Proposal
Midterm
Final

Premier League Prediction

1. Intro

The reason billions of people watch, attend, and bet on sports is due to the excitement from their unpredictability. Armed with machine learning, we hope to predict the unpredictable with the largest sports league, the English Premier League.

Our dataset is sourced from kaggle and contains Premier League match data from 2000-2022 season. The dataset includes match details such as teams, goals, and results (both full-time and half-time), along with statistics like shots, fouls, and bookings. It also records match-specific information such as attendance, referee, and key events like offsides, corners, and red/yellow cards for both teams.

Link to dataset: https://www.kaggle.com/datasets/saife245/english-premier-league

Literature Review: In the past, machine learning engineers have approached this problem using various methods such as statistical analysis, data mining, and Bayesian analysis [1]. Over time, static Bayesian networks evolved into dynamic ones that incorporated time-variable factors like team strength [3]. This included numerous metrics relating to both offense and defense. More complex models, like the FRES system, which combined rule-based systems with Bayesian networks, were developed to produce more realistic results [1]. Machine learning engineers were able to achieve an accuracy of 75.09% when predicting 2010-2013 English Premier League match outcomes, outperforming earlier models like decision trees and k-nearest neighbors [2][3]

2. Problem

The problem we are trying to solve is to predict outcomes of matches using only team statistics. The motivation is that many public datasets do not incorporate individual player statistics, and we hypothesize that team play is as important towards match outcomes as individual performances. This will be impactful because it can completely change our understanding of team/player dynamics and sports predictions.

3. Methods

Data

On the data side, since we only have per-match data, we decided it would be best to use match data features grouped by each teams previous results. We defined previous results as rolling match statistics with a few set window sizes. After some trial with different window sizes, rolling windows of size 3, 5, 10, 20, and 50, to capture both short term and long term trends among the teams. Each match statistic was aggregated for each team based upon their perspective, so for example, in a Home vs. Away team match, we would have statistics for the home team's stats, the home team's historical opponents' stats, the away team's stats, and the away team's historical opponents' stats. After this point, we refined the data by clipping outliers, and using PCA to heavily reduce dimensionality while maximizing variance to increase model performance.

The data used was captured from the dataset in the introduction, where we used data from the Premier League 2000-2001 through the 2021-2022 seasons.

Modeling

We have implemented three models. First, we implemented a simple linear model and a basic 2-layer neural network with Relu activation in the middle, and sigmoid at the end using Pytorch. Inspired by previous projects, we automated the training pipeline so that it first scaled the data using the

RobustScaler module to stabilize the data for the models, then split the data in a 5/12.5/12.5 train/validate/test split, and trained with cross-validation and automated hyperparameter search via Bayesian optimization. Our loss function is also unique, with class-balanced focal loss, using a concept of class reweighting by effective class size, which helped balance out the class imbalance inherent in homefield advantage. Additionally, a decrementing learning rate schedueler was added with Adam optimization to improve training. This helped reduce the amount of work in the training process and allowed us to quickly find the best hyperparameters, particularly for the 2-layer neural network.

Finally, for the third model, we tried a different approach with a Random Forest. Since Pytorch doesn't have an easily made Random Forest implementation, we used the scikit-learn module RandomForest. This meant that we could not use our previously mentioned pipeline, requiring manual hyperparameter tuning. We tried various combinations of hyperparameters here, finding that ~100 estimators was the sweet spot, as shown in the confusion matrices below.

4. Results & Discussion

Looking at the performace of our models, we can see the pros and cons of using the neural network over the random forest and vice versa. The best way that we can compare these models is by using various visualizations such as the loss graph and the confusion matrices of both methods.

