Data Analytics Assignment-2: Duckworth-Lewis Method

Anshul Shivhare

September 12, 2021

1 Formulation of the problem

In cricket, Duckworth-Lewis method is used to model the total runs that can be scored as a function of overs to go (u) and wickets in hand(w) as the net value of the resources. This total runs that can be scored, is denoted by the function Z.

- In the first model, Z is modeled as a function of both overs u and wickets w, using the below relationship:

  $$Z(u, w) = Z_0(w)(1 - e^{-b(w)u})$$

  where $Z_0(w)$ is the average runs scored when a team has w wickets and starts playing to set a target, and doesn't have any over restrictions. And $b(w)$ is the growth rate of the function.

- In the second model, we assume that if only one ball remains, the for any number of wickets in hand, then the average increment to the score would be the same, which is a reasonable assumption if a good batsman is on strike. Thus, the slope at $u = 0$ would be same for all w. If this slope is denoted by L, then the above relationship can be revised as:

  $$Z(u, w) = Z_0(w)(1 - e^{-\frac{Lu}{Z_0(w)}})$$

- These two models can be fitted to the given data, to see how accurately they can model the potential runs scored. This is evaluated using the *Mean square error* metric.

2 Data cleaning

The given data had lot of inconsistencies, so it had to be cleaned properly to make it more consistent before fitting to the model.

After thorough exploratory analysis of the data, I decided to clean-up the data by modifying some of the rows in a suitable way, and remove some rows altogether. It includes the following:

- There are 11 matches where the *Error.in.Data* column entry is 1. Since it was clearly mentioned that there is error in those rows (but not mentioned in which columns exactly there is error), I decided to drop the data of those 11 matches altogether.

- For match number 410557, for the first innings, I observed that there was data only after the 16^{th} over. Hence, the data from 1^{st} to 15^{th} over was missing for this particular match. Hence, I dropped the data of this one match as well.

- There exist 8 matches for which *Total.Runs* column of the last over is not equal to *Innings.Total.Runs* column. I carefully examined the data of those 8 matches, and came to the conclusion that the data of *Total.Runs* column made more sense and was more realistic. There were some inexplicable jumps / mistakes in the *Innings.Total.Runs* column for those 8 matches. Hence, for these 8 matches, I replaced the *Innings.Total.Runs* column with the last value of the *Total.Runs* column, to make it more consistent.

- I also observed that in one of the matches (match number 424849), there was difference of as many as 160 runs between the *Innings.Total.Runs* column and *Total.Runs* column. There were few other matches with huge inconsistencies, hence I dropped those few matches in order to avoid bias in the model. (Later, I also experimented by not dropping these matches and then calculating the mean squared error, to ensure that I'm not dropping any valuable data points).

- I checked whether there are any matches with missing first innings data, but I did not find any such matches.

- I also checked if the aggregate of individual over runs is equal to the *Total.Runs* column. There also I found some inconsistencies, and stuck with individual over runs where the error was very small, and dropped few matches where error was very large.

- Also checked whether *Innings.Total.Runs* column was consistent for each match, and it turned out to be consistent. (For each innings, I did not find cases of *Innings.Total.Runs* column containing different values).

2.1 Analysing interrupted matches

I also analysed the matches which seem to be *interrupted by rain*, based on the values of u and w, as follows- In a non-interrupted match, if the team loses all 10 wickets in less than 50 overs, then at some point, the *Wickets.in.Hand* column would become zero and the maximum value in the *Overs* column for that particular match would be ≤ 49. And if the team does not lose all 10 wickets in 50 overs, then in a non-interrupted match, the maximum value in the *Overs* column would be 50.

Hence, I'm putting the condition that for each match, if the maximum value in *Overs* is less than 50 *and* the minimum value in *Wickets.in.Hand* column is non-zero, then the match must be interrupted. Hence, I assumed that all such matches were interrupted in between, either due to weather or due to some reason.

But here also, I found some inconsistencies or unusual observations that didn't make much sense.

There were as many as 79 matches which seemed to be interrupted based on the above reasoning, but on closer inspection, it seems 13 of those matches were interrupted in the 49th over! And 8 and

6 matches were apparently interrupted in the 48th and 47th over, which seemed strange. Hence, assuming that there might be some mistake in data collection, (maybe multiple wickets fell in the last over of those matches and maybe they forgot to include an entry with 0 wickets remaining), I decided to neglect those matches which were apparently interrupted after 45 overs.

So I considered only those matches which were interrupted before 45 overs, as truly interrupted. 46 out of 79 matches fell in this category.

