Increasing Ad Returns in Live Cricket by Taking Advantage of Ad Receptiveness Changes

Debabrata Champati [1] **Tong Liu** [2] **Chen Qinghan** [3] **David Raj** [4] **Jonathan Simon Wagner** [5]

https://github.com/jonathanswagner/BT5153

1. Introduction

CRICKET, being one of the most popular sports in the world, counts more than 2 billion fans across the globe. Depending on the category of the match, it can last from five hours to as long as five days. Known for its exhilarating ambience, cricket matches draw in a lot of investment from the cricketing councils, sponsors, audience and media companies. Every minute in the game is crucial and a lot of strategy goes into the game from each team. Earlier, the analysis of the drivers of the game and prediction of the results could only be done by the Cricket Gurus. However, with the advent of machine learning and data sciences, it has become easier to draw insights and leverage it to predict the performance at various levels of the game by analysing past performance. Of course, there are innumerable factors that one might fail to take into account- for example changes in strategy, instantaneous decision of the players to introduce variations in their playing style, the data for which might not even be captured. However, the focus here is to generalise through few assumptions to understand the odds behind a certain outcome.

2. Cricket Overview

A match of cricket is played between two teams (eleven players each) - with one team batting first while the other team bowls. There are two players from the batting team at one time on the cricket pitch while the bowling team has all the eleven players who have different roles (for example bowlers, fielders and wicket keeper). The bowling team is supposed to carry out the specified number of bowls by rotating their bowlers. One over consists of six such continuous bowls bowled by one bowler. Depending on the type of match the team competes in – T20 or One day – the batting team gets to play 20 or 50 overs respectively. The batting team scores runs by running across the pitch or by hitting the ball across the boundary for maximum runs. The bowling team tries to minimise the runs by effective bowling, fielding or by getting the batsmen out, for example, by bowling in such a way that the batsman misses the ball and it hits the stumps. The batsman then gets replaced by another batsman from the batting team. The bowling team continues to bowl until the completion of all overs or till all the wickets have been taken. In the second innings, the bowling team gets to bat and chase the score. The number of runs scored in each ball can vary from 0-6 if scored only by bat. However, the extra runs might vary depending on whether an unfair ball is delivered by the bowler, in which case the batting team gets extra runs.

[1]A0206519E [2]A0206538A [3]A0206525L [4]A0056525E [5]A0152784X. Correspondence to: Jonathan Simon Wagner <e0024890@u.nus.edu>, Tong Liu <e0427366@u.nus.edu>, David Raj <david.raj@u.nus.edu>, Debabrata Champati <debabratachampati@u.nus.edu>, Chen Qinghan <qinghan.chen@u.nus.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

3. Project Objective

Every ball of the game is important and plays a critical role in shaping the score. The focus of the prediction model in this case is the outcome of each ball inside a game basis the number of runs scored. Each ball/delivery by the bowler can be considered an independent event which is affected by countless factors. However, there are certain important factors that we consider to explain the outcome which can be grouped as follows:

- **Player attributes and performance** Just by the design of the game, it is easy to say that the characteristics of the bowler, batsman and the attributes of the pair are important drivers for the outcome of the event. Hence, the type of bowler (for example, right hand spin or right hand fast) and his bowling statistics (recent as well as historical) along with the type of batsman (for example, right handed opener) and his batting statistics are the key factors to be taken into account
- **Team attributes and performance** A team's performance would depend on the performance of the individual players along with the synergy effects. In order

to capture this synergy, the team attributes (for example, the division of batsman-bowler in the team) are important factors in addition to the team performance statistics (recent as well as historical)

- **Match characteristics** The sequence and position of the match in the tournament series can also be a factor which captures the level of effort (and consequently, the runs scored) by a team. For example, it is expected that the level of efforts put by a team in a Tournament final match is considerably higher than during the initial league matches. We also hypothesis that the time of the match (day/night) can be a contributor to the outcome as it influences individual player performance.
- **Stadium & Weather** Last but not the least, the ambient factors such as the type of stadium (size, pitch characteristics), whether one of the teams is playing in their hometown (this is not guaranteed in IPL) and weather conditions are also expected to affect the runs scored. The amount of moisture also contributes to the speed and movement of the ball when released from the hands of the bowler.

