ADAM MICKIEWICZ UNIVERSITY IN POZNAŃ
FACULTY OF SOCIAL SCIENCES
KOGNITYWISTYKA

Gracjan Popiółkowski

Prediction Using Artificial Neural Network Models

Poznań 2023

Contents

Introduction

In this paper, I will present the results of training an artificial neural network with various parameters, which I will use to predict the outcome of a match based on data from the first 15 minutes of the popular game "League of Legends". The first 15 minutes of the game represent a critical phase, which raises the following questions: How accurately can we predict the match outcome based on key statistics of both teams? Are the first 15 minutes decisive?

CHAPTER 1

Data Presentation

1.1. Dataset

I am using the "League of Legends Diamond Games (First 15 Minutes)" dataset, made available by Ben Fattori on the Kaggle platform. Access to this dataset was provided by the game developers, Riot Games. The data was collected from nearly 50,000 matches of diamond-ranked players (one of the highest ranks) on the "NA1" server and includes key statistics from the first 15 minutes of gameplay.

1.2. Data Description

Of the 19 columns in the dataset, 4 were removed prior to further analysis:

- `id` (data number) - could interfere with the functioning of ML algorithms.
- `matchId` (individual match identifier) - has no influence on gameplay.
- `blueDragonKills` (how many dragons were killed by the blue team)
- `redDragonKills` (how many dragons were killed by the red team) - the value of this variable is 0 for both teams in every match, making it irrelevant.

Further analysis includes 15 variables, which are described below:

- `blue_win` - a categorical variable with values of 0 and 1. 1 if the blue team won, and 0 if the red team won.

The remaining columns are numerical variables:

- `blueGold` - the total amount of gold earned by the blue team, consisting of 5 players.
- `redGold` - the same for the red team.
- `blueMinionsKilled` - the number of minions killed by the blue team.
- `redMinionsKilled` - the same for the red team.
- `blueJungleMinionsKilled` - the number of jungle minions killed by the blue team's jungler.
- `redJungleMinionsKilled` - the same for the red team.
- `blueAvgLevel` - the average level of the blue team's champions at 15 minutes of gameplay.
- `redAvgLevel` - the same for the red team.
- `blueHeraldKills` - the number of Herald monsters killed by the blue team.

- `redHeraldKills` - the same for the red team.
- `blueTowersDestroyed` - the number of towers destroyed by the blue team.
- `redTowersDestroyed` - the same for the red team.
- `blueChampKills` - the total number of enemy champions killed by the blue team.
- `redChampKills` - the same for the red team.

Here are the data statistics for the blue team:

Variable	Mean	Std	Min	Mdn	Max
blueGold	26349.16	2777.30	15930.00	26167.00	39769.00
blueMinionsKilled	335.86	32.67	120.00	337.00	455.00
blueJungleMinionsKilled	83.14	14.79	0.00	83.00	164.00
blueAvgLevel	9.17	0.41	5.40	9.20	10.80
blueChampKills	11.01	4.53	0.00	11.00	50.00
blueHeraldKills	1.32	0.96	0.00	1.00	4.00
blueTowersDestroyed	0.79	1.05	0.00	1.00	11.00

Table 1.1: Mean represents the average, std - standard deviation, min - the minimum value, mdn - median, and max - the maximum value.

1.3. Data Preparation

As mentioned earlier, 4 data columns were removed. The categorical variable is the `blue_win` column. Additionally, I performed data standardization on all 14 numerical features, transforming them so that they have a mean value of 0 and a standard deviation of 1. This can improve the efficiency of the machine learning algorithms I used.

CHAPTER 2

Artificial Neural Network

`Artificial Neural Network` is a model inspired by the structure of the brain, composed of units called neurons, which are interconnected. Neurons are organized into layers, and each connection between them has a weight. The learning process involves adjusting these weights to allow the model to more accurately predict outcomes. The parameters we can influence in artificial neural networks include:

1. `Training set size` - determines the ratio of the training set to the test set, defining how much data is allocated for training the network versus testing its performance. In my analysis, the training set size is set to 80% (38,920 observations), leaving 20% (9,731 observations) for testing.

2. `Number of layers and neurons` - determines the structure of the neural network. More layers and neurons can allow the model to better understand complex relationships, but it also increases the risk of overfitting.

3. `Activation functions` - define how activations are passed through the neurons. Activation functions introduce non-linearity into the model, enabling it to model more complex relationships in the data.

4. `Loss function` - a parameter that defines how the model measures its performance during training. The loss function is minimized during training, and its choice depends on the type of problem (e.g., binary classification, multi-class classification, regression).

