CS 6375 Machine Learning - Final Project

Vedang Wartikar VPW210000
Shlok Urmil Pujara SXP220146

12 December 2022

Contents

1 Project Topic

Compare different Machine Learning models for the FIFA World Cup data set, find out the best one (try to implement it from scratch), and with help of that model predict schedule-wise matches of the 2022 World Cup. Also, find out which teams have a higher probability of qualifying for the next round and eventually winning the World Cup Title.

2 Motivation

Most of the time, Machine Learning involves building a model using the train data set and then finding out the model's accuracy using the test data set. With the FIFA 2022 World Cup currently happening at this very moment, we can compare our model's accuracy with the actual matches currently going on in Qatar. We intend to explore the application of ML in the sports domain and how it can benefit sports analysts to make decisions like strategic plays, team substitutions, changing the team's formations, etc.

3 A Quick Overview

After a few preprocessing steps, we compared different Machine Learning models (Logistic Regression, Naive Bayes, Decision Tree, KNN, SVM, and AdaBoost) across various metrics (Accuracies, Cross Fold Validation, ROC Curves, Confusion Matrices, Precision, Recall, and F1 score). The primary reason behind comparing these models beforehand is to find out the most optimal models so that we can create our own implementations for them. Furthermore, we have also used Logistic Regression as the base model to make the schedule-wise predictions for the 2022 World Cup and figured out with what probability which teams qualify for the next round.

4 About the Data Set(s)

4.a Description

We have used a total of 3 data sets. One for building our model, one for making predictions, and the other for sequentially predicting the stage-wise as per the Qatar World Cup schedule

4.a.1 Build the models

- **international_matches.csv**
  The original data set comprises of about 24,000 rows of data spanned over 32 columns. Each data point includes the details of each match. We have filtered out unwanted noisy data (handled missing values, removed matches from other tournaments, etc.). The preprocessed data set contains 5960 data points and uses 7 features for building the models. The features and classes are discussed in the following section.

4.a.2 Make the predictions

- **fifa_rankings_2022.csv**
  This CSV consists of the latest rankings of the countries allotted by the Fédération Internationale de Football Association (FIFA). We have used these rankings to make schedule-wise predictions for the matches.

4.a.3 Follow the official schedule

- **world_cup_teams.csv**
  It contains the teams as per the schedule and groups of matches and we will be building our way to the winner by making probabilistic predictions on the matches following this official schedule.

4.b Features

After applying some preprocessing techniques, we have used the following features for predicting the target class - the result of the match

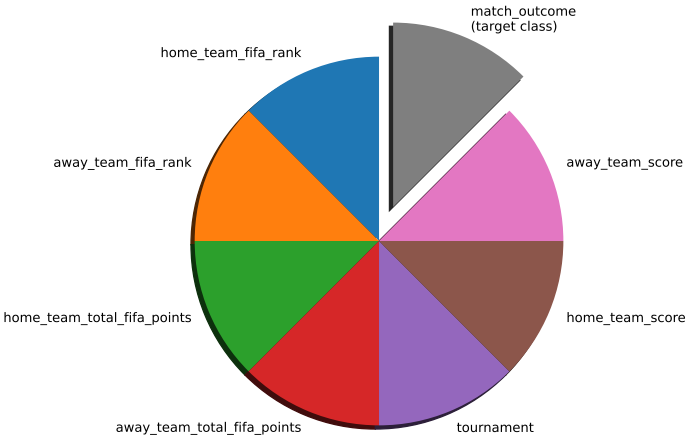


Figure 1: Features

We have combined the features related to ranks into average rank and rank difference, and the home/away team points into a combined net point feature. The match outcome (result) has been computed from the goal score difference of the 2 teams.