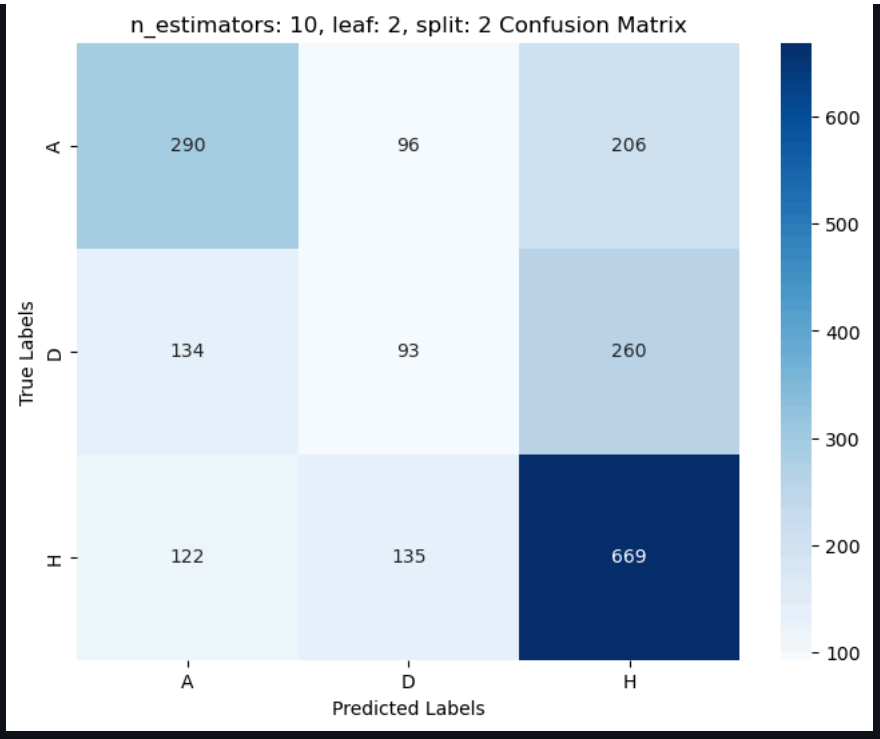
Starting with the neural network from our midterm, we can see that our train loss started to reduce drastically for the first 20 epochs, after which it generally plateaued at a little less than loss=0.6. It is important to note that this loss is still rather high. Looking at our testing loss, the values reduced for the first 5 epochs after which it plateaued at around 0.9 which is much higher than our train loss. This can signify overfitting, which was also a problem in our midterm checkpoint. To combat this we tried to add better preprocessing methods to utilize more of our data features, however the overfitting problem still persisted.

Looking at the Random Forest approach, this approach was chosen due to the rather poor performace of the neural network. When looking at the performance of this machine learning model in comparison to the neural network, we can use the confusion matrix. The confusion matrix of the random forest shows that the model is able to predict most of home game wins correctly and away game wins correctly. However, it was very poor on estimating draws.

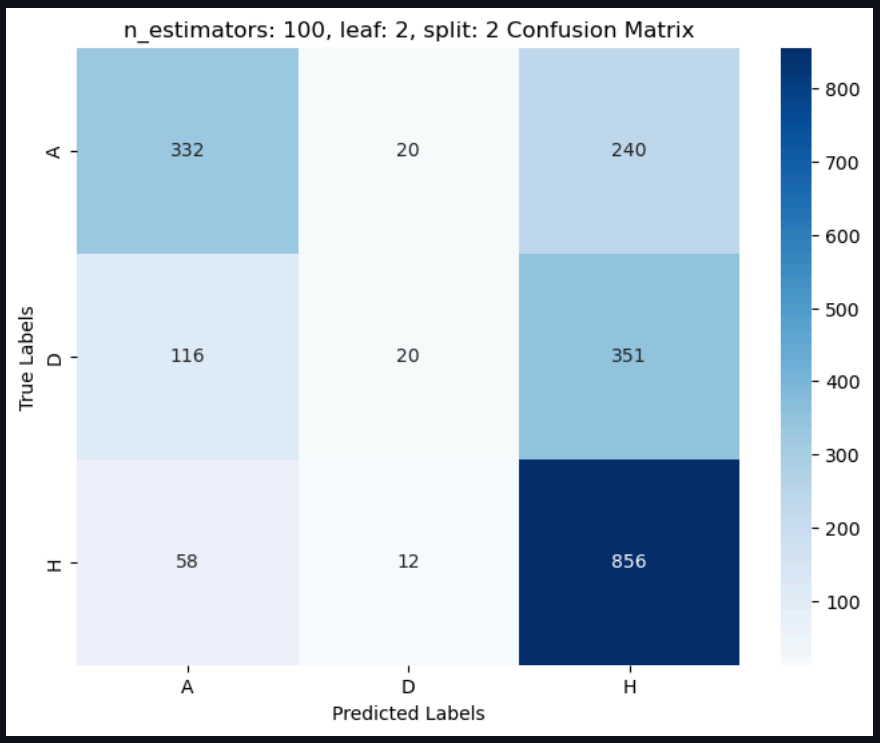
Lastly, looking at the linear classifier, we can see similar problems to the neural network, however this model works a bit worse than the neural network. The training loss is around 0.65 and the test loss is slightly below 0.9. The simlarities in the results may be attributed to the similarity of the models, as a linear classifier is essentially a one layer neural net while our neural network is a 2 layer neural net. Due to both of these models being rather shallow, the results are quite similar.

