CS-UY 4563 Final Project

Prediction of Soccer games through Machine Learning

Chloe Atchabahian

Anson Lee

May 2, 2022

Abstract

This project aimed to predict the outcome of a soccer match given all of the information available prior to that match occurring. This information consisted of the FIFA attributes of the team and each of the 22 players, as well as some personal traits of the players. We used three different algorithms to attempt to build a good model for this objective. Our benchmark was the bookkeeper odds, which were available in the database, and are accurate 53% of the time. Both logistic regression and SVM models built were able to satisfy this goal. However, neural networks did not produce acceptable results. This is likely due to insufficient training data.

Introduction

Our project aimed to be able to predict, with reasonable accuracy, the outcome of a soccer match based on the teams and players. To do this, we pulled a database from kaggle.com containing entries for 25000 matches, as well as data on each player and team. However, due to missing data, 4000 of these matches were pruned from the final dataset. We used three different machine learning algorithms to model the behavior of the dataset: logistic regression, SVM, and a neural network. Both SVMs and Logistic regression models produced acceptable results, but neural networks did not.

Preprocessing

The sourced database contained several tables. The main table was the *Matches* table, which included the score, date, as well as every player on either team. A different table included stats on each player, including their birthday and a number of statistics taken from the *FIFA* series of games. The feature vector (X) was constructed as a list of the team attributes as well as the attributes of each player. They were organized as home or away. The y values were one-hot encoding of the winner, i.e. a home team win is represented as [1, 0, 0], an away team win is represented as [0, 0, 1], and a draw is represented as [0, 1, 0].

The benchmark was the bookkeeper odds. Indeed, the database included odds for 10 different bookkeepers, and it was found that these odds are correct 53% of the time. As such, we set 53% as our target goal, with the understanding that if we were to be able to consistently beat the bookkeepers then we would drop out of college and make a living off of sports betting.

Unsupervised Analysis

PCA:
Due to the large number of initial features, the Principal Component Analysis (PCA) model was used to reduce the dimension of the feature vector and thus increase the speed of learning. Using the algorithm in the sklearn.decomposition.PCA library, the number of features was reduced from 730 to 399 features in the minmax X matrix and 472 features in normalized X matrix, while maintaining 95% of the variance in the original 730 features. Due to the superior performance of the minmax X matrix, it was chosen to train all the models in the supervised learning part of the project.

Initially, we attempted to capture 99% of the variance in the original X matrix. However, the dimensionality reduction ability of PCA was significantly reduced. Namely, 590 features are required in the minmax X matrix and 628 features are required in the normalized X matrix. With the significant increase in the amount of features, the slight difference in variance alone cannot justify the increase of the variance threshold to 99%.

