Machine Learning and Sports Management with Professional Fighters League

Brandon Bergstrom, George Smith, Mark Paradis, Charles Vanleuvan

IST 718 Big Data Analytics

Contents

Introduction 2
Specification 3
Observation 3
Analysis 6
Recommendation 11
References 12
Appendix 12

Introduction

Mixed Martial Arts is a sport where two competitors with backgrounds in different techniques fight each other in an octagon using versatile strategies involving striking (punching, kneeing, kicking, wresting, sambo, karate, kickboxing and jiu jitsu). A fighter can win through knock out, technical knockout, submission (i.e., the loser forces the referee to stop the fight), or the display of superior skills in both defense and attack to win more points from the judges.

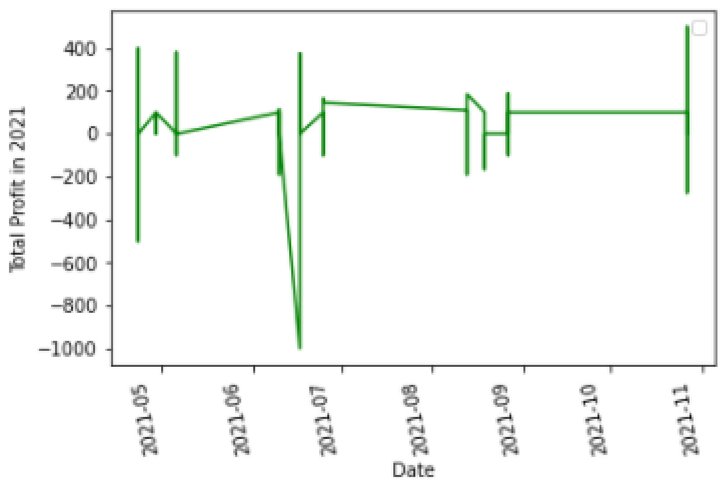
The project will be about a professional sports league called Professional Fighters League (PFL). The project will expand our horizons for utilizing this league's data. This project will be the groundwork for producing a user data source of what individuals think. The project or dataset in use will become enormous in the future.

We want to get informational data on fighters in the PFL and analyze the features of PFL fighters. This will hopefully identify a way to see if the fighter's data will predict fights better than an overall analysis by hypotheticals. Occasionally, someone may guess closer to the true answer, but, as you repeat the experiment, the same person is never consistently better than a model. We have the outcomes and statistical information on every fight in the PFL, and we can build a machine learning or a statistical model to produce a profit.

Mixed Martial Arts is a growing sport collecting more and more data. Doing this project will allow the data collection to continue without having performance issues in the future, and it will ensure an orderly structure protecting the data's integrity. The easy retrieval and updating of data, efficiency, data consistency, data integrity, speed, and security are optimal for this project's future. The business impact of creating a model on the PFL will could be to create a second income stream. This will give the stakeholder, who is our group, an easy way to manipulate and clean the data for machine learning purposes in Python programming language. In the end, our goal is to make a profit on the 2021 season using the historical data to train and test a model. We will be making a $100 bet on underdogs. For favorites, we will bet whatever money is needed to win $100. For example, if the odds are -150, it tells us we must place a $150 bet to win $100, giving us a total of $250 dollars.

In our model script, if we start with $2,000, it is shown that we would have made over **$5,000** throughout the 2021 season. This is only risking $100 or betting to win $100 on each predicted bet. In the below graph, it shows a timeline of our profitability. The model was 80% accurate on fights in 2021.

```
5412
```

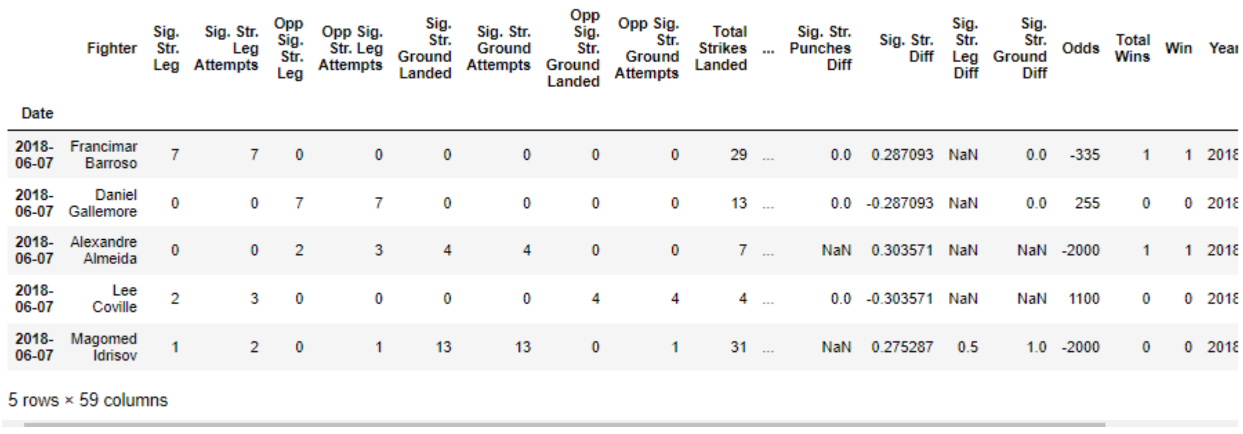


Specification

The project goal is determining whether we can make a profit on individual fights using historical data for the PFL. Our goal will be measured on the data provided by the PFL website. We have scraped the historical data for all years using Python programming.