4.c Target Classes

We want to predict whether the team will qualify for the next round or not. In order for the team to qualify, it needs to score goals (usually with a higher score difference). We have processed the data such that if the score difference between team A and team B is greater than 0, team A wins and qualifies for the next round, else we consider it as a Loss. If both teams A and B score equal goals, we consider that a loss for team A (team A is the home team) needs to have a valid goal score difference to get points in the first stages to clear the initial rounds.

```
# function determines the match outcome for a given datapoint
        def transform_match_outcome(score_difference):
            if int(score_difference) > 0:
                return 1
            else:
                return 0
```

5 Comparison between Algorithms (scikit-learn)

5.a Algorithms used

- Logistic Regression
- Decision Tree
- Naive Bayes
- K Nearest Neighbor
- Support Vector Machines
- AdaBoost

5.b Train / Test Accuracies

Computing Train/Test accuracies based on the train and test errors for the above models, we can see the following graph,

Figure 2: Train / Test Accuracies compared across various models

From the above figure, it is evident that Logistic Regression, Naive Bayes, and SVM perform equally well on the train and test set. Decision Tree appears to have high accuracy on the train set (overfits) but fails to accurately capture the predictions on the test set. Logistic Regression and Naive Bayes have relatively good accuracies as expected because we are making predictions on target classes that are categorical (binary classification).

5.c 10-Fold Cross Validation

We are running k-fold cross validation (k=10) and checking the average performance of the models by dividing the data into 10 different blocks and iteratively running the models on 10 combinations of train/test (9:1) sets.

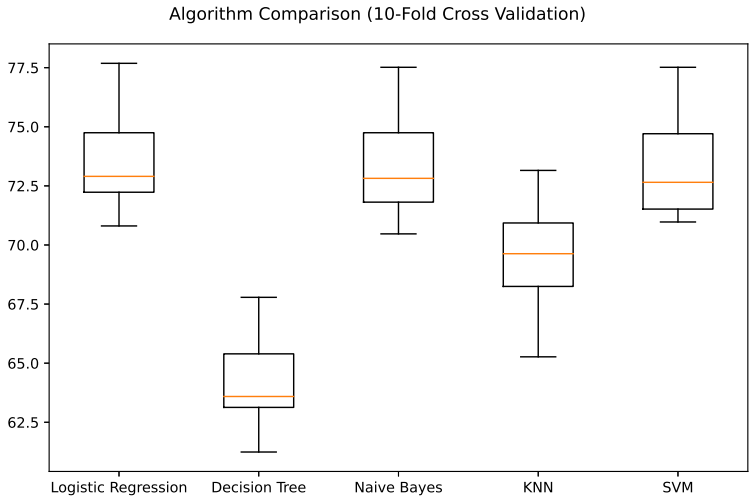


Figure 3: Algorithm comparison over 10-fold cross validation

We can see that Logistic Regression and Naive Bayes perform really well with cross validation with the median accuracy being roughly around 73%. Decision Tree and KNN produce relatively average results over cross validation.

5.d ROC/AUC Curves

Generating ROC curves for the following algorithms, we get the below plot

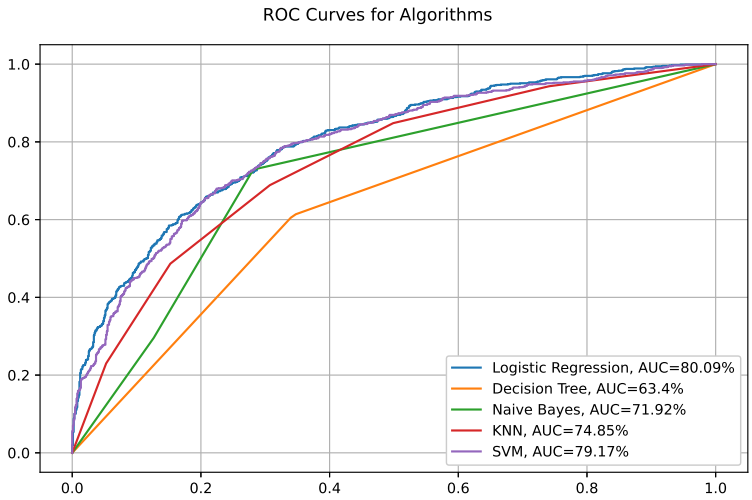


Figure 4: Comparison of ROC curves for different algorithms

We can see that Logistic Regression has the highest Area Under the Curve (AUC). SVM performs equally well. Naive Bayes has a relatively less area than the 2 models.

