Predicting NFL Scores using Sportsbook Odds

Nick Colvin

CS 6375 Fall 2022

Introduction

The dataset used for this prediction problem comprises 4219 NFL games from the beginning of the 2007-2008 season to the most recently completed week (Week 13) of the 2022-2023 season. It includes regular season and playoff games. According to the source website, data is sourced from various offshore and Nevada based sportsbooks. Results are updated weekly.

Methods

Two models were used for this problem: MLPRegressor from scikit-learn and my own implementation. The data was reorganized into a more interpretable format for the models. Team names were encoded into integer labels, and every sample was normalized according to the test and training sets. The training set size was 75% of the overall dataset size, and was selected arbitrarily for brevity by choosing a split that performed well with both models. Hyperparameters were also chosen in an ad hoc manner: different configurations of 1-3 hidden layers consisting of 10-200 neurons using either tanh, sigmoid, or ReLU activation functions and MSE or MAE loss functions were tested. Ultimately, a single hidden layer of 100 neurons using ReLU activation was used along with a learning rate of 0.003 over 100 training iterations for the following analysis of features.

In order to judge the relative importance of each feature to the model predictions, the models were trained on the same dataset each time with a single feature removed. This process was repeated 10 times in order to calculate averages and standard deviations for the scores, records, and profits using the modified datasets.

Additionally, bar charts were generated in order to gain insight as to which features were weighed more heavily by my implementation of the model. Each bar represents a sample, and the 25 best predictions are shown. The size of each section represents the norm of the product of feature weights and sample value. These charts were generated after the performance table, and do not reflect the weights for those models. However, they provide similar examples of what a single training session might generate. Features were removed in the same order as in the table.

