2017 – 2020 (Pre-Covid) Wicket Analysis of Virat Kohli: When does Virat Kohli get out?

Mohit Soni\*, Sundar Srinivasan†

\*Tesla STEM High School, Redmond, USA. Email: [mohitksoni@outlook.com](mailto:mohitksoni@outlook.com)

†[Insert Heading] University of Washington, Seattle, USA. Email: [Insert Email]

***Abstract* – With an international career longer than a decade, and a total of 70 international centuries, the wicket of Virat Kohli is a significant achievement for a bowler. Therefore, it is important to analyze the wickets of Virat Kohli so patterns can be found, if there are any. Using a database from kaggle.com as well as data on cricbuzz.com, multiple features were engineered such as the line, length, speed, bounce, etc. of the delivery via web scraping and text parsing. In addition to features about each delivery, data was collected on the location of the game, format of the game, etc. The engineered features were used to aid in the procedure of classifying deliveries to wicket/non-wicket instances for Virat Kohli. During this process, the major problem that came up was that of imbalanced data as non-wicket balls greatly outnumber wicket balls for Virat Kohli. The hindrance made it difficult to create a high accuracy model therefore providing no conclusive evidence for deliveries that may get Virat Kohli’s wicket. Even without convincing results, this study may be able to motivate other Machine Learning researchers to learn from this study’s mistakes/successes and find definite results.**

1. **BACKGROUND**

Virat Kohli, born in New Delhi, India on November 5th, 1988, is an Indian international Cricketer. Learning Cricket in his birthplace, Delhi, Kohli developed into a talented player and represented Delhi in Ranji Trophy, a tournament that serves as the steppingstone to the Indian international team. Kohli’s first major success came in 2008 when he captained the India Under-19 team to victory at the 2008 Under-19 World Cup. Soon after this achievement, he was selected to play for the Indian international team. Since his debut in August of 2008 when he was a 19-year-old, Kohli has had a tremendous international career scoring approximately 23,693 runs with an average of 53.72 runs per match across all formats of the game, an average better than the all-time top scorer, Sachin Tendulkar. Being only the third batsman ever after Sachin Tendulkar and Rahul Dravid to score over 20,000 runs in the international arena and a winner of the ICC World Cup, Kohli has been deemed as the greatest batsman of modern-day cricket. In addition to his batting achievements, Kohli has also served as India’s captain across all 3 formats (T20, ODI, Test) from 2013 to 2022. If not for the right hander, the Indian international cricket team may never have been able to develop an aggressive and optimistic attitude in its playing style which in turn may have restricted the team’s success that it claims of today.

**Fig. (1)** Virat Kohli (Virat)

1. **DATA PREPARATION APPROACH**

Data Acquisition

The data used in this research was acquired from kaggle.com (link is available in the Works Cited, Section VI). The dataset is managed by Raghuvansh Tahlan and includes all matches – International, IPL (Indian Premier League, BBL (Big Bash League), and PSL (Pakistan Super League) – from 2017 until pre-covid 2020, covering over 1200 Cricket matches. “INTERNATIONAL\_MATCH.csv” serves as the file in which superficial information is contained for every match i.e., Name of the teams playing, unique team ID, venue, unique venue ID, match date, match result, and most significant of all is the Match Number, which can be used to obtain more data on each singular match. The “COMMENTARY\_INTL\_MATCH” folder contains ball-by-ball information for every match (i.e., what happened on each delivery, commentary, total runs on delivery, score, bowler name, batsman name), which can be accessed via the naming convention “XX\_COMMENTARY.csv” where “XX” is the Match Number. Thus, for one match, the Match Number must be found which can then be used to retrieve more data for that match.

Data Formatting

After downloading the data from kaggle.com and loading “INTERNATIONAL\_MATCH.csv” into Python via the pandas command read\_csv(), the file was used to filter out every international India match. Every India match had a unique Match Number which was extracted from the file as a pandas Series. As a preliminary formality, the Series containing Match Numbers was checked for duplicate values (there were none) as well as null values (there were none). Additionally, the Match Numbers were casted from floats to integers (1062574.0 🡪 1062574) so later, the usage of these Match Numbers to access other files becomes simpler. The retrieved Match Numbers were used to open all the files in the “COMMENTARY\_INTL\_MATCH” folder that corresponded to international matches of India. One Match Number was removed from the Series because it did not have a corresponding file in the “COMMENTARY\_INTL\_MATCH” folder. Deliveries played by Virat Kohli were extracted from the opened individual match files and were all appended to a new data frame. From the “INTERNATIONAL\_MATCH.csv”, data regarding the location of each match as well as the format of each match was also added to the new data frame. From the new data frame, deliveries when Virat Kohli got run out were removed as the motive of the research is to help a bowler get Virat Kohli out; however, most of the time, bowlers are not involved in run outs.

