THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

Stochastic Dynamic Programming :

Optimal Strategy for Decision Review System in Cricket

by Huizhen Tan

School of Information Technology and Electrical Engineering

The University of Queensland

Submitted for Master of Dara Science Capstone 2

2020-11-01

Abstract

Decision Review System is very important in a cricket match especially for the test match. One single review can even change the result in a tight match. However, the historical data shows that the ability to use DRS reviews correctly has been quite poor and wasn't improved much since DRS was launched. Therefore, this project aims to use stochastic dynamic programming to simulate the cricket match and find the optimal strategy for using DRS reviews. We use some web scraping techniques to acquire the data from a website called ESPNcricinfo. Some data cleansing and data merging process is carried out before we use a rule-based classifier to classify every historical deliveries into some categories. After the classification, we estimate the probabilities which represent the distribution of real world cricket events, and insert them into the SDP model as parameters. The expected run score returned by the SDP model is really closed to the average run score from historical data. What's more, we extract the optimal DRS strategy, and it's basically a confident threshold, indicating how confident one should be for the opportunity in order to take the DRS review. Other than that, we analyze the impacts for batting and bowling team's reviews, the first and second reviews, and different DRS iterations. At the end, we discuss the sensitivity of the model parameters, the impact DRS have done to the test match and the problem of DRS usage estimation.

Content

1 Introduction

This project aims to analyze the previous use of decision review system (DRS) in test cricket and find an optimal DRS strategy based on historical data. In cricket, DRS, a technology-based system, is used against ambiguous decisions made by umpires. Similar system such as the Hawk-eye is also used in tennis. Players can take a DRS review to challenge umpire's decision if they think the umpire made a wrong decision. However, unsuccessful reviews are limited and precious, and sometimes a successful review (or an unsuccessful one) can change the result of a match. Therefore, in order to make the most rational choice, the optimal strategy should not only consider the trade-off between the current benefit and risk , but also the possible benefit in the future. In this section, I'll introduce the DRS history and Australia's current DRS strategy. Then I'll talk about the motivation and the aims of this project.

1.1 DRS history

DRS was first tested in 2008 and officially launched by the International Cricket Council (ICC) in 2009. Since then, DRS had evolved to several generations with slightly different rules. The first generation, which is from November 24th 2009 to January 1st 2013, allows 2 unsuccessful reviews for both team in every inning. And from then to September 28th 2017, the second generation allows both team to reset to 2 reviews every 80 overs. After that is the third generation, which cancels the review reset rule and allows both team to keep the review chance if the DRS shows that it's an umpire's call decision. The SDP model in this project is based on the 3rd DRS Iteration.

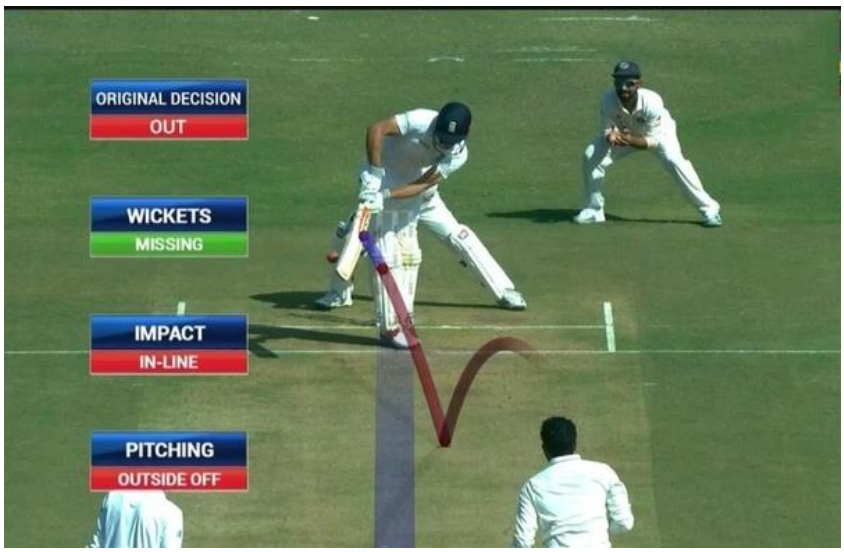

Figure 1.1 Decision Review System [1]

1.2 Current Strategy

As experience have been gained, cricketers have mastered the art of calling referrals. According to Ashok Kumar Shaw (2019) [2], the use of DRS is more about when not to review, rather than the opposite, as reviews are limited. He illustrates some scenarios when players are more likely to make wrong DRS decisions. For example, when there's a desperate need of wickets or when reviews are used against top batsmen, costly errors are easily made. So referrals should be taken only if the team is confident about their challenges based on the delivery. According to an interview, Australia makes its current DRS strategy pretty simple. Under this strategy, only the bowler and wicketkeeper will have a proper say on reviewing caught behinds, while in LBW appeals the point fielder will also be used ("Aussies review DRS strategy", 2019) [3].

1.3 Motivation

Australia makes its strategy simple because the captain has only 15 seconds before he makes a decision. This will probably ignore the possible impact of the decision in the future due to lack of thinking time. However, the optimal strategy should take into account the future impact of the current decision, because we want an optimal strategy for the whole match. More importantly, sometimes a review can completely

change the result of the match. A famous example is the third test match of the Ashes 2019 series. It was a tight match between Australia and England. At the end of the third from the last over, although Australia was leading, England only needed 8 runs to overturn the match. Both team were under great pressure. So when an turned down LBW (leg before wicket) decision was given to England's batsman, Australia took a desperate review and lost the last review. However, in the next over, the umpire gave another turned down LBW decision. The replay on the television showed that this was a poor decision and could be overturned on DRS. But Australia had no reviews left. They lost the match in the end. The captain of Australia, Tim Paine, was considered to be a terrible decision maker since then. However, if there's a more detailed DRS strategy, which can quantify the potential impact of the current decision, to guide him to make the correct choice, Australia may be able to avoid such dramatic failure.

1.4 Project Aims

In order to determine the optimal DRS strategy, we first need to find a way to mimic the test match. Since it's a Markov Decision Process where the outcomes are partly random and partly controlled by a decision maker, we need to estimate the probabilities for random events such as a turned down caught dismissal, an up held LBW dismissal or a specific number of gained runs from historical data. Therefore, for each delivery in each match, we intend to classify it into some categories such as whether the wicket falls, whether it's a LBW dismissal, whether it's an opportunity to review, what's the confidence level of taking a review and so on. By solving this classification task in section 2, we estimate the probabilities of different outcome for a delivery. We then explore the data and extract some statistics in section 3. After that, we use stochastic dynamic programming to simulate a test match inning under some assumptions, which we will discuss more in the section 4. Then in section 5, we calculate the confidence threshold at a specific state in the inning by comparing the expected value of reviewing or not. This confidence threshold will be the optimal DRS strategy we are looking for. What's more, in section 6, we tune different parameters

or change the assumptions to see how these variation affect the optimal strategy. Some further issues about the SDP model are also discussed in section 6.

In conclusion, the project aims can be split into five parts:

- Section 2: Data processing and Delivery Classification
- Section 3: Explored Data Analysis
- Section 4: Inning Simulation using Stochastic Dynamic Programming
- Section 5: Optimal Strategy Determination
- Section 6: Further Discussions

2 Data Acquisition and Processing

In this section, I'll introduce the dataset we use in this project, and illustrate how we acquire and process the dataset. At the end I'll talk about how we solve the classification task by using a rule-based decision tree together with some inappropriate methods we've tried.

2.1 Dataset

The data we need sits on the ESPNcricinfo website, which is a cricket news website who features news, articles, live coverage of matches, and even information of historical matches. It records nearly everything about a cricket match, such as the match notes (figure 2.1), the scorecard, and even commentary for each delivery (figure 2.2).

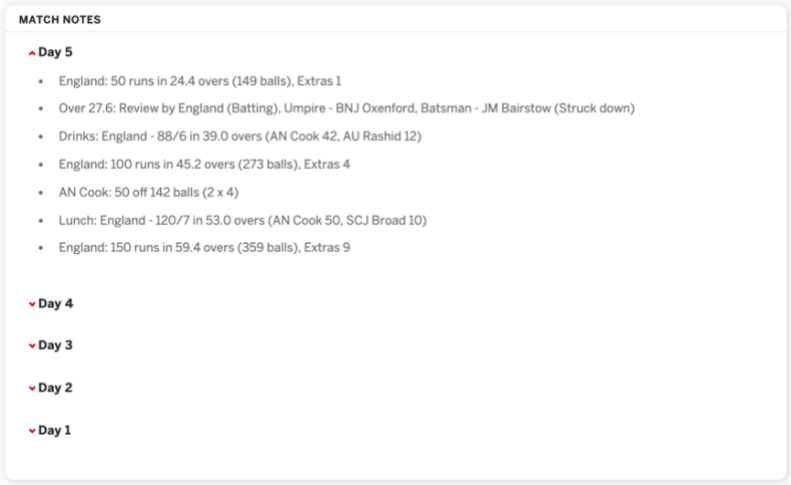

Figure 2.1. Match Notes

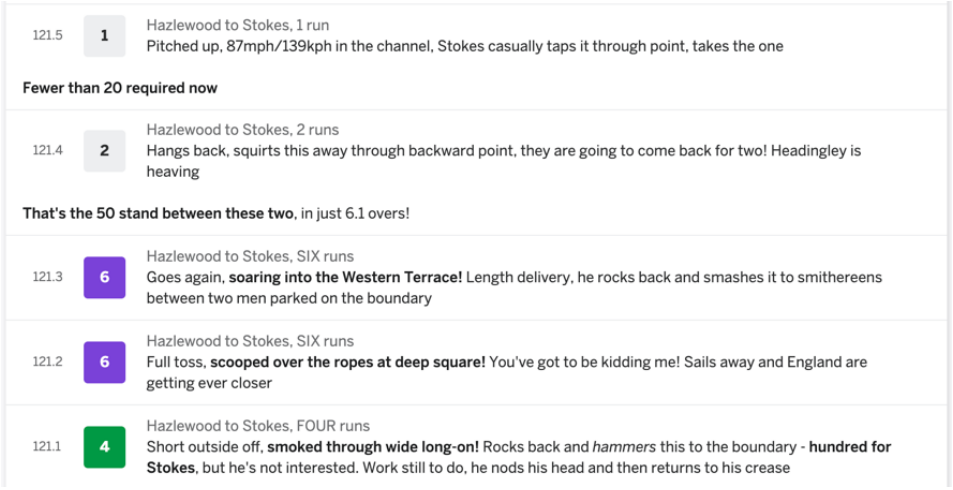

Figure 2.2. Commentary Data

The commentary and the match notes are what we need because the classification is mainly based on the commentary data, and match notes is where the actual DRS review results recorded. So we need to use web scraping techniques to extract the data.