4. Setting

Modern day cricket has three main variants – One day (where each team has 50 overs to play), Test (played for 5 days with alternate rounds of batting and bowling) and T20 (where each team has 20 overs to play).

Indian Premier League (IPL) is based on T20 matches where 8 teams (each team consisting of 7 Indian and 4 overseas players) are competing against each other every year during April/May. This format of the game is quite popular as the thrill and playing aggression is generally higher in shorter matches. The popularity is evident with more than tens of thousands of viewers flocking the stadiums for each game. For the application of our analysis, we choose the IPL data due to the following reasons:

- **Data Availability** One of the major reasons we choose to use IPL data is the availability of detailed information for all the years (since it started in 2008) from various open source parties. The website also has a lot of data on the player level and team level statistics for each year which we consider in our model to make it richer. Also, the matches are played only during the Summer season, hence the bias due to season affecting the outcome is also eliminated
- **Consistent Strategy** As this is a 20 overs match (per team) the full match generally lasts for 4-5 hours. With the limited overs, the obvious effort of each team is to play aggressive and score maximum runs possible within the small number of overs. In longer matches, there can be variations in the strategy and speed of scoring runs.

  The teams will also use their experienced bowlers more consistently than introducing newer bowlers because of the limited number of overs.

  The stamina and enthusiasm of the players can also be expected to remain intact in a short game as compared to a longer game.

  IPL teams are a mix of Indian and overseas players. Hence, the bias related to country rankings in cricket changing over time (which influences the match outcomes) is also eliminated.

  All in all, by sticking only to IPL tournaments we expect to reduce the strategy and stamina related noise from our predictive model.
- **Richness of Data** With the league matches being played in a Round-Robin format, there is a higher chance of having more data points for one combination of a batsman-bowler, which is one important combination in predicting the runs scored.

5. Business Use Case

The business use case chosen for our project is the area of advertisement effectiveness optimisation. It is standard for cricket matches, which are shown on TV, to display an advertisement after an event is happening within a match and the companies paying for these ads have an incentive to maximise the return on investment (ROI) for these advertisements. On average, ad slots in the Indian Premier League cost US$7.000 per second[Team, 2017] and the minimum ad length is roughly 10 seconds so for the course of this report we are assuming one ad slot to cost US$70.000. With that high of a cost per ad slot, advertisers are constantly seeking ways to optimise their customer engagement to make sure their ROI for these ad slots are at least slightly positive. Measuring the ROI for advertisement in general is extremely tough as one never knows what leads the customer to buy a certain brand or product and of course, the same applies for advertising during live sports. For our purpose, the assumption was made that live sports advertisement on average do not generate a positive ROI so they only generate a revenue exactly as high as the incurred cost.

This however is only the average effectiveness of advertisements. With a ROI of 0% no company would be interested in paying for expensive ad slots but the fact they are still buying these is based on some more recent studies. In 2017, a study by Yahoo concluded that customers' advertisement receptiveness increases by roughly 40% if the customer is feeling *upbeat*[Tan, 2017], meaning enthusiastic or happy. Advertising companies are therefore trying to catch the cus-

tomer when his or her receptiveness to the ad is the greatest and hence increase the ROI of the ad.

For placing ads during sports events, it can therefore be inferred that the most profitable ad slots are the ones immediately subsequent a major event such as a high run score in cricket. After a major event, spectators - either live inside the stadium or in front of the TV - usually feel upbeat, either because their team obtained a high score or simply because of the heated up atmosphere of the game.

With a score forecasting model, companies could therefore profit from this change in mood and the corresponding ad receptiveness by strategically selecting the most profitable ad slots before the game and increase their return on investment significantly. It is assumed that companies with a strategic important for the sport of cricket and with some influence on the broadcaster are able to pick the ad slots that they want. This is usually the case for major corporate partners of a particular league.

