Julius Randle Point Predictions

Anthony Elkommos Youssef, Benjamin Wolff, Shaan Hossain

Abstract

Predictions have been a crucial part of sports betting for a long time. More accurate predictions can help sports bettors make more informed decisions. In this project, we investigate three different machine learning models to try and predict the number of points Julius Randle, a professional NBA player, will score on any particular game given data from the previous games. The first method is Linear Regression, the second method is Ridge Regression, and the third method is a feed forward neural network.

Introduction

Since the beginning of organized sports, sports betting has been an important aspect of the game. Today, this “sport” has become so attractive that according to Statista, one of leading providers of market and consumer data, over 200 billion U.S. Dollars were spent on sports betting just in 2019. [5] With the emergence of sports betting applications and websites that provide a number of different betting options, the task of predicting such sports outcomes has become even more valuable. By having the tools to make more accurate predictions on the number of points a player will score in a game, an inexperienced bettor can potentially beat even the most experienced of bettors.

Sports bettors have long tried to “beat the book” and make predictions with a higher likelaspects of player-specific bets in basketball: points scored. We intend to create a regression model that can predict the number of points Julius Randle will score in a given NBA game. A model that could reliably “beat the book” could provide important information about the game of basketball and what contributes to the success of a player. Similarly, such a model could be very useful and profitable for a basketball bettor who wishes to succeed in bets relating to Julius Randle's points in a game. Ultimately, our project serves to answer an important question: can we use a regression model to, given data for a basketball player, predict the number of points that player will score for a given game?

Technical Approach

In the next three sections, we will provide specific details on each of the three approaches which we have used in order to approach and solve this difficult problem:

Method 1: Linear Regression Model

Serving as a basepoint, we decided to implement a Linear Regression Model as it's a simple model that is able to make highly accurate predictions when the independent and dependent variables have a linear relationship. [3]