Results

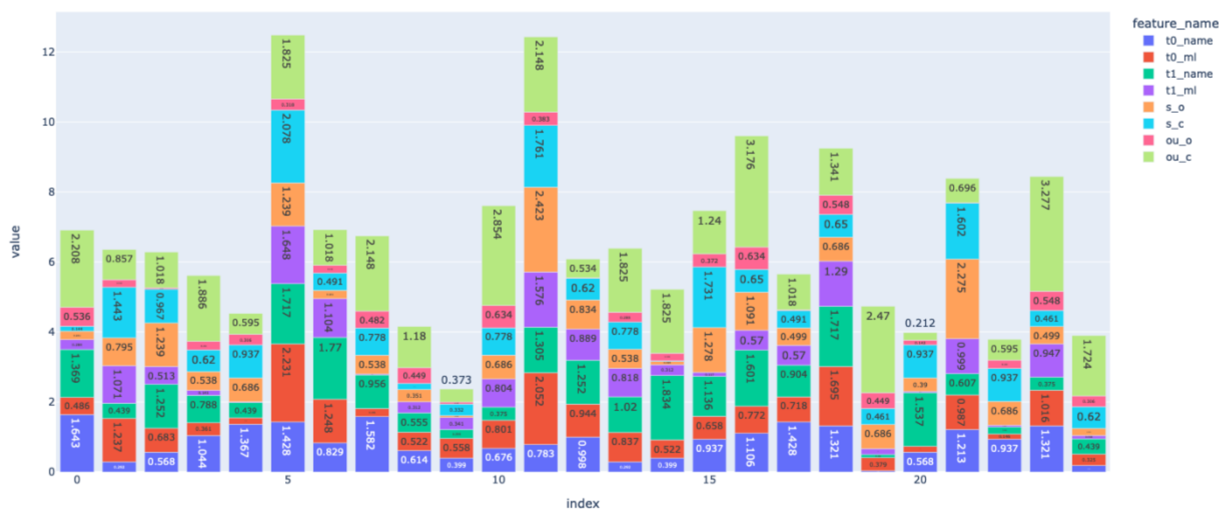
Scikit-learn MLPRegressor

Removed Feature	R2	MAE	MSE	ML Record	ML Profit	Spread Record	Spread Profit	Over/Under Record	Over/Under Profit
None	0.029 stddev=0.149	0.763 stddev=0.009	0.971 stddev=0.149	670 331 3 51	-35.5 stddev=6.81	478 459 20 98	-23.7 stddev=13.8	480 468 10 97	-32.2 stddev=17.5
Advantage	0.033 stddev=0.121	0.764 stddev=0.008	0.967 stddev=0.121	675 330 3 47	-33.3 stddev=6.77	474 465 20 96	-34.1 stddev=15.7	489 474 11 81	-29.0 stddev=21.6
Name	0.054 stddev=0.053	0.759 stddev=0.005	0.946 stddev=0.053	676 329 3 47	-43.9 stddev=7.13	454 445 17 139	-32.7 stddev=17.8	466 474 10 105	-50.1 stddev=21.1
ML	-0.038 stddev=0.094	0.791 stddev=0.006	1.04 stddev=0.094	551 390 3 111	-19.3 stddev=14.8	515 475 22 43	-7.1 stddev=15.8	476 474 10 95	-40.5 stddev=21.4
Spread Open	0.078 stddev=0.036	0.758 stddev=0.004	0.922 stddev=0.036	679 325 3 48	-24.3 stddev=8.59	480 438 19 117	-1.76 stddev=15.2	480 473 10 92	-37.0 stddev=18.0
Spread Close	0.044 stddev=0.044	0.763 stddev=0.004	0.956 stddev=0.044	682 329 3 41	-31.4 stddev=8.17	472 455 18 110	-26.2 stddev=21.3	482 476 10 87	-37.6 stddev=21.1
Total Open	0.113 stddev=0.005	0.758 stddev=0.003	0.887 stddev=0.005	677 326 3 49	-26.8 stddev=12.0	480 450 18 106	-14.2 stddev=19.8	491 563 9 92	-16.3 stddev=19.6
Total Close	0.031 stddev=0.027	0.769 stddev=0.003	0.970 stddev=0.027	679 334 3 39	-38.7 stddev=6.69	479 439 20 116	-3.46 stddev=13.9	506 482 11 56	-21.5 stddev=14.7