To achieve this outcome and maximise the revenue associated with an advertisement, the score prediction only needs to decide between a major and non-major event before the game starts and does not need to be concerned with slight variations within these two groups making the model more robust and more accurate.

6. Dataset

6.1. Overview

The necessary data needed to train the model consists of data covering match outcomes, player attributes and match attributes. The main data source is from Kaggle[1] which provides the outcome of each round in a cricket match and the attributes of those matches. The Kaggle dataset comes in 2 structured csv files which are:

Matches – Contains high level details of all matches between 2008 to 2019 such as team names, date, city played in, score details and best player. In total there are 756 records.

Deliveries – Contains detail of each throw in a match found in *Matches* (linked by id) such as batsman name, bowler name, various details of runs made and fielders involved.

From both data files above, initial exploratory data analysis shows that there are 15 teams, which played 756 matches in 41 venues from 2008 to 2019. In these matches, there are 179,078 deliveries, involving 516 unique batsmen and 405 unique bowlers.

At this point, the data is still lacking in terms of details about the players. In cricket, players have special attributes that makes them more effective in a match especially if their skill is able to counter their opponent's skill. For example, if a bowler is able to bowl with a spin using their right hand, they are more effective against a batsmen who hits with their left hand. Hence acquiring player attributes would be useful for training the model.

[1] https://www.kaggle.com/manasgarg/ipl

Table 1. Exploratory Data Analysis Overview

	Bowler	Batsman	Non-Striker	Team
Unique	405	516	511	15

To acquire details regarding the players, API extraction was done from *cricapi.com*. The API requires using a player ID (PID) to make the call. However the deliveries dataset does not have the PID as the PID is unique to cricapi.com. To get around this, the full list of PID was scraped together with the player names. This serves to match the players from deliveries to their corresponding PID using the player name as the key. With the list of PIDs, a loop was created to make http requests (using the request python library) with each PID and the response was stored into a dataframe. From the dataframe details such as batting style, bowling style and playing role was acquired for each player, if available, in the deliveries dataset.

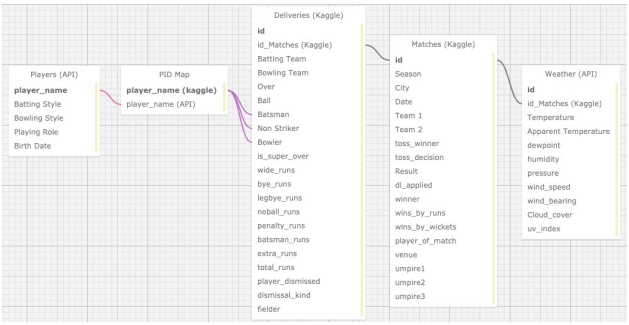
Weather conditions were also considered in the training of the model. Depending on the humidity and match conditions, it could favour the bowler or the batsmen. For example, a softer ground due to higher humidity dampens the toss of a bowler, slowing the speed of the bowl and making it easier for the batsmen to achieve a better outcome. To acquire weather conditions, the IPL website *iplt20.com* was scraped using UI Path for additional match attributes like time, date and location. UI Path is a Windows application that mimics mouse clicks and keyboard inputs on a computer based on a designed workflow. UI Path was used to access the website, navigate to certain pages and identify html elements which contained the time of a particular match that coincides with the matches dataset. Once identified the information is scraped and stored. To join the scraped data with the matches dataset, the data and teams were used as keys.

With the match location, date and time for each match available, API extraction from openweathermap.org was performed to acquire the weather conditions of each match. With the various data acquisition techniques performed, the complete dataset can be summarised as in figure 1.

7. Methodologies

The objective is to predict a significant event happening within a cricket match as accurately as possible to maximise revenues generated by advertisements. This is essentially a

Figure 1. Dataset Overview



binary classification problem. A non-major event (class 0) is defined as batsman run score equals to 0 or 1, whereas a major event (class 1) is a score above 1.

The whole process to tackle this problem includes: data pre-processing, feature engineering, standardisation, sampling as well as modelling. Below sections will discuss each step in more detail.