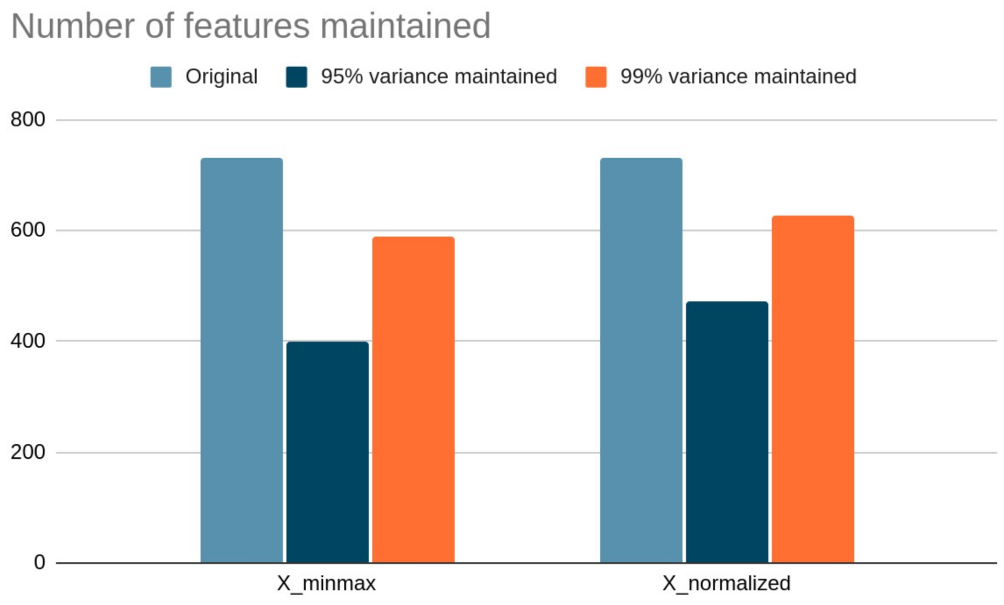

Figure 1: Number of features maintained after running PCA

Supervised Analysis

Logistic regression:
The data was first preprocessed as per described before. A number of models with different combinations of parameters in the sklearn.linear_regression.logistic_regression object were tweaked to test the effectiveness of each model. At first, a model with all the

default settings except near-zero regularization (a high C) was used. Then, different regularization parameters were experimented. Both L1 and L2 regularization attempted, with C=1, C=.1, and C=.0001.

Then, a multinomial algorithm was used instead of the original one-versus-all method; similar experimentation on different regularization parameters also took place, though with L2 regularization only due to the previous similar performance of L1 when compared to L2.

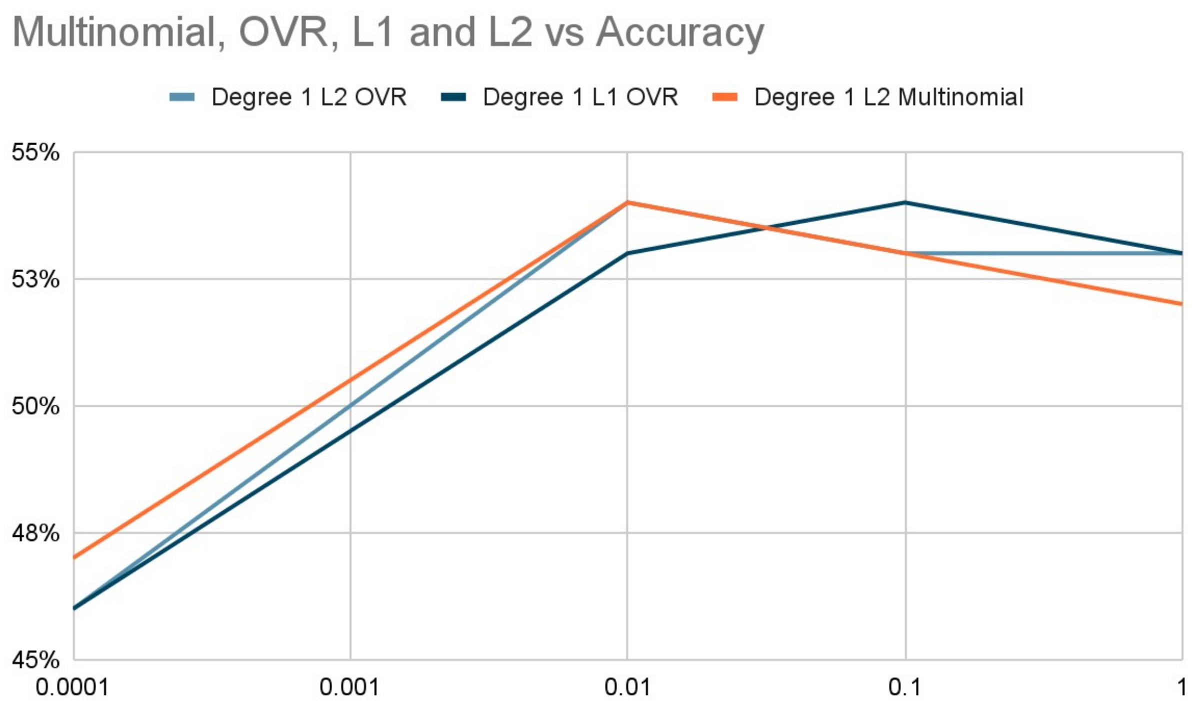

Figure 2: Impact of OVR/ Multinomial and L1/L2 on accuracy of the model

Polynomial feature transformation was also attempted, using the second, third, fourth, and fifth degree. Due to the inherent large number of features in X vector, Nystroem method for kernel approximation was used to speed up the runtime of training the polynomial features. [1] As the multinomial method does not seem to bring too much improvement on the accuracy, OVR was used in all of the polynomial transformations modeling.