2.2 Data Acquisition

To scrape the data from webpages, we need to use some web scraping techniques. Usually there are two type of webpages: static and dynamic. The main difference between them is that dynamic webpage can change its display according to user's behavior while static webpage can't. Because there are too much commentary data, the webpage loads new commentary when the user scrolls down the webpage. So we are dealing with nearly infinite scrolling pages here. As shown in figure 2.3 [4], we need to use the web dev tool to check HTTP requests which receive the hidden data, and then scrape it directly from the JSON content returned by the server. To be more specific, we use Python and the well-known HTML parsing Python package Beautiful Soup, together with some code from Github [5] to do the web crawling work.

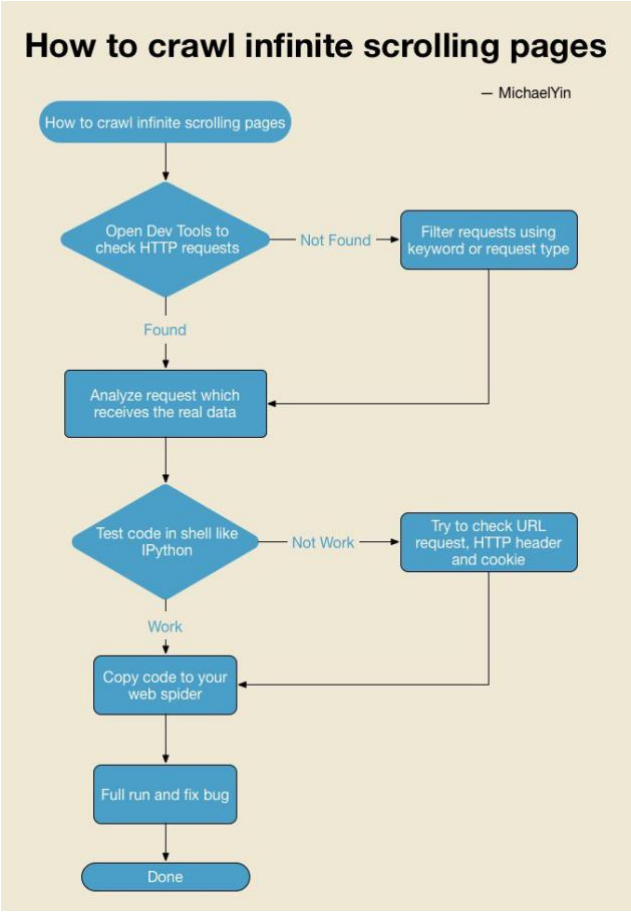

Figure 2.3. Process of crawling infinite scrolling pages

We scrape the data into two types of data frame, one is the match notes data frame, with approximate 36,000 rows and 18 columns where each row is a piece of match note, like in figure 2.1. Another is the commentary data frame for each match, with approximate 1500 rows and 20 columns where each row represents a delivery. And there are 450 matches in total, which makes the concatenate table with 650000 rows.

2.3 Data Cleansing

Although we have scraped the data into data frames, the original data frame is not in a tidy and useful form yet. So we need different types of data processing to make the data frame ready for use.

2.3.1 Match Notes Data

The match notes data need data processing because the original data frame only has a few useful features. Some important features are also missing and need to be extracted.

- Add Additional Features

Interesting features need to be extracted from the *NoteText* column, which records the text of the match note. Specifically, we only focus on match notes which record the outcome of actual reviews. We notice that the majority of useful *NoteText* are in such form:

Figure 2.4. Useful *NoteText* Form

Apparently, useful features can be extracted from this piece of text, such as whether this was a review, which team took the review, whether the review success, when the review happened, and who's the batsman and umpire. However, some of the *NoteText* didn't record the team, in which case we need to manually extract the information.

- Remove Duplicate Records

What's more, we encounter duplicate match notes caused by the website's programmer, in which case we just simply keep the first record.

- Find Innings Number

In order to combine this data frame with the commentary data frame, we need to join them base on their match ID, innings number and overs number. But the inning number is missing in this data frame. Luckily, the match note records both team's innings in a nice form, which can be easily identified. However, if the first batting team A (by default, team A is the first batting team) has a big lead in the match, the other team B may bat the second and the third inning consecutively. And if team B can't bat more runs than team A after the third inning, the match will be over without the fourth inning. Therefore, we define a logical method to automatically calculate the innings number for all possible situations. The algorithm basically search the index of the first inning for both team forwardly and the second inning backwardly, and then calculate

the inning number for each match note based on the index. But again there are some records error need to be handled by hand as well.

2.3.2 Commentary Data

The original commentary data frame also need processing. With a larger size, this data frame is even messier than the match notes.

- Remove Useless Matches

First of all, matches with no DRS usage or really short text should be removed because they can't provide the information we want. And if these matches are included in the data, the outcome will be biased.

- Reconcile Over Number

The match information from the website is not maintained by only one programmer, so they might have different recording criteria. Specifically, some programmer record the over number from x.0 to x.5 while others record from x.1 to x.6. So we have to keep all records in a consistent form.

- Add No-Ball Column

In cricket, a no-ball is an illegal delivery to a batsman, in which case an extra delivery is given, with the same over number as the no ball delivery. Therefore, we add a no-ball column to distinguish no-ball and its next delivery, in order to avoid duplicates when joining two data frames.

2.3.3 Merging and Tagging

Now that both data frames are processed, we merge them into one table, and start tagging. Tagging is the process of defining multiple binary-categorial labels for each delivery based on its commentary. The following labels are what we put on the data.

- Review Reference : *True* if any review related word is mentioned in the commentary.
- Negative Review Reference : *True* if the delivery mentions 'review', but the review is not actually requested.
- Challenge Reference : *True* if the commentary mentions 'challenge'.

- Appeal Reference : *True* if the commentary mentions a positive appeal or shout, and also doesn't contain negations or exclusions such as 'half', 'muted' and 'died'.
- LBW Reference : *True* if the commentary mentions any leg before wicket related word.
- Caught Reference : *True* if the commentary mentions 'caught' or 'catch'.
- Not Out Reference : *True* if the commentary mentions a not out in the delivery.
- Replay Reference : *True* if the commentary mentions 'replay'.
- Conditional Refer Reference : *True* if the commentary mentions 'refer', and it's not a refer for boundary issues, like whether it's a four or a six run.
- Irrelevant Replay Reference : *True* if the commentary mentions irrelevant reasons for a replay.

After tagging, we have enough labels which enable us to classify every delivery. To be more specific, for every delivery, we want to know whether it's a review opportunity, whose review opportunity it is, and what's the confidence level of the opportunity.

2.4 Rule-based Classifier

A rule-based classifier is a decision tree with pre-defined nodes. Usually a decision tree is used in machine learning and the nodes are learned by maximizing the purity of the split child nodes. However, in this method, the nodes are defined in advance. The reason why we use this method is that, the data is not labelled in terms of the classes we need. More importantly, we are actually creating these classes by definition, which is consistent with this method. We classify deliveries in terms of three different labels: whether it's a review opportunity, whose review opportunity, and confidence level of the opportunity.

- Whether Review Opportunity

Figure 2.4 shows how the rule-based classifier identifies whether a delivery is a review opportunity. A red node indicates a review opportunity while a green one represents a negation. A yellow node is a flag mode, which means we can't classify the delivery. However, if a delivery has no explicit expression about being a review opportunity, it's

likely to be a negation. Even if it is a review opportunity, there will be very few such cases. So we also consider yellow nodes as not review opportunities.

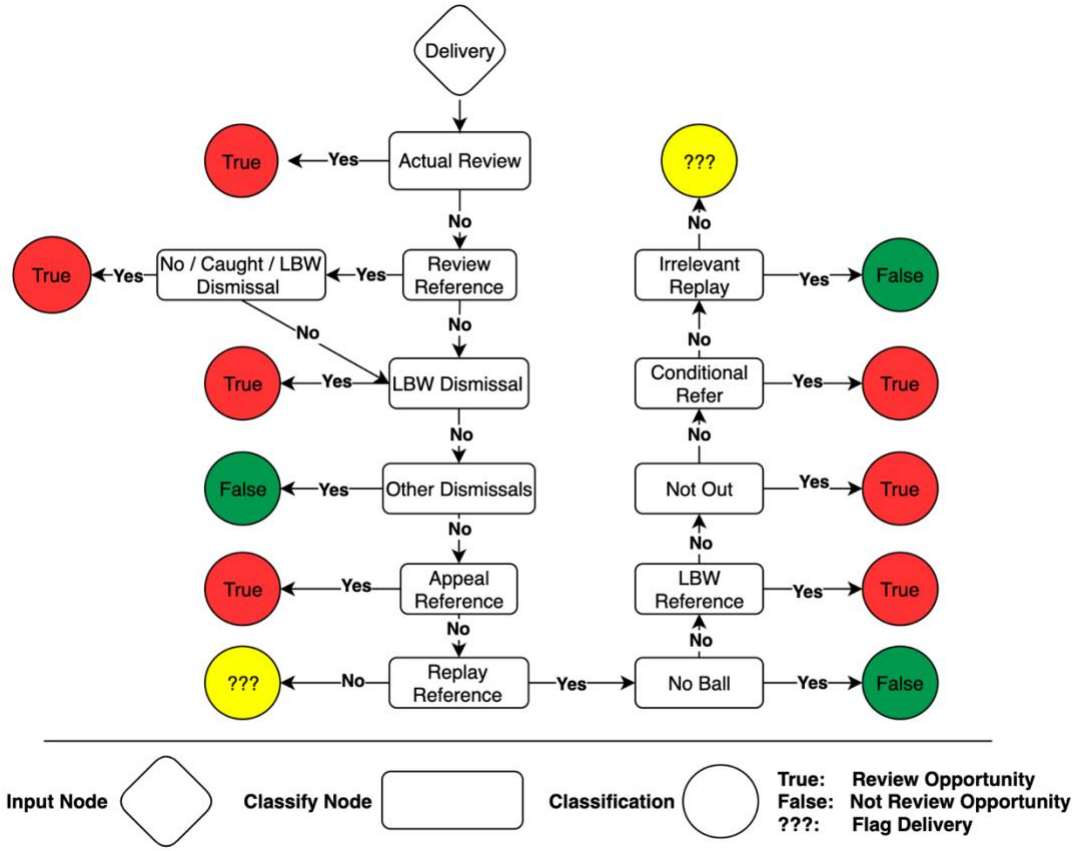

Figure 2.4. Rule-based Classifier

The ideas under this classifier are a bit complicated. First of all, if a delivery is an actual review, then it's a review opportunity. Otherwise, if it mentions 'review', and itself is an interested dismissal (caught or LBW) or not a dismissal, then it can also be a review opportunity. After that, delivery which mentions 'review' and is LBW dismissal, caught dismissal or not a dismissal, is filtered out. Then we again look at LBW dismissal and other dismissals. We are saying that, although the delivery doesn't mention 'review', it's still a review opportunity if it's a LBW dismissal. But if it's another type of dismissal, then it's not a review opportunity. By now the obvious opportunities are almost filtered out, we then identify less obvious ones. If the delivery mentions 'appeal', then it's an opportunity. Then we look for deliveries which relates to 'replay'. Usually a replay represents an opportunity, but not a replay for no balls. And it's an opportunity

if it's related to 'LBW', 'not our' or if it's a conditional referral, which means it's not a replay for boundary identification. At the end, if it's a irrelevant replay, we consider it as not review opportunity. What's left is flagged as unclassified delivery. It seems that we consider some words with overlapping meanings. Actually it might be the case, but we just want to cover as many situations as possible.