$$Y = \beta_0 + \beta_1 X$$

Simple Linear Regression Equation

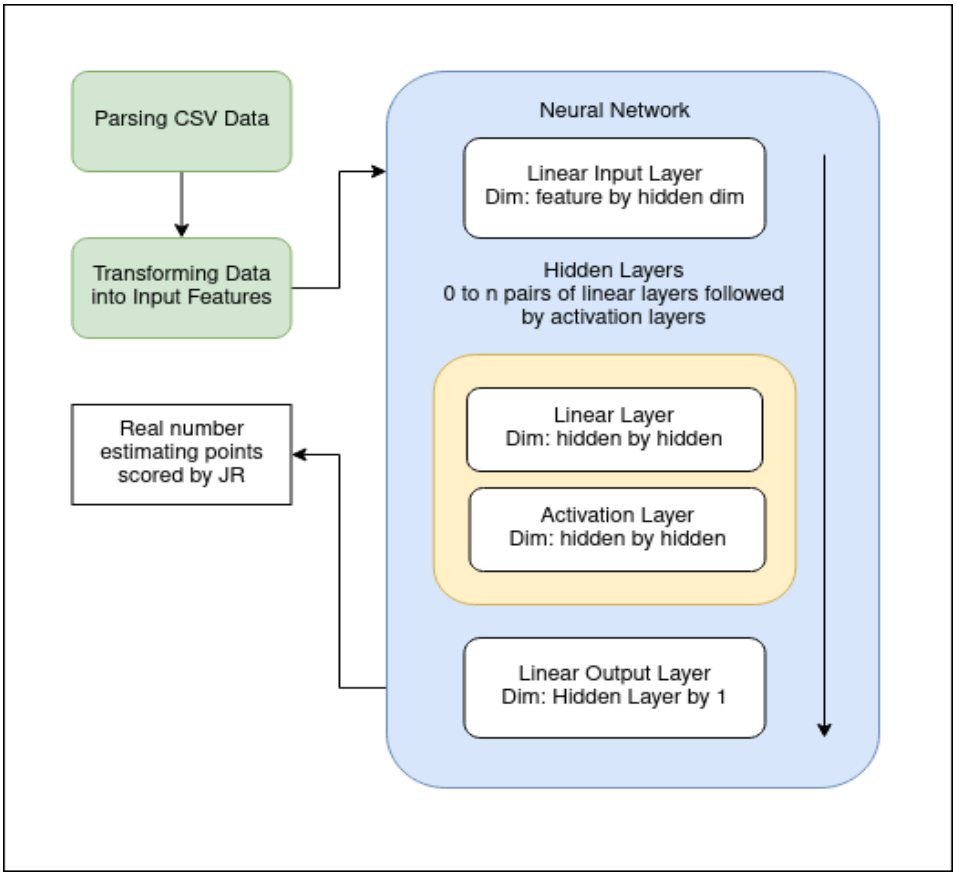
Method 2: Ridge Regression Model

One problem that emerges when using Linear Regression is over-fitting. To solve this issue, we decided on Ridge Regression as our second model since it uses the L2 Regularization technique. With such a large number of data points to consider, Ridge Regularization can be very effective since it is a model that can deal with multicollinearity and a number of observations (m) that is inferior to the number of independent predictor variables (n). By using Ridge Regression, we are able to regularize the coefficient estimates which are produced by Linear Regression. [2]

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2$$

The L2 Regularization Loss Function

Method 3: Neural Model



Given the complexity and non-linear nature of the data from basketball games, we decided to use a neural network since it performs well when given enough data. We should note that we don't have nearly enough data to make it effective. Additionally, it has been shown that

neural nets models are excellent at modeling data with high heteroskedasticity (highly volatile and scattered data).

First, we added an input layer which is simply a linear layer that takes in the input features in a vector of dimension *x*, and transforms the data to a dimension *y*.

Next, we created hidden (computation) layers which take data of dimension *y* and contain pairs of linear and activation layers. After passing through the initial linear input layer, the data is passed to an arbitrary number of 0 to infinite pairs of (linear, rectified linear) layers as the main sequence of computation layers.

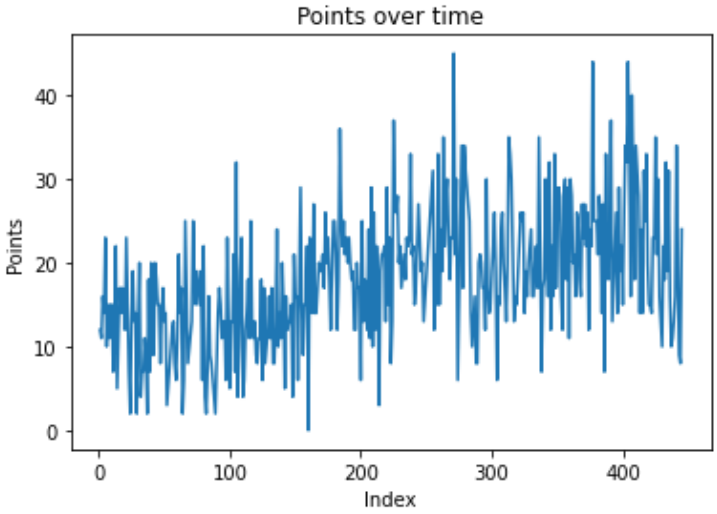
Finally, the output layer is a linear layer containing a single output neuron, and the output to our neural model as a whole is a real number point value.

One issue with neural networks is that they require a large amount of data in order to produce accurate and reliable predictions. Adding to that, the amount of data that we have found was not large enough to allow for any reliable predictions. Knowing it would be impossible to produce a neural model that can consistently, or at least usually, predict the exact number of points that Julius Randle will get, we decided to allow for a slack of 1 when calculating the accuracy of the model. [1]

Experimental Results

The Dataset

The dataset which we used to train the models was from SportsData.io, and it includes data regarding all of the games which Julius Randle played in during his career. It also contains a large number of features, such as whether the game was home or away, and all of Julius Randle's stats from the game including minutes played, assists, and shots attempted and made. The data also contains information about the opposing team as well as various betting information, such as how various sports betting websites rated Julius Randle at that moment in time and how well they predicted him to play. The dataset provided had 81 columns and 573 games as rows.