Observation

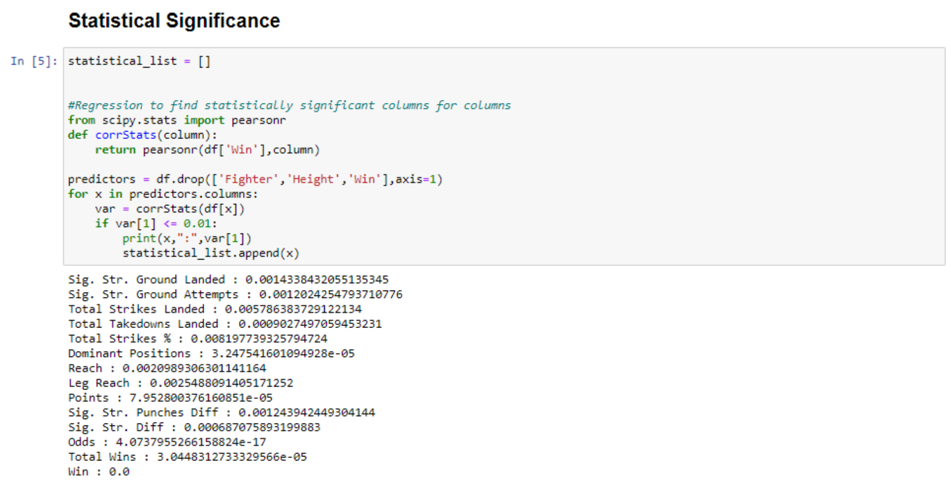
Below are a few sample rows of our dataset showing the date and different statistics at the time of the fight.



Date	Fighter	Sig. Str. Leg	Sig. Str. Leg Attempts	Opp Sig. Str. Leg	Opp Sig. Str. Leg Attempts	Sig. Str. Ground Landed	Sig. Str. Ground Attempts	Opp Sig. Str. Ground Landed	Opp Sig. Str. Ground Attempts	Total Strikes Landed	...	Sig. Str. Punches Diff	Sig. Str. Diff	Sig. Str. Leg Diff	Sig. Str. Ground Diff	Odds	Total Wins	Win	Yea
2018-06-07	Francimar Barroso	7	7	0	0	0	0	0	0	29	...	0.0	0.287093	NaN	0.0	-335	1	1	201
2018-06-07	Daniel Gallemore	0	0	7	7	0	0	0	0	13	...	0.0	-0.287093	NaN	0.0	255	0	0	201
2018-06-07	Alexandre Almeida	0	0	2	3	4	4	0	0	7	...	NaN	0.303571	NaN	NaN	-2000	1	1	201
2018-06-07	Lee Coville	2	3	0	0	0	0	4	4	4	...	0.0	-0.303571	NaN	NaN	1100	0	0	201
2018-06-07	Magomed Idrisov	1	2	0	1	13	13	0	1	31	...	NaN	0.275287	0.5	1.0	-2000	0	0	201

5 rows × 59 columns

Next, we wanted to see what values were statistically significant and see what features are important to the dataset for our predictor (the Win column). We made a for loop and function on the Pearson package to check for statistical significance in our dataframe. We only printed and appended our features with a statistical significance of 99%.