7.1. Data Pre-Processing

As described already, there are four main datasets for this project:

- high-level match dataset
- ball by ball deliveries data for IPL from 2008 to 2019
- players performance matrix acquired by API extraction
- weather indicators

The individual datasets were strategically joint by using several keys such as player names and IDs or match IDs. After merging, the master dataset is down to the granularity of each delivery (ball). For each row of the master dataset, it contains details for four perspectives: match (match id, teams names, venue), deliveries (inning, over, ball number), players performance (bowler, batsman and non-striker style and statistics), and weather (humidity, temperature, cloud cover).

One thing to note here is that the players performance were given per player per year. So one player could correspond to zero, one or multiple sets of statistics from different years. In order to join this with the deliveries data and avoid leaking performance data from the observed period, the previous year's player statistic with respect to each match was used. If the player did not have any statistic for the year prior (i.e. he did not participate in the IPL that year), this would result in a N/A entry. This is especially relevant for new player participating in the IPL for the first time since they won't have any historical statistics. For these players, only their main attributes and playing styles that do not change over the years were present.

For further pre-processing, several columns were in the form of strings so these either had to be converted to categorical variables or only the presence of certain keywords was converted into an indicator variable. For example, the original bowling style columns contained information such as *left-arm medium*, *right-arm fast* or *right-arm googly*, which is a combination of speed variations, and different spin skills. Thus, we extracted the keywords from combination and fit into several dummy columns including *googly*, *left-arm*, *right-arm*, *offbreak*, *legbreak*, *fast*, *medium*, *slow*.

With regards to data imputation, the main assumption is that players with no player statistics in the year prior to the respective match are new and/or inexperienced players, hence their respective statistics were filled with the 25th percentile making them not the worst players on the field but definitely below average. This data imputation was a necessary step since roughly 30% of the deliveries have no player statistics filled out, meaning 30% of deliveries were performed by players who did not participate in the IPL before. This is a significant portion of the dataset, hence it can not be simply removed.

The remaining rows with null values present were omitted from the dataset since their influence can be neglected.

7.2. Feature Engineering

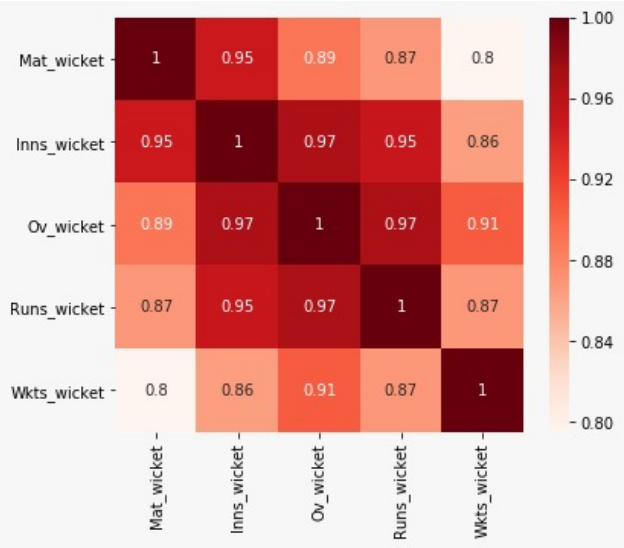
During feature engineering, using correlation plots, several high correlations between the chosen x variables were identified. We want to avoid having multivariate correlation mainly for two reasons.

- Collinear features in the dataset unnecessarily increasing the dimensionality and slows down the calculation speed
- It may dilute the feature importance, making collinear features less informative, thus lead to wrong or biased model interpretation

Below in figure 2 is an example for one of the highly corrected variable groups. The correlations among the fives variables are all above 0.8. Instead of making the decision about which variables to delete, PCA was applied. In our case, we set the percentage of variance we would like to preserve to 90%, which lead to one component being constructed by the PCA algorithm.