We cleaned up this data, removing columns we deemed irrelevant to determining points scored. We converted string values such as “Home” and “Away” and null values to numerical ones (in the described case, “home” was matched to 1 and “Away” was matched to 0). We performed a one-hot encoding for the team that Julius Randle was playing against

in order to best utilize this data. Additionally, we performed data manipulation to create new features, such as the number of days since Julius Randle was last injured. Lastly, since we would not have access to some of these values before a given game starts (for values such as points and minutes played), we used many of the features given to create averages over the past 5 (or any number k) games that would allow us to use this data to make predictions for the results of the game.

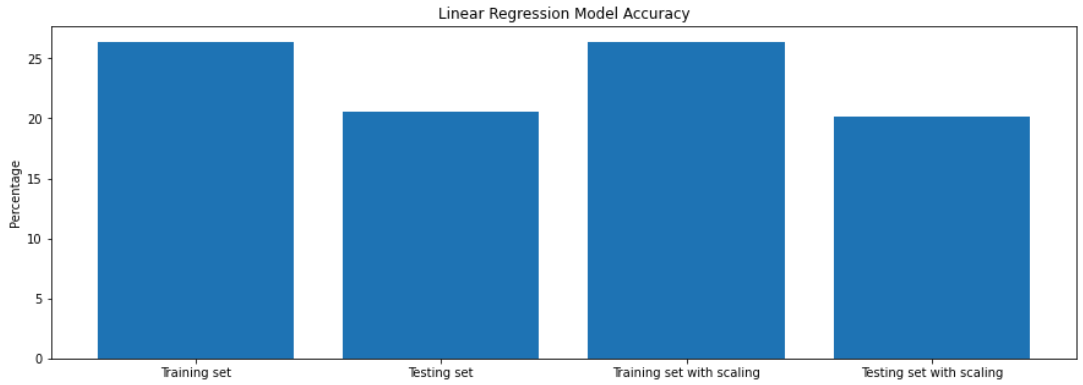
After all the data cleaning and manipulation, we had 45 features (which includes 30 features used from one-hot encoding for the opponent) and data from 425 games. Not including the opponent name features, the features used are:

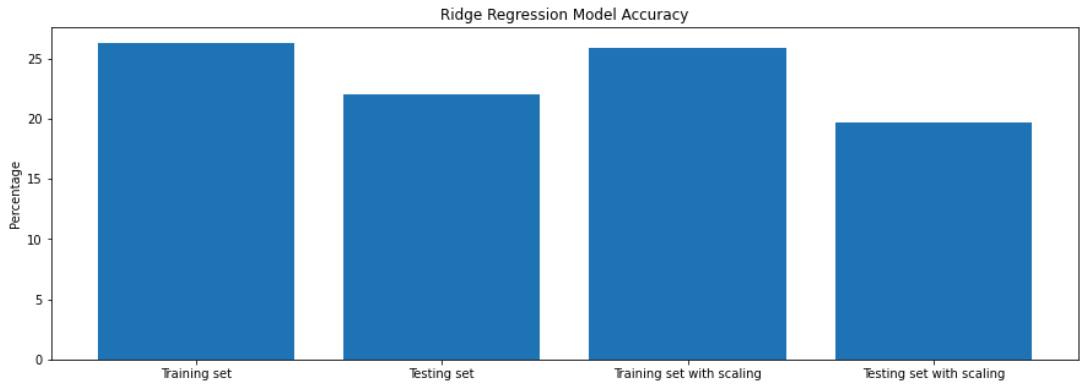
- HomeOrAway': Binary value representing if Randle's team is home (1) or away (0)
- 'OpponentRank': Numerical value representing the rank of the opposing team
- 'DaysSinceInjury': Number of days since Randle was last injured
- 'FantasyPointsAvg': Average Fantasy Points obtained over the past 5 games
- 'FantasyPointsFanDuelAvg': Average FanDuel Fantasy Points obtained over the past 5 games
- 'FantasyPointsDraftKingsAvg': Average DraftKings Fantasy Points obtained over the past 5 games
-
- 'FantasyPointsYahooAvg': Average Yahoo Fantasy Points obtained over the past 5 games
-
- 'FantasyPointsFantasyDraftAvg': Average Fantasy Draft Fantasy Points obtained over the past 5 games
- 'FanDuelSalaryAvg': Average FanDuel fantasy "Salary" given over the past 5 games
- 'DraftKingsSalaryAvg': Average DraftKings fantasy "Salary" given over the past 5 games
- 'YahooSalaryAvg': Average Yahoo fantasy "Salary" given over the past 5 games
- 'MinutesAvg': Average minutes played over the past 5 games
- 'UsageRatePercentageAvg': Average usage rate (of the basketball) over the past 5 games
- 'PointsAvg': Average points scored over the past 5 games