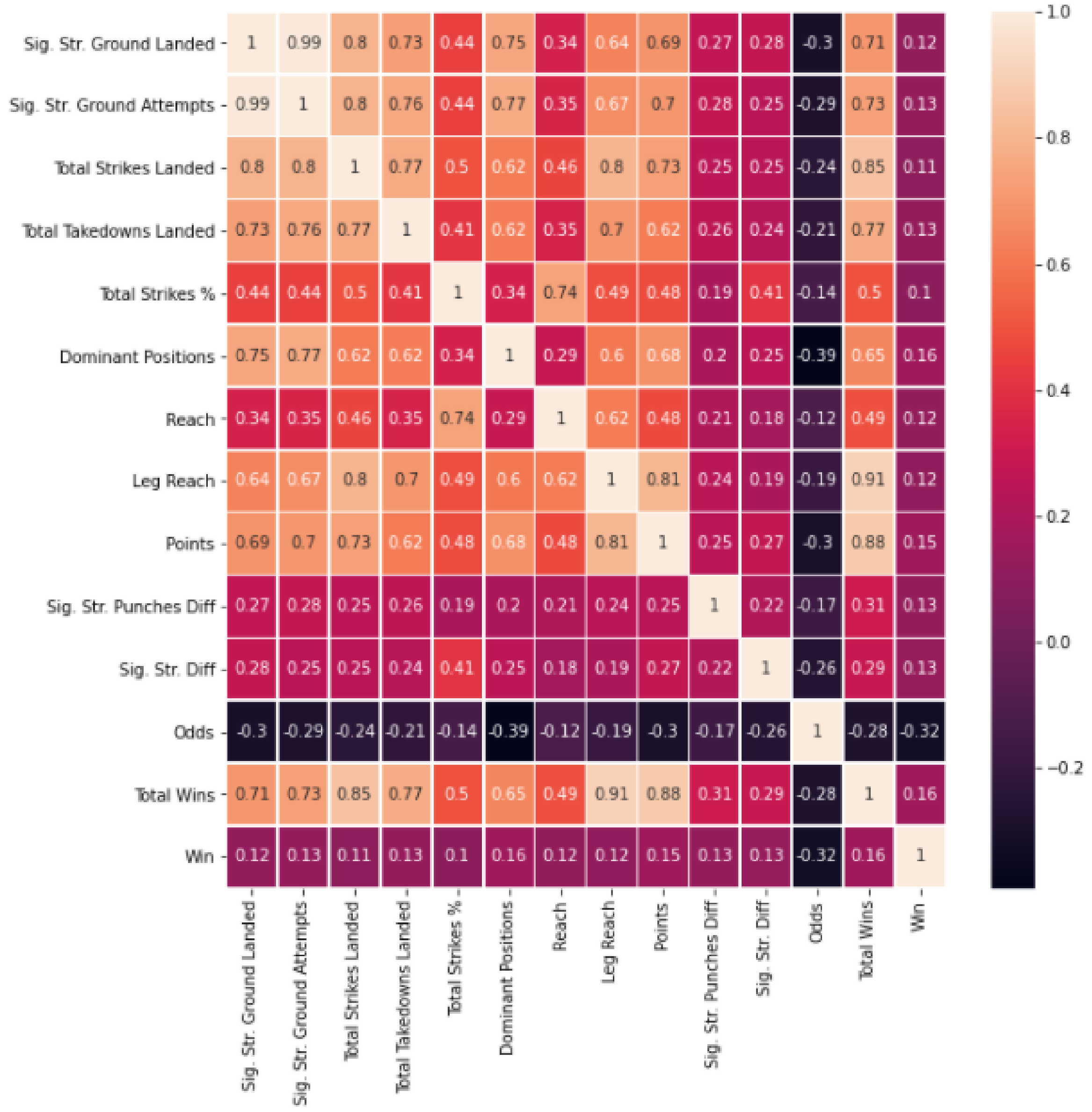


Statistical Significance

```
In [5]: statistical_list = []


        #Regression to find statistically significant columns for columns
        from scipy.stats import pearsonr
        def corrStats(column):
            return pearsonr(df['Win'],column)

        predictors = df.drop(['Fighter','Height','Win'],axis=1)
        for x in predictors.columns:
            var = corrStats(df[x])
            if var[1] <= 0.01:
                print(x,":",var[1])
                statistical_list.append(x)
```

```
Sig. Str. Ground Landed : 0.0014338432055135345
Sig. Str. Ground Attempts : 0.0012024254793710776
Total Strikes Landed : 0.005786383729122134
Total Takedowns Landed : 0.0009027497059453231
Total Strikes % : 0.008197739325794724
Dominant Positions : 3.247541601094928e-05
Reach : 0.0020989306301141164
Leg Reach : 0.0025488091405171252
Points : 7.952800376160851e-05
Sig. Str. Punches Diff : 0.001243942449304144
Sig. Str. Diff : 0.000687075893199883
Odds : 4.0737955266158824e-17
Total Wins : 3.0448312733329566e-05
Win : 0.0
```

For us to make a profit, we will need to ensure we are accurate in our dataset. We made a correlation map to see if any of our highly statistically significant features had strong correlation to the Win column Unfortunately, nothing has a strong relationship with the dependent variable, as can be seen below.

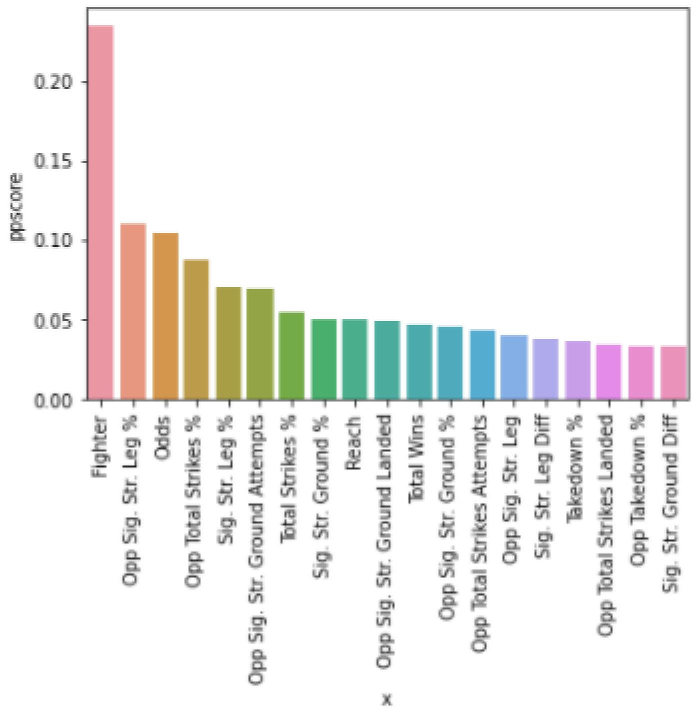


After this, we graph the Predictive Power Score which also helps us identify the most important features.

Find patterns in the data: The PPS finds every relationship the correlation finds — and more. Thus, you can use the PPS matrix as an alternative to the correlation matrix to detect and understand linear or nonlinear patterns in your data. This is possible across data types using a single score that always ranges from 0 to 1.

Feature selection: In addition to your usual feature selection mechanism, you can use PPS to find good predictors for your target column. Also, you can eliminate features that just add random noise. Those features sometimes still score high in feature importance metrics. In addition, you can eliminate features that can be predicted by other features because they don't add new information. You can also identify pairs of mutually predictive features in the PPS matrix — this includes strongly correlated features but will also detect non-linear relationships.