5.e Decision Trees with AdaBoost

We have also compared the results from the Decision Tree for depths [1, 20] with boosted Decision Trees. The following is the result,

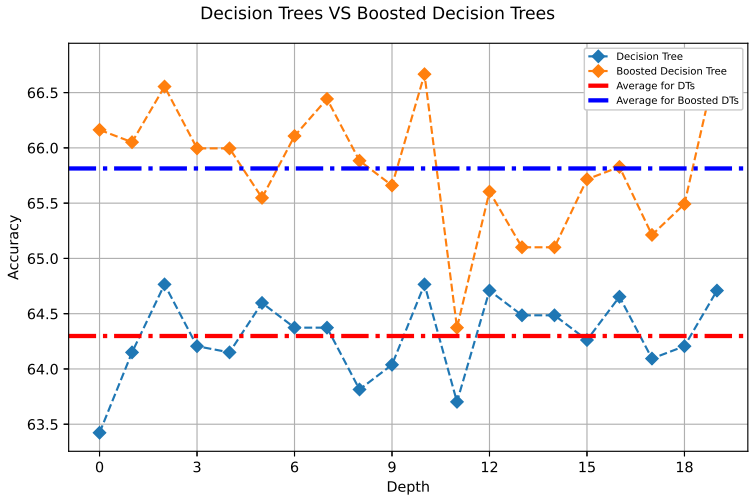


Figure 5: Effect of Boosting on Decision Trees

Looking at the plot, we can see that after boosting the Decision Tree Classifier using AdaBoost, the model performs relatively better. The average accuracy over the 20 depths is about 2% higher after applying AdaBoost. This accuracy is still less than our top 2 performing models - Logistic Regression and Naive Bayes.

5.f Confusion Matrix, F1, Precision and Recall

Logistic Regression:

	precision	recall	f1-score
Loss/Draw	0.72	0.73	0.72
Win	0.73	0.71	0.72
accuracy			0.72
macro avg	0.72	0.72	0.72
weighted avg	0.72	0.72	0.72

Confusion matrix:

	Loss/Draw	Win
Loss/Draw	654	238
Win	260	636

Decision Tree:

	precision	recall	f1-score
Loss/Draw	0.62	0.66	0.64
Win	0.64	0.61	0.62
accuracy			0.63
macro avg	0.63	0.63	0.63
weighted avg	0.63	0.63	0.63

Confusion matrix:

	Loss/Draw	Win
Loss/Draw	586	306
Win	352	544

Naive Bayes:

	precision	recall	f1-score
Loss/Draw	0.72	0.71	0.72
Win	0.72	0.73	0.72
accuracy			0.72
macro avg	0.72	0.72	0.72
weighted avg	0.72	0.72	0.72

Confusion matrix:

	Loss/Draw	Win
Loss/Draw	631	261
Win	241	655

KNN:

	precision	recall	f1-score
Loss/Draw	0.69	0.69	0.69
Win	0.69	0.69	0.69
accuracy			0.69
macro avg	0.69	0.69	0.69
weighted avg	0.69	0.69	0.69

Confusion matrix:

	Loss/Draw	Win
Loss/Draw	618	274
Win	279	617

SVM:

	precision	recall	f1-score
Loss/Draw	0.72	0.73	0.72
Win	0.73	0.72	0.72
accuracy			0.72
macro avg	0.72	0.72	0.72
weighted avg	0.72	0.72	0.72

Confusion matrix:

	Loss/Draw	Win
Loss/Draw	648	244
Win	252	644

Looking at the above metrics, we can see that Naive Bayes has the highest chance of correctly predicting the true label, while Logsitic Regression surpasses Naive Bayes in correctly predicting the other label. Looking at the overall statistics, Logistic Regression performs really well compared to other algorithms.

