# Background

The UEFA Champions League is a yearly international club soccer tournament featuring the best teams in Europe. Several machine learning models have been used to predict the results of sports games. Bunker and Thabtah [1] found that using an ANN with backpropagation performs with a slightly higher accuracy than expert predictions, beating ESPN sportscasters 75% to 63% accuracy when predicting NFL games. According to Zhang et al [2], RNNs can be unable to capture long-term dependencies. They recommend the use of LSTM, a variant of RNN, to get more accurate results. Yue et al [3] discuss the use of a Bayesian approach called GLICKO to predict strength of tennis players and therefore outcomes of tennis matches. In our particular use case, the unique format of the Champions League allows for more complex predictions of team knockout stage performance based on domestic league and group stage performance.

# Problem

We want to create a model that accurately predicts the team knockout stage. Despite soccer being the most popular sport worldwide, more money is put into analytics for the NBA and NFL in the United States. The field of predicting soccer matches is scarce. Additionally, since UEFA only has European teams, it receives less resources than other soccer events such as the World Cup. An accurate model could help teams allocate their resources, knowing which factors are important for a winning team. It can also allow a team to see where weaknesses lie, allowing them to improve on their deficiencies.

# Data Sourcing, Cleaning, and Processing

We obtained our data from [UEFA](https://www.uefa.com/uefachampionsleague). Using this resource, we collected data from 1981 to 2021 on club teams and how they finished (winner, runner-up, etc.), points scored by the winner of the game, and points scored against the winner of the game. We also took note of the location of the game, but we did not use that data in the first round. We did use this data for our second model. In addition, we obtained country coefficients for each team that have proven to be a good predictor of team performance. We used these coefficients for our data in the second model. After doing this, we cleaned the data, as some years used a point system similar to what is used in World Cup brackets. We also filled in some missing information and made the years uniform. Once we finished cleaning the data, we moved on to pre-processing. In order to fit the data to our model, we manipulated the scores, changing it from points scored for each team to the point difference between teams. This allows our model to predict the point difference when given 2 teams without knowing the winner in advance. Then, we parsed the team names and converted them into numerical values that our model could understand. We also input every team’s home city and used this in conjunction with our data to determine whether a match was home or away. For our first model, we split the training and testing data randomly. For our final model, we instead split the data by time, saving the most recent years of the tournament for testing.

# Methods

For our first model, we implemented a Recurrent Neural Network (RNN). RNNs excel at time series analysis (predicting future results based on past data) because they are able to capture temporal dependencies. This method of implementation should be able to provide significant outputs based on our training data. In order to implement the RNN, we utilized the TensorFlow library. The library has pre-existing architecture for implementing a simple RNN, GRU, and LSTM, so we can alter the type of neural network we’re using in future implementations without significantly altering the code or data. This allows us to get initial results in our first iteration while maintaining flexibility for later iterations.

Our second model used an RNN implemented with TensorFlow, for the same reason we used it for the first model. We used two LSTM layers with batch normalization, making some changes to the model and data input to get more accurate predictions. First, we added an LSTM layer and increased the size of the LSTM layers in our model to better handle our more complex data. We also added batch normalization to stabilize the training process. Then, when handling the data, we normalized the features to prevent a small subset of the features from dominating all the others. These additional layers and normalization improved the training process for the new model, and allowed it to make more advanced predictions rather than the +/- 1 goal predictions from the first iteration.

# Results

## Model 1

In our first model, we predicted the score differential primarily through past games. Our mean squared error on our training set was 0.9976, with the loss per epoch being around 0.999 every iteration. The model yielded similar results on the testing data, which was obtained through a random sample. In future iterations, we hope to use the most recent years as our testing data and past years for training.

Graph 0

Figure 0

To tune our hyperparameters, we looked at the affect of different values on our model. As seen in Figure 0, the training loss decreases drastically after Epoch 15 while the validation loss increases. This is an indication of overfitting. So, along with some trial and error, we decided on 15 epochs.

Graph 1

Figure 1

Graph 2

Figure 2

Based on the statistics obtained, it does seem that the past performance has a strong indication on how a team will perform in the future. This is somewhat demonstrated through a plot of the original data. As shown in Figure 1, the same teams tend to appear every year in the tournament, and those same teams tend to win.

Graph 3

Figure 3

Figure 3 shows the relative accuracy of our model. We used our model to predict the results of matches, matching up each team with all of the other teams. We recorded the number of times our model predicted a win for the each team, and scaled the wins to reflect the average number of matches a team played in our dataset. As seen in Figure 3, our model predicted similar results to the actual results for about half of the teams. Teams such as AC Milan, Bayern Munich, Liverpool, and Marseille were drastically different.

However, the model does need improvements. Currently, the model is able to predict the score differential when two teams are inputted, but that score is typically within +/- 1 goal. With future iterations, we hope to investigate our normalization of features and create a model which will produce scores similar to those seen in a real soccer match. In addition, the accuracy of the model is held back by its limited input features.

## Model 2

**## ADD model 2 analysis and comparison here**

Graph 4

Figure 4

Figure 4 shows the relative accuracy of our second model. We used our model to predict the results of the knockout stages in 2020 and 2021. Since this data was left out of training our model, this information was new to our model. We fed our model the match-ups of the 2020-2021 knockout stages and recorded it’s predictions. As can be seen in Figure 4, our new model is much more accurate than our previous one. Every prediction from our model was within +/- 1 win of the real results, with the exception of Liverpool. This is most likely because Liverpool has historically performed extremely well in the EUFA Cup, even up to recent years. So, it’s lack of wins in 2020 or 2021 was not predicted by the model.

If we were to continue this project, there are a few more factors we would consider. This includes looking into the amount of players that are returning from the previous season, the distance each team travels to the stadium, and individual games from the season.

## Conclusions

**##ADD conclusions**

## Contributions

| **Task** | **Primary Team Member** | **Secondary Team Member** |
| --- | --- | --- |
| Model Selection | Jakob |  |
| Data Sourcing/Cleaning | Kenny | Andrew |
| Data Pre-Processing | Erin | Jakob |
| Model Coding | Jakob | Mary-Kate |
| Data Visualizations | Mary-Kate | Kenny |
| Model Analysis | Erin |  |
| Video Presentation |  |  |
| Website Updates | Mary-Kate |  |

## 