**IPL Score Prediction**

**Introduction :**

The Indian Premier League (IPL) is a fast-paced cricket tournament brimming with excitement and strategic complexities. Predicting its outcomes presents a captivating challenge due to the dynamic nature of the game and the influence of various factors. This project delves into the world of IPL score prediction, leveraging data analysis and machine learning techniques to unveil the underlying patterns hidden within player performance, team compositions, and historical trends. By harnessing the power of data, we aim to develop a model that can forecast match scores with greater accuracy, providing valuable insights for cricket enthusiasts and potentially even aiding strategic decision-making within the IPL ecosystem.

Since the dawn of the IPL in 2008, it has attracted viewers all around the globe. A high level of uncertainty and last-minute nail-biters have urged fans to watch the matches. Within a short period, the IPL has become the highest revenue-generating league in cricket. In a cricket match, we often see the scoreline showing the probability of the team winning based on the current match situation. This prediction is usually done with the help of data analytics. Before, when there were no advancements in machine learning, predictions were usually based on intuition or some basic algorithms.

Here I have developed an IPL score predictor using the existing data. The prediction model is going to predict if A team is batting against B team and if the X runs has been taken in 6 overs and Y wickets have fallen then what will be the predicted score at the end of 20 overs. That is the goal of the project.

**Tools and Technologies:**

* Python for data analysis and model building
* Streamlit web app enables users to predict total runs between teams using current runs and wickets.

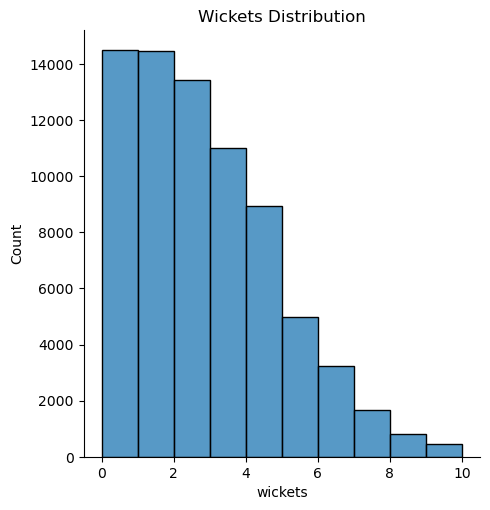
### **About the Dataset:**

The data for the use case is taken from an online repository and the data is from 2008 to 2017. This project is to understand the power of sports analytics and how machine learning can be used in the sports domain for predictions. This data consists of 73507 rows and 15 columns.

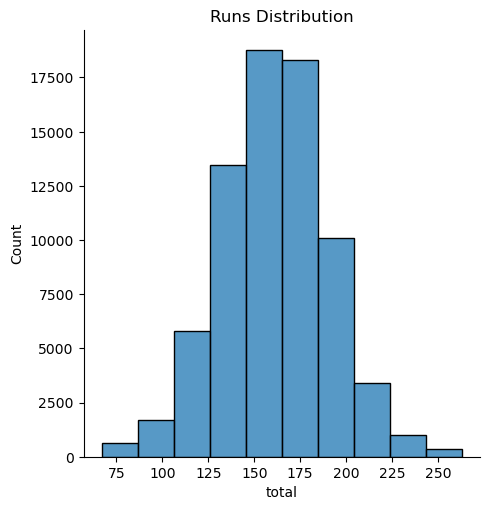
### **Data Structure:**

This CSV has 73507 rows and 15 columns.

* mid - match id
* date - when matches are played
* venue - place where matches are played
* bat\_team - batting team
* bowl\_team - bowling team
* batsman - batsman
* bowler - bowler
* runs - runs scored
* wickets - wickets
* overs - next 3 are based on this
* run\_last\_5 - runs scored in last 5 overs
* wicket\_last\_5 - wickets in last 5 overs
* striker - batsman playing as main 1
* non-striker - batsman playing as runner up - not main 0
* total - total score (target variable)

Initially I have done some EDA to understand the dataset. And made some distribution plots like Wicket distribution and Runs distribution.  