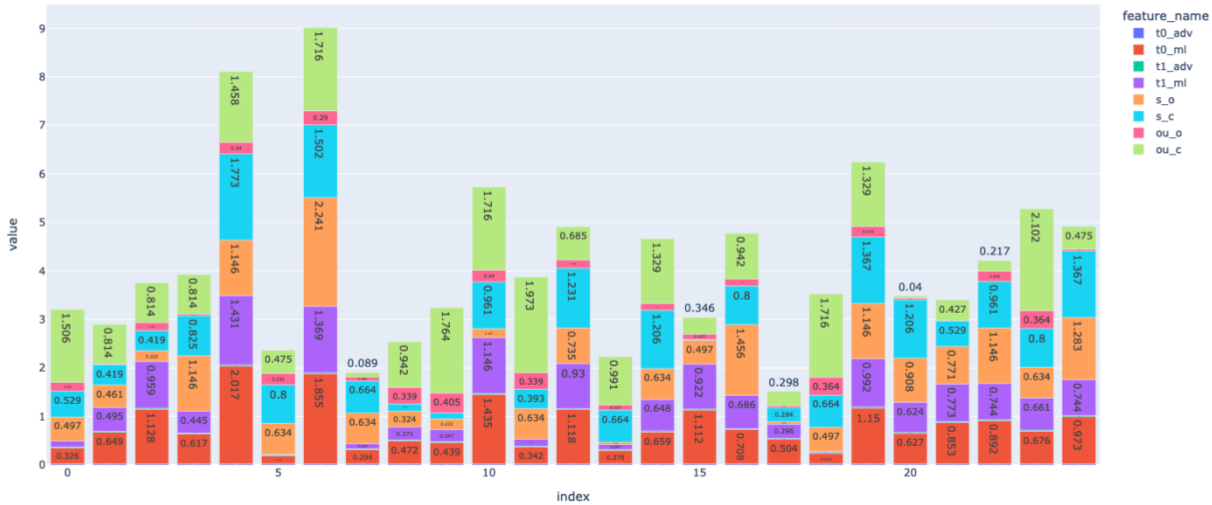
Own Implementation

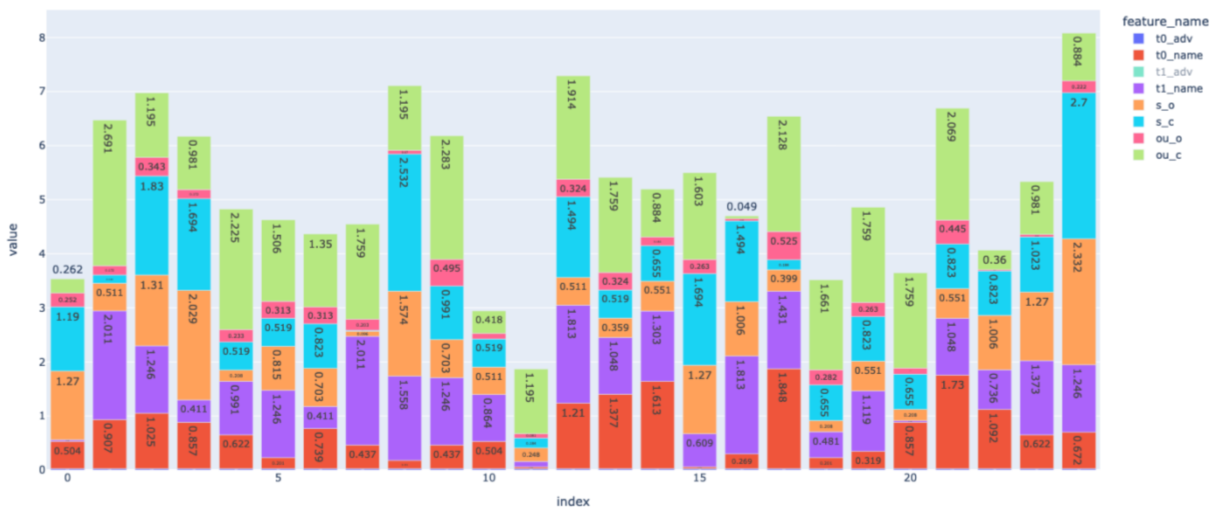
Removed Feature	R2	MAE	MSE	ML Record	ML Profit	Spread Record	Spread Profit	Over/Under Record	Over/Under Profit
None	0.116 stddev=0.003	0.753 stddev=0.001	0.884 stddev=0.003	678 329 3 45	-35.6 stddev=7.21	448 456 17 134	-49.3 stddev=17.3	483 447 8 117	-8.50 stddev=11.7
Advantage	0.084 stddev=0.012	0.754 stddev=0.001	0.916 stddev=0.012	669 336 3 46	-54.7 stddev=3.40	440 453 17 145	-53.4 stddev=16.9	510 491 11 43	-27.1 stddev=7.87
Name	0.096 stddev=0.005	0.751 stddev=0.0006	0.904 stddev=0.005	692 337 3 23	-44.0 stddev=2.56	404 408 14 229	-41.2 stddev=9.86	515 502 12 26	-34.0 stddev=4.74
ML	0.038 stddev=0.023	0.780 stddev=0.002	0.962 stddev=0.023	525 369 3 157	-18.8 stddev=8.88	511 481 23 39	-16.9 stddev=9.89	496 477 9 74	-26.3 stddev=9.19
Spread Open	0.108 stddev=0.006	0.750 stddev=0.001	0.892 stddev=0.006	670 333 3 49	-46.8 stddev=4.41	454 433 18 150	-21.0 stddev=14.2	507 489 11 48	-27.8 stddev=5.96
Spread Close	0.096 stddev=0.016	0.753 stddev=0.002	0.904 stddev=0.016	683 344 3 25	-51.1 stddev=3.94	443 462 20 130	-59.9 stddev=17.7	512 496 10 37	-30.5 stddev=8.23
Total Open	0.113 stddev=0.002	0.751 stddev=0.0008	0.887 stddev=0.002	673 332 3 47	-46.9 stddev=5.41	447 456 18 134	-49.2 stddev=14.0	522 491 10 32	-16.4 stddev=8.81
Total Close	0.029 stddev=0.021	0.767 stddev=0.003	0.971 stddev=0.021	681 336 3 35	-48.1 stddev=4.55	433 440 17 165	-45.9 stddev=14.7	535 465 10 45	20.6 stddev=8.44

