Assignment 2 - Technical Report

Chenglong Li

1 Introduction

Expected Assist (xA) is an analytic measure in soccer that evaluates the quality of a pass by estimating the likelihood that it will result in a goal, known as the expected goals (xG) of the subsequent shot. xA gauges a player's contribution to creating scoring opportunities, irrespective of whether the shot ultimately results in a goal. This metric is crucial for assessing a player's effectiveness in orchestrating shots, providing a nuanced perspective on their role in the team's offensive strategy. By leveraging xA, we can more accurately evaluate players' performances and strategic approaches to offense, which aids in player development, game planning, and talent scouting. For my analysis, I utilized **2018 Premier League** data and employed a **neural network** model to compute xA values. I pinpointed the top five players with the highest xA over the match and also identified the player with the highest xA in other matches. The model code is posted at github repository.

2 Data

2.1 Dataset Introduction

The data refer to season 2017/2018 of five national soccer competitions in Europe: Spanish first division, Italian first division, English first division, German first division, French first division. In addition, it includes data from the World Cup 2018 and the European Cup 2016, which are competitions for national teams. The dataset is saved in **JSON format**, and in the 2018 Premier League dataset, the first entry and the meaning of each label are shown in Table 1.

2.2 Feature engineering

Expected Assist (xA) assesses the quality of a pass by estimating the likelihood that it leads to a goal. To calculate the xA value, I analyze data from the pass before the shot (ball B) and the shot itself (ball A), including their positions, shooting distance, and angles, as well as the distance and angle between the two balls. These parameters are used as features in the model, with the success of the shot at scoring a goal serving as the target.

Table 1: Dataset Entry Description

Label	Data	Description
eventId	8	Identifier for the type of event.
subEventName	Simple pass	Type name of the subevent.
tags	`['id': 1801]`	List of tags providing additional event details.
playerId	25413	Identifier for the player associated with the event.
positions	`['y': 49, 'x': 49, 'y': 78, 'x': 31]`	Start and end positions of the event on the field.
matchId	2499719	Identifier for the match containing the event.
eventName	Pass	Type name of the event.
teamId	1609	Identifier for the player's team.
matchPeriod	1H	Match period when the event occurred.
eventSec	2.758649	Time in seconds when the event occurred during the match.
subEventId	85	Identifier for the type of subevent.
id	177959171	Unique identifier for the event.

The shooting distance is calculated according to the following formula

$$\text{Distance} = \sqrt{x^2 + c^2}$$

c represents the vertical distance between the shot and the midline of the pitch (Y=50).

The shooting angle is calculated according to the following formula

$$\theta = \begin{cases} \arctan\left(\frac{7.32 \cdot x}{x^2 + c^2 - (7.32/2)^2}\right) & \text{if } \arctan\left(\frac{7.32 \cdot x}{x^2 + c^2 - (7.32/2)^2}\right) > 0 \\ \arctan\left(\frac{7.32 \cdot x}{x^2 + c^2 - (7.32/2)^2}\right) + \pi & \text{otherwise} \end{cases}$$

The distance between A and B is calculated using the following formula

$$\text{A\_B\_Distance} = \sqrt{(A_x - B_x)^2 + (A_y - B_y)^2}$$

The directional angle from ball B to A is given by

$$\text{B\_A\_Angle} = \arctan 2(B_y - A_y, B_x - A_x)$$

To handle the features above, I use **StandardScaler** normalisation with the following formula

$$z = \frac{(x - \mu)}{\sigma}$$

where x is the original data, μ is the sample mean (average) and σ is the sample standard deviation.

Based on the feature engineering above, I have obtained the model's input features and the model's target, as shown in the Table 2.

Table 2: Features and Target of Model

Features	$A_x, A_y, A_c, B_x, B_y, B_c$, Shot_Distance, Shot_Angle, A_B_Distance, B_A_Angle
Target	Goal (Probability of Goal = 1)

3 Method

To calculate the value of xA, I used a simple feedforward neural network. This neural network essentially consists of two layers of linear processing units, each enhanced with an activation function to add non-linear processing capability, enabling it to handle and learn complex data patterns.

A FNN is formed by one input layer, one or more hidden layers and one output layer. Each layer of the network is connected to a following layer. The structure of FNN is shown in Figure 1.

