Grant Jenson
Jackson Hendrix
Jess Elliott

Basketball Betting Bot

Machine Learning Final Project

5/4/2023

Introduction

The Basketball Bet Bot is aimed at helping betters predict two main aspects of a basketball game – whether the home team will win or not and the number of points they will score.

Dataset

The dataset came from Kaggle and contained 62,400 rows and 54 features pertaining to basketball games played from the 1946 season up until March of 2023. It had many features such as whether the home team won or lost, the free throw percentage for the home and away team, and the team matchup. However, not all those features were needed to predict the two questions at hand. In exploring the data, it was noted that not all the statistics were consistently tracked, like rebounds and steals, until 1985. Thus, the dataset was reduced to 41,073 rows which now contained statistics for all games played from the 1985 season onwards.

Question 1

To determine `home points`, the relevant features needed to be used in the model. To decide which features were to be used, 11 scatterplots were created to visualize the relationship between the home points and the various other features – including between away points and home points. It was interesting to see that there was a strong correlation between these two features, with most of the values ranging between 50 and 150. Through these plots, the features chosen were - rebounds, steals, assists, blocks, and turnovers of both the home and away teams. `Home points` formed the actual predicated values dataset.

Before the data could be split into training and testing, the values of the selected features needed to be normalized so that every feature could have a standard scale. Z-score normalization was used, not only to achieve this standard scale but to allow for easier interpretation of features and faster convergence.

As the last step of data preprocessing, the feature and `home points` datasets were split into games from 1985 – 2015, which became the training set (76%), and games from 2016 – 2023, which became the testing set (24%).

The two models used to determine this question were Linear Regression and Regression Neural Networks, as the output feature was numerical and continuous.

<u>Model 1 - Linear Regression</u>

With the use of the scikit-learn library, a Linear Regression model was developed with the inclusion of a bias term. After the model was trained, the weight vector produced from the model was used in correlation with the general knowledge of the selected features, to ensure that the effect each feature had corresponded to the impact it would have on home points. For example, making sure that `away steal` was a negative value. The model then predicted `home points` using the testing feature set. An evaluation of the model's performance led to surprising results.

The root mean squared error (RMSE) measures the average difference between a predicted value and an actual value. The model produced an RMSE of 11.89, which means our model is on average 10 points off the actual home score – not the best range when money is on the line, but not terrible either.

The mean absolute percentage error (MAPE) measures the average percentage difference between a predicted value and an actual value. The model produced a MAPE of 8.31% which, according to literature, suggests an adequately performing model.

The coefficient determinant (R^2) measures how well the model fits the data, with 0 being poor and 1 being best. The model produced an R^2 value of 0.17, which suggests the opposite of what our performance values have shown so far. The model went from performing mediocrely to not great by any means. While the R^2 value should not be the only value considered when determining performance, having a value that low raises some concerns.

Model 2 - Regression Neural Network

With the linear regression model not performing as well as expected, more hope was put into the neural network model. After some trial and error, a neural network containing 2 hidden layers each having 50 neurons, using the sigmoid function for the activation function, running 1000 iterations of stochastic gradient descent on the data was developed using the scikit learn library. The neural net was trained on the feature training data and then the feature testing data was used for prediction.

But alas, the results were not what was hoped for. The neural net performed worse than linear regression almost every time the model was executed. It had an RMSE of 12.23, MAPE of 8.59%, and R^2 of 0.12.

Question 2

To determine whether `home win or loss`, the features chosen had to correlate with winning or losing but not be directly related to a result – for example, the `home` or `away points` or `field goals home`. Thus, the features selected were rebounds, steals, assists, blocks, and turnovers of both the home and away teams. `Home win or loss` would form another dataset, that held W for a win or L for a loss.

The win or loss dataset was then converted into a column of 0's and 1's where 0 was a loss and 1 was a win. This binary representation was used so that the data would be compatible with the sigmoid function and to help in the classification of the probabilities between 0 and 1.

The feature data was again normalized so that every feature could have a standard scale. Z-score normalization was used, not only to achieve this standard scale but to allow for easier interpretation of features and faster convergence.

As the last step of data preprocessing, the feature and `home win or loss` datasets were split into games from 1985 – 2015, which became the training set (76%), and games from 2016 – 2023, which became the testing set (24%).

The two models used to determine this question were Logistic Regression and Classification Neural Networks, as the output feature was categorical – either a win or a loss.

Model 1 - Logistic Regression

A Logistic Regression model was developed using the scikit-learn library with the inclusion of a bias term. After the model was trained, the weight vector produced from the model was used in correlation with the general knowledge of the selected features, to ensure that the effect each feature had corresponded to the impact it would have on whether the home team won or lost. The model then predicted 0 (loss) or 1 (win) using the testing feature set.

The model performed much better than linear models. The training accuracy of the model was 79.38% and the testing accuracy was 78.87%. These values suggest the model is classifying 78.87% of the unseen data correctly, which is good. However, only predicting 79.38% of the seen data correctly is lower than expected, as it should be performing better on training data.

Model 2 - Classification Neural Network

Since the logistic regression model performed well, the expectation was that the neural network might perform a little better. So, after some trial and error, a neural network containing 2 hidden layers each having 50 neurons, using the sigmoid function for the activation function, running 1000 iterations of stochastic gradient descent on the data was developed using the scikit learn library. The neural net was trained on the feature training data and then the feature testing data was used for prediction.

After evaluation, the neural net had a testing accuracy of 79.24% and the training accuracy was 78.87%. This is surprising as these values indicate this neural net performed slightly worse on the training data than the logistic model but achieved the same accuracy on the testing data.

Discussion

The evaluation of the models for each of the questions left significant room for improvement. In the future, it would be interesting to see how the linear models might be affected by including other features in the dataset, or how using a different function or model entirely would affect the accuracy.

Analysis of why the linear regression model had the low R^2 score was completed by plotting the predicted y and actual y against various features. It revealed that the predicted y values did not fit the data very well, often having a smaller range than the actual y values (*See Figure 1*). This is something that could be explored further as ways of increasing the range and accuracy. This could take the form of feature engineering or a better function for feature scaling (regularization).

To conclude, would Betting Bot earn millions if it gambled in the NBA leagues? Maybe not, but would it be interesting to see how it performed with a few tweaks in comparison to others? Most definitely.

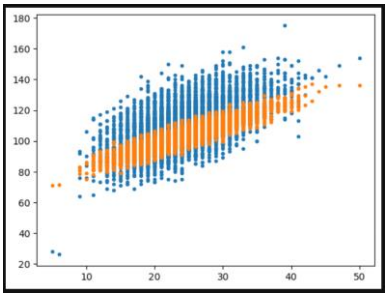

Figure 1: Graph showing the predicted y values (orange) and actual y values (blue) for assists of home.