Furthermore, instead of converting the team name directly into dummy variables, the teams are binned into 3 distinct tiers. The evaluation is based on domain expert judgement

Figure 2. Correlated Player Statistics



and is mostly related to the public perception of each team[2]. This binning helps reducing the number of variables since not every team needs to be OneHotEncoded during the modelling process and we end up with only 3 variables

An additional new dummy variable is created to represent if one of the competing teams is playing in their home stadium. Due to the nature of the IPL, this is not guaranteed and only happens *by accident*. The assumption is however, that teams playing at home might have an advantage due to better motivation and crowd support.

In order to bring together all features and transform every column with the respective transformer needed, the DataFrameMapper was chosen, which works in a similar but more streamlined manner as the ColumnTransformer. The DataFrameMapper accepts a pandas dataframe as input and is able to apply different transformers to each feature as specified. It can then be combined with other FeatureTransformers together in a FeatureUnion.

By leveraging the FeatureTransformer, five new features are created based on manipulation of existing features concerning the player statistics. The logic behind these features is to emphasise the relative strength of two players combined. To achieve this, the ratio of the same metric for bowler as well as batsman was used to express the relative strength of players for a particular ball. A very high or very low number would indicate a high imbalance in strength and hence more likely results in very high or low results in terms of run score. One of these relative strength features and its calculation is presented in (1)

$$\frac{Runs_{run}}{Runs_{wicket}} = rel\_runs \tag{1}$$

where:

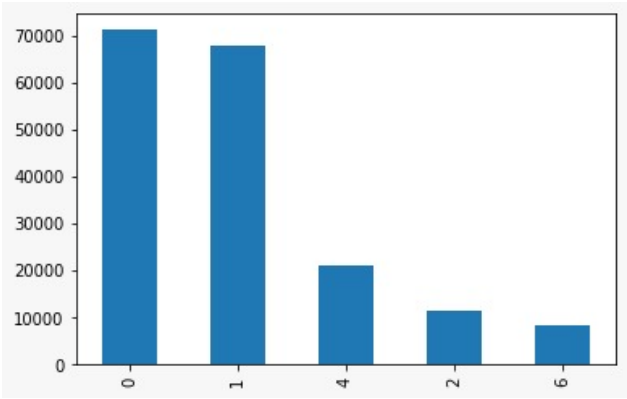
$Runs_{run}$ = Number of runs scored by the batsman so far

$Runs_{wicket}$ = Number of runs conceded by the bowler

7.3. Sampling Techniques

The problem is a binary classification since we want to detect major events happening within a cricket match. As stated before, the y variable, which is the batsman run score, is classified into two categories. Class 0 is defined as batsman run score equals to 0 or 1, whereas class 1 corresponds to a score above 1. Figure 3 below shows the score distribution before transformation where a significant imbalance can be observed. After binning, the dataset still features an extremely high unbalanced class profile, with 77% of observations in class 0 and only 23% in class 1

Figure 3. Distribution of Batsman Score



Based on our business objective, we are more concerned with the minority class (being the major events). Without sampling, the classifiers would not be incentivised to learn anything as the classifier would be able to achieve an accuracy of 77% by simply choosing class 0 for all observations. Therefore, sampling techniques need to be explored.

Several techniques (SMOTE, RandomUnderSampler, RandomOverSampler) and combinations of techniques were explored using a logistic regression as baseline classifier. The most promising sampling happens with a combination of SMOTE and RandomUnderSampler[3]. The corresponding sampling strategy will also be tuned in the following advanced classifiers. Every sampler takes an additional float metric as an input which is the sampling strategy. It is telling

[2]Note that this is not a result leakage from past matches since these tiers are historically established tiers that have not changed during the observation period used for this dataset.

[3]The evaluation metric for this choice will be further explained in the next chapter.

the sampler to what extent to undersample or oversample the minority group. The given float corresponds to the desired ratio of minority class samples over majority class samples after the sampler has been applied. This can be expressed as a formula in (2).

$$sampling\_strategy = \frac{N_{rm}}{N_{rM}} \tag{2}$$

where:

N_{rm} = Number of samples in the minority class after resampling

N_{rM} = Number of samples in the majority class after resampling

The optimal sampling strategy was seen to be very different between different classifiers during the tuning process. In general, classifiers either preferred a combination of upsampling with a ratio of 0.6 and a successive under-sampling with a ratio of 0.9 or having both sampling ratios at 0.9.