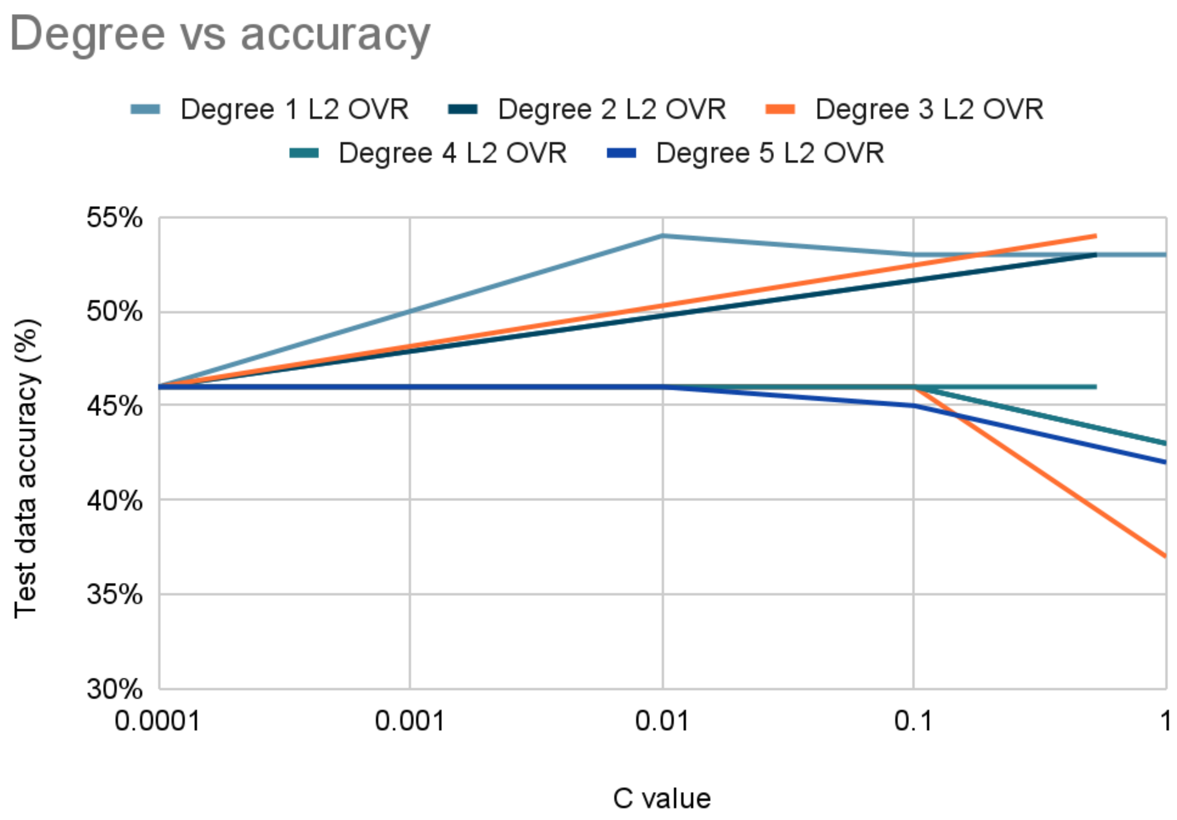

Figure 3: Degree of polynomial transformation vs Accuracy

<u>Support Vector Machine (SVM):</u>
Initially, a linear SVM model with all default settings was used by calling the SVC object from sklearn.svm library. Then, the regularization setting was changed to .01 and .0001 for experimentation. A C value higher than 1 is observed to run significantly longer than its decimal counterparts; therefore, the attempts on C were restrained to only these three numbers in all SVM models. [2]

Kernel SVMs, such as linear, polynomial, and radial-based function (RBF) kernels, were used to attempt to capture non-linear patterns by transforming the feature vector.