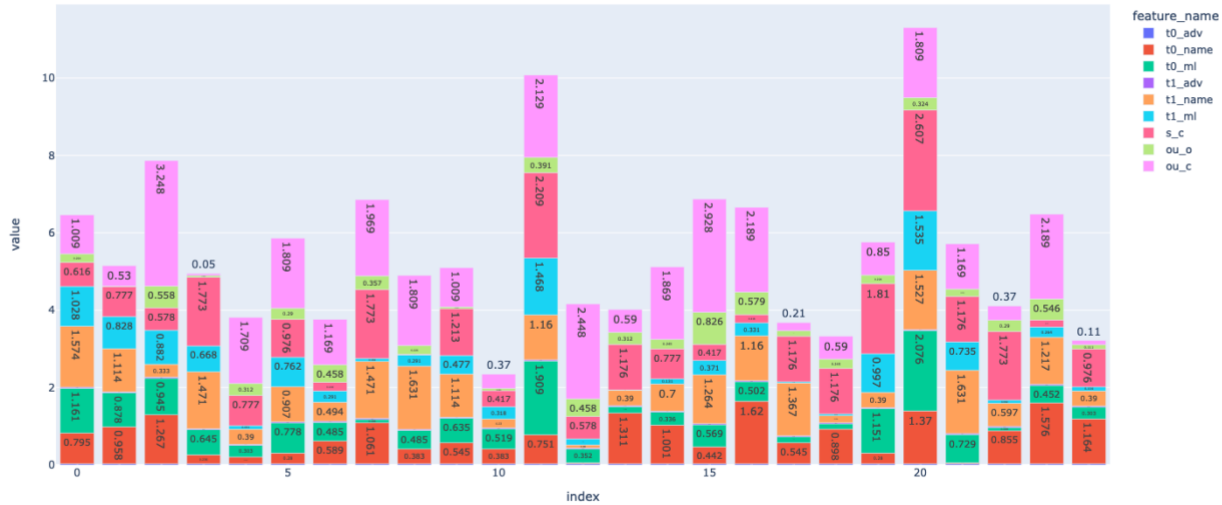
feature_name
t0_adv
t0_name
t0_ml
t1_adv
t1_name
t1_ml
s_o
s_c
ou_o
ou_c
value
index
0
5
10
15
20
2
4
6
8
10
12