Comparing the random forest's confusion matrices with the neural network's and linear classifier's confusion matrices shows that the random forest was better at predicting home wins and away wins but was worse than the neural networks at predicting draws. This can potentially be attributed to the method in which each of these algorithms work. The random forest works by using decision trees which might have a harder time detecting the subtlties of a draw, whereas a neural network may find these patterns better than a decision tree, but have a harder time making the decisions of wins. In addition, the random forest would be more robust to class imbalances which do exist in our data.

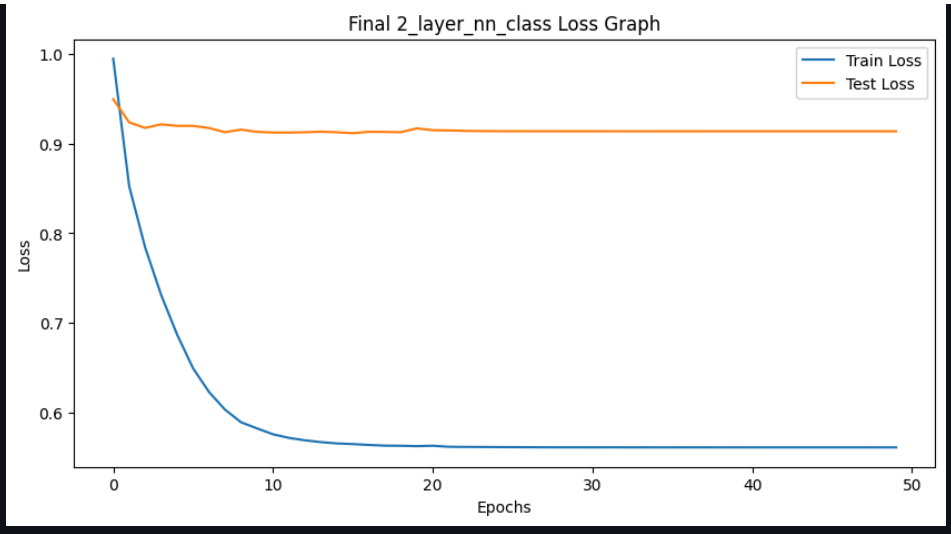
Visualizations



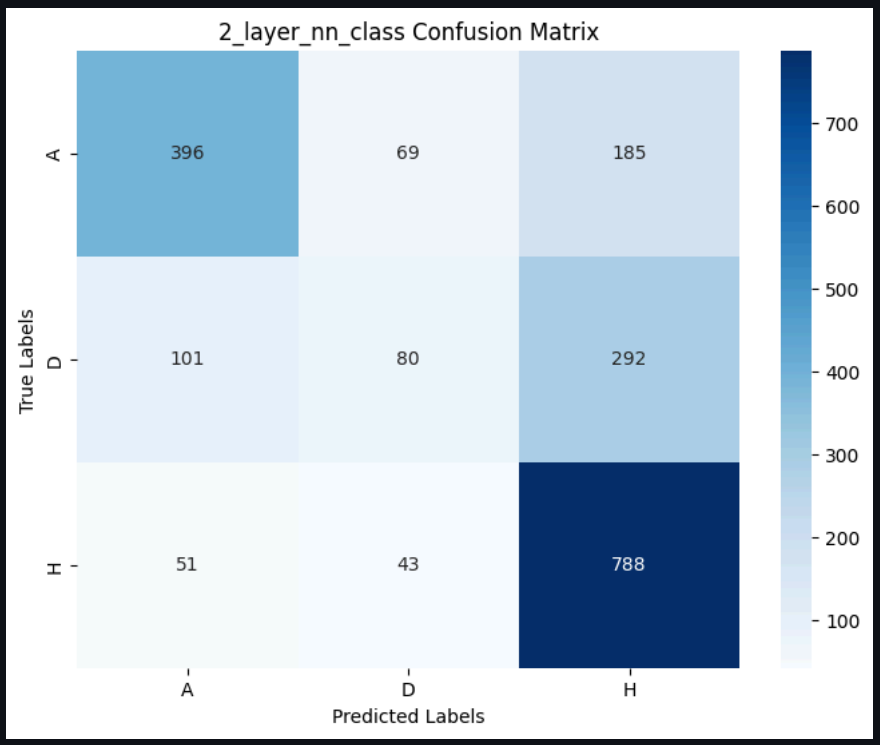
Worst Random Forest Confusion Matrix (before hyperparameter tuning)