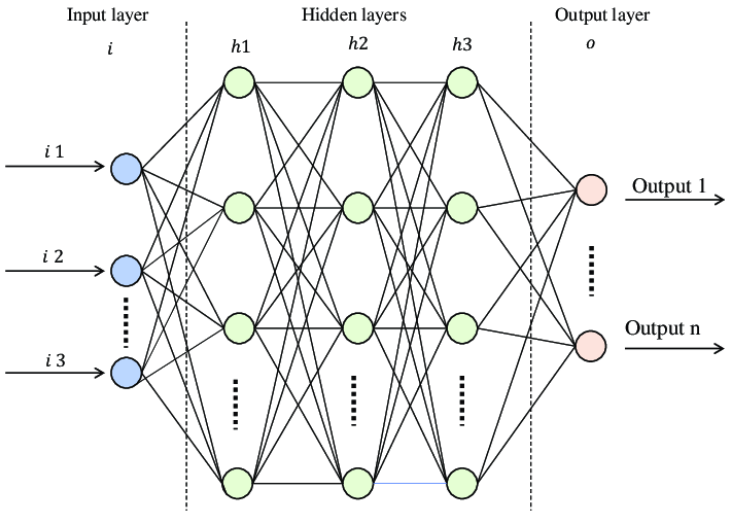

Figure 1: The architecture of a Feedforward Neural Network (FNN)

A forward feedback neural network processes data through multiple layers: an input layer that receives the data, several hidden layers that transform it using weights and activation functions, and an output layer that produces the final results.

In an FNN, each layer performs a linear transformation by multiplying the input by a weight matrix and adding a bias, followed by applying a non-linear activation function to introduce complexity and enable the network to learn non-linear patterns. Assuming an FNN has L layers, with $n^{(l)}$ neurons in layers l, and the input data is $\mathbf{x}$, the computation at layer l is represented as

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$$

where:

- $\mathbf{z}^{(l)}$ represents the linear combination result at layer l.
- $\mathbf{W}^{(l)}$ is the weight matrix connecting layer $l-1$ to layer l, with size $n^{(l)} \times n^{(l-1)}$.
- $\mathbf{a}^{(l-1)}$ is the output of layer $l-1$.
- $\mathbf{b}^{(l)}$ is the bias vector for layer l, with size $n^{(l)} \times 1$.

Next, the activation function is applied to obtain the output of the layer

$$\mathbf{a}^{(l)} = f(\mathbf{z}^{(l)})$$

where f represents the activation function, such as ReLU, Sigmoid, or Tanh.

The final output is calculated by passing the value through the activation function of the output layer. For classification tasks, the Softmax activation function is commonly used to generate a probability distribution

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{z}^{(L)})$$

4 Experiment

As a metric in soccer data analysis, the Expected Assists (xA) value quantifies the scoring probability of each assist action. In this experiment, I employ a Deep Neural Network (DNN) model to predict the probability of scoring for each shot. This choice is motivated by the significant advantages of deep neural networks in handling complex nonlinear issues, particularly their suitability for processing high-dimensional and structured data, such as the combination of factors like player positions, states, and opposition quality in soccer matches.

In terms of model design, I treat the outcome of each shot as a binary classification task: "1" represents a goal, while "0" indicates no goal. By training the neural network to identify and learn the key features that lead to scoring, the model outputs a probability value between 0 and 1, indicating the likelihood of scoring under the given offensive and defensive context. This probability is used as the predicted xA value, which provides a quantified standard to assess the quality and potential scoring outcome of each assist.

4.1 Experiment Design

In this experiment, data from the 2018 English Premier League season was utilized, specifically focusing on shot data and the events immediately preceding each shot. Through feature engineering, a total of 8,451 data points were derived from the datasets. For the purposes of training the Neural Networks model, the dataset was partitioned into three distinct subsets: 60% was allocated as the training set, 20% served as the validation set, and the remaining 20% was used as the test set.

For the neural network model the relevant settings of the parameters are given in the following Table 3.

Table 3: Neural Network Configuration

Parameter	**Value**	**Description**
Input Size	12	Number of input features
Hidden Units 1	64	Number of units in the first hidden layer
Hidden Units 2	16	Number of units in the second hidden layer
Output Size	1	Number of output unit (binary classification)
Epochs	70	Number of times the entire dataset is processed
Batch Size	10	Number of samples in each batch
Learning Rate	0.001	Step size at each iteration
Loss Function	BCELoss	Binary Cross-Entropy Loss, for binary classification tasks

4.2 Results

I collected data through this experiment and generated two figures to characterize the model's performance. Figure 2 displays the variation in accuracy across different training iterations, evaluating the model's learning progress and stability. Figure 3 is a ROC curve, which provides an intuitive view of the model's ability to distinguish between positive and negative samples by depicting the relationship between true positive rate and false positive rate.