7.4. Evaluation Metric

Before jumping to the different classifiers, this section is dedicated how to choose model evaluation metrics based on our business statement, which is advertisement effectiveness optimisation.

People are generally feeling upbeat when there is a significant event, in this case a high score runs; their receptiveness for advertisement is increased by 40%. Thus, to optimise the effectiveness of advisement, false positives for class 1 need to be minimised and true positives maximised as much as possible.

Since the class imbalance between major and non-major events is extremely high, one cannot simply maximise classification accuracy since this would lead to the classifier only being able to detect the majority class. Therefore, as an objective for the GridSearch, the F1 score of the minority class was taken. Maximising precision would lead to undesired outcomes since the model would also try to minimise classifications for the minority group and hence bring down false positives by increasing false negatives. F1 seems to be a good middle-ground here and has proven to work well for the models chosen. The modified evaluation metric can be passed to the GridSearch by using the make_scorer() function.

7.5. Pipeline Setup and Modelling

In total, the dataset contains 143,278 observations and each observation has 55 features. We split the total dataset into training and testing dataset utilising a 4:1 ratio. For modelling, a pipeline with 4 steps has been created: feature transformation, feature standardisation, combination of different sampling techniques, and finally, fitting the prediction model itself.

Table 4 shows the different functions applied for each pipeline step.

Figure 4. Pipeline Steps

INDEX	STEP	FUNCTION
1	FEATURE TRANSFORMATION	FeatureUnion (DataFrameMapper + Self-Defined Feature Construction Functions)
2	NORMALIZATION	StandardScaler()
3	SAMPLING	SMOTE() + RandomUnderSampler()
4	MODELLING	KNeighborsClassifier() LogisticRegression() DecisionTreeClassifier() LGBMClassifier() RandomForestClassifier() KerasClassifier()

For step 3 and 4, since there are various hyperparameters for exploration, we chose the final combinations for the best models using GridSearch Cross-Validation. The steps were chained using imblearn"s "Pipeline" method. Random seeds were set for non-deterministic operations for reproducibility of results.

7.6. Classifier Choice

As a first process, basic classification techniques such as *K Nearest Neighbours*, *Logistic Regression* and *Decision Tree* classifiers were applied in order to establish a baseline model and test the actual feasibility of the project. These basic algorithms are easy to apply and provide a good model interpretability, from which insights could be derived in order to tune more advanced models or help with feature extraction. As a next step, more advanced classifiers were introduced such as Random Forest, LightGBM and Neural Networks (Multi-Layer Perceptron). This advanced classifiers are supposed to lean features that were not detected by the baseline classifiers and hence bring us closer to the model objective previously specified. Besides the explicitly stated hyperparameters for each of the below models, the best sampling strategy was also searched for as the optimal value here can also differ between different models.

KNN is suitable when the relationship among features are numerous and complicated, yet the items of similar class types tend to be fairly homogenous. We tried out different values for number of nearest neighbours. Comparing to decision tree and logistic regression, much longer time is required for running KNN.

Logistic regression is one of the most commonly used machine learning algorithms for binary classification. It makes ample use of the logistic function; it is efficient and does not require too many computational resources. Furthermore, for the feature importance, it can differentiate between positive influence and negative influence, which makes the model highly interpretable. For hyper parameter tuning, we tried different values for C, which represents the inverse of regularisation strength and various solvers, including *newton-cg*, *lbfgs*, *liblinear*, *sag* and *saga*.

Decision Tree is selected as it can give intuitive and easy explanation based on tree representation. Here, we only tuned three hyperparameters, which are criterion, max_depth and min_samples_leaf, to increase performance while avoid overfitting at the same time. Export_graphviz and pydotplus are used to visualise the tree model.

