Task 2: Predicting Baseball Strikeouts

Jackson Sweet

June 6, 2025

1 Task & Lead

My task was predicting the number of strikeouts a baseball player would have in their next 10 games given the number of walks, singles, doubles, triples, and home runs in each of their last 20 games. I was the lead and sole contributor to the work done by my group on this task.

2 Methods

I experimented with four different models in Scikit-learn to try to achieve satisfactory predictions on the baseball dataset. The first model used was a single neural network, implemented with Scikit-learn's multi-layer perceptron regressor. I chose this model because neural networks are shown to generalize to a wide variety of problems.

Expanding on this, I also tried a neural network ensemble, using Scikit-learn's voting regressor to take the average of the predictions of an array of identical neural networks with different random initialization states. I learned in class and heard from my colleagues that ensembling models led to improved performance, so I chose to test this technique in addition to a single neural network.

Beyond neural networks, I also experimented with random forests and support vector machines to ensure a wider gamut of models was evaluated. Because of the simplicity of random forests and their purported high performance on high-dimensional data, I believed random forests had potential for use on the small, high-dimensional baseball dataset. The ability of support vector machines to maximize the margin between points on either side of a manifold attracted me to the model, as I surmised that a support vector regressor would exhibit good performance on the limited dataset.

Two performance metrics were used to judge the models: mean squared error (MSE) and R^2 score. These metrics were applied to both the train set and the development set. MSE was used to gauge effectiveness because it showed the average margin of error between the regressors' predictions and the true value. While MSE does not have the same units as the output data, root mean squared error is monotonically correlated with MSE and can easily be intuited from MSE by inspection. Therefore, I deemed MSE an appropriate error metric. Additionally, I used R^2 to assess the correlation between the predicted values from my models and the true values. A reasonable benchmark for R^2 was established through a conversation with Professor Sizemore, who informed me that a score of around 0.4 was satisfactory with the limited dataset.

3 Submission Model Details

The best model I devised for this task is a deep neural network ensemble. Each neural network in the ensemble has a width of thirty neurons and a depth of six layers, and the ensemble is composed of seventy-five networks. The regularization alpha value for each network is 0.01, and the activation function I chose was logistic sigmoid, as these proved to yield the best results. The best run with this model lasted ten epochs, and I elected to not run for a greater number due to diminishing returns on performance. Early stopping was turned on for the networks to prevent overfitting to the training set.

4 Results

The best model, the deep neural network ensemble, attained an MSE of around 6.67 and an R^2 score of roughly 0.50 on the train set, exhibiting a significant ability to learn the trend of the dataset. On the development set, the model attained an MSE of around 6.81 and an R^2 score of roughly 0.45, showing similar results and illustrating the model's ability to generalize to unseen data.

The next most performant model was the single deep neural network, which achieved MSE 7.50 and R^2 0.43 on the

train set and MSE 7.64 and R^2 0.38 on the development set. This performance is inferior to that of the neural network ensemble, but still exhibits good generalization. Both the neural network ensemble and single neural network showed a performance increase of roughly 2 points on MSE compared to their baseline runs. Each model appeared to have a characteristic range for network width and depth in which they performed best, and both performed best with a small regularization coefficient. The ensemble gradually improved as the number of networks increased, though it exhibited diminishing returns after a certain point.

Below the single neural network was the support vector machine, attaining MSE 8.81 and R^2 0.33 on the train set and MSE 9.71 and R^2 0.21 on the development set. Despite use of the RBF kernel, the SVM exhibited inferior performance and generalization. Its R^2 on the development set barely exceeded the threshold for a statistically significant trend.

The worst model was the random forest, with MSE 12.8 and R^2 0.03 on the train set and MSE 15.4 and R^2 -0.25 on the development set. Even with 1000 decision tree estimators used in the random forest, the model showed abysmal performance, with terrible generalization and a negative R^2 value on the development set, illustrating a trend toward incorrect predictions. No support vector machines or random forests showed significant improvements beyond the baseline run. For a side-by-side comparison of model performance, see the figures below.

5 Distribution of Work

My group of four chose to have one person work exclusively on a given task. As such, I was the only one to contribute meaningful work to the baseball task. However, I was provided with skeleton code for my baselines by my partner Zuriahn Yun, who began work on his task before I began mine. This foundation greatly expedited the process of attaining baseline performance and beginning the process of experimentally improving my results.