Stump the Data - DATATHON

# Project Report

# on

# Cricket Match Prediction using

# Machine Learning

# Submitted by

# Team Name : Akatsuki

# Name of participants :

# 1. Harsh Bhogal

# 2. Amit Singh

# 

1. **Introduction**

Cricket is a game of millions of factors and derivatives of these factors, and general ability and performance of both the individual players and the entire team, and field conditions, and weather conditions play a very major and crucial role in changing the result of a game. As the process of data gathering has grow fast in sports, incorporating ML into cricket has been a valuable way to make accurate predictions, for example performance of the players, the result of a match or even the so called expected run rate. That is why the assessment of these factors is quite likely to support teams, coaches, and analysts in decision-making that impacts team results positively. This project aims to develop an ML model that uses historical cricket data to predict [specific prediction task: The parameters in question are usually aspects like player performance, results of the match or runs rate. Some of the features on the dataset include: Players – runs scored, wickets, batting average, Matches – ground, tossing side, teams, External conditions – weather, pitch type. We also plan to extract patterns and trends out from this data by preprocessing this data and implementing sophisticated machine learning algorithms. By predicting the match parameters and clusters, the model can help coaches understand which factors have a pervasive impact on the result. Besides, the predictive factors indicating the outcome providing the highest coefficient, can become a valuable tool for defining strengths and weaknesses of the team strategy, increasing preparation of the players, as well as improving decision-making during the actual game. Due to the increased trend towards data analysis, this project captures the possibility of applying ML to cricket both in analytical and playing aspects.

**Keypoints:**

* Data Preprocessing
* Feature Selection
* Model Architecture and Techniques
* Hyperparameter Tuning
* Results and Insights from Model Performance
* Conclusion

1. **Data Preprocessing Steps**
   1. **Overview of the Dataset**

The dataset has details of cricket matches played in the past with details on player statistics, match statistics and results. They are: player statistics such as runs or wickets or strike rate, match information including the venue of the match as well as the toss and match results such as wins or losses, players scores.

* 1. **Handling Missing Data**

Some columns lacked certain values, especially where maybe information about player injury or some conditions about the match. We handled this by: Estimating the values in the gaps by using the average or, preferably, the median. Replacing missing-valued features with K-Nearest Neighbors Algorithm, the mode value of the referencing feature i.e., venue.

* 1. **Encoding Categorical Data**

Qualitative variables like “team”, “venue”, “toss decision”, and ‘opponent” had to be transformed using on One-Hot Encoding. This enabled us to transform non-numeric fields into a state that would be easily process by machine learning algorithms.

* 1. **Scaling and Normalization**

For numeric features, player strike rate, run rate, and wickets we applied StandardScaler to the data. It helps enhance the accuracy of results because features with different scales do not influence the model.

* 1. **Train-Test Split**

For the analysis we split the data in to 80% of training data and 20% of test data. It enabled us to determine the efficiency of the model based on data it has not come across.

1. **Feature Selection**
   1. **Features Considered**

Initially, the dataset had features such as: Score cards that may show how many runs in how many balls/movies the player scored, how many wickets the player took, the player’s batting average. Additional match information, such as the ground where the match is to be played, which team won the toss, home team, away teameters. External factors (for instance climate, type of pitch) .

* 1. **Feature Engineering**

Additional features were created, such as: Match sharpness (determined as the average score derived from the player’s performance of the last five games). Home ground (dummy variable used to test if a team is playing at home venue) Average of runs scored per match (from runs and wickets taken)

* 1. **Feature Selection Method**

Feature Selection Method Using Recursive Feature Elimination (RFE), we established the features that have greater significance to model complications. As stated, it progressively eliminates unimportant attributes which direction us to the most significant ones. The top features selected were: Player form Venue Toss decision Team strength must be calculated from the player averages.

1. **Model Architecture and Techniques**
   1. **Chosen Model**

We experimented with several models, including:

* Random Forest
* XGBoost
* Number Representation – feature = logistic Regression (suitable for binary classification of match outcomes).
* Linear Regression (for inferring run rates)

After the performance comparison, we chose [ insert the best model, such as XGBoost] because this algorithm is stable when working with large numbers of features and has protection against overfitting.

**4.2 Model Details**

Random Forest: An ensemble technique which forms a number of decision trees during training and then presents the class most frequently output (classification) or mean of the predictions (regression).

XGBoost: A progression of gradient boosting especially used in structured data.

Logistic Regression: Most beneficial where the goal is the classification of two results such as win/loss or scores in a game.

Linear Regression: Used for making continuous outcome predictions for events such as run rates or even the performance of players.

**4.3 Libraries/Frameworks**

Titanic dataset with features and target value features engineering.

XGBoost that stands for eXtra Gradient Boosted decision trees.

For data manipulation, Pandas and NumPy application.

**5. Hyperparameter Tuning**

**5.1 Tuning Approach**

For the above selected models, we used GridSearchCV to determine the most suitable hyperparameters. For Random Forest and XGBoost, the key hyperparameters tuned were:

* Number of estimators (trees)
* Maximum depth of the trees
* Learning rate (for XGBoost)
* Minimum samples per split

**5.2 Final Hyperparameters**

After tuning, the final hyperparameters were:

* + - * **XGBoost**
* Number of estimators: 100
* Maximum depth: 6
* Learning rate: 0.1
* Random Forest:
* Number of estimators: 150
* Maximum depth: 8

1. **Results and Insights from Model Performance**

**6.1 Evaluation Metrics**

The following metrics were used to evaluate the model:

* + - * **Accuracy**: For match outcome, the model was [insert value] accurate in the prediction of the outcomes.
      * **Precision and Recall**: For binary classification such as win or lose, precision and recall where measured to be [insert values].
      * **RMSE**: For ‘continuous’ type predictions such as a player’s performance or run rate, the RMSE was [insert value].
      * **Confusion Matrix**: On the classification problems, the confusion matrix provided information on how well the model was performin on the events of wins and losses.

**6.2 Insights and Observations**

* + - * **Player form** was most significant in determining predictions on games followed by the venue factor since recent performances define results.
      * **Toss Decision** was moderate as teams won the toss had a slight advantage.
      * The **Venue** feature concluded that factor that Counts In was important especially in the home team ;- the home teams won more matches.
      * Linear model was well suited to predict **Run Rates** as high correlation was observed between actual and predicted Run Rate, having correlation co efficient [insert value].
  1. **Touchpoints and Turning Points**

While the model performed well, there are areas for improvement:

* + - * **Additional Data**: Adding in weather data or even more exact pitch conditions might help with that.
      * **Feature Engineering**: Other more sophisticated approaches, such as sentiment analysis of the player or team’s morale could be introduced.
      * **Model Complexity:** Other more complex models such as deep learning architectures could be looked at for example in regards to predicting a player performance.

1. **Conclusion**

Altogether, we discovered an ML model employing historical cricket data to assess the following: match outcomes or player performance. In reality, the model was a very basic one with a lot of scopes for further improvements when refined with more data and more complex feature engineering.

**GitHub Repository Link:** [**git@github.com:Harsh456B/Cricket-Prediction.git**](file:///C:\Users\VICTUS\Downloads\git@github.com:Harsh456B\Cricket-Prediction.git)

**where I have to place my GitHub account: Harsh456B**

**my profile link:** [**https://github.com/Harsh456B**](https://github.com/Harsh456B)