From the given results in Figures 1-4, this model shows great performance tied with the Logistic Regression model with an optimal test accuracy of 67.3%. Figure 2's hyperparameter tuning plot shows the interaction between the number of estimators and the maximum depth of the trees. The optimal hyperparameters were 150 trees and a maximum depth of 30. From the decision tree stump to max depth of 10+, we can see that the performance increases drastically as expected. Compared to the decision tree model, random forest aggregates multiple trees, which is likely why it gave a better accuracy score on the test set.
Figure 4's confusion matrix shows 376 true positives and 99 false positives for Fighter1, and 185 true positives and 174 false negatives for Fighter2. This is very similar to the logistic regression and SVM models, as we can tell by looking at Figure 1.
This model performed similarly to the logistic regression model, and it shows how the ensemble nature of the Random Forest Classifier is resistant to overfitting, which is key in an unpredictable dataset like UFC fight data.

Model Comparison Summary
Across the 4 models that were evaluated on the UFC dataset, Logistic Regression & Random Forest Ensemble Classifier were the top performing algorithms with an optimal test accuracy of 67.3% and macro average precision, recall, and F1-scores of

.67, .65, and .65, respectively. Second was the SVM model with 66.8% accuracy, and the worst performer was the decision tree model with 63.4% accuracy. In conclusion, the Random Forest Classifier would stand out as the best performer due to its ensemble robustness and resistance to overfitting.

DATASET 2: HEALTHCARE DATA

Figure 5 presents the optimal test accuracy, precision, recall, and F1-score for each machine learning model applied to the Healthcare dataset.

Model	Accuracy	Precision	Recall	F-1 Score
KNN	0.445	0.44	0.44	0.44
SVM	0.365	0.36	0.36	0.36
Perceptron	0.337	0.34	0.34	0.34
RF Ensemble	0.438	0.44	0.44	0.44

Figure 6: Healthcare Dataset - Optimal Test Accuracy, Precision, Recall, & F-1 Score

Figure 6 illustrates the hyperparameter tuning results for the four machine learning models evaluated on the Healthcare dataset.

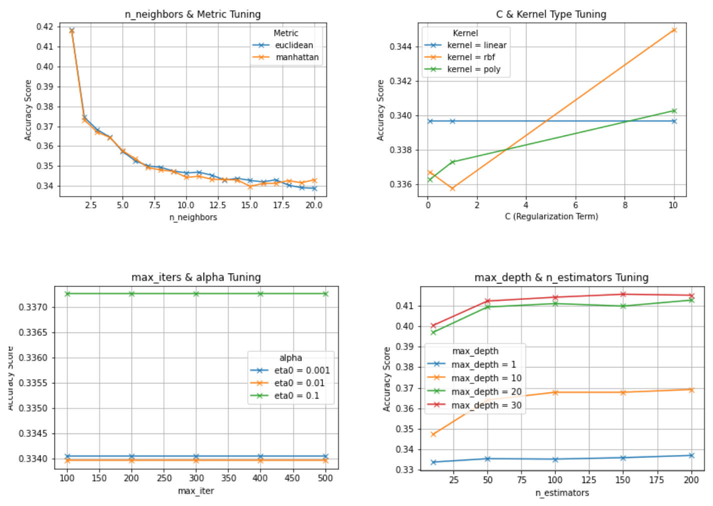

Figure 7: Healthcare Data Hyperparameter Tuning Plots

Figure 7 displays the optimal hyperparameter values identified for the four machine learning models applied to the Healthcare dataset.

Model	Optimal Hyperparam 1	Optimal Hyperparam 2
KNN	'Metric': Euclidean	'n_neighbors': 1
SVM	'C': 10	'Kernel': rbf
Perceptron	'alpha': .1	'max_iter': 100
RF Ensemble	'max_depth': 30	'n_estimators': 150

Figure 8: Healthcare Data - Optimal Hyperparameters

Figure 8 illustrates the confusion matrices generated by the four machine learning models evaluated on the Healthcare dataset.

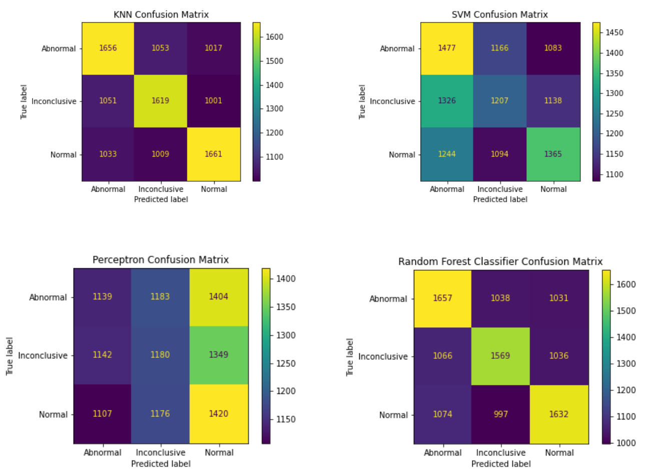

FIgure 9: Healthcare Data Confustion Matricies

K-Nearest Neighbors Analysis

From the given results in Figures 5-8, we see the KNN model shows the strongest performance, achieving an accuracy score of 44.5%. The optimal hyperparameters were n_neighbors = 1 and measuring distance using the Euclidean method. We can see in Figure 6 that as the number of neighbors the model looked at increased, model accuracy consistently declined. This is a sign that represents increasing bias in the model. The confusing matrix in Figure 8 reveals that the predictions on each class were relatively balanced, however, a large number of samples across all classes were misclassified. The reason for this could be due to the model having difficulty understanding and generalizing to the differences in each class. Overall, this model performed the best out of the 4, however, it was not a strong performance, and that can be chalked up to the dataset not having enough relevant features.