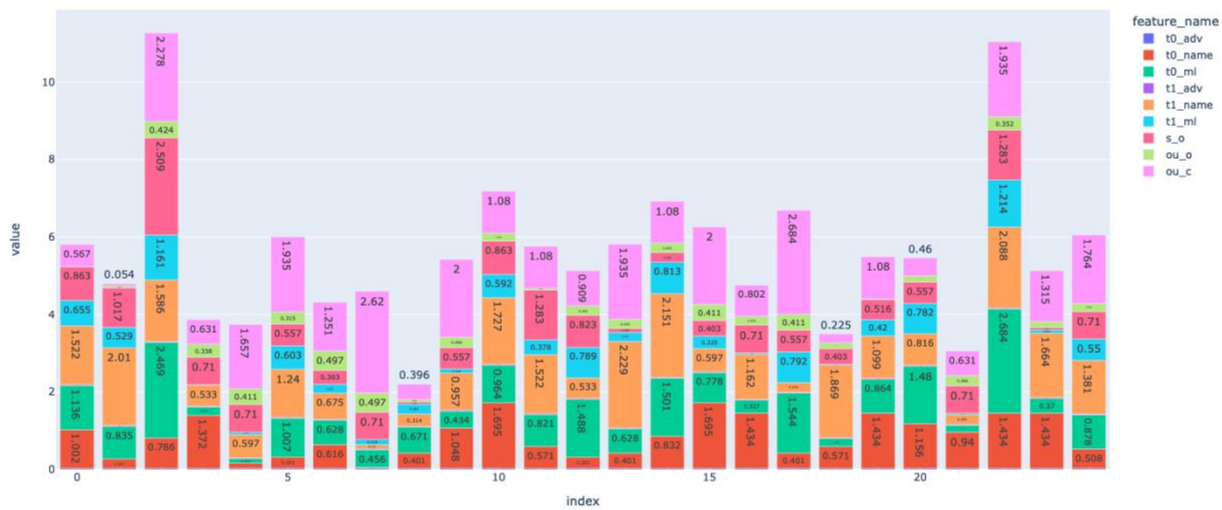


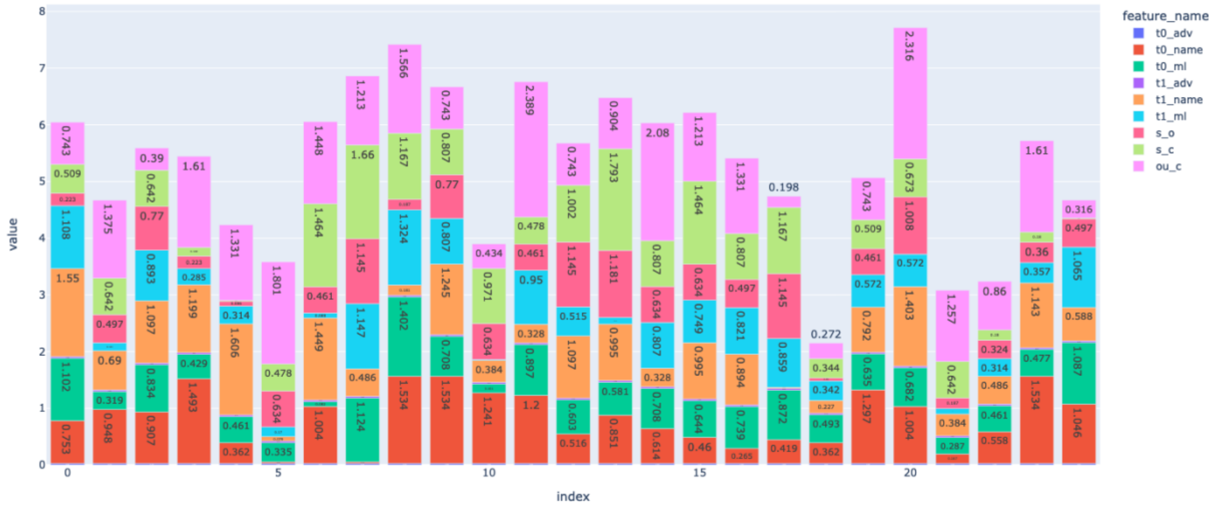
feature_name
t0_name
t0_ml
t1_name
t1_ml
s_o
s_c
ou_o
ou_c
value
index
0
5
10
15
20
2
4
6
8
10
12