Data Normalization: One can find entity structures in the data via interpreting the PPS matrix as a directed graph. This might be surprising when the data contains latent structures that were previously unknown.



In the above picture, we can see the features that are the most important according to the PPS.

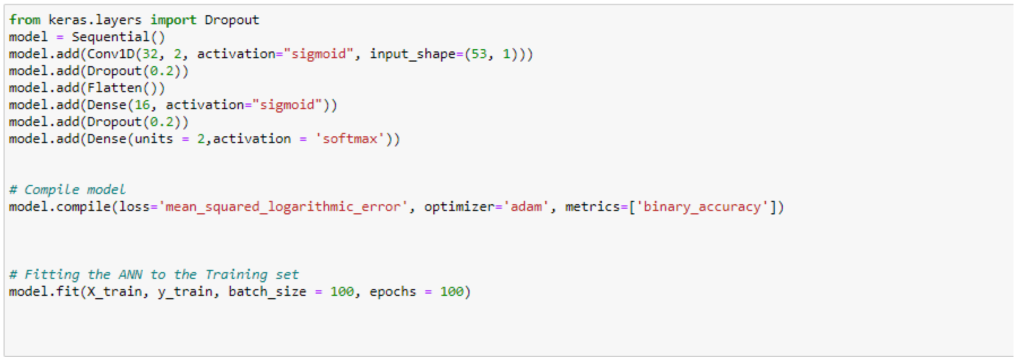
Analysis

Using our broad experience of sports, finance, and knowledge in data science, we decided to use a few models to identify the best way to produce profitability in 2021.

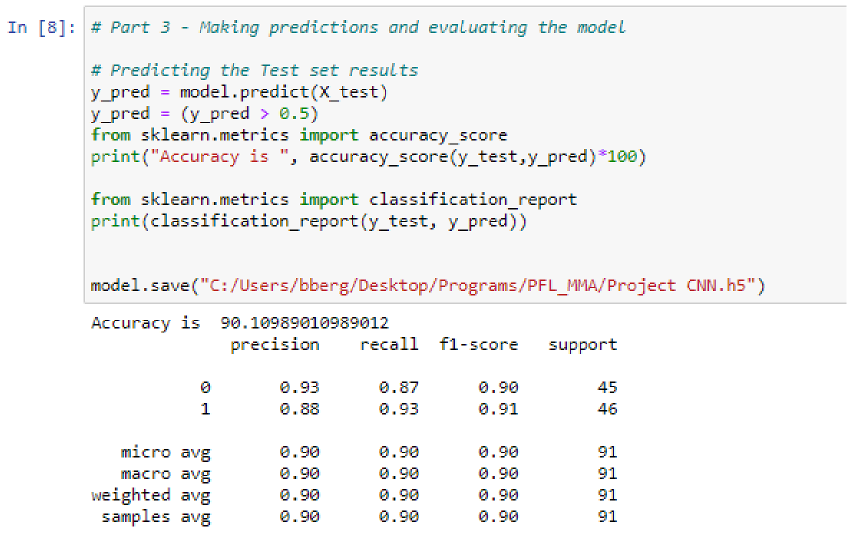
In our first example, we used a Convolutional Neural Network, which is a 1-dimensional model. Instead of reading in 2D or 3D pictures, we can use it for a structured dataset that is 1 dimensional. In 1D CNN, kernel moves in 1 direction. Input and output data of 1D CNN is 2 dimensional.

We create different layers of the model using the Kera's package. Our initial features are all 53 features except the Fighters name, Height, and the dependent variable. Our reasoning behind this is that we do not have strong indicators to help generalize the model in the training. We decided to let deep learning create its own features and see something in the dataset that we couldn't foresee.

Our activation functions for certain layers in the sigmoid function. The sigmoid activation function produces results in the range of 0 to 1 which is interpreted as the probability. Next, we use a probability activation function called softmax to produce a probability of who will win the fight for each prediction. We let the dataset learn the dataset 100 times and 100 rows at a time for each adjustment of mistakes.

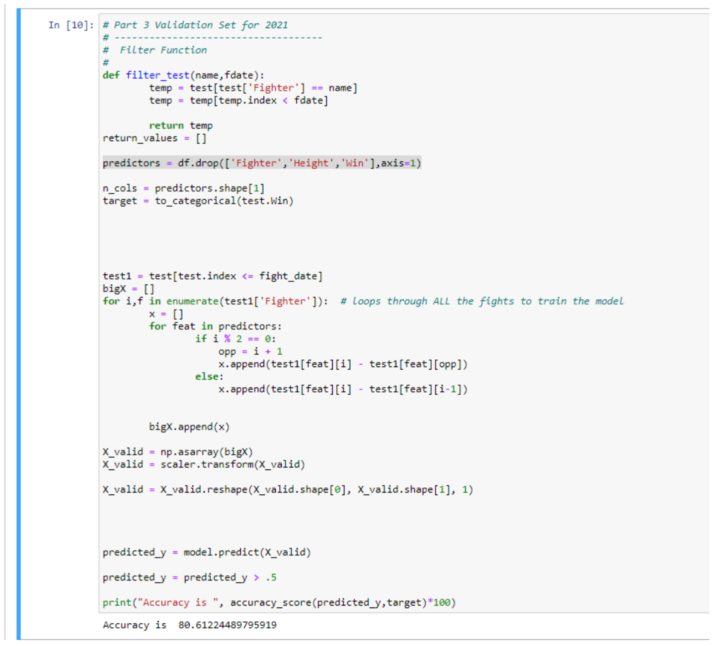


```
from keras.layers import Dropout
model = Sequential()
model.add(Conv1D(32, 2, activation="sigmoid", input_shape=(53, 1)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(16, activation="sigmoid"))
model.add(Dropout(0.2))
model.add(Dense(units = 2,activation = 'softmax'))


# Compile model
model.compile(loss='mean_squared_logarithmic_error', optimizer='adam', metrics=['binary_accuracy'])


# Fitting the ANN to the Training set
model.fit(X_train, y_train, batch_size = 100, epochs = 100)
```