5. `Optimizer` - an algorithm used to update the weights during training. Optimizers control how quickly the model learns and how well it adjusts to the training data.

6. `Performance metrics` - metrics used to evaluate the model's performance. For example, accuracy measures the percentage of correctly classified examples.

7. `Batch size` - a parameter that defines the number of training examples used for a single weight update. The appropriate batch size affects the speed of training and memory usage.

8. `Number of epochs` - a parameter that defines the number of complete passes through the training set. Too many epochs can lead to overfitting.

9. `Learning rate` - a parameter that defines how quickly the model learns. A learning rate that is too high can lead to overfitting, while a rate that is too low can result in slow progress during training.

10. `Dropout (Dropout layers)` - Dropout layers help prevent overfitting by randomly deactivating certain neurons during training. This regularization technique reduces the risk of overfitting.

In the following sections, I will present a comparison of the neural network's prediction results obtained with different sets of parameters.

CHAPTER 3

Results Analysis

3.1. The Simplest Neural Network Model

`Model 1` - this will be the simplest neural network model, which will serve as our baseline. Its specification:

- `Number of Layers` - I applied the minimal number of layers, consisting of an input layer, one hidden layer, and an output layer.
- `Number of Neurons` - a small number of neurons: 14 neurons in the input layer, 8 in the first hidden layer, and 1 neuron in the second hidden layer.
- `Activation Functions` - I used the ReLU activation function in the hidden layer.
- `Loss Function` - I used binary classification; Binary Crossentropy is a commonly used loss function.
- `Optimizer` - the popular Adam optimizer.
- `Number of Epochs` - I selected a fixed number of 10 epochs, without early stopping for the training process.
- `Batch Size` - I used the default value.

I will primarily compare the models based on `Accuracy`, which measures the percentage of correct predictions of the winning team on the test set. In our simplest model:

- `Accuracy` = 0.7881

This means the model's accuracy in predicting the winning team based on the first 15 minutes of the match is 78.81%.

3.2. Comparison of 10 Models with a Single Parameter Change

In the second part of my analysis, I decided to construct 9 models, each differing from the baseline version by only one parameter.

3.2.1. Model 2 - More Neurons in the Hidden Layer

In Model 2, I chose to increase the number of neurons in the hidden layer. This choice aims to enhance the network's ability to detect more complex patterns in the training data. Two layers were used - the first with 32 neurons, and the second with 16. The final layer, containing one neuron with a sigmoid activation function, is used for binary classification. The model was trained for 10 epochs. The performance of this model was:

- `Accuracy` = 0.7901

3.2.2. Model 3 - Fewer Neurons in the Hidden Layer

In the case of Model 3, I took the opposite approach, reducing the number of neurons in the hidden layer. A smaller number of neurons can help avoid overfitting, especially when the training data is limited. The layers of the model consist of 8, 4, and 1 neuron, with a sigmoid activation function. This model was also trained for 10 epochs. The performance of this model was:

- `Accuracy` = 0.7872

3.2.3. Model 4 - Additional Hidden Layer

In Model 4, I introduced an additional hidden layer to increase the model's capacity. The added layer has 8 neurons and uses the ReLU activation function. The model consists of three layers: two hidden layers and one output layer with a sigmoid activation function. It was trained for 10 epochs. The performance of this model was:

- `Accuracy` = 0.7885

3.2.4. Model 5 - Different Learning Rate

In the case of Model 5, I changed the learning rate of the optimization algorithm to 0.001. The learning rate controls the steps the algorithm takes to minimize the cost function. A lower learning rate can result in more precise weights, but it may also require more training epochs. The model consists of one hidden layer (8 neurons) and one output layer with a sigmoid activation function. The performance of this model was:

- `Accuracy` = 0.7874

3.2.5. Model 6 - Different Activation Function

In Model 6, I used a different activation function in the hidden layer. Instead of ReLU, I applied sigmoid in the layer with 8 neurons. Activation functions determine how signals are passed between neurons. Sigmoid is often used in the output layer for binary classification, but its use in the hidden layer can introduce subtle differences in learning. The performance of this model was:

- `Accuracy` = 0.7889

3.2.6. Model 7 - Different Batch Size

Model 7 differs from the previous ones by using a different batch size during training. Batch size refers to the number of training examples used for one weight update. Changing this parameter can affect the stability of the learning process. In this case, I used a batch size of 64, and the model consists of one hidden layer (8 neurons) and one output layer with a sigmoid activation function. The performance of this model was:

- `Accuracy` = 0.7867

3.2.7. Model 8 - Different Number of Epochs

In Model 8, I changed the number of training epochs to 20. The number of epochs defines how many times the model will pass through the entire training set. Increasing the number of epochs can further adjust the model to the training data, but it also increases the risk of overfitting. This model consists of one hidden layer (8 neurons) and one output layer with a sigmoid activation function. The performance of this model was:

- `Accuracy` = 0.7883

3.2.8. Model 9 - Dropout Layers

Model 9 uses dropout layers, which help prevent overfitting by randomly deactivating certain neurons during training. In this case, I applied a dropout layer with a dropout rate of 0.2. The model consists of two hidden layers (16 and 8 neurons) and an output layer with a sigmoid activation function. The performance of this model was:

- `Accuracy` = 0.7879

3.2.9. Model 10 - Different Loss Function

The final model, Model 10, differs by using a different loss function. Instead of binary_crossentropy, I used mean_squared_error. The loss function determines how well the model estimates how much its predictions deviate from the actual values. This model consists of one hidden layer (8 neurons) and one output layer with a sigmoid activation function. The performance of this model was:

- `Accuracy` = 0.7884

Here is a bar chart comparison of my models' accuracies:

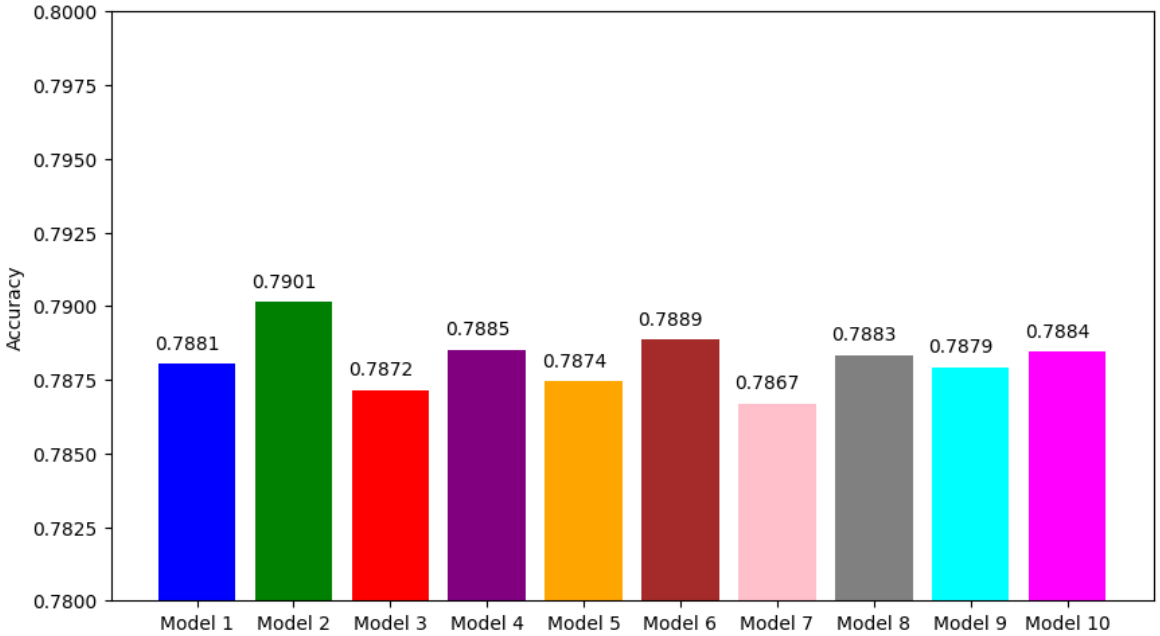

Figure 3.1: A bar chart comparing our 10 models. Model 1 was our baseline model, and Models 2 through 10 differed by only one parameter from Model 1.

The analysis of the chart indicates that the highest accuracy was achieved by Model 2, where the number of neurons in the hidden layer was increased. On the other hand, the lowest result was obtained with Model 7, where the batch size was changed. Additionally, Models 3, 5, and 9 performed worse than the baseline Model 1.

3.3. Three More Advanced Models

In the third part of my analysis, I compare three models that differ from my simplest model by more than one parameter.

