DSA4211 Group 13 Project Assignment
Predicting the winners of UFC fights

Ragnhild Laursen & Jeppe Sjørup & Tay Zhi Yao Joel & SONG Xiaohan

October 2019

Abstract

In this report, we built models mainly with two methods, one was logistic regression and the other one was Neural Network, to predict the winner in 1078 fighting competitions from Ultimate Fighting Championship(UFC) with 159 feature data for each fighting prediction.

Before the analysis, we split the data into training and testing set, and the train-test split was 7:3. For feature selection, we only kept the non-constant features and removed those features which were highly correlated, so the number of features that joint into the analysis was reduced from 159 to 135.

For Logistic Regression, we firstly did ordinary logistic regression without regularization, the model accuracy on the test set was 68.1%. Then we added L1 regularization into the model and applied cross-validation to choose the best λ. The model accuracy was therefore increased slightly to 68.6%, and the selected λ was 0.017. Furthermore, since LASSO regression was used, many predictors' coefficients were shrunk to 0, and the dimension of the features was therefore decreased to 26.

For Neural Network, we attempt to implement a feed-forward neural network on the 135 features and output the probabilities for each observation being 1 of the K classes (Here $K = 2$). We configured our model with 5 layers in all, one input layer, one output layer and three hidden layers, where we used ELU, Sigmoid and Tanh function as activation functions respectively. For model training, we used a validation split of 20%, which means we use 80% of the training data for model training and the else for validation. Furthermore, we use Adam optimizer and categorical loss entropy as our loss function and applied an early stopping callback function with the patience of 5 epochs to prevent overfitting. Lastly, we set our epoch number as 40, and batch size as 128. After implementing the Neural Network model on the testing set, we finally got an accuracy of 68.9%.

In the end, we discussed the respective pros and cons of these two models. From the perspective of prediction accuracy, the Neural Network looks better, however, sometimes, it might be justifiable to forgo the benefits of accuracy that the neural network has to offer in return for the interpretability of the logistic regression model depending on the use case.

Key words: *Logistic Regression* *LASSO* *Neural Network* *Early Stopping*

1 Introduction

In this report we look at predicting winners of fights from Ultimate Fighting Championship (UFC) based on data available prior to the start of the fight. In our investigation we use data hosted on `kaggle.com` and compiled by Rajeev Warrier [1]. The data contains information about 3592 UFC fights conducted from 1993 to 2019.

UFC is an professional American mixed martial arts organization that was founded in 1993. They host events throughout the world, where fighters from different combat sports compete in one versus one fights.[2] The fights are split up in different weight classes from straw weight (105-115 lbs) to super heavyweight (265+ lbs), with each weight class having a reigning champion. Each fight has a pre-determined maximum number of rounds, which is normally 3 or 5 with a duration of 5 minutes per round. The fights can end prematurely either by a knockout (KO), submission, or a technical knockout (TKO), in which case the fighter that gets knocked out would be declared the loser and the other fighter would be declared the winner. If no stoppage occurs the winner is selected by judges using a points system after the fight has ended.[3] The two fighters are distinguished by the color of their shorts. Each fight has red fighter and a blue fighter. Traditionally the fighter given the red shorts is considered the favorite before the fight, therefore historically the win percentage of the red fighter is 66.26% as dervived from the dataset.

2 Dataset

2.1 Data Structure

The data consist of 159 features containing various statistics known before the fight starts. The data has been processed so that each feature only contains information known before the fight takes place. Out of the 159 features, there is 142 features describing the two fighters, i.e. 71 features for each fighter. This contains their fighting stance, height, weight, age, matches won, average strikes and take downs among many other descriptive statistics obtained from previous fights. The remaining 17 features describes the format of the fight, including the weight class, gender and the maximum number of rounds. All of the feature are quantitative as they are either dummy, discrete or continuous variables. There is only one feature, which is a factor variable of two factors indicating whether the fight was a title or not, which is changed to a dummy variable.

The response is also a factor variable indicating whether the *Blue* or *Red* fighter won. The response is also changed into a dummy variable, where 1 indicates red fighter and 0 indicates the blue fighter.

2.2 Data Processing

Before analysing the data, first, we randomly assign 70% of the observations to a training set and treat the rest as our test set, which we will use to validate our models. To keep our test set independent from our training set, we only process our data using estimates based on the training set.

Afterwards, we remove the constant columns and highly correlated columns, i.e. a correlation larger than 0.95, only based on the columns from the training set. The same corresponding columns are removed from the test set. The processing above removed 23 features from our data, which leaves us with 135 features in our data.