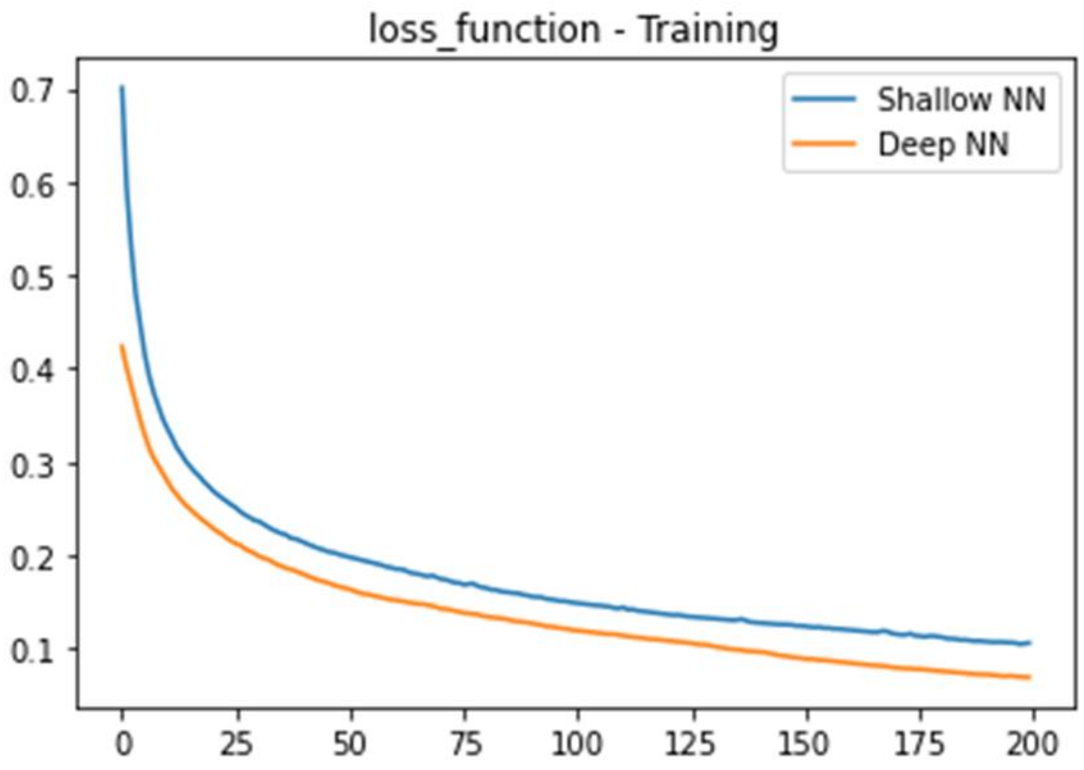
```
In [8]: # Part 3 - Making predictions and evaluating the model

        # Predicting the Test set results
        y_pred = model.predict(X_test)
        y_pred = (y_pred > 0.5)
        from sklearn.metrics import accuracy_score
        print("Accuracy is ", accuracy_score(y_test,y_pred)*100)

        from sklearn.metrics import classification_report
        print(classification_report(y_test, y_pred))


        model.save("C:/Users/bberg/Desktop/Programs/PFL_MMA/Project CNN.h5")

        Accuracy is  90.10989010989012
                      precision    recall  f1-score   support

                   0       0.93      0.87      0.90        45
                   1       0.88      0.93      0.91        46

           micro avg       0.90      0.90      0.90        91
           macro avg       0.90      0.90      0.90        91
        weighted avg       0.90      0.90      0.90        91
         samples avg       0.90      0.90      0.90        91
```

The validation set is 80% accurate in 2021 fights producing a profit of $5,412.

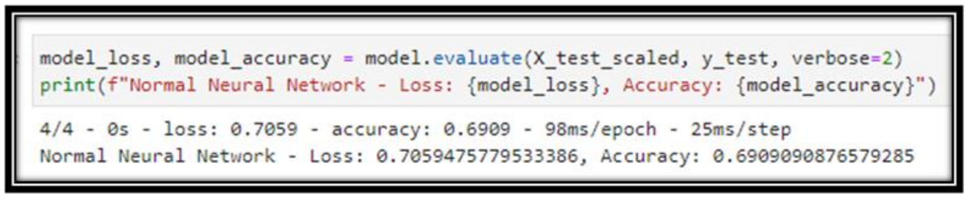


```
In [10]: # Part 3 Validation Set for 2021
         # -----------------------------------
         #  Filter Function
         #
         def filter_test(name,fdate):
                 temp = test[test['Fighter'] == name]
                 temp = temp[temp.index < fdate]

                 return temp
         return_values = []

         predictors = df.drop(['Fighter','Height','Win'],axis=1)

         n_cols = predictors.shape[1]
         target = to_categorical(test.Win)




         test1 = test[test.index <= fight_date]
         bigX = []
         for i,f in enumerate(test1['Fighter']):  # loops through ALL the fights to train the model
                 x = []
                 for feat in predictors:
                         if i % 2 == 0:
                             opp = i + 1
                             x.append(test1[feat][i] - test1[feat][opp])
                         else:
                             x.append(test1[feat][i] - test1[feat][i-1])


                 bigX.append(x)

         X_valid = np.asarray(bigX)
         X_valid = scaler.transform(X_valid)

         X_valid = X_valid.reshape(X_valid.shape[0], X_valid.shape[1], 1)




         predicted_y = model.predict(X_valid)

         predicted_y = predicted_y > .5

         print("Accuracy is ", accuracy_score(predicted_y,target)*100)

         Accuracy is  80.61224489795919
```