Based on Figure 2 and Figure 3, we know that the training accuracy begins at approximately 89.5% and gradually increases, eventually stabilizing around 90%, which indicates effective learning from the training data. However, the consistently higher training accuracy compared to the validation accuracy throughout the training process suggests the presence of overfitting. The ROC curve illustrates the trade-off between sensitivity (true positive rate) and specificity (1 - false positive rate) at different threshold settings. The curve's position significantly above the diagonal (which represents a random classifier) demonstrates that the model possesses a good level of separability.

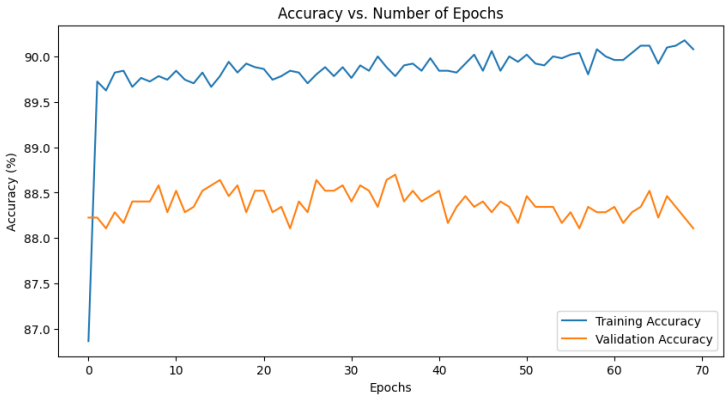

Figure 2: Training iteration versus accuracy performance

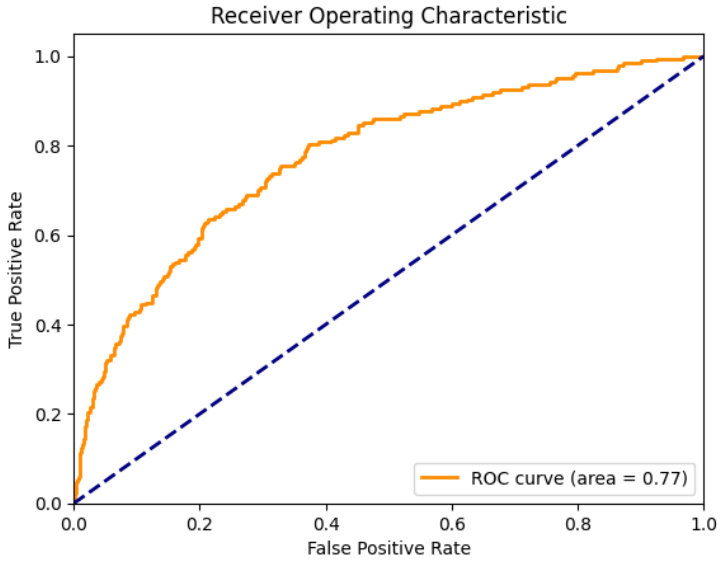

Figure 3: ROC curve

Using the trained model, I identified the top 5 players in terms of xA in the 2018 Premier League. In addition, I applied the model to other leagues to identify the players with the highest xA in each league. The resulting rankings are summarised in the Table 4 and the Table 5.

Table 4: Top 5 Premier League Players by xA in 2018

Name	**xA**
Kevin De Bruyne	9.120857
Mohamed Salah Ghaly	8.519657
Christian Dannemann Eriksen	8.06708
Riyad Mahrez	7.703627
Wilfried Zaha	7.500452

Table 5: Highest xA players in each league

Name	**League**	**xA**
Lionel Andrés Messi Cuccittini	Spain	14.410911
Florian Thauvin	France	11.362547
Alejandro Darío Gómez	Italy	9.349335
Kevin De Bruyne	England	9.120857
Philipp Max	Germany	7.235979
Lionel Andrés Messi Cuccittini	World Cup	2.432224
Romelu Lukaku Menama	European Championship	2.369976

5 Conclusion

Using data from the 2018 Premier League, a simple neural network model was trained to calculate each player's total xA for that season. After the model was trained, it was also applied to players from other leagues to calculate their xA. Ultimately, this resulted in the top five players in terms of xA in the Premier League in 2018, as well as the players with the highest xA in other leagues.