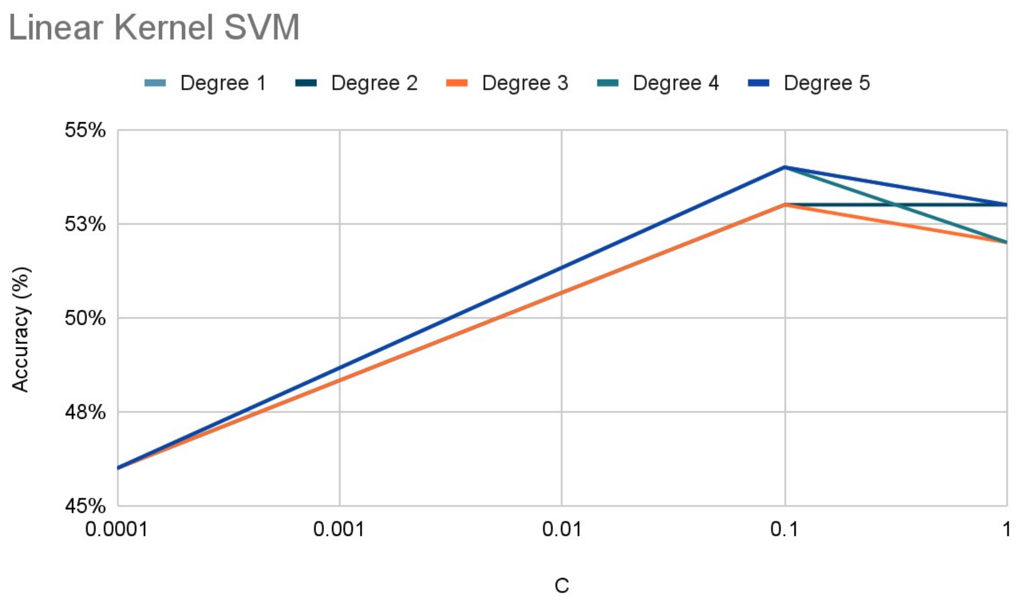

Figure 4: Performance of Linear Kernel SVM

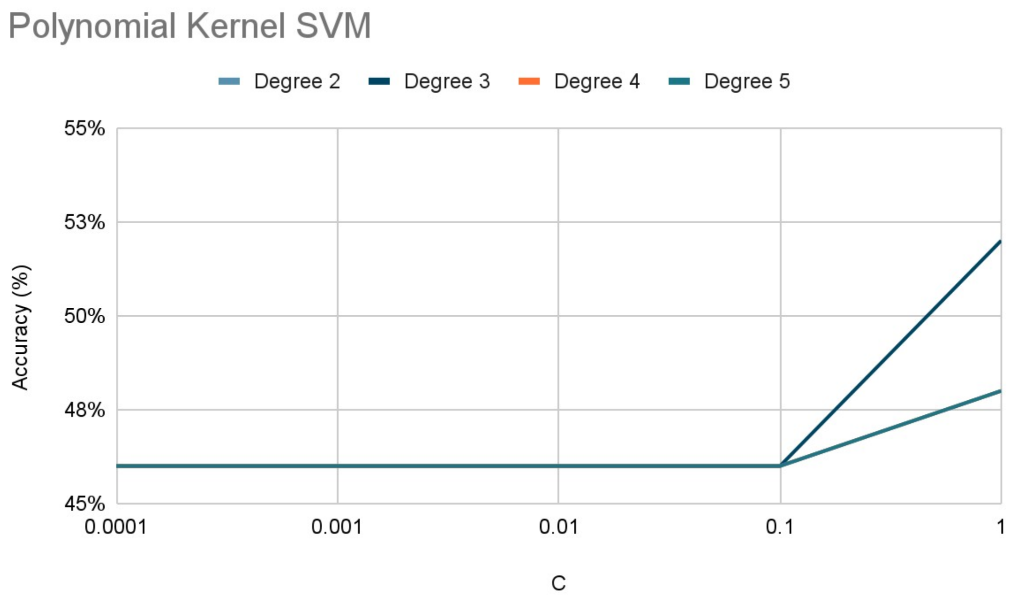

Figure 5: Performance of Polynomial Kernel SVM

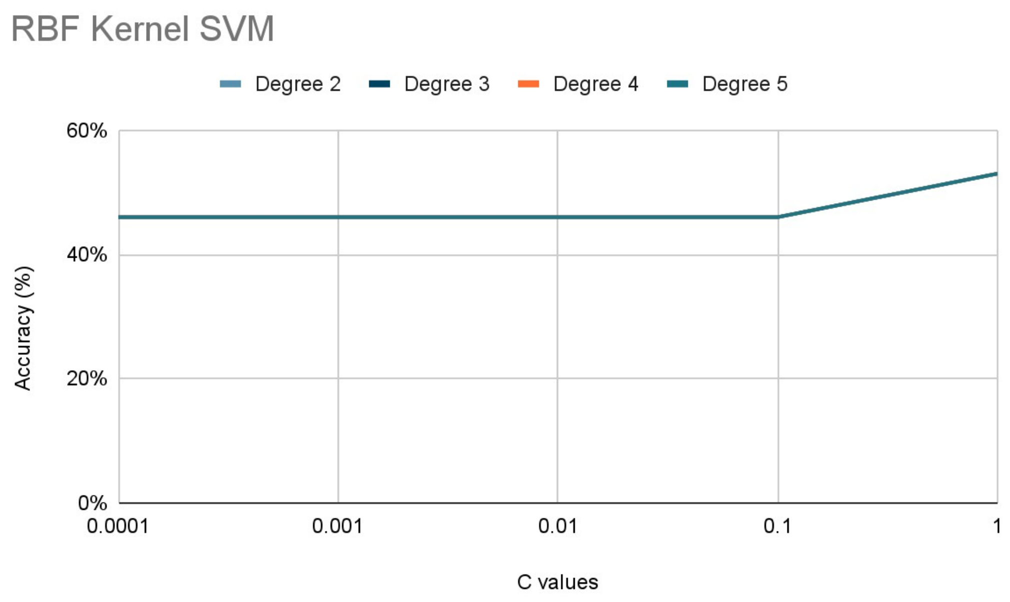

Figure 4: Performance of Linear Kernel SVM

<u>Neural Network:</u>
Initial attempts were made with arbitrary numbers of neurons: namely 100, 16, 32, and 48, as well as arbitrary depths. These results yielded accuracies ranging from 28-40%. After some research, it was found that a good initial approach was to have each hidden layer be the same size as the input vector, and to have layers such that the number of weights was between 2-10 times the number of training examples. [3] However, this proved very difficult as the number of training examples was low relative to the number of weights. Even one hidden layer of 472 neurons led to there being 224200 weights; this was a larger amount than the number of training examples and thus led to a high degree of overfitting. Even by lowering the variance threshold of the PCA and creating a smaller network, it was found that the results are not very good. The best results were found with 5-7 layers and as many neurons per layer as there are features.

Although neural networks are very powerful, it seems that they are not the right tool for this kind of problem, as there are simply not enough data points available.