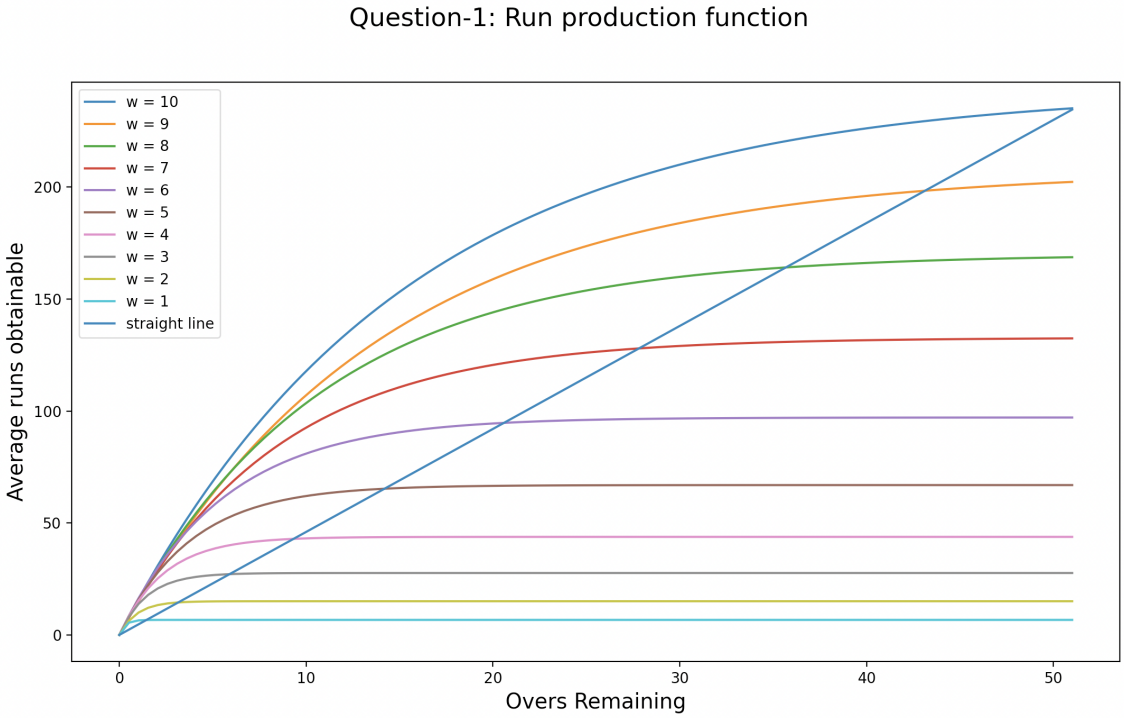
With these two matches, I tried two approaches to deal with it:

- First, I tried dropping these matches and checked the performance of the model by checking the mean square error.
- Then I tried making suitable modifications to the data to include rain-affected matches, as follows - Suppose a match was interrupted in the 30th over, then regardless of the number of wickets in hand, I modified the overs to go u as (30 - *Over*) instead of (50 - *Over*), which makes more sense for such shortened matches. Then I computed the mean square error, and observed that it gave slightly better performance.

3 Observations

3.1 First model (with 20 parameters $Z_0(w)$ and $b(w)$)

3.1.1 Plot of the run production function $Z(u, w)$



3.1.2 Values of $Z_0(w)$ (for cleaned data)

w	$Z_0(w)$
1	6.704
2	15.039
3	27.652
4	43.787
5	66.926
6	97.119
7	132.748
8	170.133
9	207.462
10	243.587

3.1.3 Values of $b(w)$ (for cleaned data)

w	$b(w)$
1	3.321
2	1.061
3	0.675
4	0.418
5	0.262
6	0.1795
7	0.1195
8	0.0938
9	0.0726
10	0.0661

3.1.4 Slopes at $u = 0$ (for cleaned data)

w	$b(w)$
1	22.27
2	15.96
3	18.67
4	18.3
5	17.53
6	17.44
7	15.87
8	15.97
9	15.06
10	16.11

3.1.5 Mean Square Error

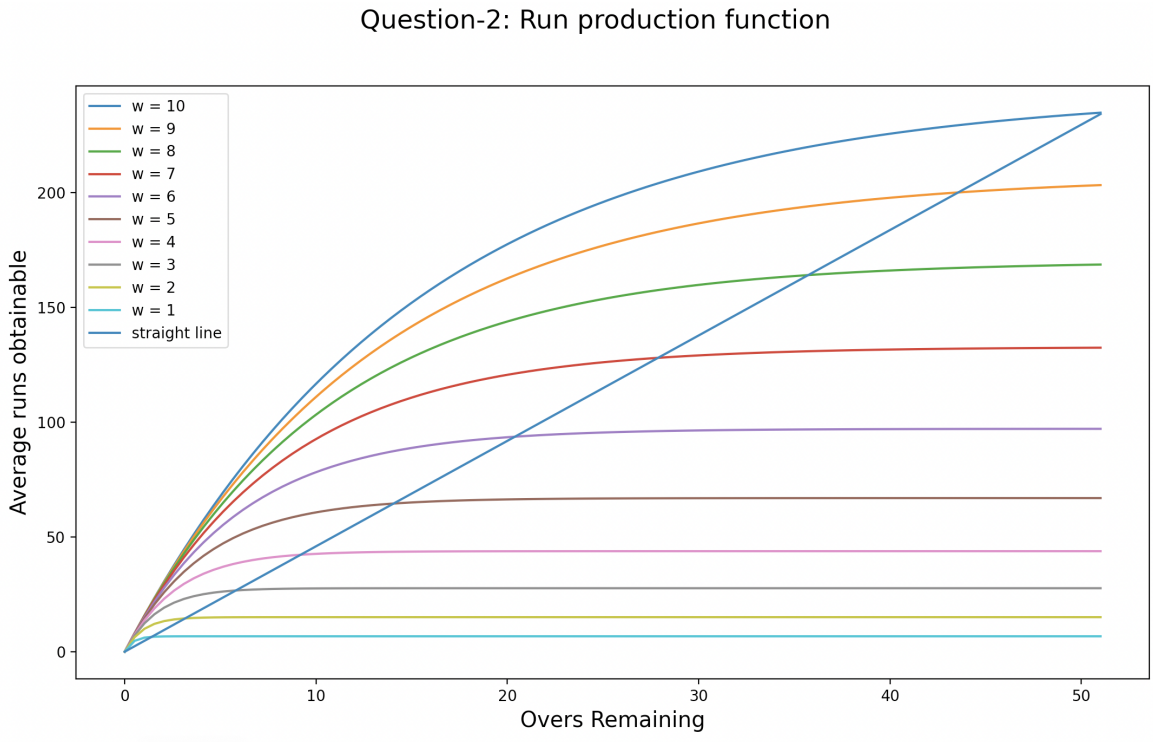
- MSE = 1717.25 for cleaned data.
- MSE = 1829.39 for uncleaned data (without modifications to original data provided on Moodle)