- Whose Review Opportunity

The classification of this label is simpler. For a reviewed delivery, we already have the information of whose review it is. For a non-reviewed delivery, if it cause a fall wicket, then it's batting team's opportunity, otherwise it's bowling team's.

- Confidence Level

In terms of confidence level classification, we define different confidence levels with our subjective feelings. For an actual review, the classification is clear. High confidence level for struck down, medium for umpire's call and low for up held reviews. As for non-reviewed delivery, if it contains strong phrases such as 'would have been overturned', we consider it as high confident opportunity. If it's a caught dismissal, we consider it high confident if it mentions 'review', otherwise it's medium confident. As for LBW dismissals, or just a reference of LBW, if it contains specific adjectives, such as 'big', 'sustained', 'huge' or 'massive', indicating a possible opportunity, then it's a medium. Also if the delivery says that it has no reviews left, it's an umpire's call, or it has a replay, then we also consider it as a medium confidence level. And the rest of them are all low confident opportunities. Clearly we only calculate the confidence level if it's a review opportunity.

By now, we have classified deliveries into different labels. We count the number of deliveries for each labels, and extract them as probabilities. There are some more probabilities needed before we build the SDP model. By definition, a high confident opportunity has a 90% probability to overturn the original decision. Such probability for medium and low confident opportunities are 50% and 10%. What's more, it's possible to result in umpire's call for a LBW review. We set this probability to 5%, 10% and 20% for high, medium and low confident opportunities respectively.

2.5 Failed Methods

We didn't choose to use this rule-based classifier at the beginning. Instead, we tried some other methods. Although they didn't work, I'll still briefly talk about them and also the reason why they didn't function well.

- Sentiment Analysis

At first, we thought we could classify the confidence level based on the sentiment of the commentary. This brings up the problem of how to quantify the sentiment of a sentence or a paragraph. We found some well-processed lexicon from the Internet, which contains sentiment scores for common words in English. By applying this method, we can have the sentiment score for each delivery, and classify them based on some sentiment score threshold. However, there are 2 main problems which make it not functional. Firstly, the words used in cricket commentary are not like those used in usual natural language processing settings, such as news or product comment. Many words used in cricket have different meanings than their usual use cases, which makes the general sentiment lexicon biased in this cricket commentary setting. Secondly, a piece of commentary can have both positive comment for batting team, and negative comment for bowling team, which makes the overall sentiment neutral and unclear.

- Multi-layer Perceptron (MLP)

Then we turned to machine learning for help. Like the usual natural language processing steps, the idea is to transfer each piece of commentary into an one hot encoding vector. Each word, which occurred in the whole commentary data, is a variable, and the value is the number of times it occurs in the comment of the delivery. Then we can train a MLP with some training data and might end up with a MLP model that can classify every unlabeled delivery. However, the problem is that we only have labelled data for actual reviewed deliveries. The model trained by this data might not be a good classifier for all deliveries, especially when non-reviewed deliveries consist of more than 99% of the data.

3 Explored Data Analysis

The data is well-processed and ready for use now. Before we build the SDP model, we first look at some statistics from the data. Charles Davis has done some statistical analysis on test matches from 2009 to 2016 [7], including DRS review results for different umpires and batsman. I intend to extend the time to 2020 and add more types of exploration.

3.1 DRS by year

drs_iteration	year	matches	review/match	overturn/match	overturn_rate(%)
1	2009	9	10.2	2.3	22.8
1	2010	20	8.9	2.7	30.3
1	2011	19	7.6	2.2	29.0
1	2012	26	10.0	2.7	26.4
2	2013	28	9.1	2.6	28.1
2	2014	28	11.0	2.5	23.1
2	2015	33	9.5	1.8	18.4
2	2016	37	11.7	3.5	29.8
2	2017	47	10.9	3.1	28.9
3	2018	47	11.0	3.0	27.0
3	2019	37	10.0	3.1	31.4
3	2020	8	7.8	2.5	32.3

Table 3.1. DRS by year

Table 3.1 shows the DRS usage per year. Noted that year 2009 and 2020 have less than 10 matches in the data, so the number might not be accurate for these two years. There seems to be no obvious increasing or decreasing trend in terms of review per match or overturn rate by year. The average review per match and overturn rate among all these years (excluding 2009 and 2020) are 9.97 and 27.24, so there's still a huge progress can be made about the DRS review.

Grouping by DRS iteration, the average review per match and overturn rate are 8.8, 10.4, 10.5 and 28.6, 25.7, 29.2 respectively. We know that there are two reasons can cause increase in review per match. One is the change of DRS rules, the other is the

increase of overturn rate. Comparing the average values for the 1st and the 2nd DRS Iterations, the 2nd DRS iteration has higher reviews per match but a lower overturn rate, which implies that, people are using DRS reviews worse in the 2nd DRS Iteration. The increase of reviews per match is simply because of the change or DRS rules (They allow DRS reviews reset every 80 overs). As for the 2nd and the 3rd DRS Iterations, they have nearly the same reviews per match, but the 3rd DRS Iteration has higher overturn rate. This implies that people use reviews better, but the rules reduce the available DRS reviews in a match in the 3rd DRS Iteration.

3.2 DRS by team

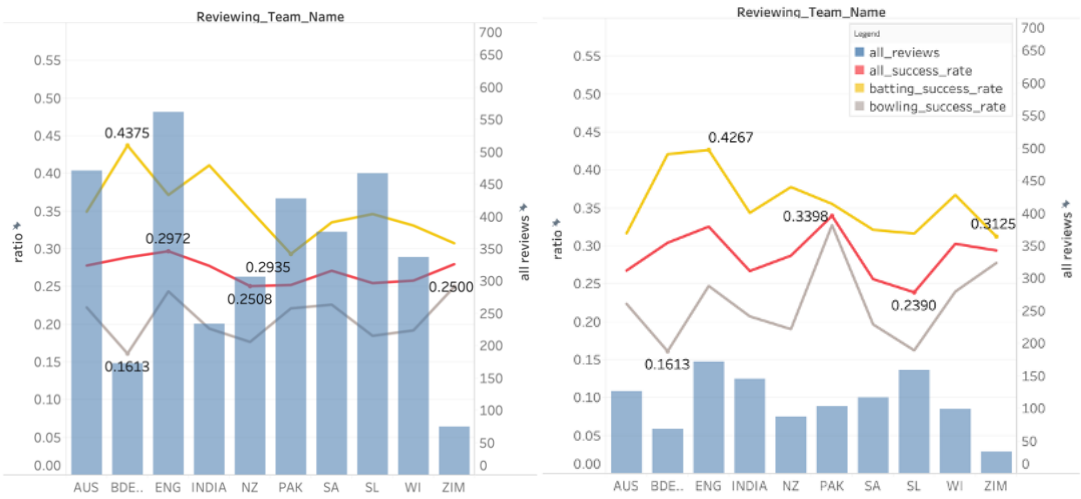

Figure 3.1. DRS by team (left: all DRS iterations, right: the 3rd DRS iteration)

Figure 3.1 shows the DRS usage by different teams. The abbreviation name from the left to right are Australia, Bangladesh, England, India, New Zealand, Pakistan, South Africa Sri Lanka, West Indies and Zimbabwe. The bar height indicates the number of reviews have been taken, and the yellow, red and grey line indicates the batting review success rate, overall review success rate and bowling review success rate respectively. In general, batting review success rate is higher than bowling's.

Looking at all historical data, England takes the most reviews, having the highest overall review success rate as well, while New Zealand has the lowest overall review success rate. Bangladesh is the best in batting review, but the worst in bowling review.

Pakistan is the worst batting review team while Zimbabwe is the best bowling review team.

When it comes to the latest DRS Iteration, Pakistan becomes the best overall review team, with the highest bowling review success rate, while Sri Lanka is the worst overall review team with a really low bowling review success rate. What's more, comparing to the batting review success rate in the left figure, England has a huge progress in improving their batting review skills, which makes it the best batting review team in the 3rd DRS Iteration.

3.3 DRS by over

We also want to see the DRS usage in terms of overs, which sort of represents the period of an inning. Noted that innings can have number of overs from 50 to 220. Therefore, only innings with overs between 100 and 130 are considered in the following figures.

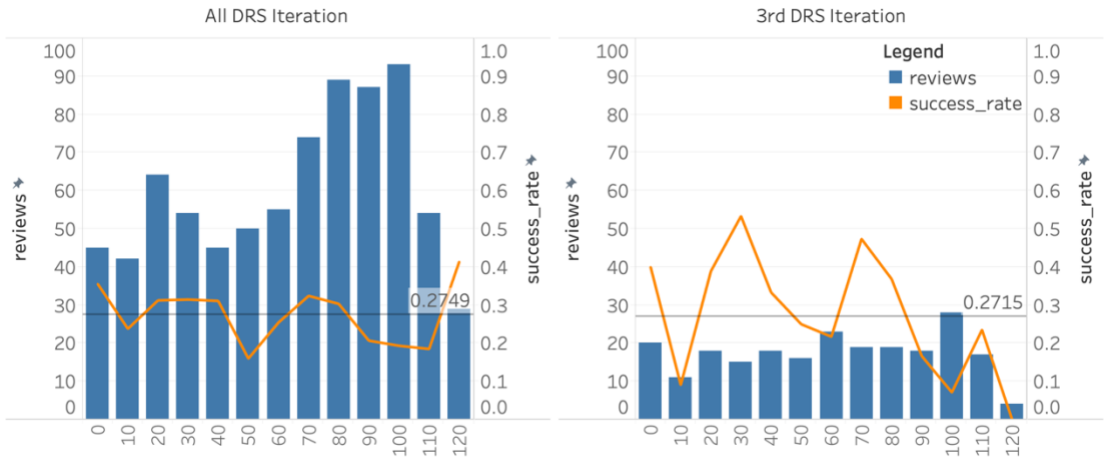

Figure 3.2. DRS by over (left: all DRS iterations, right: the 3rd DRS iteration)

First we look at the number of reviews (bar) and the overall review success rate (line) by over in terms of all DRS iterations and only the 3rd DRS Iteration. From the left, most DRS reviews are taken at 80, 90 and 100 overs, which is almost the end of the inning. However, the overturning rate is quite low, especially for 90 and 100 overs. This kind of implies that, it's not a good idea to save reviews to the end. Surprisingly, the overturing rate at the beginning of the inning is higher than the average number. As

for the 3rd DRS Iteration case, less data makes the success rate line more twisted. It seems reviewing at the beginning, 1/3 and 2/3 of the inning is a good idea.