Last, we scale the data based on the scales found from the training data. This includes finding the mean and standard error of each feature in the training data and then scaling each observation in the data using these parameters.

When referring to the data in the analysis below, we are always working with processed data i.e. correlated and constant features removed and the data is scaled. This also includes the split between training and test set, which remain the same through the report.

3 Methods and Results

The goal of our analysis is twofold. Primarily we want predict the winner of fights. This implies that we should perform binary classification of our data. The baseline of this classification is predicting the favorite (red) as the winner. Secondly we want to identify important features related to winning fights.

In our investigation of the data we have chosen logistic regression and neural network. For logistic regression we first fit a model based on all the features. Afterwards we try to fit a new logistic regression model, where we regularize our parameters using lasso. As we use lasso, the remaining coefficients also represent the most important features. The logistic regression has a linear decision boundary, so afterwards we try to predict neural network which has a highly non-linear decision boundary to see if this gives a better prediction.

3.1 Logistic Regression

Logistic regression is a soft classifier that models the probabilities of a given data point belonging to each of K classes. In the case of the UFC data set we want to model whether the winner is *Red* or *Blue*, so $K = 2$. Letting Y denote the class and $x \in \mathbb{R}^d$ a given data point of features, then we model the probabilities by

$$
\begin{aligned}
P(Y = \text{'}\mathit{Red}\text{'} | X = x) &= \frac{\exp(\beta_0 + \beta_1^T x)}{1 + \exp(\beta_0 + \beta_1^T x)} \\
P(Y = \text{'}\mathit{Blue}\text{'} | X = x) &= \frac{1}{1 + \exp(\beta_0 + \beta_1^T x)}
\end{aligned}
$$

where $\beta_0 \in \mathbb{R}$, and $\beta_1 \in \mathbb{R}^d$ are the parameters that determine the model. The model is fit via maximum likelihood and parameters can be found via the Newton Raphson-algorithm as described in [4, p. 120]. The log-likehood, which we want to maximize is given by

$$
\ell(\beta_0, \beta_1) = \sum_{i=1}^{n} \left\{ y_i(\beta_0 + \beta_1^T x_i) - \log(1 + \beta_0 + \beta_1^T x_i) \right\}
$$

3.1.1 Non-regularized logistic

As a baseline model we fit the full data set using logistic regression. The confusion table for the prediction can be seen in table 1. The train and test error can then be calculated as

$$\text{train accuracy} = \frac{344 + 1455}{344 + 213 + 502 + 1455} = 0.7156$$
$$\text{test accuracy} = \frac{109 + 626}{109 + 91 + 252 + 626} = 0.6818$$

(a) Training results

	Blue	Red
Blue	344	213
Red	502	1455

(b) Test results

	Blue	Red
Blue	109	91
Red	252	626

Table 1: Confusion tables for logistic regression using all 135 features.

3.1.2 Regularized logistic regression

We see that the training accuracy is a bit higher than the test accuracy, when doing ordinary logistic regression. This is an indication that we have overfitted our train data. One way to mitigate this is using regularized logistic regression. Here, the weights are penalized in the loss function, restricting them from becoming to large. In lasso regularization, the optimal estimates of $\beta_0 \in \mathbb{R}$ and $\beta_1 \in \mathbb{R}^d$ is found by maximizing the following expression

$$\ell(\beta_0, \beta_1) - \lambda \sum_{j=1}^{d} |\beta_{1j}|$$

where $\ell(\beta_0, \beta_1)$ is the log-likelihood function of our data.

In order to fit the model, the parameter λ has to be decided on. In order to make this selection we use crossvalidation. We used 10 folds in our crossvalidation. That is we split the training set in 10 equally sized parts, and fit the data using 9 of those parts, and then predicting on the last set. Then we repeat this until every part has been used as validation set. For choosing the sequence of λ's we rely on `glmnet`'s default implementation. That is we set λ_{max} equal to the value in which the first weight is included in the model and the next values equal to the point where the next weights are included until λ_{min} as the λ in which all coefficients are included in the model. Then we do a grid search in the region $[\lambda_{min}, \lambda_{max}]$ to find the best λ to use. The results from cross-validation can be seen in fig. 1. By using the one-standard error rule we obtain an optimal lambda of $\lambda_{1se} = 0.01726505$. Using this λ our model reduces to only 26 different features, as the weights for all the other features have been set to 0. Next we want to evaluate this model on the test set. The results from this predicting can be seen in table 2 and the training and test errors can be computed from the confusion tables as:

$$\text{train accuracy} = \frac{335 + 1452}{335 + 211 + 516 + 1452} = 0.7108$$
$$\text{test accuracy} = \frac{46 + 693}{47 + 24 + 315 + 693} = 0.6855$$

(a) Training set

	Blue	Red
Blue	335	211
Red	516	1452

(b) Test set

	Blue	Red
Blue	46	24
Red	315	693

Table 2: Confusion tables for regularized logistic regression, effectively using only 26 different features.

