MAIS 202 Deliverable 2

Antoine Dangeard
Thomas Inman

Problem Statement:

We are creating a baseball game score difference predictor. The goal of the model is to predict the score difference (positive for home team win) of a baseball game, given the teams up-to-date stats.

Data Preprocessing:

We are working with two datasets: MLB team statistics and retrosheet game log. The MLB team stats were retrieved using web scraping, and the game log was downloaded. The datasets are then cleaned, combined, and preprocessed. The cleaning stage consists of removing the unnecessary elements from our data; for instance,we use (date, team names, scores) per game from the game log, so we removed the excess data and reformatted it. The combining stage includes matching games with team stats, so that the teams relative stats are “up-to-date” for the given game. The preprocessing stage standardizes our features.

Machine Learning Model:

1. Our model is the same as in deliverable 1, a polynomial regression model. Our model takes team statistics (such as hits, SLG, OBP, etc.), standardizes the stats, and creates their polynomials. I.e. each feature will have a respective polynomial of n^{th} degree (we will treat the degree as a parameter).
2. We decided to split our data into three sections with training (50% of games), validation (25%), and test (25%). We chose this split to allow more training and validation data for the model.
3. Since we have a large quantity of data (g Games, m features, nth degree) we have around g*m*n elements of data, with around 90,000 games, 16 features, and 10th degree. This results in longer commuting times. Ideally, we would like to implement PCA into our model for dimensionality reduction. This would allow similar stats to be combined into an abstract space, which would reduce the amount of computation required for our model to perform. Thus decreasing the time cost of running the model.
4. We are testing our model with MSE. To get a better understanding of our model’s accuracy we are testing the accuracy of the game winner. This allows us to have a better benchmark on how our model should be performing. At the very least, our model should get 50% (random guessing probability) of the game winners correctly. Our model performed with an MSE of ~18.76 which is around RMSE of 4.3 points. This is a decent result for the first attempt. The model’s winning accuracy demonstrates our models' performance better. An accuracy rate of 55.84% on the testing data shows that our model is performing slightly better than average. The model can be improved with the steps mentioned in 5), which has potential to decrease MSE and increase the winner accuracy.

```
===== SCORE =====
Mean squared error: 18.759967999537054
Model predicted outcome correctly 8437 times out of 15110 games.
(55.84% Accuracy)
=================
```

5. The next steps of our model is to tune the hyperparameters, implement PCA, and review the data we want to use. The hyperparameters we wish to alter include the learning rate, regularization rate, and polynomial degree. To find the most efficient and well performing values, we will perform a grid search over the hyper-parameters. Implementation of PCA will be crucial for our model to perform efficiently. There will likely be correlations between the data which would allow for PCA to be successfully applied. Some features can be modified/added to the team stats such

as pitching matchups vs batting avg; specifically, having batting average as a feature depending on the pitcher's throwing arm. Also, depending on the results of PCA, some features may be removed which do not have a big impact on the result of the game.