Another model that we used was a Tensorflow – Deep Neural Network. We experimented with both shallow and deep neural networks and found better results using a deep neural network at the expense of increased computing time.

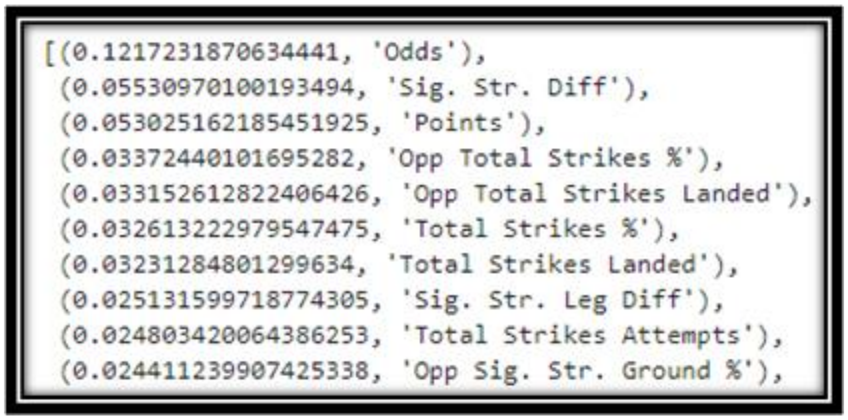


This model used 52 features and produced an accuracy of 69%.



```
model_loss, model_accuracy = model.evaluate(X_test_scaled, y_test, verbose=2)
print(f"Normal Neural Network - Loss: {model_loss}, Accuracy: {model_accuracy}")

4/4 - 0s - loss: 0.7059 - accuracy: 0.6909 - 98ms/epoch - 25ms/step
Normal Neural Network - Loss: 0.7059475779533386, Accuracy: 0.6909090876579285
```

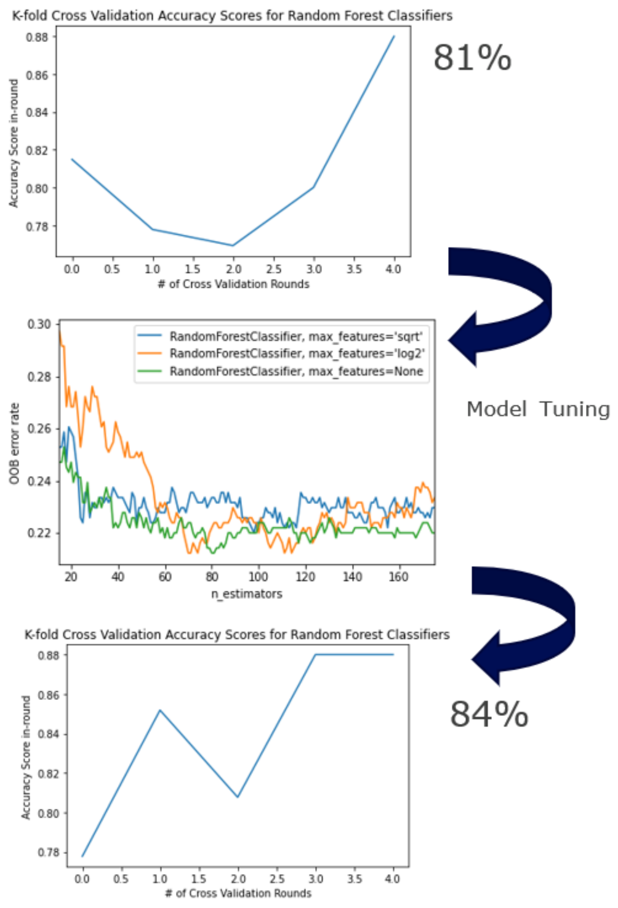
The top ten features are listed below.



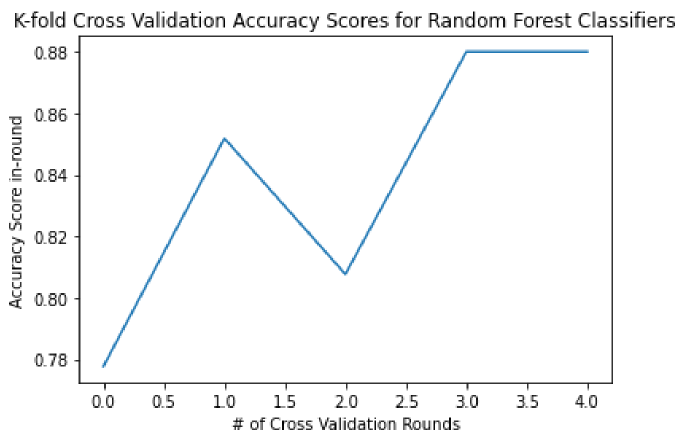
```
[(0.1217231870634441, 'Odds'),
 (0.05530970100193494, 'Sig. Str. Diff'),
 (0.05302516218545192 5, 'Points'),
 (0.0337244010169528 2, 'Opp Total Strikes %'),
 (0.033152612822406426, 'Opp Total Strikes Landed'),
 (0.03261322297954747 5, 'Total Strikes %'),
 (0.0323128480129963 4, 'Total Strikes Landed'),
 (0.02513159971877430 5, 'Sig. Str. Leg Diff'),
 (0.02480342006438625 3, 'Total Strikes Attempts'),
 (0.02441123990742533 8, 'Opp Sig. Str. Ground %'),
```