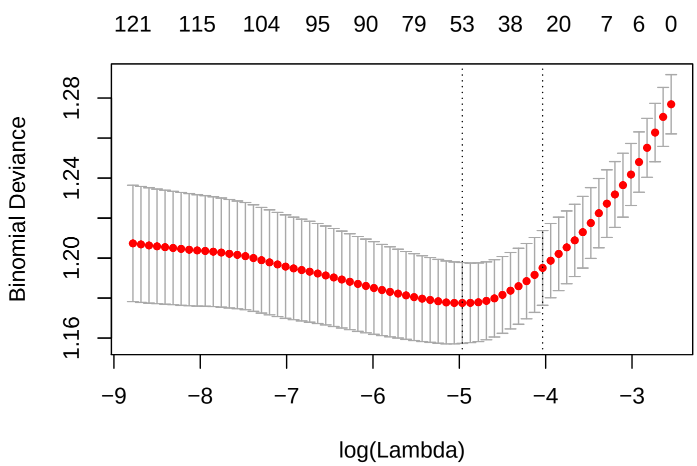

Figure 1: Crossvalidation results for logistic regresssion. The leftmost vertical line indicates the λ for which the crossvalidation error was lowest and the rightmost vertical line indicates the λ one standard error from the minimum value.

3.1.3 Important features

Lasso regularization also preforms feature selection as it sets coefficients to zero. For the cross-validation plotted in fig. 1 there is new weights included for each value of λ. From this method we obtain that the most important features are

3.2 Neural Networks

Neural networks is also a soft classifier as logistic regression, which means it returns probabilities for each observation being in one of the K classes. The difference from logistic regression is that the decision boundary will be highly non-linear in neural networks. The neural network is therefore expected to perform better than logistic regression if the decision boundary between the two winners are actually non-linear.

3.2.1 Feed-forward Neural Network

In this report, we attempt to implement a feed-forward neural network on the 135 features and predict the winner of fights. Information in feed-forward neural networks move only in one direction, from the input nodes into the hidden nodes and finally into the output nodes. As the response is categorical with 2 levels; "Red" and "Blue", we utilized one hot encoding on the response.

3.2.2 Network Shape

The network shapes and dimensions that we used can be seen and illustrated in the table below.

Layer(type)	Output Shape	Number of parameters
input31 (InputLayer)	(None, 135)	0
dense102 (Dense)	(None, 50)	6800
dense103 (Dense)	(None, 30)	1530
dense104 (Dense)	(None, 30)	930
dense105 (Dense)	(None, 2)	62

- Input Layer, 135 units. We use 135 units in our input layer because the dimensions of our input data is 135.
- Hidden Layer 1, 50 units with ELU activation
- Hidden Layer 2, 30 units with Sigmoid activation
- Hidden Layer 3, 30 units with Tanh activation
- Output Layer, 2 units with Softmax activation. We use 2 units in the output layer because we utilized one hot encoding during conversion of the output variable. Since the variable Winner is in binary, this will lead to a dimension of 2 after one hot encoding. We use Softmax function to convert our output into probabilities that sum to 1.

3.2.3 Model Attributes

We use Adam optimizer and categorical loss entropy as our loss function. The metric that we use is accuracy as we are doing categorical analysis. We also utilize an early stopping callback function. It monitors validation loss with a patience of 5. This means that if there is no improvement in the training after 5 epochs, training will be stopped. The motivation to use an early stopping callback function is to prevent overfitting. Lastly, an epoch of 40 and batch size 128 is used with a validation split of 20% during the training of the neural network.

3.2.4 Model Training

As mentioned above, we use a validation split of 20% in training the neural network. This means that we do a 20% split of the train data and we train our neural network using 80% of the train data and validate it on 20% of the train data. This way, we will not be doing forward looking into our test data during training and we only use the test data for predictions.

3.2.5 Results

The plot of the training/test accuracy and loss can be seen below. As we had used an early stopping callback function, we prevented the neural network from over-fitting and potentially giving us worst predictions.