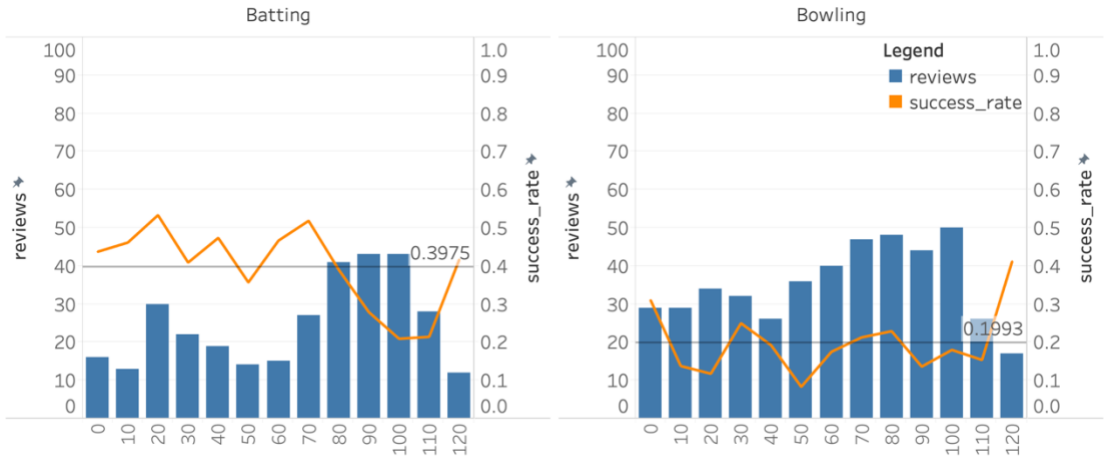

Figure 3.3. DRS by over (All DRS iterations)

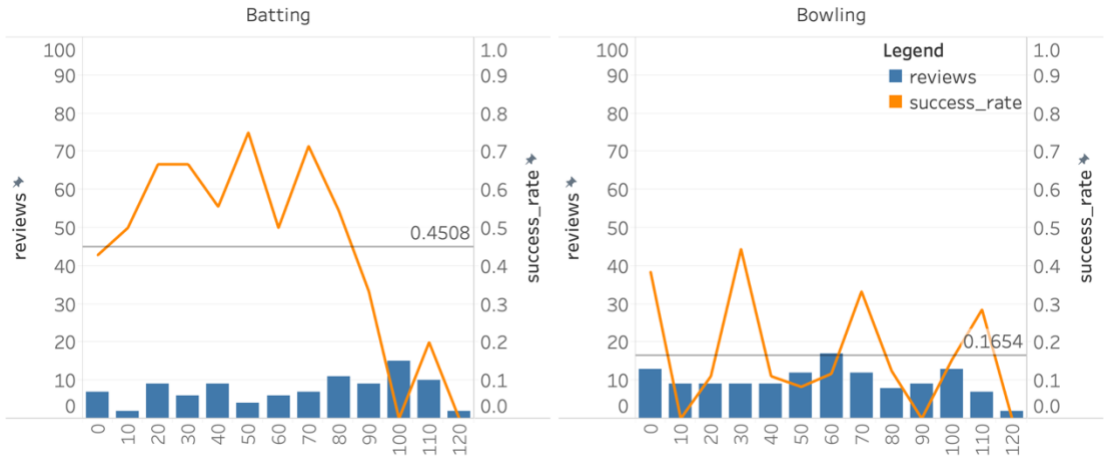

Figure 3.4. DRS by over (the 3rd DRS Iteration)

Next we look at the batting team review versus bowling team's. In terms of all the data, there are less batting reviews than bowling, but batting reviews has higher success rate. We already know that the overall review success rate is low at 90, 100, and 110 over. This conclusion is even more solid in terms of batting review. In the 3rd DRS Iteration, the difference is even larger. The success rate of batting review can reach 50% to 70% for most of the time. However, when near the end of the inning, it can be less than 30%, even 0%.

3.4 DRS by wicket

The number of remaining wickets is another measure for inning progress. An inning starts at 10 wickets and ends at 0. The x axis is sorted in a descending order to make it more intuitive.

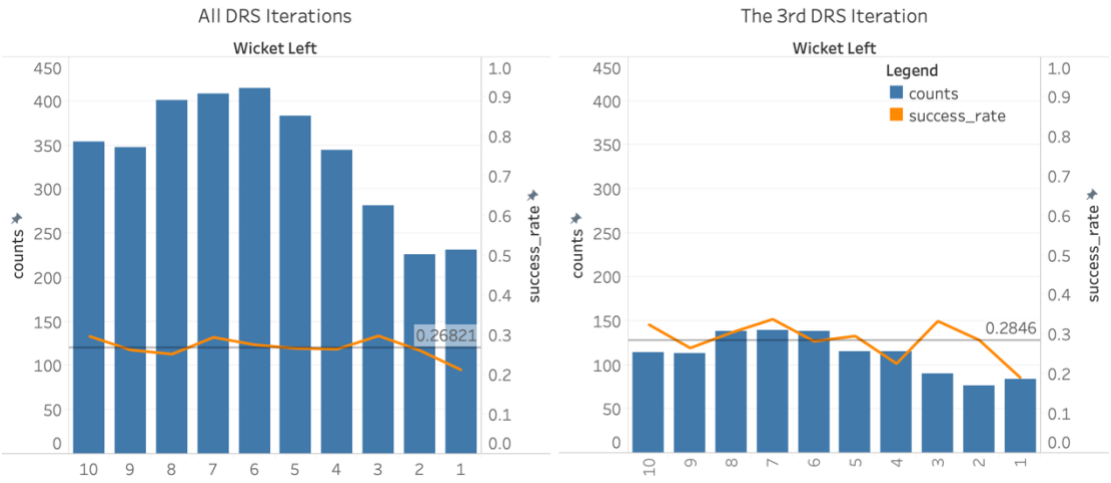

Figure 3.5. DRS by wickets (all DRS vs the 3rd DRS)

Comparing to figures from DRS by over, the review success rate line is less twisted. One reason might be that more match data is included so that the curve is softened. Another reason is that, the number of remaining wickets is a different metric than overs. For example, one might still have 10 wickets left, but he is already at the 50th over. Or one might still at the 20th over, but already lost 3 wickets. However, this still implies that the review success rate at the beginning is higher than the end.

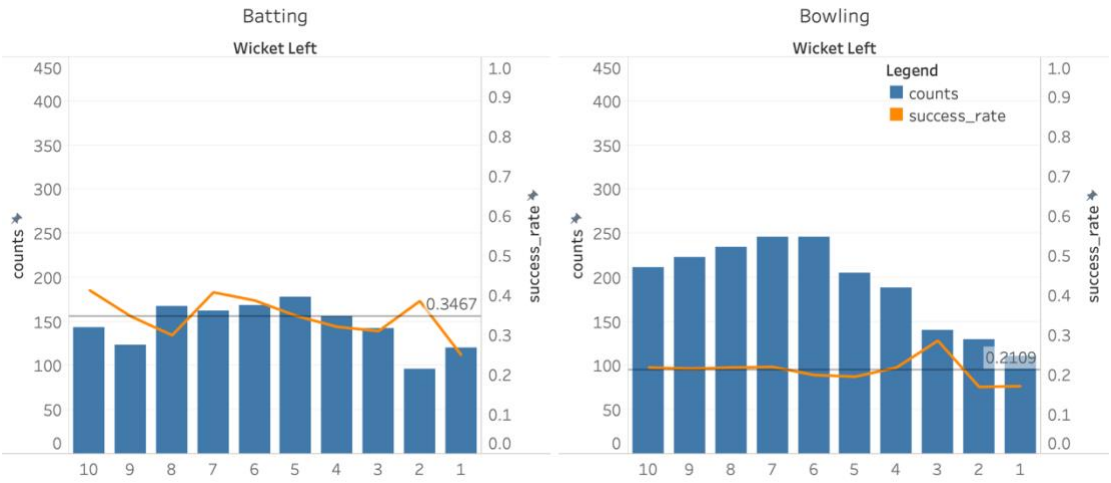

Figure 3.6. DRS by wickets (all DRS)

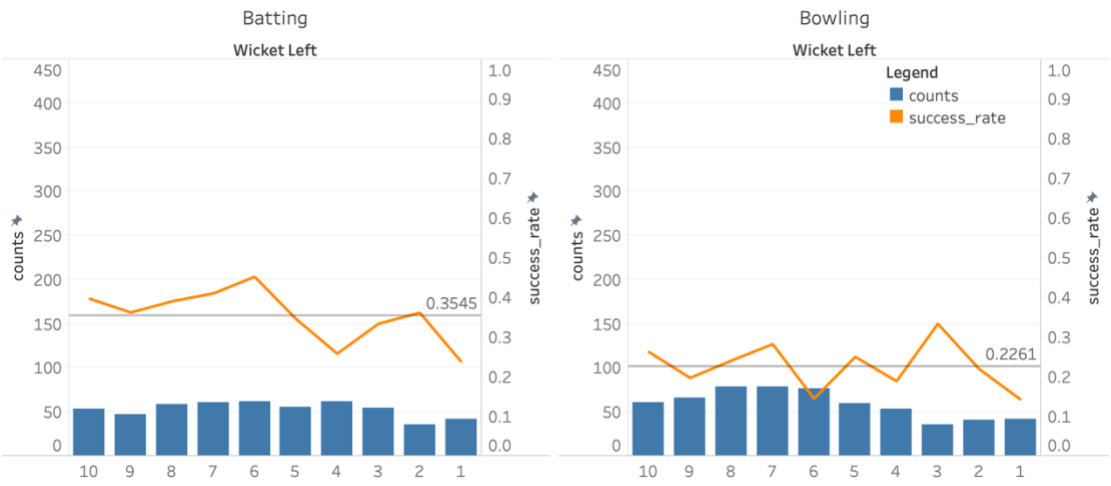

Figure 3.7. DRS by wickets (the 3rd DRS)

Figure 3.6 and 3.7 shows the DRS by wickets in terms of batting and bowling team. From figure 3.6, neither team prefer to save DRS reviews to the very end. This is a bit different than the conclusion we have when looking at figure 3.3. In my perspective, this figure is more reliable. What's more, the bowling team prefers to use DRS review at the first half of the inning. This is consistent with the idea that, bowling team wants to bowl out all wickets as soon as possible. As for figure 3.7, it has similar distribution and trend as figure 3.6, only less obvious because of less data.

4 Stochastic Dynamic Programming

4.1 Theory

Dynamic programming (DP) is a common technique for solving optimization problems, and stochastic dynamic programming (SDP) is a specific type of DP, where the world is stochastic and actions might be non-deterministic. In order to make DP functional, the problem should be able to be represented in discrete stage and state, where stage is usually time, indicating steps of the problem, and state contains the information of the world at some stage. The problem should also have a Markovian transition model, where the value function at present only depends on the current state, rather than the historical state sequence. The core concept of DP is closely related to the Bellman Equation [8](shown below), which is widely used in artificial intelligent, control theory, economy and other topics in applied mathematics.

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a)U(s')$$

In this formula, s is state, a is action, $A(s)$ is the available action set under state s, s' is the next possible state taking action a from state s with probability $P(s'|s,a)$, γ is a discounted factor. $U(s)$ is the utility/value at state s, and $R(s)$ is the reward at state s.