Lastly, another successful model our team used was a Random Forest model. This model used 54 features and matched up well with features and our model type. Random Forest algorithms create many individual trees from randomly selected subsets of the dataset. Each of these individual trees generates a classification/regression of objects within the subset of data. For each tree, the prediction error on the out-of-bag portion of the data is recorded (error rate for classification, MSE for regression). Then, the same is done after permuting each predictor variable. The difference between the two is then averaged over all trees and normalized by the standard deviation of the differences. If the standard deviation of the differences is equal to 0 for a variable, the division is not done (but the average is almost always equal to 0 in that case). Following this, a model has been built, splitting 20% of the data for testing to produce results with high accuracy.

Using the default model parameters (100 estimators and no maximum tree depth), the initially developed model produced 81% accuracy. In random forest modeling, only a subset of the training data is used to build out the trees (this is known as bootstrapping). The tree model developed on this subset can then be applied to the unused training data and an accuracy score can be generated. This is known as the Out of Bag error rate. OOB error rate visualization shows the effect an additional tree (or other tuned parameter) has on the model training performance. From this, we can see where the error rate stabilizes for the number of estimators and pick a good value. The number of estimators was varied from 15 to 175 (I.e., the number of trees in each forest) and the maximum depth of each tree was varied from the square root of sample size, log-base two of sample size, and no maximum depth. The OOB error rate testing is shown below:



The most accurate model developed in training had 80 trees, and no maximum depth to the tree. Five-fold cross validation testing resulted in an average accuracy score of 84% for correctly predicting whether a fighter would win a fight based on fighting features. The 5 fold cross validation scores are shown below:



Recommendation

Our recommendation would be that we can create a profit of $5,412 using only $100 for underdogs and betting to win $100 on favorites. If we increase our exposure, it increases our risk, but it would increase our profits, too.

In our initial result, if we had $2,000 and bet enough to win $100 or risked $100, it would be a 5% risk per bet (100 divided by 2,000). In this same instance, we would have grown the account by 270%. For example, if we had a $100,000 fund, we would be pricing equivalently. We would have made $170,600 if our account was larger, and we placed equivalent risk on each bet like we did in our dataset.

Our more successful random forest model had an 84% fight prediction success rate. Typically, 55-60% is considered best in class for predicting game outcomes using purely sport data. This means the addition of the MoneyLine odds provides an "x-factor" that vastly improves our model. However, profitability is entirely dependent on the average odds from all bets wagered. Model accuracy rate must be higher than the breakeven success rate to consider this a profitable, valid betting strategy. The average odds for all fighters in the data set is –190 (1.52 decimal odds). This gives a breakeven win rate of 65.75%. Our models outperform the sportsbooks by 18.25%. Overall, if we keep this pace, we can see successful profit going forward and our biggest impact will be to optimize our profit and reduce how much we are wrong. We do not have a ton of frequency like a day trader as fights only happen once a week. This means we have less opportunity to produce profit, so we need to be extremely accurate.