Data Exploration

Chart, box and whisker chart

Description automatically generatedChart, bar chart

Description automatically generated

**Fig. (3)** Virat Kohli average runs for each dismissal type

**Fig. (2)** Virat Kohli average runs against different teams

Feature Engineering

Graphical user interface, text, application

Description automatically generatedBased on the existence of keywords in the commentary acquired from the individual match files from the “COMMENTARY\_INTL\_MATCH” folder, the line, length, bounce, and the location of where each ball traveled in the field was approximated. The speed of each ball was estimated via web scraping the espncricinfo.com website and finding information about the bowling style of the bowler. Based on the bowling style, the speed of the bowler was projected by using a random number generator in a specified range. For example, a bowler with a bowling style of “Fast Medium” will usually bowl in the range of 128 to 139 km/hr. Similarly, a bowler with a bowling style of “Spinner” will usually bowl in the range of 70 to 95 km/hr. In the end, after engineering the additional features, a final data frame was created which included the Line, Length, Location, Speed, and Bounce of the ball. The Bowling Style of the bowler was also added to the data frame as well as the Game Location, Format, and a final column, representing the target variable, distinguishing whether the delivery was a wicket ball (1) or not (0).

**Fig. (4)** espncricinfo.com page for Mustafizur Rahman. The “BOWLING STYLE” was extracted using web scraping. (Mustafizur)

Data Preparation for Logistic Regression

The first requirement of Logistic Regression is that the target variable must be binary. In terms of the final data frame, the target variable, representing whether the delivery was a wicket ball, was binary, passing the requirement.

Chart, bar chart

Description automatically generated

**Fig. (5)** Checking Binary Condition

The second requirement of Logistic Regression is the absence of missing/unreasonable values. The Bowling Style feature had 255 null values. This occurred because the feature was engineered through web scraping the espncricinfo.com site which did not contain information for some bowlers. As the Speed feature was derived from the Bowling Style feature, the Speed feature also had 255 null values represented as -1’s. Thus, the 255 rows out of the 11,135 total rows were removed to eliminate all null values leaving 10,880 rows of data. In addition to the null value problem, the observed minimum of the Speed feature was 0 km/hr which is an unreasonable value as the lowest speed value as set by the speed ranges was 40 km/hr. As a result, all the rows where the speed was equal to 0 km/hr were removed. Still, the minimum value of the Speed Feature was 1 km/hr, again an unreasonable value. Similar to before, all the rows where the speed was equal to 1 km/hr were removed. After these deletions, the minimum of the Speed feature became 40 km/hr which was a reasonable value.

The third requirement of Logistic Regression is to have all features (both categorical and numerical) represented as numbers. The only numerical feature in the final data frame was Speed, all other features were categorical – Format, Game Location, Line, Length, Delivery Location, Bowling Style, and Bounce. The features Format, Line, and Length were converted to numerical values using One Hot Encoding as they did not have several unique values and they did not store ordinal data. However, the same technique of One Hot Encoding could not be used for Game Location and Bowling Style since they had too many unique values. To reduce the unique values for the Game Location feature, instead of storing cities, the feature was converted to store countries, which cut down the unique values from 56 to 8. Similarly, by grouping together different bowling styles, the unique values of the feature reduced from 24 to 6. The reduction made both features, Game Location and Bowling Style, eligible for One Hot Encoding. Finally, the remaining features, Delivery Location and Bounce, were not eligible for One Hot Encoding as the Delivery Location feature had too many unique values which could be cut, and the Bounce feature stored ordinal data. Thus, these remaining features were converted to numbers using Label Encoding.

The fourth requirement of Logistic Regression is to not have any dependency between separate features. To clarify, if one feature is One Hot Encoded into several different features, dependency between those features is valid as they are not “separate features”. As per Fig. (6), the features Length: Good and Bounce are the only separate features that have a trace of dependency. Since the feature Length: Good is created from One Hot Encoding, it is related to other features and therefore needs to be present in the data frame. Thus, the Bounce feature is removed.

Chart

Description automatically generated

**Fig. (6)** Independence Plot between features

The final requirement of Logistic Regression is to have at least 50 values per every predictive feature or in other words:

Where is the # of features and is the # of datapoints

The final data frame passes this requirement as:

1. **CLASSIFICATION APPROACH**

The target variable is extremely imbalanced with 99% of the values as 0 (non-wicket delivery) and 1% of the values as 1 (wicket delivery). Therefore, several approaches were carried out to combat this problem of imbalance:

Chart, pie chart

Description automatically generated

**Fig. (7)** Target Variable Pie chart

Logistic Regression Classification

1. Over-Sampling Minority Training Data
   1. Split data into train & test data
   2. Oversampled training data using RandomOverSampler() from imblearn.over\_sampling
2. Under-Sampling Majority Training Data
   1. Split data into train & test data
   2. Undersampled training data using RandomUnderSampler() from imblearn.over\_sampling
3. Both Over-Sampling and Under-Sampling Training Data
   1. Split data into train & test data
   2. Oversampled minority class until it is 20% the size of majority class
   3. Undersampled majority class
4. Penalizing Logistic Regression more for mistakes on Minority Class
   1. Split data into train & test data
   2. Set Logistic Regression model with class\_weight = ‘balanced’

After applying techniques to resolve the problem of imbalanced data, the Logistic Regression model was trained on the training data and was tested on the test data. Results can be seen in the (IV) Section.