In general, the SDP formula (shown below) is a bit different than the Bellman Equation. It doesn't have a discounted factor, or you can consider it with discounted factor 1. And the reward function (negative reward equals to cost) are also based on the action you take, so the maximum symbol is outside of both value and reward functions.

$$V_t(s_t) = \max_{a_t \in A(s_t)} \left(R_t(s_t, a_t) + \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) V_{t+1}(s_{t+1}) \right)$$

Following the formula, in order to build a SDP model, we need to define the stage and the state, as well as the valid actions. We also need a transition function, and a value function. And the objective will be to maximize the value function at starting state.

4.2 Model Assumptions

There are many things going on in a test cricket match. So before we can build the SDP model, some assumptions need to be made to simplify the model.

- Terminating States

Test cricket has indefinite stages. Although it will definitely finish at some point, it could last for a really long time if you are lucky enough. So technically speaking, the stage can have infinite values. This is the computational problem of DP known as the dimensionality curse **Error! Reference source not found.**. In order to avoid this situation, the model will only simulate an inning with at most 1200 deliveries, which is 200 overs. It means that, other than losing all 10 wickets, the inning will also terminate when it reaches the 1200th delivery, even if it still have several remaining wickets. Therefore the model has finite states.

- LBW and Caught dismissals

In test cricket, reviews and replays can also be used under other situations such as run out dismissals or no balls. However, LBW and caught dismissals are the most common situation for DRS reviews, so the model will only consider these two types of dismissals.

- The 3rd DRS Iteration

The model will simulate the test under the 3rd DRS Iteration. So for a LBW review, other than success and failure, there's also an possible outcome as umpire's call. In this case, the original decision will still hold, but the reviewing team won't lose their review chance.

- Confidence Level

As mentioned in 2.4, we assume there are three types of confident level for a review opportunity. The high confident one has the probability of 90% to overturn and 5% to result in umpire's call. The corresponding probability for medium and low is 50%, 10% and 10%, 20%.

- Average Run Score

In practical, batting team may gain 1 or 2 run by running between wickets, or 4 or 6 run by striking the ball to the boundary in each delivery. In our model, we instead use an average run score, which is calculated from the data, for each delivery.

- Parameters based on remaining wicket

For parameters and probabilities calculated from data, we assume some of them are constant during the whole inning, while others differ as the number of remaining wickets reduce. Changeable parameters are the average run and the probability of a wicket falling.

4.3 Model Structure

In this cricket problem, the basic representations for stage, state and actions are the same as what Dr. Forbes proposed [9]. We consider stage t as the t^{th} delivery. State s is a tuple $(w, bat, bowl)$ where w is the number of remaining wickets, bat and $bowl$ are the number of batting and bowling team's remaining reviews. Actions are to review or not. The main recursive value function is shown below. What's more, $P_w(w)$ is the probability of a wicket falling, while $P_{-w}(w)$ is the probability

of a wicket standing. They both change with w, so does the average run $run(w)$. P_y^x indicates the probability of team x's y dismissal given a wicket falling or standing. The value of the value function is the cumulative run score from stage t and state $(w, bat, bowl)$ to the end of the inning. The transition function is embedded in four functions $BatLBW$, $BatCaught$, $BowlLBW$, and $BowlCaught$, where the model makes the decision of whether review or not.

$$V_t(w, bat, bowl) = \begin{cases} 0, if\ t = 1200\ or\ w = 0 \\ P_w(w) \times \begin{pmatrix} P_{lbw}^{bat} \times BatLBW(t, w, bat, bowl) + \\ P_{caught}^{bat} \times BatCaught(t, w, bat, bowl) + \\ \left(1 - P_{lbw}^{bat} - P_{caught}^{bat}\right) \times V_{t+1}(w-1, bat, bowl) \end{pmatrix} + \\ P_{-w}(w) \times \begin{pmatrix} P_{lbw}^{bowl} \times BowlLBW(t, w, bat, bowl) + \\ P_{caught}^{bowl} \times BowlCaught(t, w, bat, bowl) + \\ \left(1 - P_{lbw}^{bowl} - P_{caught}^{bowl}\right) \times \left(V_{t+1}(w, bat, bowl) + run(w)\right) \end{pmatrix} \end{cases}$$

The interpretation is clear. If it reaches 1200th delivery or 0 wicket, the inning terminates and returns 0 run score. Otherwise, it has two possible outcomes: a standing or falling wicket. On the one hand, if it's a falling wicket, then it has three further possible results: an LBW or caught review opportunity for batting team, or no review opportunities at all. On the other hand, if it's a standing wicket, then there can be bowling team's LBW or caught review opportunity, or no opportunities at all. The batting team gains an average run if the wicket stands and no review opportunities for bowling team.

$$BatLBW(t, w, bat, bowl)$$

$$= \begin{cases} V_{t+1}(w-1, bat, bowl),\ \ if\ bat = 0 \\ \sum_{P_{opp}, P_{success}, P_{umpire} \in Confidence} P_{opp} \times \max \begin{Bmatrix} V_{t+1}(w-1, bat, bowl), \\ P_{success} \times \left(V_{t+1}(w, bat, bowl) + run(w)\right) + \\ P_{umpire} \times V_{t+1}(w-1, bat, bowl) + \\ \left(1 - P_{success} - P_{umpire}\right) \times V_{t+1}(w-1, bat-1, bowl) \end{Bmatrix} \end{cases}$$

$$BatCaught(t, w, bat, bowl)$$

$$= \begin{cases} V_{t+1}(w-1, bat, bowl), if\ bat = 0 \\ \sum_{P_{opp}, P_{success} \in Confidence} P_{opp} \times \max \begin{Bmatrix} V_{t+1}(w-1, bat, bowl), \\ P_{success} \times \left(V_{t+1}(w, bat, bowl) + run(w)\right) + \\ (1 - P_{success}) \times V_{t+1}(w-1, bat-1, bowl) \end{Bmatrix} \end{cases}$$

$$BowlLBW(t, w, bat, bowl)$$

$$= \begin{cases} V_{t+1}(w, bat, bowl) + run(w), if\ bowl = 0 \\ \sum_{P_{opp}, P_{success}, P_{umpire} \in Confidence} P_{opp} \times \min \begin{cases} V_{t+1}(w, bat, bowl) + run(w), \\ P_{success} \times V_{t+1}(w-1, bat, bowl) + \\ P_{umpire} \times \left(V_{t+1}(w, bat, bowl) + run(w)\right) + \\ \left(1 - P_{success} - P_{umpire}\right) \times \left(V_{t+1}(w, bat, bowl-1) + run(w)\right) \end{cases} \end{cases}$$

$$BowlCaught(t, w, bat, bowl)$$

$$= \begin{cases} V_{t+1}(w, bat, bowl) + run(w), if\ bowl = 0 \\ \sum_{P_{opp}, P_{success} \in Confidence} P_{opp} \times \min \begin{cases} V_{t+1}(w, bat, bowl) + run(w), \\ P_{success} \times V_{t+1}(w-1, bat, bowl) + \\ (1 - P_{success}) \times \left(V_{t+1}(w, bat, bowl-1) + run(w)\right) \end{cases} \end{cases}$$

As the formula above shows, when batting team has review opportunities, the model compares the expected value of reviewing and the value of not reviewing. Then it chooses to take the action with the maximal value. If batting team has no reviews left, they simply lose one wicket. On the contrary, when it's bowling team's opportunity, they aim to minimize the value. And if bowling team has no reviews left, batting team gain the run score. In the formula, P_{opp} is the probability of a specific confident opportunity occurring. And $P_{success}$ is the probability of overturning the original decision under a specific confident opportunity. For LBW dismissals, there's also a P_{umpire} for umpire's call situation. Table 4.1 below shows the possible transitions for reviewing.

	Result	**Wicket**	**Run Score**	**Review**
Batting	Success	keep	keep	keep
Team's	Umpire's Call	lose	lose	keep
Review	Fail	lose	lose	lose
Bowling	Success	lose	lose	keep
Team's	Umpire's Call	keep	keep	keep
Review	Fail	keep	keep	lose

Table 4.1. Transitions for reviewing

5 Results

5.1 Expected Run

Given the model, we simply run $V(t = -1, w = 10, bat = 2, bowl = 2)$, and the model will recursively explore the value of further states until it comes to the end of an inning. Then it backdates the value up to the starting state, and gives us an expected run score. The parameters in the model are calculated based on complete innings (the 1st , 2nd and 3rd innings in a test match) under all DRS Iterations.

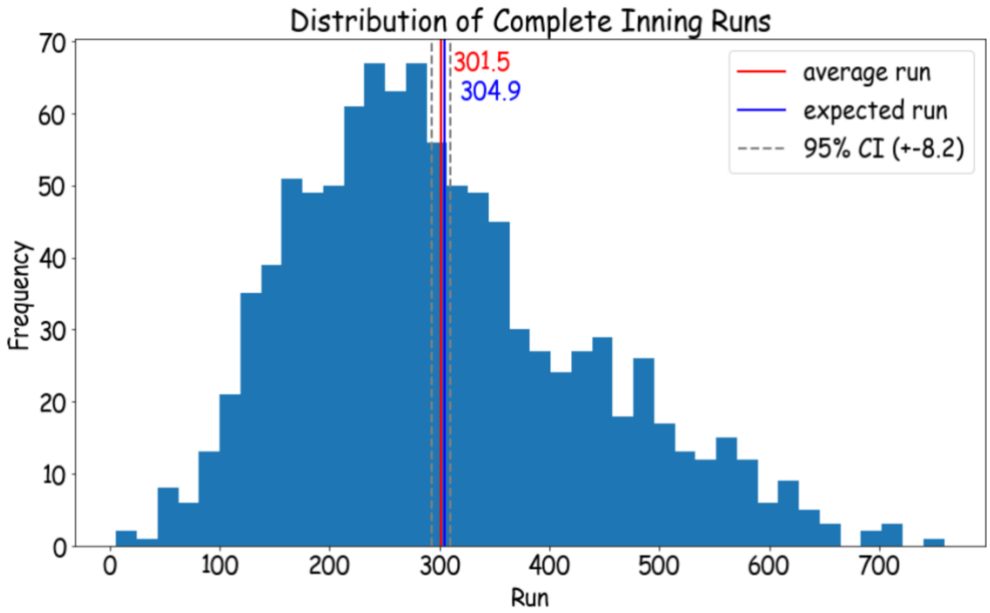

Figure 5.1. Innings Run Distribution

Figure 5.1 shows the distribution of inning run scores, with the mean of 301.5, and standard deviation of 132.9. The expected run calculated from the SDP model is 304.9, which is 3.4 larger than the average run. But it still lies in the 95% confident interval for the average run, which is [293.3, 309.7]. Therefore the SDP model is quite accurate in terms of the expected value.

5.2 Optimal Strategy

The expected run score gives us a sense about whether the model is correct. But in practice, we want a strategy to guide the decision making process. To get the optimal strategy, we first need to define what is the rational choice for a review opportunity.