SVM Analysis

From the given results in Figures 5-8, we can see that the SVM model underperformed relative to the other models, achieving an accuracy of 36.5%, with precision, recall, and F-1 scores coming in at 36%. The optimal hyperparameters were C=10 and kernel = ‘rbf’. The reason ’rbf’ was chosen as the optimal kernel, even though the polynomial achieved higher accuracies at a higher C value, on the 5-fold cross validation, the rbf kernel had more stable performance across the different regularization strengths. Similar to the KNN model, the confusion matrix shows there was high misclassification in all of the classes. This implies the SVM model failed to find effective decision boundaries between classes, even after using the non-linear RBF kernel. The SVM model was hypothesized to work very well, however, it failed to generalize well, possibly due to the noisy features hindering the model's ability to establish an appropriate hyperplane.

Perceptron Analysis

From the given results in Figures 5-8, we can see that the Perceptron model achieved the weakest performance with an accuracy of only 33.7% and precision, recall, and F-1 scores of 34%. The model struggled to capture any meaningful pattern in the data, and the optimal hyperparameters obtained from tuning were alpha = .1 and max_iter = 100. An alpha of .1 in this model was relatively high, which indicates more regularization, making the model simpler and leading to potential underfitting. Possibly that is what occurred, and that is why the performance on this model was the worst. Having undefitting with a lower alpha may indicate the model was struggling to fit this complex dataset due to too much noise. A max_iterations of 100 indicates that beyond 100, the model plateaued and was not able to improve accuracy, possibly due to struggling to learn because of limitations with linear separability. Once again, the confusion matrix indicates there was a large amount of misclassification amongst all classes, which limited the model's ability to draw any effective decision boundaries between non-linearly separable groups.

Random Forest Analysis

From the given results in Figures 5-8, we see that the Random Forest model achieved a relatively strong accuracy score of 43.8%, outperforming
g both the SVM and Perceptron models. This can be attributed to the fact that Random Forest is good at reducing overfitting and is effective in noisy, high-dimensional datasets like this one. The optimal hyperparameters of max_depth = 30 and n_estimators = 150 tell us that the performance of the model improved consistently up to 150 trees before plateauing, as can be seen in Figure 6. The tuning plot for this model shows that a max_depth = 30 yields the best results, suggesting a deeper tree structure was necessary to capture the data, but not so deep that it would cause overfitting. The confusion matrix showed that the random forest model has notably lower misclassification than the SVM and Perceptron models, indicating a more reliable performance.

Model Comparison Summary

Across the 4 models evaluated on the healthcare dataset, KNN and Random Forest were the top performers. Both models achieved comparable accuracy scores and macro-average precision, recall, and F-1 scores. Even though all models did not have great performance, possibly due to an inconclusive dataset with weak features and too much noise, both KNN and Random Forest performed similarly in terms of the metric. Overall, Random Forest would be the more reliable choice due to both its ability to relatively effectively capture the complex patterns in the data, and for having a greater robustness to noise and overfitting.

VI. CONCLUSION

With more time and resources, several improvements can be made to enhance these models. For the UFC dataset, one key improvement I would make is to incorporate a fighter's most recent performances with greater weight, allowing the model to better reflect current form and momentum rather than treating all past fights equally. Additionally, integrating a real-time API to constantly scrape and update the dataset with new fight outcomes would ensure the model remains relevant and up to date. As for the healthcare dataset, a major limitation was the lack of relevant features. More extensive feature engineering is needed, as the current dataset does not contain enough informative features to make accurate predictions on the labels.

In conclusion, this project demonstrates how the effectiveness of machine learning algorithms can vary depending on the quality of the dataset they are applied. In the binary classification data using UFC fight data, Logistic Regression and Random Forest emerged as the top-performing models, achieving an optimal test accuracy of 67.3% and showing strong generalization through cross-validation and balanced class performances. These models did a good job of handling the high-dimensional data. In contrast, the multiclass classification data using the healthcare data yielded lower performance overall, with KNN and Random Forest being the most effective, still with low overall performance. The healthcare data results show that even ML models cannot compensate for noisy or weak features. Overall, the results of this project showcase how important it is to conduct data preprocessing, relevant feature engineering, and algorithm selection to build a reliable machine learning model for both sports analytics and healthcare diagnostics.

VII. REFERENCES

Magnus, Alex. *UFC Fight Statistics July 2016 - Nov 2024*. Kaggle, 2024, https://www.kaggle.com/datasets/alexmagnus24/ufc-fight-statistics-july-2016-nov-2024.

Pedregosa, Fabian, et al. *"1.4. Support Vector Machines."* Scikit-learn: Machine Learning in Python, scikit-learn.org, https://scikit-learn.org/stable/modules/svm.html.