IPL Score Predictor

Team: Sai Meghana Kolla, Sai Santan Bharadwaj Mudumba

COMP 520 – Artificial Intelligence

Abstract

The cricket score prediction models proposed in the recent past focused on few aspects either with the past statistics or with the live feed of the match. In this project, the IPL score predictor, We broadened the horizon by predicting the score of a match by covering the maximum aspects without over fitting the mode. In this project, we created our own dataset, trained the model, evaluated the model and presented the model with an User interface.

1. Introduction

Cricket is the second most popular sport played majorly in Subcontinent, England, Australia, South Africa, and West Indies. Cricket is mostly played in three formats viz, Test, ODI and T20. IPL is a franchise-based T20 tournament based in India in which 8 teams compete with each other in the round-robin stage followed by playoffs. We have prepared a model that predicts IPL match scores using machine learning accurately.

In the recent past, researchers created models to predict the cricket scores, out of which a popular method Duckworth-Lewis is formulated and adopted by ICC in the 1990's which is used to date. However, it is often criticized for the shorter formats like T20. Few other models are developed in recent years, each model has its limitations. One of such types predicts the score based on the recent form and the past encounters between the teams which shouldn't be the ideal model, as in cricket a team can win irrespective of previous records. Another model requires the game to be in progress and uses the current game scores to predict the score without the knowledge of the playing teams.

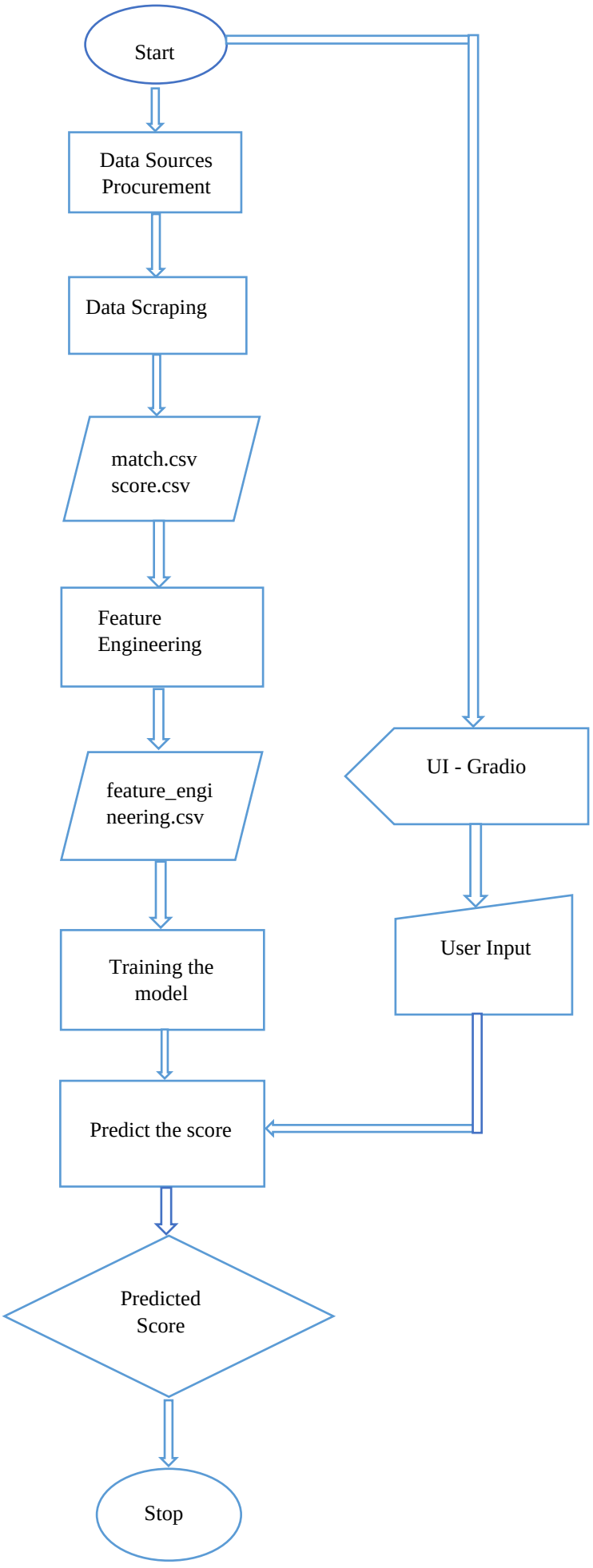
We created a model to address the above limitations by considering both current match statistics and the team data.