Random Forest Classification

Similar to Logistic Regression, the same approaches were carried out to combat the problem of imbalance with the Random Forest Classification Algorithm as well:

1. Over-Sampling Minority Training Data
   1. Split data into train & test data
   2. Oversampled training data using RandomOverSampler() from imblearn.over\_sampling
2. Under-Sampling Majority Training Data
   1. Split data into train & test data
   2. Undersampled training data using RandomUnderSampler() from imblearn.over\_sampling
3. Both Over-Sampling and Under-Sampling Training Data
   1. Split data into train & test data
   2. Oversampled minority class until it is 20% the size of majority class
   3. Undersampled majority class
4. Penalizing the Random Forest model more for mistakes on Minority Class
   1. Split data into train & test data
   2. Set Random Forest model with class\_weight = ‘balanced’

After applying techniques to resolve the problem of imbalanced data, the Random Forest model was trained on the training data and was tested on the test data. Results can be seen in the (IV) Section.

1. **RESULTS**

|  |  |
| --- | --- |
| Logistic Regression Approach 1  Accuracy = 0.5068997240110396  Confusion Matrix: [[1085 1059]  [ 13 17]]  Not out accuracy = 0.98816029143898  Out accuracy = 0.015799256505576207  ROC\_AUC Score: 0.5765080845771144 | Logistic Regression Approach 2  Accuracy = 0.5068997240110396  Confusion Matrix: [[1085 1059]  [ 13 17]]  Not out accuracy = 0.98816029143898  Out accuracy = 0.015799256505576207  ROC\_AUC Score: 0.5765080845771144 |
| Logistic Regression Approach 3  Accuracy = 0.6039558417663293  Confusion Matrix: [[1301 843]  [ 18 12]]  Not out accuracy = 0.9863532979529946  Out accuracy = 0.014035087719298246  ROC\_AUC Score: 0.524922263681592 | Logistic Regression Approach 4  Accuracy = 0.6011959521619136  Confusion Matrix: [[1294 850]  [ 17 13]]  Not out accuracy = 0.9870327993897788  Out accuracy = 0.015063731170336037  ROC\_AUC Score: 0.5586909203980099 |

|  |  |
| --- | --- |
| Random Forest Approach 1  Accuracy = 0.9655013799448022  Confusion Matrix: [[2099 45]  [ 30 0]]  Not out accuracy = 0.9859088774072334  Out accuracy = 0.0  ROC\_AUC Score: 0.5694651741293532 | Random Forest Approach 2  Accuracy = 0.5354185832566697  Confusion Matrix: [[1144 1000]  [ 10 20]]  Not out accuracy = 0.9913344887348353  Out accuracy = 0.0196078431372549  ROC\_AUC Score: 0.5849657960199004 |
| Random Forest Approach 3  Accuracy = 0.9001839926402944  Confusion Matrix: [[1954 190]  [ 27 3]]  Not out accuracy = 0.9863705199394245  Out accuracy = 0.015544041450777202  ROC\_AUC Score: 0.5547419154228855 | Random Forest Approach 4  Accuracy = 0.9714811407543699  Confusion Matrix: [[2112 32]  [ 30 0]]  Not out accuracy = 0.9859943977591037  Out accuracy = 0.0  ROC\_AUC Score: 0.580185012437811 |

1. **CONCLUSION**

In this paper, we have used Virat Kohli’s batting data to find specific deliveries that may have a high chance of taking Kohli’s wicket. However, due to the problem of the severe imbalance in the target variable as well as the superficial approach taken when feature engineering, the results of both the Logistic Regression and Random Forest model are not convincing. Nevertheless, other researchers may be inspired by this paper and may want to perform further work learning from the study’s mistakes and successes to find convincing results.

1. **WORKS CITED**

ESPN Digital Media Pvt Ltd. (n.d.-a). *Mustafizur Rahman profile and biography, stats, records, averages, photos and videos*. ESPNcricinfo. Retrieved July 22, 2022, from https://www.espncricinfo.com/player/mustafizur-rahman-330902

ESPN Digital Media Pvt Ltd. (n.d.). *Virat Kohli profile and biography, stats, records, averages, photos and videos*. ESPNcricinfo. Retrieved July 22, 2022, from https://www.espncricinfo.com/player/virat-kohli-253802

Tahlan, R. T. (2021, April 13). *CRICKET SCORECARD AND COMMENTARY DATASET*. Kaggle. Retrieved May 22, 2022, from https://www.kaggle.com/datasets/raghuvansht/cricket-scorecard-and-commentary-dataset