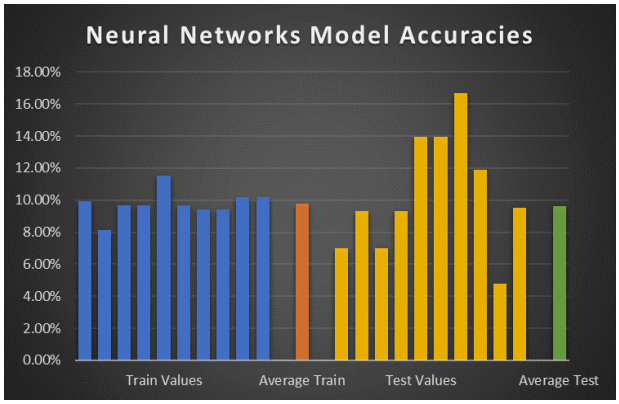
Implementation

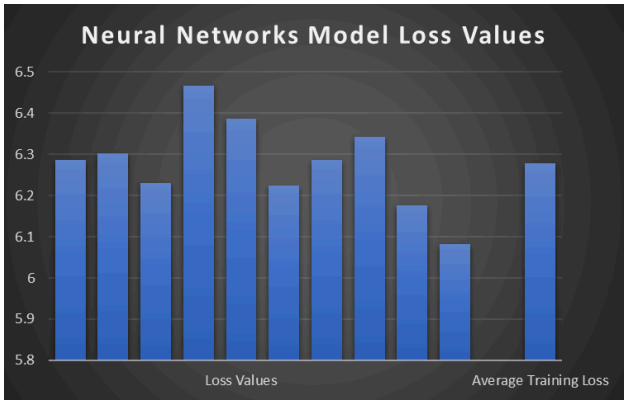
We implemented linear regression and ridge regression using the Scikit-learn library's built-in functions for these models. We trained the models on the features as well as features scaled with Scikit-learn's MinMaxScaler (that scales the data to a range from 0 to 1) for comparison. We used Scikit-learn's explained_variance_score to attempt to get another measure of the accuracy, but we mostly used the built-in score function for these models, which measures the r-squared value of the model with the data.

For the ridge regression model, we performed hyperparameter tuning using Scikit-learn's GridSearchCV, which we can use to compare various different values of ridge regression's hyperparameter alpha: the "regularization strength, [which] improves the conditioning of the problem and reduces the variance of the estimates." [4]









For the neural network, we used Pytorch. We started with Shaan's previous homework assignment and built on top of the modular design. There's a custom class that starts by defining all of the layers in the model. The forward method determines what happens on a forward pass. The network outputs a real number value as determined by the final activation function.

For the data we started with the same preprocessing scaling used for the two previous models, but we noticed that our values converged even closer to the mean target value, so we omitted the scaling step.