- Definition of Optimal Strategy

Figure 5.2. Actions and Results for a review opportunity

Figure 5.2 shows possible actions and their results when it comes to a review opportunity. Thanks to the model, we have the expected run score for all these states. Apparently, it's better to review if the expected value for reviewing is larger than the expected value for not reviewing. Noted that the expected value of reviewing is a combination of the expected value of successful and failed reviews (not including the umpire's call case).

$$k \cdot E(successful\ review) + (1-k) \cdot E(failed\ review) = E(not\ review)$$

By making them equal to each other, we can get a threshold k where reviewing and not reviewing make no difference (shown in the formula above). And this threshold k, which can be considered as the confidence threshold, is the optimal policy.

- Insights of Optimal Strategy

Figure 5.3 and 5.4 below show the confidence threshold for batting and bowling team respectively. Each subplot is for a different *w*, where the x axis is the delivery *t*, and the y axis is the confidence threshold. The confidence threshold for each *(bat, bowl)* pair is displayed in different color.

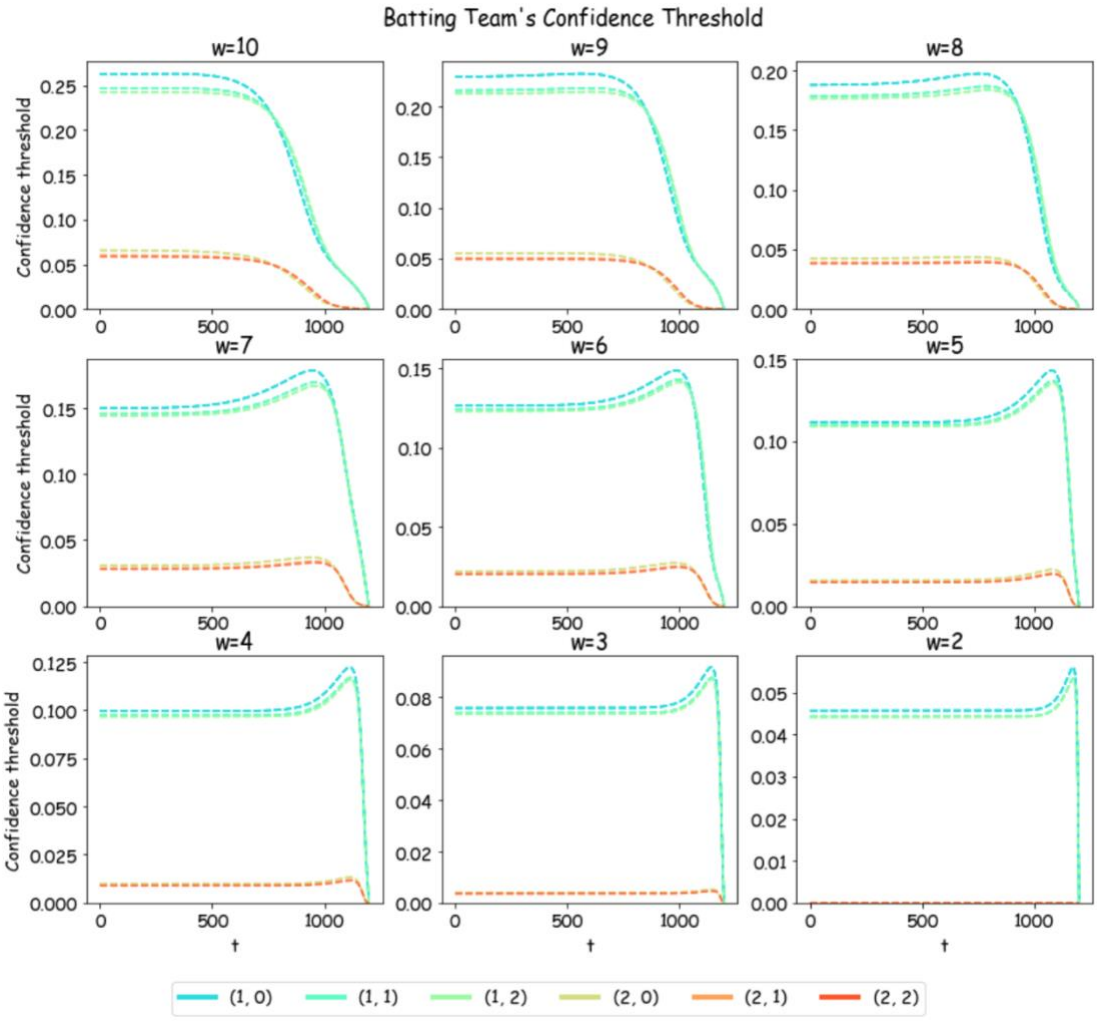

Figure 5.3. Confidence Threshold for Batting Team

From figure 5.3, we see that the confidence threshold for batting team is pretty low when they have 2 reviews. At the beginning of an inning, where *t=0* and *w=10*, you only need to be at least 6% confident about the review opportunity to review, which is basically saying, just take the DRS review once you have a review opportunity. When batting team only have 1 review left, the confidence threshold for *w=10* and *t=0* increase to 25%, where it's no longer rational to review at low confident opportunities (10% confident). However, you should always take the review if a medium or high confident opportunity is met. Moreover, if you have 2 wickets left and also 2 reviews left, then just take the review once there's a chance. Comparing different *(bat, bowl)* pairs, it seems that the number of bowling team's DRS reviews doesn't affect the strategy so much, especially when bowling team have 1 or 2 reviews left. Generally

speaking, as the inning going forward (*t* increasing) and the wicket falling, the confidence threshold decreases. Part of the curve may looks like a straight line, but it's still decreasing in a really small amount. There's a peak in the curve when *w<=7*. Actually if you look closer, *w=8, 9* is also the case. This is because the inning terminates at the 1200th delivery. So to make sure it can get to the end of the match, the model sacrifices redundant wickets and saves reviews for emergency.

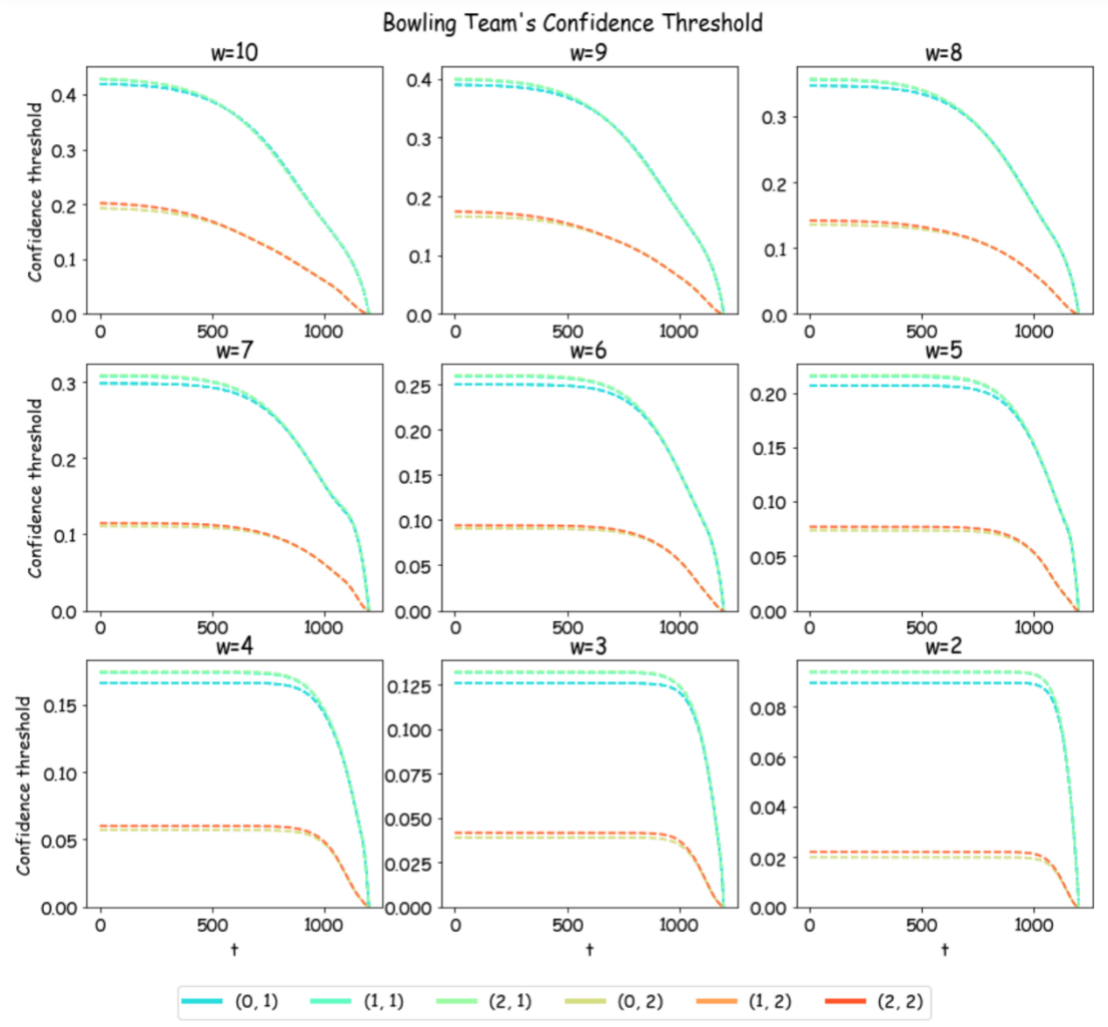

Figure 5.4. Confidence Threshold for Bowling Team

From figure 5.4, at *t=0* and *w=10*, the confidence threshold for bowling team is 20% when they have 2 reviews left, and 42% when they only have 1. Comparing to the batting team's number, the bowling team need to be more confident to take a review. The reason is that, bowing team has more review opportunities than batting team. So

they need to be more careful on DRS decisions. Other than that, bowling team's confidence threshold has some common properties as batting team. The confidence threshold decreases, as *t* grows and *w* decreases. And the number of batting team's review also doesn't affect bowling team's strategy much. But the confidence threshold is slightly higher if batting team has more reviews.

In order to take a closer look at the effect of the number of remaining wicket, I combine curves with different *w* values into one single plot. Figure 5.5 and 5.6 are the confidence threshold for batting and bowling team with *(bat=2, bowl=2)*.