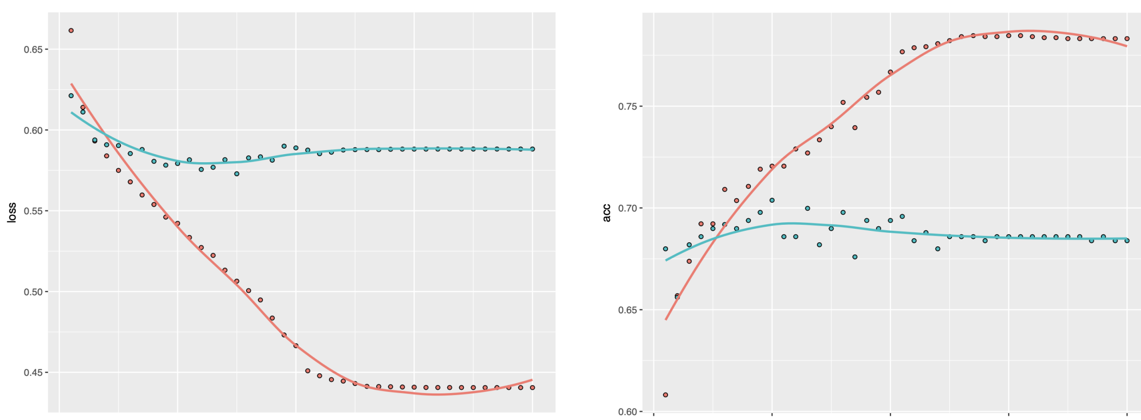

Figure 2: Training/Valid Loss and Training/Valid Accuracy

The green line in the plot above indicates the validation loss/accuracy and the red line indicates the training loss/accuracy. The x-axis is the number of epochs which is 40 in this case.

As our predictions are numerical probabilities, we use a threshold of 0.5 in deciding which categories they belong to. Having a predicted probability greater than 0.5 would result in the category being assigned to Red, else it would be assigned to Blue.

We construct a confusion matrix below to visualize our predicted directions as compared to the actual directions. The test accuracy is also shown below.

(a) Predicted VS Test

	Blue	Red
Blue	133	107
Red	228	610

$$\text{test accuracy} = \frac{133 + 610}{133 + 107 + 228 + 610} = 0.6892393$$

The results of the neural network leads to an accuracy of 68.9%.

4 Discussion

In this session, we would compare these two methods we used and discuss their pros and cons.

According to the result, we got about 0.68 for the accuracy of logistic regression and 0.69 for the accuracy of the Neural Network. So we could see, although the Neural Network outperforms slightly, the prediction impacts of these two methods were generally similar.

As for the reason of the difference, it should result from the decision boundary for this classification issue was not linear so that the classification method with linear boundary like logistic regression may not perform as well as methods with nonlinear boundary like Neural Network.

Although the Neural Network outperformed, it has relatively more complex model fitting process and is difficulty in feature interpretation, which results from the original features have been merged and transformed to thousands of indicators after passing through all the hidden layers, so that it would be hard for us to tell which factors would be essential.

Also, we need to note that Neural Networks require much more data compared to other traditional machine learning methods to have significant results. In the current data set used, we have about 3592 rows of data which might be insufficient for the neural network.

However, In logistic regression, especially with LASSO, the original features were not transformed and the dimension was also reduced significantly from 135 to 26. Therefore, we can easily tell which features are more important, which may be more efficient as we can use the model to guide the fighters in their training.

5 Conclusion

Generally speaking, logistic regression and Neural Network have their pros and cons in this case. Perhaps it might be justifiable to forgo the benefits of accuracy that the neural network has to offer in return for the interpretability of the logistic regression model depending on the use case.

From the perspective of a trainer training his fighter for the competition, certainly the logistic regression model would be more helpful as it gives a clearer interpretation of the features and provides the fighters with a direction to focus on to increase their odds of winning.

However, from the perspective of perhaps someone that is betting on the game, the focus would be aiming to make the prediction more accurate and accuracy would be imperative here. Hence, the Neural Network would be a more optimal choice.

References

[1] Rajeev Warrier, UFC-Fight historical data from 1993 to 2019, https://www.kaggle.com/rajeevw/ufcdata

[2] History of the UFC https://www.ufc.com/history-ufc

[3] Unified rules of mixed martial arts https://www.ufc.com/unified-rules-mixed-martial-arts

[4] Hastie, Tibshirani and Friedman (2009). The Elements of Statistical Learning (2nd edition), Springer-Verlag. 763 pages.