After the model has been defined, we have a method called train which takes in a model, features, target values, and a number of epochs. We left this method outside of the main

class to allow different loss functions and optimizers. The train method returns the final training loss and accuracy values.

In order to tune our hyperparameters we needed a form of cross validation. We implemented k-fold validation, so we simply pass our entire training data and allow it to break it up into k-folds, train on all but one and do so for every combination of k-1 folds. The final fold is used for validation. Our k-fold implementation returns the averages for training accuracy, training loss, and testing accuracy. It also returns the list of all of these values for each fold.

We experimented with a variety of hyperparameters. Here's a list of the hyperparameters and values we tried. We left the best values we found as defaults in the script and all of the graphs shown are for those values

We did not tune the number of previous games to consider for each sample in the input features, but this would be for future experimentation. 10 was a reasonable number. We decided to keep this consistent with the other regression models. We also did not experiment with folds for validation. There are more hyperparameters than listed to consider, but these are the ones we experimented with. In the table below, the first row is the hyperparameter, the second row is the range of values we tried with the steps in that range, and the third row is the best value we found. For epochs we also tried 1,000, and 10,000 just to observe any differences.

Games to look back	Pairs of hidden layers	Nodes per hidden layer	Activation function in layers	Epochs to train model	Folds for validation
10-10	1-5 Step=1	10-100 Step=10	ReLU, tanh, sigmoid	50-250 Step=50	10-10
10	3	50	ReLU	150	10

Conclusion

Ultimately, our models did not provide great results. However, based on the heteroskedasticity of the data and the complexity of the problem, such results were not completely unexpected. The ridge regression provided the best results, which makes sense since it is an optimization of linear regression that suits this problem well, and although a neural network can be a very effective technique, we likely do not have enough data points to take advantage of a neural model. Additionally, since the motivation of our chosen project was driven by the implications for sports betting, it is still possible that our models could be used to "beat the book"; further exploration could be performed to determine how our predicted results compare to the over/under value for Julius Randle's points and how this compares to the actual results.

Participants Contribution

Shaan:

- Implementing neural model, tuning hyperparameters, and generating results for that regression class.
- Starting implementation for logistic and linear regression, which became ridge and linear regression with Ben's work
- Slack accuracy method for evaluating models
- Average over last n values method for converting parsed data into input features
- Writing and revising reporting data pertaining to neural network including hyperparameter table and graph
- K-fold validation implementation for neural network

Anthony:

- Formatting, cleaning, normalizing, and parsing data to be used for the models
- Adding additional data points such as the number of days since last injury
- Researching and picking models (raised issues relating to the Logistic Regression model that was initially used)
- Helping in calculating the accuracies for the models
- Used matplotlib.pyplot for the creation of the graphs for the Linear Regression and the Ridge Regression model and Excel for the creation of the graphs for the Neural Network
- Creating, writing, and compiling the text, diagrams, and models for the report

Benjamin:

- Parsing the data using Pandas
- Cleaning the data, removing null/NaN values
- Finishing implementation for both regression models
- Tuning hyperparameters for ridge and linear regression models
- Implementing cross validation for ridge and linear regression models
- Creating a plot for the points scored over time, data analysis

References

“A beginner's Guide to Neural Networks and deep learning,” Pathmind. [Online]. Available: https://wiki.pathmind.com/neural-network. [Accessed: 18-Dec-2021]. [1]

“Chapter 335 ridge regression - ncss-wpengine.netdna-ssl.com.” [Online]. Available: https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCS/Ridge_Regression.pdf. [Accessed: 18-Dec-2021]. [2]

“Linear regression,” IBM. [Online]. Available: https://www.ibm.com/topics/linear-regression. [Accessed: 18-Dec-2021]. [3]

“Sklearn.linear_model.Ridge,” scikit-learn. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html. [Accessed: 18-Dec-2021]. [4]

“Topic: Sports betting worldwide,” Statista. [Online]. Available: https://www.statista.com/topics/1740/sports-betting/. [Accessed: 19-Dec-2021]. [5]