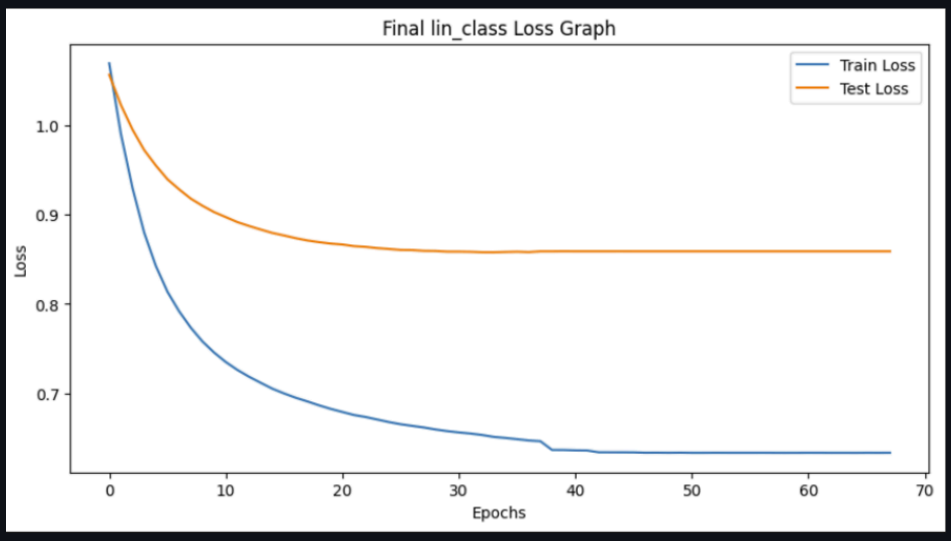
Best Random Forest Confusion Matrix (after hyperparameter tuning)