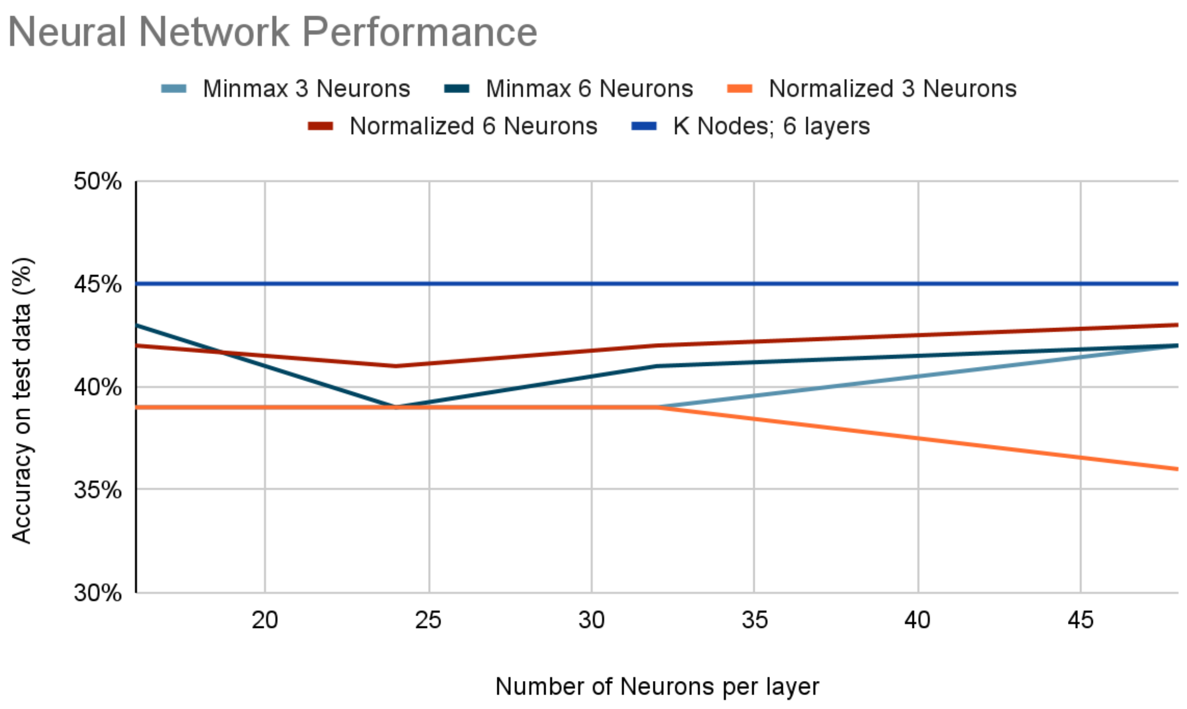

Figure 5: Performance of Neural networks

Results

Table 1: Table of results of Logistic Regression

Logistic Regression					
C	Degree	OVR or Multinomial	L1 or L2	Accuracy % (Training, Testing)	F-score (Training, testing); [home, draw, away]
100000000	1	OVR	L2	55%, 53%	[0.67, 0.15, 0.53], [0.66, 0.10, 0.50]
1	1	OVR	L2	55%, 53%	[0.67, 0.15, 0.53], [0.66, 0.09, 0.50]
.1	1	OVR	L2	55%, 53%	[0.67, 0.10, 0.52], [0.66, 0.08, 0.51]
.01	1	OVR	L2	54%, 54%	[0.67, 0.02, 0.51], [0.67, 0.02, 0.51]

.0001	1	OVR	L2	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
1	1	OVR	L1	55%, 53%	[0.67, 0.13, 0.52], [0.66, 0.09, 0.50]
.1	1	OVR	L1	54%, 54%	[0.67, 0.01, 0.52], [0.67, 0.01, 0.51]
.01	1	OVR	L1	53%, 53%	[0.66, 0, 0.49], [0.66, 0, 0.49]
.0001	1	OVR	L1	46%, 46%	[.63, 0, 0], [.63, 0, 0]
1	1	Multinomial	L2	55%, 52%	[0.67, 0.18, 0.53], [0.66, 0.12, 0.49]
.1	1	Multinomial	L2	55%, 53%	[0.67, 0.15, 0.52], [0.66, 0.10, 0.50]
.01	1	Multinomial	L2	54%, 54%	[0.67, 0.04, 0.52], [0.67, 0.04, 0.52]
.0001	1	Multinomial	L2	47%, 47%	[0.63, 0, 0.07], [0.63, 0, 0.06]
1	2	OVR	L2	53%, 43%	[0.67, 0, 0.50], [0.59, 0, 0.15]
.1	2	OVR	L2	52%, 46%	[0.66, 0, 0.42], [0.63, 0, 0.03]
.01	2	OVR	L2	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
.0001	2	OVR	L2	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
1	3	OVR	L2	54%, 37%	[0.67, 0, 0.51], [0.54, 0, 0.20]
.1	3	OVR	L2	52%, 46%	[0.66, 0, 0.43], [0.63, 0, 0.04]
.01	3	OVR	L2	46%, 46%	[0.63, 0, 0],