1.1 Our Contributions

- We identified the potential data sources
- Performed web scrapingto generate data sets.
- Feature engineered the dataset.
- Trained model and evaluated its accuracy.
- Provided a UI interface for the user can input the values and predict the scores.

2. Material and Methods

Block Diagram


Start
Data Sources Procurement
Data Scraping
match.csv score.csv
Feature Engineering
feature_engineering.csv
Training the model
Predict the score
Predicted Score
Stop
UI - Gradio
User Input

2.1 Dataset Description

We created two datasets, one for match results and one for ball by ball stats. Even though the existence of similar datasets in Kaggle, the reason for creation is that they contain a lot of noisy data with features that aren't necessary data for our model. We can remove the unnecessary columns from the data set by data cleaning; however, we want to extend this model to accommodate a few other models which we discussed in the conclusion part, so as we extend the model we need to accommodate new features into the dataset and the size of the dataset keeps increasing drastically.

The process of creating a dataset started by identifying the potential data sources. We came across various websites like cricbuzz, IPLT20.com, However, ESPNCricInfo has all the information we needed. So we decided to perform Web Scraping on ESPNCricinfo.

We have used scrapy a high-level framework for web crawling and scraping to acquire the matches.csv dataset. We have crawled the ESPN Cricinfo website to get the matches' information. We have acquired the data from the 2019 season which has 60 matches.

For the ball by ball data set we used urllib3 a user-friendly HTTP client for Python. We have acquired ball by ball data for all the 60 matches from an API of ESPN which has a commentary for all the matches. From the comments, we acquired the necessary data to build the scores.csv website. Then both datasets were merged to form one dataset feature_engineering.csv as part of data cleaning.

2.2 Methodology

Once the dataset is created, we have performed data cleaning. Even though we had an option to skip the data cleaning by creating datasets that are required for the model, but we have decided to extend the model by adding more features.

Dataset with numeric values can be directly used in training the model, but the system cannot process the strings, they have to be converted to numeric values. As our dataset contains the columns of string type we need to convert them into numeric values, We made use of label encoder "SKLearn.Preprocessing.LabelEncoder" for this purpose which assigns a unique numeric value for each string.

As the data contains values with a massive difference like the value of over will be between 0 to 20, while the value for total will be 140 approximately and there are thousands of records, training data with these irregular distributions may result in skew problem i.e, higher values knockdown (zeroed down) the smaller values. To handle this issue we used a scalar. There are several scalars available like MinMax scalar, We used Standard Scalar for this model which uses standard normal distribution.

Splitting of the dataset into training and testing sets plays a crucial role in the model. Biased information results in an overfitting problem. The split must be randomly done. We used “SKLearn.model_selection.train_test_split" with a training set of 67% and a test set of 33%.

Once the data is split, we used the training data to train the model using Random Forest Algorithm which is an ensembling technique that builds many (forest) decision trees. The other ways we dealt with is using linear regression and the Sequential model. However, due to the large dataset, we faced an overfitting issue with linear regression and sequential model we tried to handle it using Early stopping which degraded the performance of the model. So we used the random forest model as it checks the overfitting at each decision tree and regresses them to provide the final prediction.

2.3 User Interface

Gradio:

We are using gradio a python package for UI. It provides the programmer with a default interface in which the required inputs fields can be added. The inputs read from the UI will be sent to the function defined in the definition of the gradio interface which returns the predicted score to the interface. The interface will the launched when the submit method is clicked in the UI after inputting the required values.

Displaying the score after predicting the score with the given input values.

3. Performance Evaluation

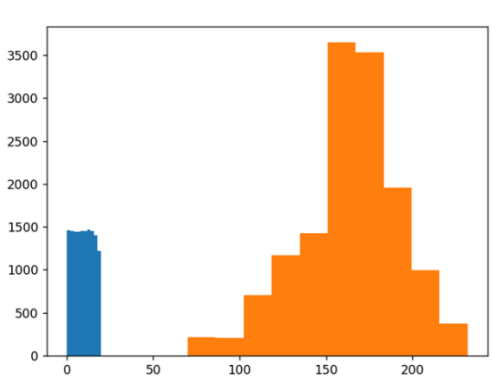
	Predict	Actual
0	150.860000	149
1	150.850000	151
2	176.800000	177
3	160.800000	161
4	108.000000	108
5	163.648333	133
6	96.400000	96
7	136.040000	155
8	172.880000	173
9	121.000000	121