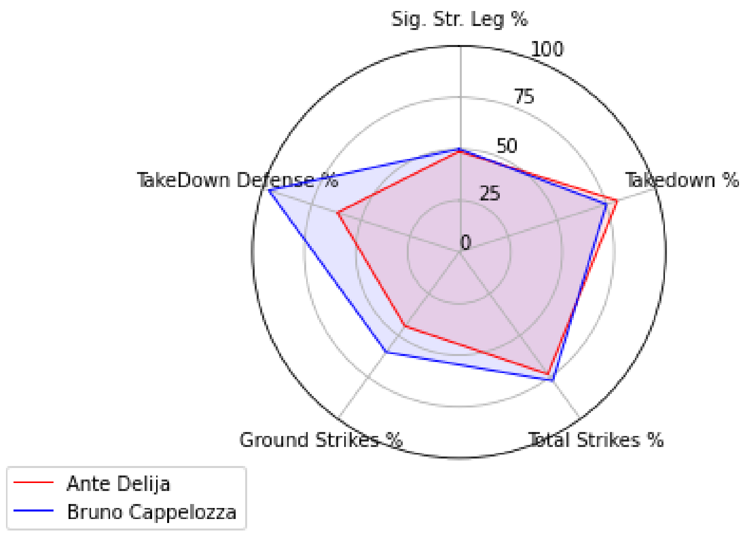
References

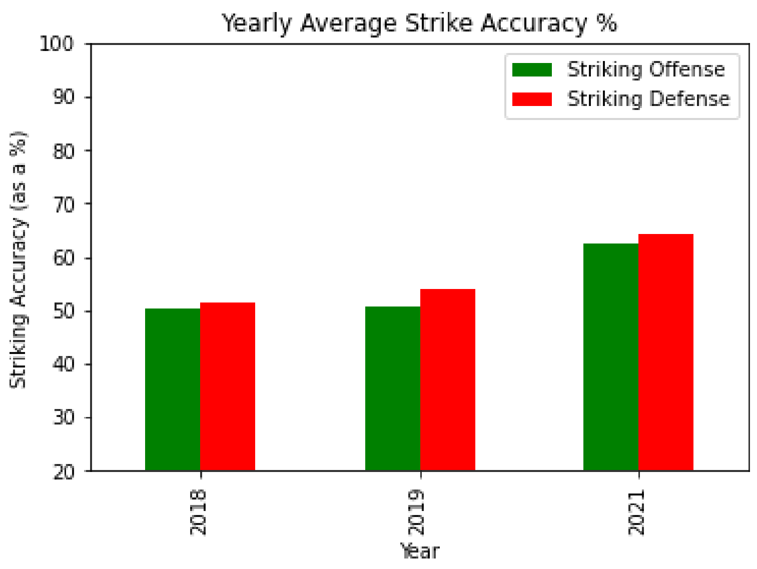
PFL Website for Scrapping Fight Data

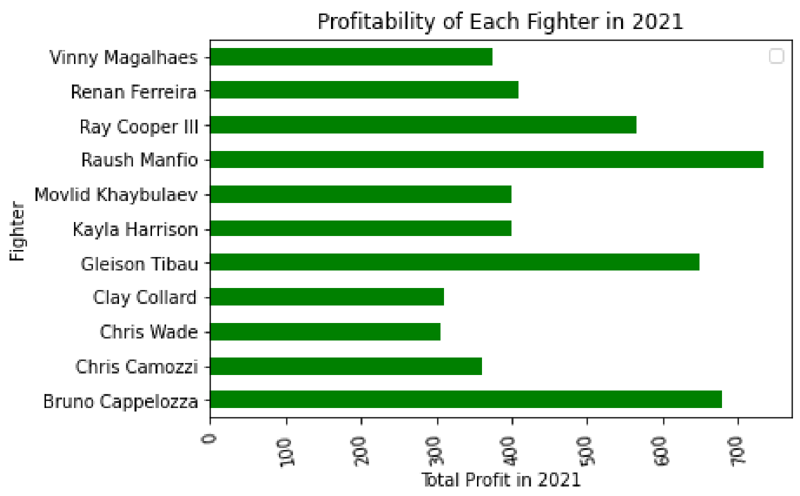
"PFL." n.d. Pflmma.com. Accessed December 12, 2021. https://pflmma.com/.

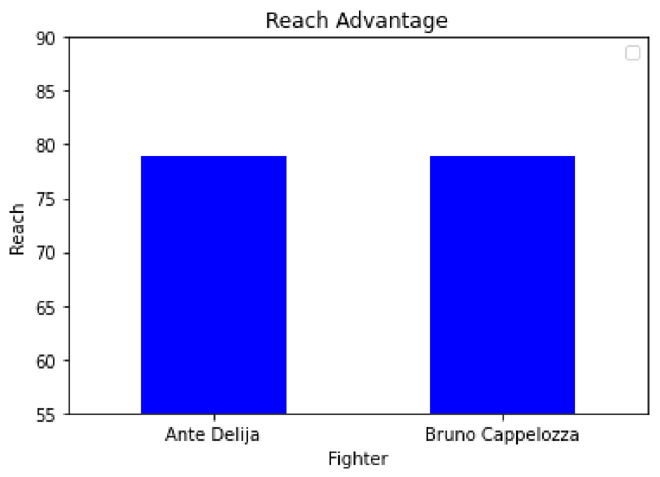
Appendix

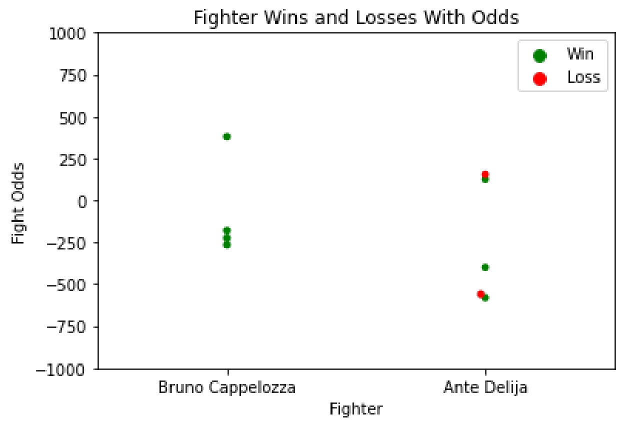
Extra Visualizations:




Yearly Average Strike Accuracy %
Striking Offense
Striking Defense
Striking Accuracy (as a %)
100
90
80
70
60
50
40
30
20
2018
2019
2021
Year


Profitability of Each Fighter in 2021
Vinny Magalhaes
Renan Ferreira
Ray Cooper III
Raush Manfio
Movlid Khaybulaev
Kayla Harrison
Gleison Tibau
Clay Collard
Chris Wade
Chris Camozzi
Bruno Cappelozza
Fighter
0
100
200
300
400
500
600
700
Total Profit in 2021


Reach Advantage
90
85
80
75
70
65
60
55
Reach
Ante Delija
Bruno Cappelozza
Fighter


Fighter Wins and Losses With Odds
1000
750
500
250
0
−250
−500
−750
−1000
Fight Odds
Bruno Cappelozza
Ante Delija
Fighter
Win
Loss