					[0.63, 0, 0]
.0001	3	OVR	L2	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
1	4	OVR	L2	54%, 43%	[0.67, 0, 0.51], [0.60, 0, 0.16]
.1	4	OVR	L2	52%, 46%	[0.66, 0, 0.43], [0.63, 0, 0.07]
.01	4	OVR	L2	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
.0001	4	OVR	L2	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
1	5	OVR	L2	54%, 42%	[0.67, 0, 0.50], [0.59, 0, 0.14]
.1	5	OVR	L2	52%, 45%	[0.66, 0, 0.42], [0.62, 0, 0.02]
.01	5	OVR	L2	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
.0001	5	OVR	L2	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]

Table 2: Table of results of SVM

SVM					
Degree	Type of kernel	C	# of support vectors (home, draw, away)	Accuracy % (Training, Testing)	F-score (home, draw, away)
1	Linear	1	[5388 4020 3959]	55%, 53%	[0.67, 0, 0.52], [0.66, 0, 0.48]
1	Linear	0.01	[5116 4020 4033]	54%, 54%	[0.67, 0, 0.51], [0.67, 0, 0.50]

1	Linear	0.0001	[5289 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
2	Linear	1	[5388 4020 3959]	55%, 53%	[0.67, 0, 0.52], [0.66, 0, 0.48]
2	Linear	0.01	[5092 4020 4016]	54%, 53%	[0.67, 0, 0.52], [0.66, 0, 0.49]
2	Linear	0.0001	[5251 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
3	Linear	1	[5403 4020 3928]	55%, 52%	[0.67, 0, 0.53], [0.65, 0, 0.47]
3	Linear	0.01	[5092 4020 4016]	54%, 53%	[0.67, 0, 0.52], [0.66, 0, 0.49]
3	Linear	0.0001	[5251 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
4	Linear	1	[5485 4020 3911]	55%, 52%	[0.67, 0, 0.54], [0.66, 0, 0.46]
4	Linear	0.01	[5098 4020 4029]	54%, 54%	[0.67, 0, 0.51], [0.66, 0, 0.51]
4	Linear	0.0001	[5276 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
5	Linear	1	[5417 4020 3985]	55%, 53%	[0.67, 0, 0.53], [0.66, 0, 0.49]
5	Linear	0.01	[5098 4020 4029]	54%, 54%	[0.67, 0, 0.51], [0.66, 0, 0.51]
5	Linear	0.0001	[5276 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
2	Polynomial	1	[6548 4017 4462]	74%, 48%	[0.79, 0.59, 0.73], [0.63, 0.11, 0.36]
2	Polynomial	0.01	[6482 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]

2	Polynomial	0.0001	[5604 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
3	Polynomial	1	[6875 4020 4513]	89%, 52%	[0.89, 0.83, 0.92], [0.65, 0.06, 0.46]
3	Polynomial	0.01	[6576 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
3	Polynomial	0.0001	[5315 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
4	Polynomial	1	[7287 4020 4595]	84%, 48%	[0.85, 0.79, 0.86], [0.64, 0.03, 0.23]
4	Polynomial	0.01	[7269 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
4	Polynomial	0.0001	[5633 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
5	Polynomial	1	[7278 4020 4596]	82%, 48%	[0.84, 0.76, 0.84], [0.64, 0.01, 0.18]
5	Polynomial	0.01	[7240 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
5	Polynomial	0.0001	[5506 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
2	RBF	1	[6402 4020 4314]	73%, 53%	[0.79, 0.49, 0.75], [0.66, 0.05, 0.49]
2	RBF	0.01	[6089 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
2	RBF	0.0001	[5285 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
3	RBF	1	[6402 4020 4314]	73%, 53%	[0.79, 0.49, 0.75], [0.66, 0.05, 0.49]
3	RBF	0.01	[6089 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]

3	RBF	0.0001	[5285 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
4	RBF	1	[6402 4020 4314]	73%, 53%	[0.79, 0.49, 0.75], [0.66, 0.05, 0.49]
4	RBF	0.01	[6089 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
4	RBF	0.0001	[5285 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
5	RBF	1	[6402 4020 4314]	73%, 53%	[0.79, 0.49, 0.75], [0.66, 0.05, 0.49]
5	RBF	0.01	[6089 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]
5	RBF	0.0001	[5285 4020 4600]	46%, 46%	[0.63, 0, 0], [0.63, 0, 0]