3.1.6 Few observations

- As one would naturally expect, we have $Z_0(10) > Z_0(9) > Z_0(8) > ... > Z_0(2) > Z_0(1)$. There is more potential to make runs when we have more wickets in hand.
- We also observe that the slope of the curves become flatter and flatter as u increases. This makes sense, because with a limited number of overs in hand, it would not make much difference if we had 100 overs in hand or 200 overs in hand, as the slope would be very flat in that region, so there would not be much difference in runs scored.
- We also observe that the curves for smaller w become flatter much faster than the curves for larger w (as u increases). For example, the curve for $Z(u, 3)$ becomes very flat at around 10 overs, but the curve for $Z(u, 10)$ is still growing till 50 overs, which signifies that there is lot more potential to score runs when more wickets are hand. Overs are valuable only when there are sufficient wickets in hand.

3.2 Second model (with 11 parameters - $Z_0(w)$ and L)

3.2.1 Plot of the run production function $Z(u, w)$ (for cleaned data)



3.2.2 Values of $Z_0(w)$

w	$Z_0(w)$
1	6.704
2	15.039
3	27.652
4	43.787
5	66.926
6	97.119
7	132.748
8	170.133
9	207.462
10	243.587

3.2.3 Value of L

After fitting the different curves, I got the optimal value of $L = 15.8965$.

3.2.4 Slopes at $u = 0$ (for cleaned data)

w	Slope at $u = 0$
1	15.8965
2	15.8965
3	15.8965
4	15.8965
5	15.8965
6	15.8965
7	15.8965
8	15.8965
9	15.8965
10	15.8965

3.2.5 Mean Square Error

- MSE = 1719.39 for cleaned data.
- MSE = 1831.38 for uncleaned data (without modifications to original data provided on Moodle)

3.2.6 Few observations

The observations for this model are very similar to the observations in the first model, except for the fact that the slopes are same at $u = 0$ for all w, and much less number of parameters are required to represent the second model, hence the computation time is faster too.

4 Structure of the code

The different functions in the code are briefly described as below:

- **prepareDataFromCSV:** This function takes the name of the CSV file as string input, processes the columns of the data, and prepares it for curve fitting in the other functions. It returns the processed data, as well as the raw data of the first innings.

- **computeZ0:** This function takes the first innings data as input, and computes the value of Z_0 for each w by taking the mean of runs scored when there are w wickets in hand.

- **DuckworthLewis20Params:** This function takes the CSV file path as string input, and internally uses the above two functions to get the processed data as well as the values of Z_0. Internally, a nested function *fn* is also declared, which is the objective function to optimize for the 20 parameters $Z_0(w)$ and $b(w)$. After declaring the function *fn*, we are doing the optimization using scipy.optimize library's *minimize function* for each w. Finally the mean square error is computed, printed and the Z and b values are returned.

- **DuckworthLewis11Params:** Similar to the 20 parameters function, this function takes the CSV file path as string input, and internally uses the above two functions to get the processed data as well as the values of Z_0. Internally, a nested function *fn* is also declared, which is the objective function to optimize for the 11 parameters $Z_0(w)$ and L. After declaring the function *fn*, we are doing the optimization using scipy.optimize library's *minimize function*. Finally the mean square error is computed, printed and the Z and L values are returned.

- **plotQ1 and plotQ2:** These two functions take as input the 20 parameters and 11 parameters respectively, and plot all the 10 curves on the same figure using matplotlib library.

- **computeSlopeQ1 and computeSlopeQ2:** These two functions take as input the 20 parameters and 11 parameters respectively, and compute the slopes of the curves for each w at $u = 0$.