Random Forest is a representative for bagging methods. It is a mixture of multiple, relatively independent Decision Trees that combine together to enhance model performance and give better predictions. Compared to Decision Tree Classifier, Random Forest Classifier is less likely to overfit and more robust. Several important parameters, such as max_depth, min_samples_split and n_estimators are searched.

LightGBM is one of the most commonly used gradient boosting algorithms. General idea for gradient boosting is to update model based on gradient descent, which can be proven equals to residual error. Depending on the different mechanisms to determine the best tree split, there are two common types, which are LightGBM and XGBoost. LightGBM uses a novel technique of Gradient-based One-Side Sampling (GOSS) to filter out the data instances for finding a split value. XGBoost uses pre-sorted algorithm & Histogram-based algorithm for computing the best split. LightGBM is chosen here as it can achieve similar (if not higher) accuracy with much faster speed compared to XGBoost. Among the many hyperparameters for lightGBM, n_estimators, learning rate, num_leaves and max-depth are probably the most important ones for our model. Learning rate can extensively affect the accuracy, and number of estimators, leaves and max depth affects the training speed and model overfitting.

Neural Networks was assumed to be the most promising machine learning tool in this case before starting the project. Deep Learning is especially relevant when there is a high-dimensional input, the need for non-linear feature transformations as well as a discrete output which can be framed as a classification problem. All this is given in the project objective and dataset described above. For Neural Networks, there are certain hyperparameters that will need to be set and tuned such as the learning rate, activation function, variation of gradient descent (e.g. standard, stochastic, mini-batch), number of hidden layers, number of neurons in each layer and lastly the loss function. The specific application of Neural Networks in our case is a binary classification. For this reason, the last activation function will be represented by a sigmoid function as it can predict probabilities between 0 and 1. The appropriate loss function in this case would be a binary_crossentropy loss.

7.7. Model Results and Analysis

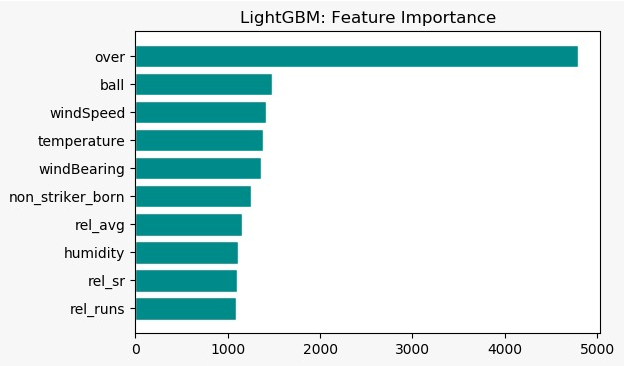
Table 5 below shows a summary of our results and model comparison. For F1 score, precision and recall, those are indicators only for the minority class (major event happening). LightGBM, Logistic Regression and Random Forest achieves the highest accuracy. KNN actually gives the best F1 score, while Neural Network shows highest precision.

Figure 5. Model Evaluation (metrics for class 1)

Model	Accuracy	F1 Score	Precision	Recall
KNN	0.54	0.34	0.25	0.51
Logistic Regression	0.63	0.32	0.28	0.38
Decision Tree	0.49	0.32	0.23	0.52
LightGBM	0.74	0.18	0.32	0.13
Random Forest	0.66	0.26	0.26	0.27
Stacked Neural Network	0.57	0.25	0.45	0.33

For Feature importance, the top 10 features from Logistic Regression, Random Forest and LightGBM are quite consistent and the majority is overlapping. Thus, only important features for LightGBM are shown below.

Figure 6. LightGBM Feature Importance



The most important feature is the index of over and ball. The batting team aims to score maximum runs which is possible when they have enough wickets to face all the 20 overs. It is seen that the propensity to lose wickets increases with more aggressive shots. The strategy of a team on when to play aggressive within the twenty overs (120 balls) heavily influences the score of the balls in that order. For example, one team might start with a defensive approach and play smaller shots in the first 10 overs to not lose out on wickets followed by bigger shots in the second 10 overs.

