**ICC 2019 Cricket World Cup Prediction using Machine Learning**

The 2019 ICC Men’s Cricket World Cup is ready to begin on Thursday (30th May). This 12th edition of the Cricket World Cup will run for almost one and a half month in England and Wales. The tournament will be contested by 10 teams who will be playing in a single round-robin group, with the top four at the end of the group phase progressing to the semi-finals.

Predicting the future sounds like magic whether it be detecting in advance the intent of a potential customer to purchase your product or figuring out where the price of a stock is headed. If we can reliably predict the future of something, then we own a massive advantage. Machine learning has only served to amplify this magic and mystery.

**Applications**

The main objective of sports prediction is to improve team performance and enhance the chances of winning the game. The value of a win takes on different forms like trickles down to the fans filling the stadium seats, television contracts, fan store merchandise, parking, concessions, sponsorships, enrollment and retention.

**Data**

Real world data is dirty. We can’t expect a nicely formatted and clean data as provided by Kaggle. Therefore, data pre-processing is so crucial that I can’t stress enough how important it is. It is the most important stage as it could occupy 40%-70% of the whole workflow, just to clean the data to be fed to your models.

I scraped three scripts from Cricbuzz website comprising of rankings of teams as of May 2019, details of the fixture’s of 2019 world cup and details of each team’s history in previous world cups. I stored the above piece of data in three separate csv files. For the fourth file, I grabbed ODI dataset for matches played between 1975 and 2017 from Kaggle in another csv file. In this file, I removed all the data from 1975 to 2010. This was done as the results of the last few years should only matter for our predictions. Since I didn’t get the data for 2018 and 2019 so this model might not be that accurate but still I believe this gives a fairly good idea. Then I did manual cleaning of the data as per my needs to make a machine learning model out of it.

**Environment and tools**

1. Jupyter Notebook
2. Numpy
3. Pandas
4. Seaborn
5. Matplotlib
6. Scikit-learn

I followed the general machine learning workflow step-by-step:

1. Data cleaning and formatting.
2. Exploratory data analysis.
3. Feature engineering and selection.
4. Compare several machine learning models on a performance metric.
5. Perform hyper-parameter tuning on the best model.
6. Evaluate the best model on the testing set.
7. Interpret the model results.
8. Draw conclusions and document work.

Diagram

Description automatically generated

Machine learning workflow

**CODE:**

I started by importing all the libraries and dependencies.

Then I loaded the csv file containing the details of each team’s history in previous world cups. I also loaded the csv file containing the results of matches played between 2010 and 2017.

Table

Description automatically generated

1. **Data cleaning and formatting**

Next, let’s display the details of matches played by India.

Table

Description automatically generated

I continued by creating a column to display the details of matches played in 2010 and taking it as a reference for future work.

Table

Description automatically generated

**2. Exploratory data analysis**

After that, I merged the details of the teams participating this year with their past results.

Table

Description automatically generated

I deleted the columns like date of the match, margin of victory, and the ground on which the match was played. These features doesn’t look important for our prediction.

Table

Description automatically generated

**3. Feature engineering and selection**

This is probably the most important part in the machine learning workflow. Since the algorithm is totally dependent on how we feed data into it, feature engineering should be given topmost priority for every machine learning project.

**Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data.**

**Advantages of feature engineering**

· **Reduces Overfitting**: Less redundant data means less opportunity to make decisions based on noise.

· **Improves Accuracy**: Less misleading data means modeling accuracy improves.

· **Reduces Training Time**: fewer data points reduce algorithm complexity and algorithms train faster.

So continuing with the work, I created the model. If team-1 won the match, I assigned it label 1, else if team-2 won, I assigned it label 2.

Table

Description automatically generated

Then I converted team-1 and team-2 from categorical variables to continuous inputs using pandas function *pd.get\_dummies.*This variable has only two answer choices: team 1 and team 2. It creates a new dataframe which consists of zeros and ones. The dataframe will have a one depending on the team of a particular game in this case.

Also, I separated training and test sets with 70% and 30% in training and validation sets respectively.

A picture containing diagram

Description automatically generated

Train/Test Split

**4. Compare several machine learning models on a performance metric**

I used Logistic Regression, Support Vector Machines, Random Forests and K Nearest Neighbours for training the model.

Random Forest outperformed all other algorithms with 70% training accuracy and 67.5% test accuracy.

**The random forest combines hundreds or thousands of decision trees, trains each one on a slightly different set of the observations, splitting nodes in each tree considering a limited number of the features. The final predictions of the random forest are made by averaging the predictions of each individual tree.**

RFs train each tree independently, using a random sample of the data. This randomness helps to make the model more robust than a single decision tree, and less likely to overfit on the training data.

**5. Perform hyperparameter tuning on the best model**

Training set accuracy: 0.700  
Test set accuracy: 0.675

The popularity of the Random Forest model is explained by its various advantages:

* Accurate and efficient when running on large databases
* Multiple trees reduce the variance and bias of a smaller set or single tree
* Resistant to overfitting
* Can handle thousands of input variables without variable deletion
* Can estimate what variables are important in classification
* Provides effective methods for estimating missing data
* Maintains accuracy when a large proportion of the data is missing

**6. Evaluate the best model on the testing set**

Let’s continue. I added ICC rankings of teams giving priority to higher ranked team to win this year.

Next, I added new columns with ranking position for each team and slicing the dataset for first 45 games since there are 45 league stage games in total.

Table

Description automatically generated

Then I added teams to new prediction dataset based on ranking position of each team.

Table

Description automatically generated

After that, I added scripts for getting dummy variables and added missing columns compared to model training dataset.

Table

Description automatically generated

**7. Interpret the model results**

Finally, the below code is for getting the results for each and every league stage match. This is the final function to predict the winner of ICC Cricket World Cup 2019.

I ran the function for semi-finals prediction.

**New Zealand and India  
Winner: India**

**South Africa and England  
Winner: England**

Hence the two finalists are India and England which is quite evident as they are considered the favourites to win this year. Also, they are first and second ranked team in ICC rankings.

**8. Draw conclusions and document work**

Finally on running the main function.

**India and England**

**Winner: England**

**According to this model, England is likely to win this World Cup.**