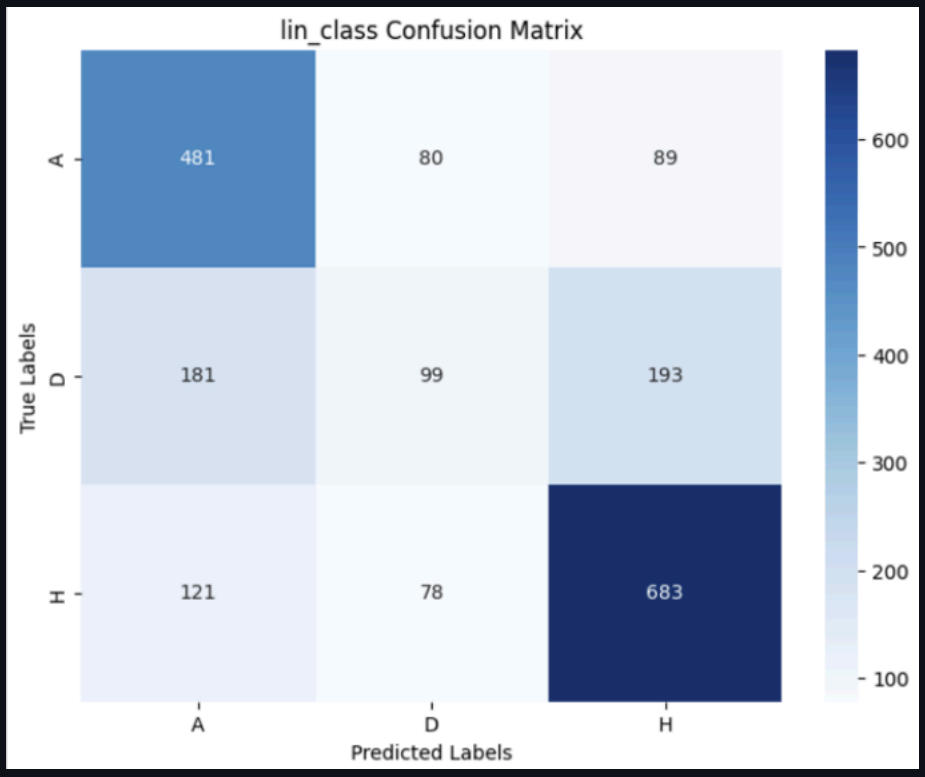
2-Layer Neural Network Loss Graph



2-Layer Neural Network Confusion Matrix



Linear Classifier Loss Graph



Linear Classifier Confusion Matrix

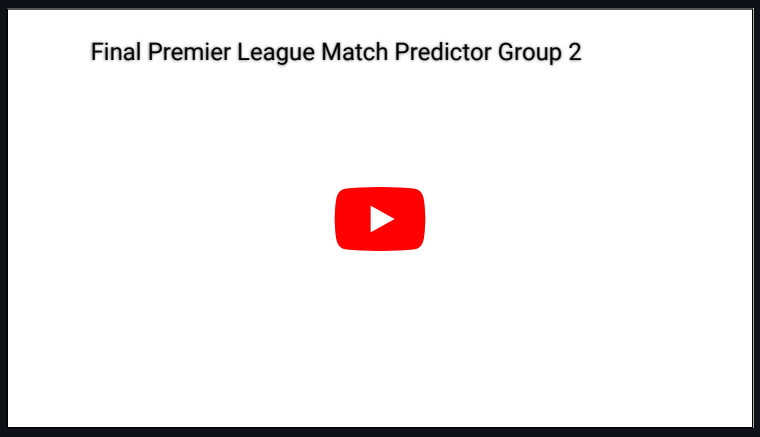
5. References

[1] R. Baboota and H. Kaur, "Predictive analysis and modelling football results using Machine Learning Approach for English Premier League," International Journal of Forecasting, https://www.sciencedirect.com/science/article/abs/pii/S0169207018300116 (accessed Oct. 1, 2024).

[2] S. Rana, D. Vasudeva, and Amol, "Digital Library Home: Information, theory, coding and cryptography: Solution mannual; 2nd ed.," Premier League Match Result Prediction using Machine Learning, http://www.ir.juit.ac.in:8080/jspui/handle/123456789/5644 (accessed Oct. 1, 2024).

[3] N. Razali1, A. Mustapha1, F. A. Yatim2, and R. A. Aziz1, "IOPscience," IOP Conference Series: Materials Science and Engineering, https://iopscience.iop.org/article/10.1088/1757-899X/226/1/012099/meta (accessed Oct. 1, 2024).

6. Video



7. Planned Responsibilities

Final Gantt Chart ML Group 2 : Fall

GANTT CHART

PROJECT TITLE	Premier League Match Predictor

					PHASE ONE									
					Sep 30									
TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	M	T	W	R	F	S	U	M	T	W
Project Proposal														
Introduction	Patrick	10/1/2024	10/4/2024	4		x	x	x	x					
Problem Definition	Pranav	10/1/2024	10/4/2024	4		x	x	x	x					
Methods	Will	10/1/2024	10/4/2024	4		x	x	x	x					
Potential Results & Discussion	Sid	10/1/2024	10/4/2024	4		x	x	x	x					
References	Jayden	10/1/2024	10/4/2024	4		x	x	x	x					
Video Recording	Jayden	10/1/2024	10/4/2024	4		x	x	x	x					
GitHub Page	Sid	10/1/2024	10/4/2024	4		x	x	x	x					
Model 1														
Data Sourcing and Cleaning	All	10/7/2024	11/8/2024	5								x	x	x
Model Selection	All	10/7/2024	11/8/2024	5										
Data Pre-Processing	Jayden	10/7/2024	11/8/2024	5										
Model Coding	Will	10/7/2024	11/8/2024	5										
Results Evaluation and Analysis	Pranav & Sid	10/7/2024	11/8/2024	5										
Midterm Report	All	10/7/2024	11/8/2024	8										
Model 2														
Data Sourcing and Cleaning	All	11/22/2024	12/3/2024	3										
Model Selection	All	11/22/2024	12/3/2024	1										
Data Pre-Processing	Will	11/22/2024	12/3/2024	2										
Model Coding	Will	11/22/2024	12/3/2024	2										
Results Evaluation and Analysis	Pranav	11/22/2024	12/3/2024	2										

Final Contribution Table ML Group 2 : Sheet1

	Name	Proposal Contributions
	Jayden Causey	Visualizations and Graphs
	Patrick Guo	Gantt chart, Contribution table, README, Presentation, Recording
	Pranav Kuppili	Analysis of Results and Discussion
	Will Lash	Algorithm implementation, Away Team Preprocessing
	Sid Parikh	Decision Tree Classifier

Sheet1