Based on the feature importance chart, we can see that weather indicators such as windspeed, wind bearing, temperature and humidity also have a high impact on the final classification. This is in line with our previously stated assumptions about external conditions influencing the players and the possibility that some players and their respective playing style is able to cope better with unusual conditions such as high temperature, humidity or a wet pitch.

Last but not least, the features created by ourselves about the player stats, such as the ratio of average_run and runs_run between batsman and bowlers, form another group of important features displaying that the ball outcome is indeed very dependent on the relative strength of the two players competing on the field.

In general, this feature importance chart gives us very reasonable and sensible result, which corresponds to our expectations and domain knowledge.

8. Model Selection and Conclusion

The final model selection was not based on any model result such as accuracy, f1-score or precision but rather the direct application on the business use case.

The business use case is to maximise ad returns as much as possible which means, false positives for class 1 need to be minimised and true positives maximised as much as possible. The model that is able to do so, will generate the highest profit in excess of a random guess model.

The random guess model is a random binomial distribution with the probability distribution obtained from the training data. Roughly 23% of the training data outcomes is a 1, hence this is the probability used for the random classifier as well.

From the results of a self-defined cost function (cost_function()) assigning a cost and return to false positives and true positives, it can be seen that the second version of the LGBMClassifier was able to generate the highest excess profit over a random guess using a classification threshold of 0.45.

The **excess profit on the training dataset is 5.82% per ad slot** while the **excess profit accounts to 2.88% on the test set** previously created.

With an average cost of one ad slot of US$70.000, this accounts for roughly US$2.100 per ad slot that the company could be generating in additional profit and therefore increase the advertisement ROI above 0%. In the highly competitive space of live sports advertisement this is surely a big win that companies could generate by applying the proposed machine learning algorithm and choosing their ad slot strategically before the event is starting.

9. Limitations and Further Opportunities

The models and their application described in the previous chapters rely on certain distinct assumptions and rough calculations. Before productionisation of the proposed machine learning algorithm, these might have to be further validated and adjusted based on the individual assumptions of the company employing the algorithm.

One of these critical assumptions is the increase in ad receptiveness during an upbeat mood. Current studies show a 40% increase in ad receptiveness but this might differ from one industry to another since some products are more targeted at emotional buyers than others. The greater ad receptiveness and higher willingness to consume might hold true for lower-value consumer goods but not necessarily for higher-value lifetime purchases such as homes, cars or the likes. Customers tend to be more rational for these purchases and might not be influenceable that easily by a high score in their favorite sport. It should therefore be treated with caution before simply employing this algorithm cross-industry.

Another assumption was that on average the ROI for live sports advertisements is around zero, currently only recovering the cost. This again might not hold true in some occasions and companies might be able to still generate a positive ROI without the model, increasing the value of the model even further since they wouldn't have anything to

lose anymore once a false positive occurs. At the moment, the model imposes a financial penalty on false positives since false positives (advertising after a non-major event) create a loss for the advertiser.

Something else that was assumed in the analysis is that advertisers are able to select their ad slots as per their convenience and they are also able to do so on relatively short notice before the game once the players and their order is announced. This right might only be available for very long-term and high profile sponsors of the IPL or of a prominent team and would have to be further checked and validated.

What the model is currently not taking into account is the overall budget constraint of advertisers. Further work could be done here to combine the obtained probability results with a linear programming model restricting the number of ad slots bought during a single game but also constraining the overall advertising budget per IPL season.

With respect to the actual modeling process there are also some things that can be looked at during future work. One example is the utilisation of player statistics. At the moment, the player statistics are joined with the deliveries by taking the year prior's statistic per player. Further work could be done here to take an average of the cumulative sum of prior performance or maybe even a weighted average, weighing higher the more recent statistics.

References

Tan, E. (2017). Reaching consumers in the right mood could make digital ads 40% more effective. https://www.campaignlive.co.uk/article/reaching-consumers-right-mood-digital-ads-40-effective/1427225 [Accessed: 23/04/2020].

Team, O. (2017). Here's how much it costs to advertise on tv during ipl games. https://officechai.com/stories/cost-of-ads-during-ipl/ [Accessed: 23/04/2020].