3.3.1. Model 11

Model 11 is a network with two hidden layers containing 32 and 16 neurons, respectively. It uses the `relu` activation function in the hidden layers and `sigmoid` in the output layer, which is commonly used in binary classification problems. The model compilation uses `binary_crossentropy` as the loss function. To prevent overfitting, EarlyStopping was applied with `monitor='val_loss'`, `patience=3`, and `restore_best_weights=True`. This model was set for 50 epochs but stopped after 8 epochs. The performance of this model was:

- `Accuracy` = 0.7902

3.3.2. Model 12

Model 12 is an extension of the previous one, increasing the number of neurons to 64 in the first hidden layer and 32 in the second. This increases the model's capacity, enabling

it to learn more complex patterns in the training data. The same activation functions, loss function, and optimizer were used as in Model 11. EarlyStopping was also applied, and the model stopped after 7 epochs. The performance of this model was:

- `Accuracy` = 0.7900

3.3.3. Model 13

Model 13 is a neural network with two hidden layers containing 16 and 8 neurons. This model has fewer neurons than the previous two, which can help avoid overfitting, especially when the training data is limited. The activation functions, loss function, and optimizer remain the same as in the previous models. It was trained for 20 epochs without EarlyStopping. The performance of this model was:

- `Accuracy` = 0.7908

Summary

In conclusion, in my work, I used various neural network models to predict match outcomes in the game "League of Legends" using data from the first 15 minutes of gameplay. The best result I was able to achieve was with Model 13, with an accuracy of:

- `Accuracy` = 0.7908

Below is a comparison of all 13 models:

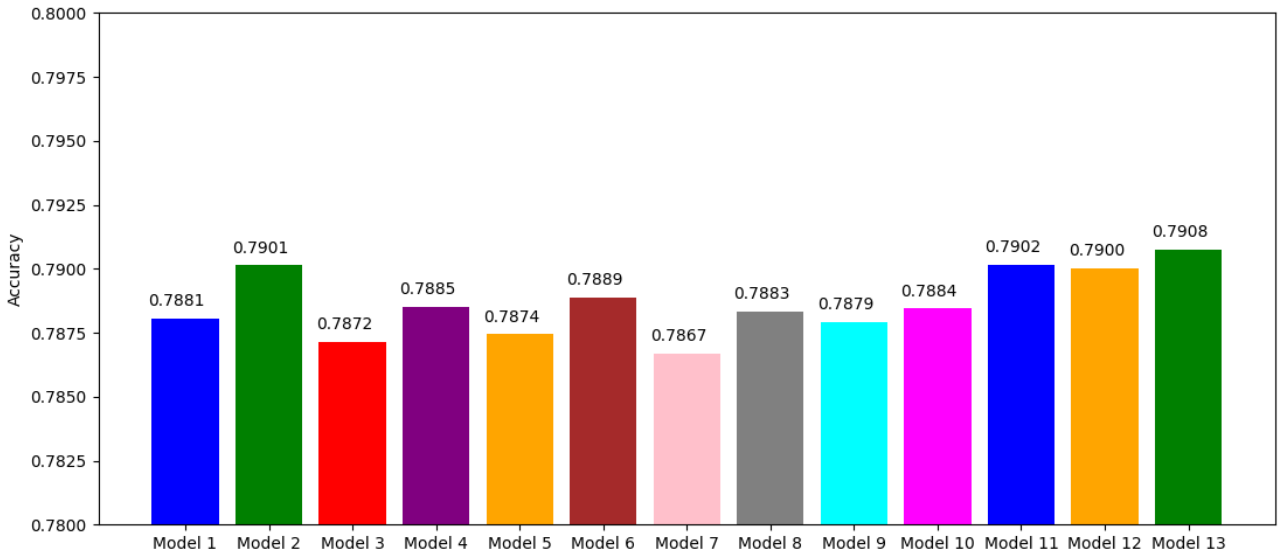

Figure 3.2: A bar chart comparing all 13 models described in this report. Model 1 was our baseline model, Models 2 through 10 differed by only one parameter from Model 1, and Models 11, 12, and 13 were the most advanced models.

I believe that the neural network performed well, achieving satisfactory prediction results. However, in no configuration was I able to exceed 80%, which suggests that predicting the outcome of a match based solely on the first 15 minutes is difficult with an accuracy higher than 80%. Exceeding this threshold might require adding more features to the dataset. However, the 14 features included represent the most important game parameters that players can influence in the first 15 minutes. It is worth noting that a full match in "League of Legends" typically lasts between 20 to 50 minutes, so achieving an accuracy level of 80% is already quite high.

Bibliography

- https://pl.wikipedia.org/wiki/Sie%C4%87_neuronowa
- https://www.statsoft.pl/textbook/stneunet.html#artificial