From the above observation we can observe the count of runs per wicket. Maximum of the wickets were fallen in the initial stages of the match. The highest bar appears to be at x=2, indicating that this wicket count might be the most frequent. Overall, this graph provides a visual representation of how many matches resulted in a specific number of wickets falling.



From the above graph we can observe that the graph is similar to normal distribution. Maximum teams scored 140 to 180 runs.

Here we can see that the columns mid, date, venue, batsman, bowler, stricker, non striker won't provide any relevant information for our model to train. After removing irrelevant columns because here we are predicting the score for 20 overs but not for the batsman, bowler or stricker.

Here I have only performed building models with consistent teams as we know these data is till 2017. So there are new teams that have popped up like gujarat lions and LSG (lucknow supergiants) etc. After removing all the data other than consistent teams we have 52566 rows and 8 columns. Here I have removed all the first 5 overs of every match and the data after removing data is 39177 rows and 8 columns.

And performed one hot encoding and column transformation ( It is used to transform the data when we have mixed data types like numerical and categorical data). Here we are using a transformer one hot encoding for the first two columns of the data and leaving the remaining columns as the same ( assuming [o,1] as index of column). Then converted the transformed data into a Numpy array in a new dataframe with transformed columns and changed the column names of the transformed columns.

**Algorithms used :**

* Linear Regression
* K-Nearest Neighbor Regression
* XGBoost Regressor
* RandomForest Regression
* Decision Tree Regressor
* Support Vector Machine

| S.No | Algorithm Name | Training set accuracy | Testing set accuracy |
| --- | --- | --- | --- |
| 1 | Decision Tree Regressor | 99.98% | 86.62% |
| 2 | Linear Regression | 65.25% | 65.19% |
| 3 | Random Forest Regression | 99.08% | 93.46% |
| 4 | Support Vector Machine | 56.70% | 56.24% |
| 5 | XGBoost Regressor | 88.66% | 85.04% |
| 6 | K Nearest Regressor | 86.60% | 77.13% |

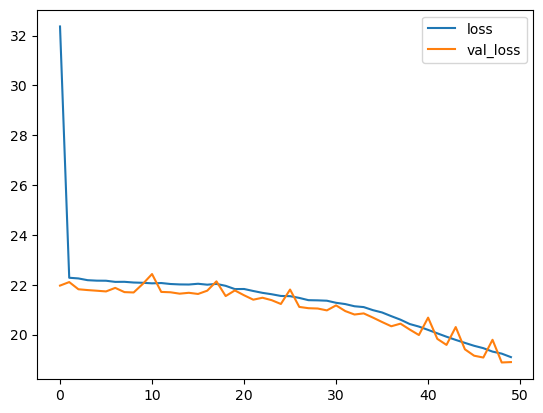
**Random Forest** performed the best, closely followed by **Decision Tree**. So we will be choosing Random Forest for the final model.

In the End i have imported a machine learning model. And Streamlit for making predictions based on the previous data.

**Deep Learning Algorithm :**

Here I have dropped some features which do not give more information for our model building. Using label encoding to our categorical features and encoded their values. Performed min-max scaling on our input features to ensure all the features are on the same scale. This scaling is performed to ensure consistent scale to improve model performance.

Here I have defined a neural network using tensor flow and keras for regression. Here we have used hidden layers with 512, 216 units with ReLu activation and the output layer with a single neuron using linear activation. The model is compiled for regression using Huber loss function and Adam optimizer.



After the training we have stored the training and validation loss values to our neuron network during the training process and plotted a graph using the stored values. From the graph we can observe x-axis represents no of epochs, y-axis represents the loss value. Here the loss gets decreasing in both training and validation loss which means the model is learning from the training data and generalizing well to unseen data in validation set predicted using the trained neural network on the testing data.

Here I have created an interactive widget using an ipy widget to predict the score based on user input for venue, batting, bowler, striker and non striker. I have created a dropdown widget to select the values of the above mentioned feature. Then I have added a predicted score button widget. Whenever the button will be clicked, the predicted score function will be called and it decodes the user selected values to their original categorical values. Encodes and scales these values to match the format used in model training and uses the trained model to make a prediction based on the user’s input and displays the predicted score.