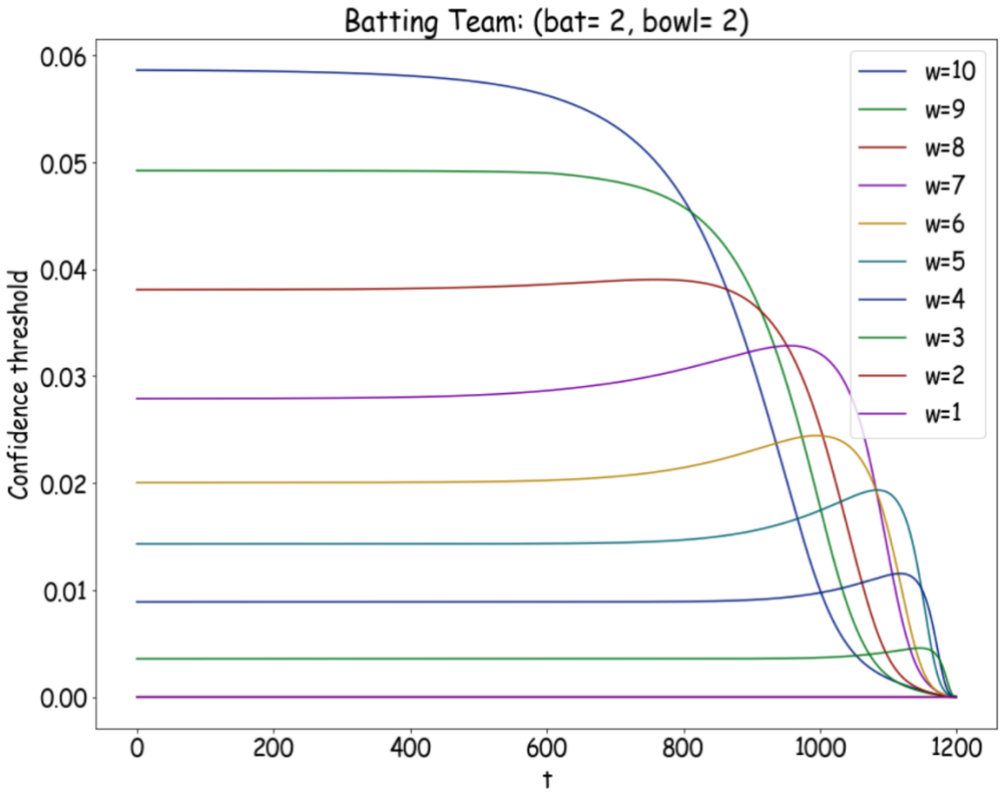

Figure 5.5. Confidence Threshold for Batting Team at (bat=2,bowl=2)

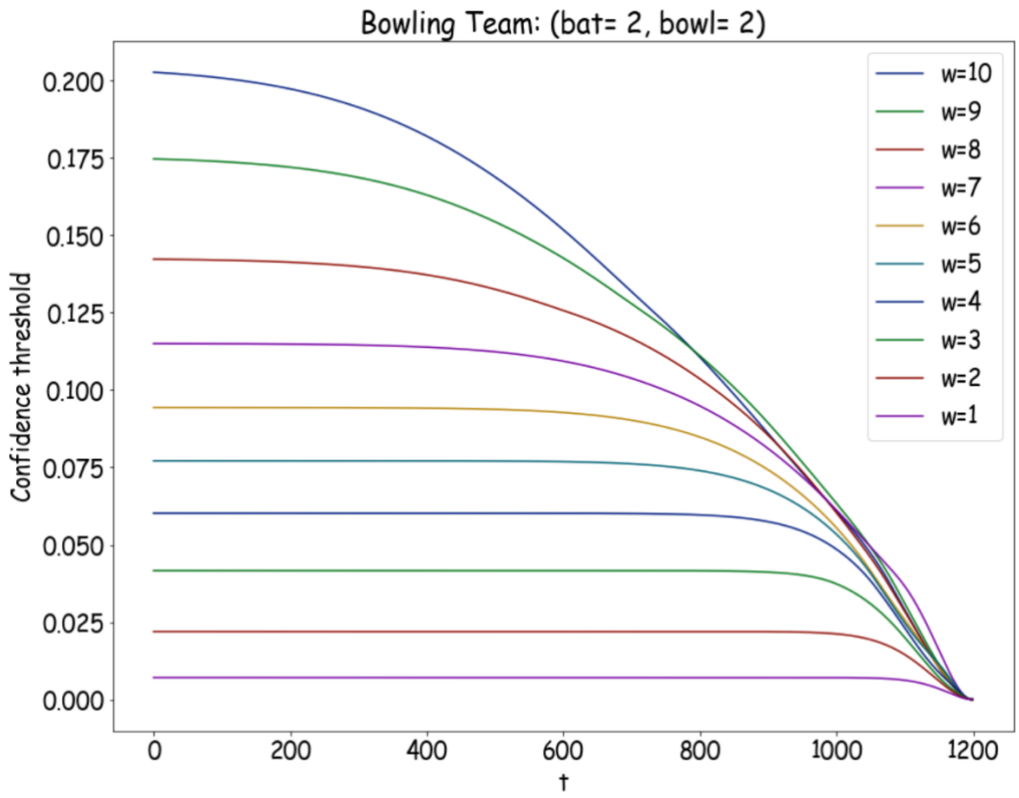

Figure 5.6. Confidence Threshold for Bowling Team at (bat=2,bowl=2)

Figure 5.5 shows some really interesting insights. A conventional thinking is that, the more wickets I have, the less confident I should be to take the review. Because with more remaining wickets, I am 'safer' and able to take more risks. However, the result from the SDP model implies that this isn't always the truth. As you can see in figure 5.5, before *t=800*, the more remaining wickets, the more confident it needs to be to take a review. And in figure 5.6, this is almost true for all t values. This is the opposite to the conventional thinking!

However, the confidence threshold becomes messy after *t=800* in figure 5.5. I wonder whether this relates to the termination at 1200th delivery. So I set the termination to *t=2400* and get figure 5.7.

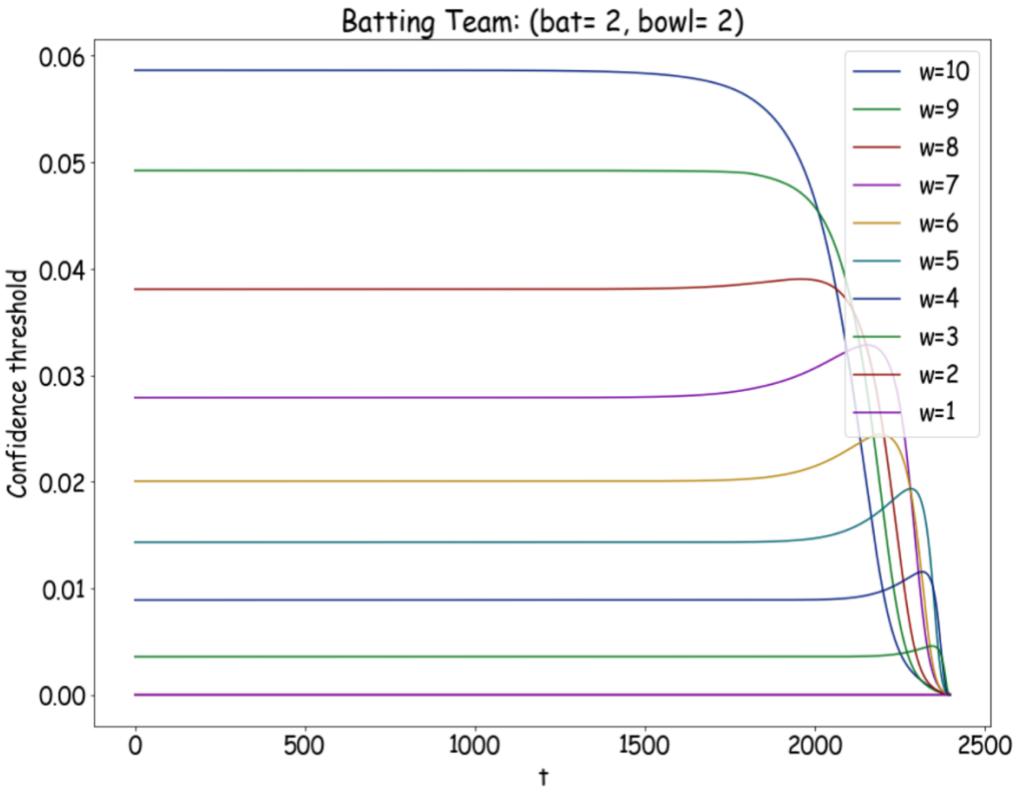

Figure 5.7. Confidence Threshold for Batting Team at (bat=2,bowl=2), t=2400

Comparing this figure to figure 5.5, they are quite similar. At the beginning, the confidence threshold is almost the same, and it gets messy after *t=2000*. Therefore, we can conclude that, the reason why confidence threshold becomes messy is because the inning terminates at some point, which makes DRS reviews less important at the end of an inning. For an infinite inning, the confidence threshold will have the same value as when *t=0.* Therefore, in practice, we can always assume there are 1200 more deliveries to go. And the confidence threshold at *t=0* will be the actual confidence threshold.

6 Discussions

In this section, we discuss some findings and problems given the SDP model. We compare the expected run score under different circumstances and check the sensitivity for parameters in the SDP model. We also look at how DRS have affected the test match, and talk about a problem when estimating DRS usage.

6.1 Expected Run Comparison

Comparison can be made between the expected run score at different states or different DRS rules, in order to show their impacts.

Case	State	Expected Run	Difference to baseline	Description
1	(-1,10,2,2)	305	0	Baseline / 3rd DRS
2	(-1,10,1,2)	302	-3	Bat=1, Bowl=2
3	(-1,10,0,2)	288	-17	Bat=0, Bowl=2
4	(-1,10,2,1)	315	10	Bat=2, Bowl=1
5	(-1,10,2,0)	346	41	Bat=2, Bowl=0
6	(-1,10,2,2)	306	1	1st DRS
7	(-1,10,2,2)	303	-2	2nd DRS
8	(-1,10,3,3)	300	-5	COVID-19 DRS

Table 6.1. Expected Run Comparison

From table 6.1, we set the baseline case 1 as when both team have 2 reviews and SDP under the 3rd DRS Iteration. We first look at the impact when batting or bowling team has 1 or 0 reviews. When batting team has 1 and 0 review, the expected run decrease 3 and 17 respectively. In terms of bowling team, the expected run increase 10 and 40 corresponding to them having 1 and 0 reviews. It seems that bowling team's reviews have greater impact on the expected run. And the second review is more important than the first one.

In cases 6 and 7, we look at the impact made by different DRS rules. The 1st DRS Iteration is almost the same with the 3rd, only without the umpire's call situation. Therefore the expected run for it is really closed to the baseline, with 1 run higher. As for the 2nd DRS Iteration, by resetting 2 reviews every 80 overs, the result is 2 runs lower. However, this is only true when everything else is fixed and only DRS rules is changed. In practice, this is definitely not the case. Players will modify their behavior based on different rules, which explicitly change all kinds of probabilities in the SDP

model. In fact, the average run score under different DRS Iterations from historical data is shown in figure 6.1 below. As you can see, compared to the 3rd DRS iteration, the average run for the 1st DRS increases about 18 runs, and the 2nd DRS iteration increases even more, about 31.5 runs. It seems that under the 2nd DRS iteration, resetting reviews makes innings last longer, with more runs per inning.