10 Sample values Predicted using Random Forest Model.

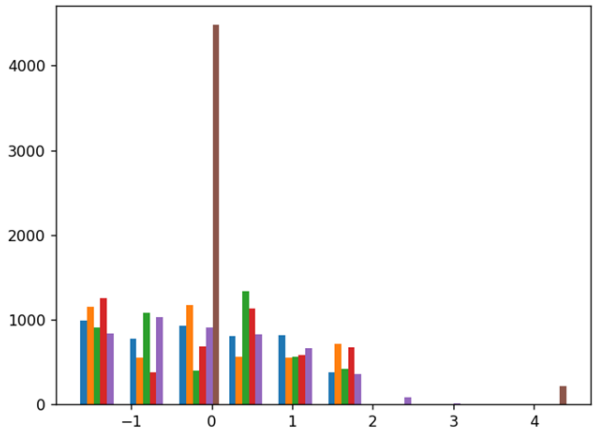
```
R square value: 93.80228055665738
```

4. Experimental Analysis

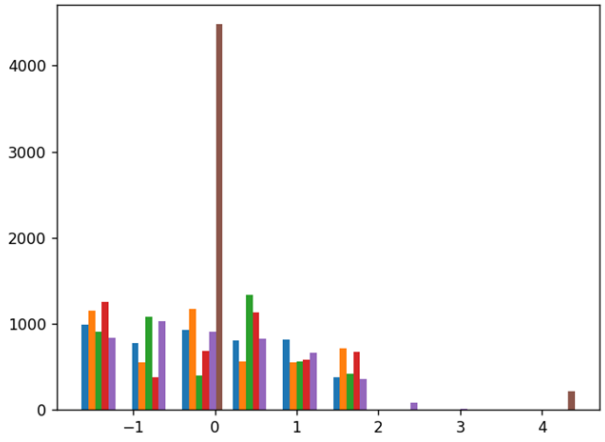
Below pic is a graph in which overs represented in blue and runs in orange which is to be scaled before using for training

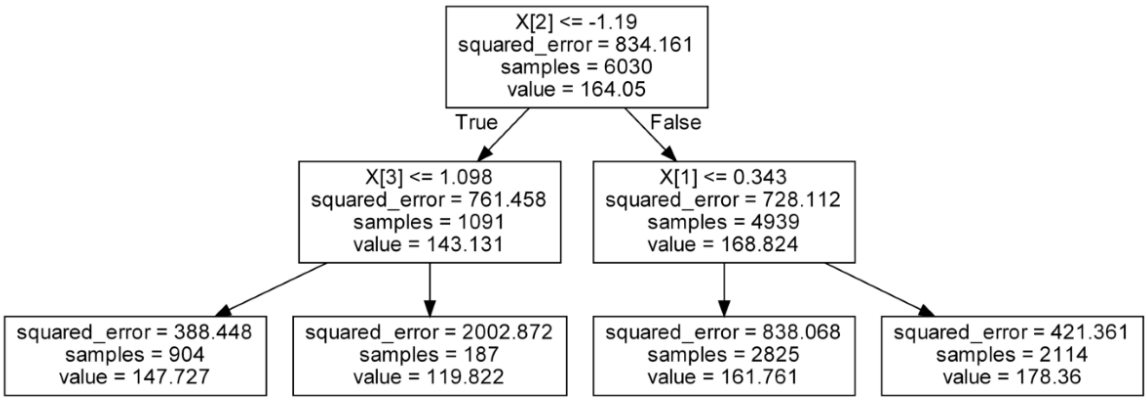


Total runs column before scaling



Total runs after scaling





The above image is a branch of a tree in a forest of Random Forest Regressor. For the purpose of visualization, we have considered the number of trees in the forest as 5, Maximum depth of the tree as 2

In general case, the forest can have 100's of tree and there won't be a limit on the depth of the tree, but it is hard to visualize.

5. Conclusion

To conclude, we trained the IPL prediction model with the dataset created and evaluated the model to ensure overfitting. The model hadn’t compromised the performance and can cope up for further extensions to predict the winning team, Power play score, Death over score by making code level changes in dataset set creation.

Furthermore, this model can be used as Cricket score predictor for T20I and ODI formats by providing the corresponding data source.