Table 3: Table of results of Neural Networks

Neural Networks					
# of neurons per layer	# of hidden layers	Type of fitting	# of weights	Accuracy %	F-score (home, draw, away)
472	6	Normalized (472 features)	1338120	45%	(0.42, 0.42, 0.41)
399	6	Minmax (399 features)	956403	46%	(0.42, 0.41, 0.41)
16	3	Normalized (472 features)	8112	39%	(0.4, 0.38, 0.39)

16	3	Minmax (399 features)	6944	39%	(0.41, 0.36, 0.39)
16	6	Normalized (472 features)	8880	42%	(0.39, 0.38, 0.39)
16	6	Minmax (399 features)	7712	43%	(0.41, 0.40, 0.40)
24	3	Normalized (472 features)	12552	39%	(0.39, 0.40, 0.40)
24	3	Minmax (399 features)	10800	39%	(0.39, 0.37, 0.38)
24	6	Normalized (472 features)	14280	41%	(0.41, 0.40, 0.40)
24	6	Minmax (399 features)	12528	39%	(0.4, 0.36, 0.38)
32	3	Normalized (472 features)	17248	39%	(0.40, 0.40, 0.40)
32	3	Minmax (399 features)	14912	39%	(0.39, 0.39, 0.39)
32	6	Normalized (472 features)	20320	42%	(0.41, 0.40, 0.40)
32	6	Minmax (399 features)	17984	41%	(0.39, 0.39, 0.39)

48	3	Normalized (472 features)	27408	36%	(0.38, 0.38, 0.38)
48	3	Minmax (399 features)	23904	42%	(0.4, 0.39, 0.4)
48	6	Normalized (472 features)	34320	43%	(0.41, 0.40, 0.41)
48	6	Minmax (399 features)	40816	42%	(0.40, 0.40, 0.40)

<u>Conclusion</u>

The logistic model's accuracy ranged from 37% to 54% in the out-of-sample error. Out of all the samples that reached the highest accuracy, it appears that the f-score of the draw scenario is significantly lower than the others, meaning the accuracy is compensated by the poor performance of predicting draws. This is also a very prevalent trait in most overfitted or over-regularized models, where the f-score would maintain for home win scenarios but the f-score for not only draw, but also away win, which are dropped to zero or near-zero.

SVM models are relatively similar in the behavior of overfitted and over-regularized models of having heavily imbalanced f-scores and consequently a much lower accuracy score than its peers. Importantly, both polynomial and RBF kernel SVM has an in-sample error as high as 89%, with the rest at 70-80% for C=1. However, all of the models only had an out-of-sample 53%, which is commonplace in both SVM and logistic models. It could be improved by perhaps further increasing the C, although it could mean the model will take much longer to run as a higher C costs significantly more time to train. [2]

Two of the three models used to predict the outcome of a match were effective. The only one that was not was a neural network. Many different combinations of layers and nodes were tried, but none could cross 50% accuracy. In some cases, this was likely due to overfitting of the data, but in others, it is likely that the neural network was better able to model noise in the data than the actual outcome. Even when shuffling the data, this still happened. Moreover, even in cases where there were not enough weights for overfitting to happen, the training accuracy was much higher than the test accuracy. The SVM and Logistic regression models were, however, able to reach the 53% threshold of the bookkeeper accuracy.

References

[1] 6.7. Kernel Approximation. (n.d.). Scikit-Learn. Retrieved May 2, 2022, from https://scikit-learn.org/stable/modules/kernel_approximation.html

[2] 1.4.5 Support Vector Machines: Tips on Practical Use — scikit-learn 0.20.3 documentation. (2018). Scikit-Learn.org. https://scikit-learn.org/stable/modules/svm.html

[3] Warren S. Sarle. (2002). comp.ai.neural-nets FAQ. Faqs.Org. http://www.faqs.org/faqs/ai-faq/neural-nets/part1/preamble.html