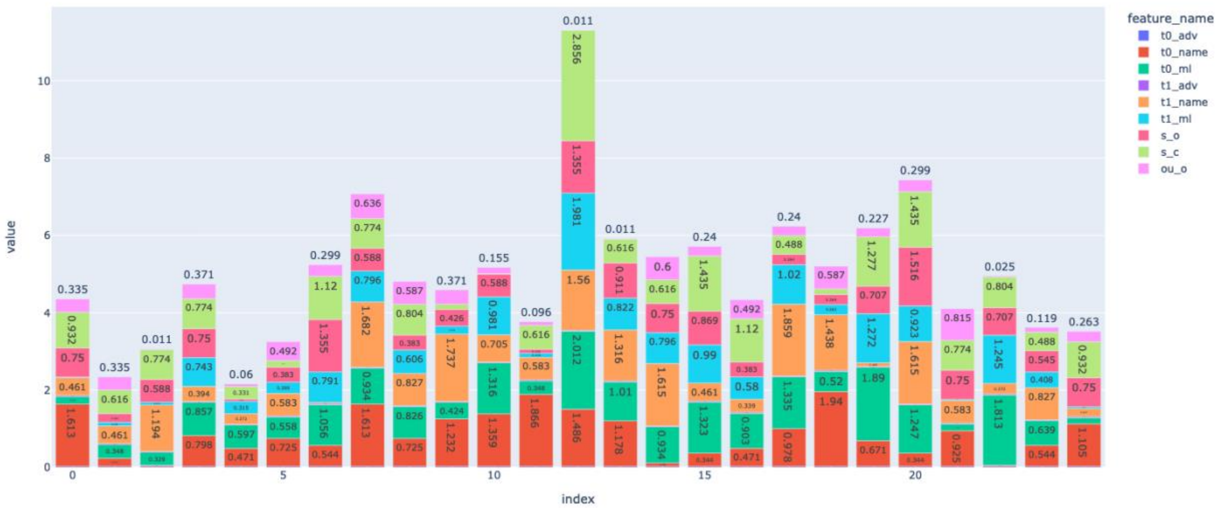

feature_name
t0_adv
t0_ml
t1_adv
t1_ml
s_o
s_c
ou_o
ou_c
value
index


feature_name
t0_adv
t0_name
t1_adv
t1_name
s_o
s_c
ou_o
ou_c
value
index


value
index
0
2
4
6
8
10
5
15
20
feature_name
t0_adv
t0_name
t0_ml
t1_adv
t1_name
t1_ml
s_c
ou_o
ou_c


value
index
0
2
4
6
8
10
5
15
20
feature_name
t0_adv
t0_name
t0_ml
t1_adv
t1_name
t1_ml
s_o
ou_o
ou_c


feature_name
t0_adv
t0_name
t0_ml
t1_adv
t1_name
t1_ml
s_o
s_c
ou_c
value
index
0
5
10
15
20


feature_name
t0_adv
t0_name
t0_ml
t1_adv
t1_name
t1_ml
s_o
s_c
ou_o
value
index
0
5
10
15
20

Analysis and Conclusion

The most important features for my model on average were the moneyline, team IDs, closing total, and closing spread. This coincides with lower R2 values when those features were removed from the dataset. However, while scores were worse, returns were better without moneyline in the dataset. It was also a bit surprising that MAE was lower without team names, but an explanation is that the model can afford greater degrees of freedom with respect to the other features. The biggest surprise from my model was an average profit of over 20 units betting the total when the closing total was removed. This was the only positive return made by any of the models, and a reasonable standard deviation suggests it was not entirely by chance.

The MLPRegressor from Scikit-learn also appeared to place the greatest weight on the moneyline, and removing it from the dataset was the only time the R2 score was lowered. Removing the opening total seemed to have a positive effect on R2, MAE, and MSE. Moreover, removing either opening or closing total from both models made a positive impact on returns. This is peculiar behavior that could be explained by the models having greater freedom to predict different totals. Overall, both models performed reasonably similar with respect to scores and returns, which is not surprising given their similar implementations. However, the standard deviation of returns was lower for my model than Scikit-learn's.

One drawback of my model was training time. This was a bottleneck in terms of getting the best performance. Ideally, these models would have been averaged over 10 sessions using 1000 iterations, however it simply took too long. There did not appear to be any evidence of overfitting even with such a large number of iterations. Another possible improvement would be some sort of loss function to factor in the returns on the training set. Lower error did not necessarily imply higher returns, so it seems like a reasonable consideration.