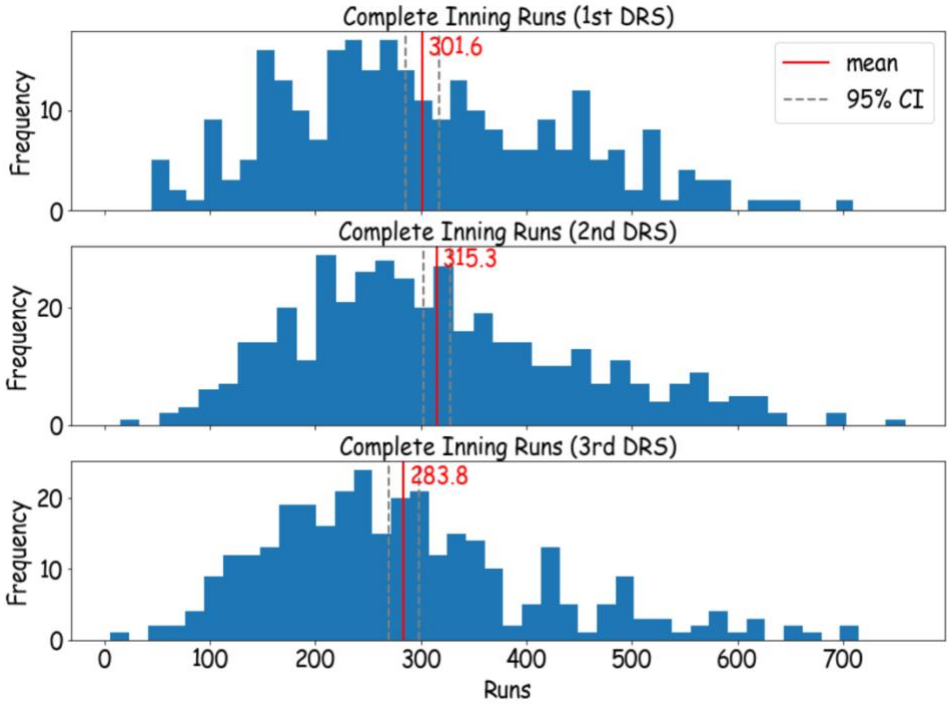

Figure 6.1. Distribution of Runs under different DRS Iterations

Case 8 is a new DRS rules proposed due to lack of experienced umpires during COVID-19 period. Both teams will have 3 reviews per inning. It seems that the bowling team will benefit under the new rules.

6.2 Sensitivity Analysis

Although most of the probabilities are calculated from the data, they may not be the same as the potential distribution in the real world. So it's better to check the sensitivity for these parameters.

- Probability of wicket falling

Table 6.2 shows how changing the probability of a wicket falling affects the expected run and confidence threshold. As mentioned in 5.2, the actual confidence threshold can be considered as the confidence threshold at *t=0*. So the confidence threshold in the table below is at *t=0* and *w=10.*

Change Rate	**Expected Run**	**Bat Conf Threshold (%)**		**Bowl Conf Threshold (%)**	
		(2,2)	(1,2)	(2,2)	(2,1)
0.1	695.2	0.5	4.5	22.4	43.1
0.3	635.2	4.1	18.9	25.4	44.5
0.5	523.9	5.2	22.7	27.4	47.7
0.75	392.5	5.7	23.8	24.5	46.4
1	304.9	5.9	24.2	20.3	42.9
1.5	208.9	6.0	24.8	14.6	36.1
2	158.4	6.1	25.0	11.8	31.1
5	62.7	6.4	25.7	5.4	17.1
10	29.2	6.5	26.1	1.8	9.9

Table 6.2. Sensitivity of P_w

As shown in table 6.2, changing P_w can affect the expected run much and the bowling team's confidence threshold as well. However, batting team's confidence threshold only fluctuates a bit. As P_w grows larger, the expected run becomes lower, and batting team's confidence threshold increases, while bowling team's decreases. But this is not always the case. I also test for extreme cases where change rate is very large or small. It seems that as the change rate increases, batting team's confidence threshold increases first and then fixes at around 6.5% and 26% for (*bat=2,bowl=2*) and (*bat=1,bowl=2*). However, as change rate increases, the bowling team's confidence threshold increases first but decreases later.

- Probability of batting team's opportunity

In this part I change both probabilities synchronously because they are both opportunities for batting team. As you can see, the change of this parameter doesn't affect bowling team's confidence threshold much, but affects the expected run and batting team's confidence threshold in a reasonable amount. When there are more batting team's review opportunities, the expected run increases, so does the batting team's confidence threshold.

Change Rate	**Expected Run**	**Bat Conf Threshold (%)**		**Bowl Conf Threshold (%)**	
		(2,2)	(1,2)	(2,2)	(2,1)
0.5	296.4	2.3	14.6	19.8	42.4
0.75	300.7	4.1	19.8	20.0	42.7
1	304.9	5.9	24.2	20.3	42.9
1.5	313.5	8.5	31.9	20.8	43.4
2	322.0	10.6	38.1	21.2	43.8

Table 6.3. Sensitivity of P_{LBW}^{bat} and P_{Caught}^{bat}

- Probability of bowling team's opportunity

Similar to the former part, here I change the probabilities of bowling team's review opportunity and leave other fixed.

Change Rate	**Expected Run**	**Bat Conf Threshold (%)**		**Bowl Conf Threshold (%)**	
		(2,2)	(1,2)	(2,2)	(2,1)
0.5	321.5	6.1	25.0	11.9	31.1
0.75	312.8	6.0	24.6	16.0	38.1
1	304.9	5.9	24.2	20.3	42.9
1.5	291.6	5.6	23.6	28.3	49.0
2	280.7	5.4	23.0	34.9	52.6

Table 6.4. Sensitivity of P_{LBW}^{bowl} and P_{Caught}^{bowl}

Comparing to the former part, changing the probabilities for bowling team doesn't change batting team's confidence threshold much. As bowling team's opportunities become larger, the expected run decreases and bowling team's expected run increases.

- Probability of high, medium and low confident opportunities

These parameters are not very sensitive, because the overall probability for them is very small. E.g. the probability of a high confident LBW opportunity equals to $P_w \cdot P_{LBW}^{bat} \cdot P_{high}$, which is really small. Actually, their sensitivity mostly depends on their prior probability.

6.3 DRS Impact Analysis

From 6.1, we conclude that bowling team's review is more powerful than batting teams, which means bringing in the DRS should decrease the expected run score in an inning. In order to confirm this conclusion, we check the average run score on historical matches where DRS wasn't used in. And only complete innings are considered as well.

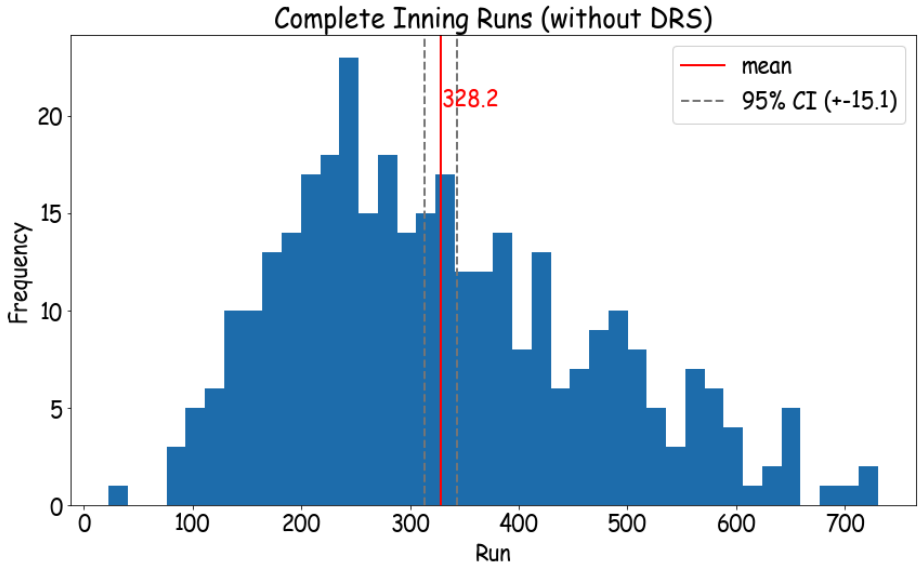

Figure 6.2. Distribution of Inning Run without DRS

From figure 6.2, the average run score from historical DRS-free matches is 328.2 which is larger than 301.5, calculated by matches with DRS. Therefore, bringing in DRS decreases the expected run from an inning.

6.4 Estimation of DRS usage

At first we proposed to compare the current team DRS usage to the optimal DRS strategy, in order to get insights about the current strategy. We thought maybe the confidence threshold can be considered as the success and failing rate for the optimal DRS strategy. However, this idea doesn't work. Although it's reasonable to say that, reviewing on a 10% confident opportunity will have 10% possibility to overturn the decision, no one will take a DRS review on such low confident opportunity under most of the time. Therefore this metrics can't be used to estimate DRS usage. Although we couldn't come up with an idea to extract the current DRS strategy, we figure another problem, which is about how to estimate DRS usage.

According to Andrew Wu (2019), Australia's captain Tim Paine is really bad at making DRS decisions with a wrong review rate of 87.1% while England's Joe Root is the best with a wrong review rate of 69.4%[10]. Andrew is judging captains' DRS decision making ability by the DRS success rate. However, we think this might not be the correct measurement. Apparently, if one only reviews for high confident opportunities, he/she will have a high review success rate for sure. But it doesn't mean this is a good strategy. In practice, comparing the review success rate is intuitionistic. However, DRS strategy with high success rate is not equivalent to the optimal strategy.

Reference

[1]. How does the DRS in cricket work [image] (2019). Retrieved from https://www.quora.com/How-does-the-Decision-Review-System-DRS-in-cricket-work

[2]. Shaw, A. K. (2019). The trick of DRS: When, where and how to use it? Retrieved from https://www.cricwizz.com/blog/2019/09/04/tricks-drs-use/

[3]. Aussies review DRS strategy. (2019). Retrieved from https://www.theaustralian.com.au/sport/cricket/aussies-review-drs-strategy/news-story/e59e2a48703696d6f3e8ee660c6d58bb/

[4]. How to crawl infinite scrolling pages using Python [image] (2019). Retrieved from https://www.accordbox.com/blog/how-crawl-infinite-scrolling-pages-using-python/

[5]. Hussainht. (2019, July 30). Match-data-scraper [Python file from Github]. Retrieved from https://github.com/hussainht/espncricinfo-scraper

[6]. Russel, S. & Norvig, P., (2010). Making Complex Decisions. *Artificial Intelligence, a modern approach* (pp. 645-655). Upper Saddle River, New Jersey: Pearson Education Inc.

[7]. Charles, D. (2017, June 1). The art of the review. Retrieved from https://www.espncricinfo.com/story/_/id/19835497/charles-davis-analyses-use-drs-players-teams

[8]. Bellman, R. & Dreyfus, S. (1962). *Applied Dynamic Programming*. Princeton: Princeton University Press.

[9]. Forbes, M. (2020). *Optimal Strategies in Cricket: An Operations Research Approach [PowerPoint slides]*. Unpublished manuscript, MATH7232, University of Queensland, St Lucia, Australia.

[10]. Andrew, W (2019, August 27). Devil's number: Paine no match for Root when it comes to DRS. Retrieved from https://www.smh.com.au/sport/cricket/devil-s-number-paine-no-match-for-root-when-it-comes-to-drs-20190827-p52l1z.html