5.g Inference

After comparing the above algorithms, we can clearly see that Logistic Regression and Naive Bayes perform better than almost all of the other algorithms. We expected them to have more accuracy over the others as the problem at hand is a binary classification problem. In the next section, we have coded our own implementation

for Logistic Regression and Naive Bayes and compared our models with the existing logic from the scikit-learn library. We have also built decision trees for depth [1, 20] using the code from our class assignments and compared it with the the library.

6 Algorithms Used (Own Implementation)

6.a Logistic Regression

6.a.1 Implementation Details

We have implemented logistic regression using the gradient descent algorithm. Results are generated over a total of 100 iterations with a learning rate of 0.2. Using a class, a small plugin-like library has been created for Logistic Regression using these 3 main functions:

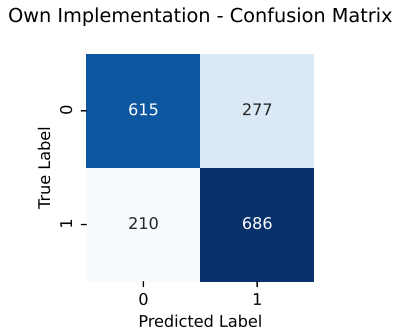
- **sigmoid(x):** This method takes a value and applies the sigmoid function to it.

$$y_{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

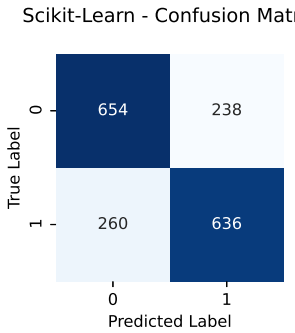
- **fit(X, y):** It takes feature set X and labeled output of training data y as inputs, trains the model, and finds the best weights for each feature.
- **predict_example(X):** Here, we take a single data entry of test data as X and find its output(label) by using the sigmoid function and the best weights.

6.a.2 Comparison with scikit-learn

After fitting the train data sets and making predictions on the test sets, we can see that our implementation with the accuracy of 72.76% performs slightly better than scikit-learn's implementation (72.15%).



(a) Confusion Matrix 1



(b) Confusion Matrix 2

Figure 6: Comparison of Own Implementation VS sklearn

Computing the True Positive Rate (TPR) and False Positive Rate (FPR), we get

$$TPR_{own} = \frac{TP}{TP + FN} = \frac{686}{686 + 210} = 0.77 \quad (1) \qquad FPR_{own} = \frac{FP}{FP + TN} = \frac{277}{277 + 615} = 0.31 \quad (2)$$

$$TPR_{sklearn} = \frac{TP}{TP + FN} = \frac{636}{636 + 260} = 0.71 \quad (3) \qquad FPR_{sklearn} = \frac{FP}{FP + TN} = \frac{238}{238 + 654} = 0.27 \quad (4)$$

Looking at the TPRs and FPRs, we can see that our implementation has a higher chance of correctly predicting the positive classes.

6.b Naive Bayes

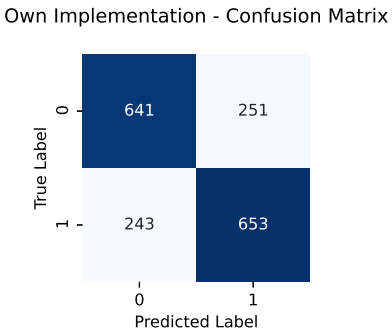
6.b.1 Implementation Details

We have implemented Naive Bayes from scratch. Firstly, we have preprocessed continuous data into discrete data by assigning different classes to values that are greater than the mean and different classes to values that are less than the mean. Using a class, a small plugin-like library has been created for Naive Bayes using these 3 main functions:

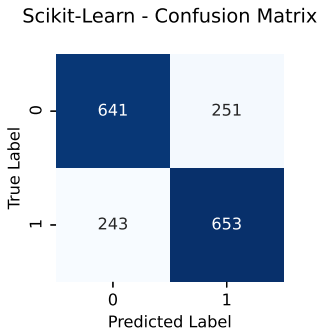
- **partition(x):** In this method, we partition the column vector x into subsets indexed by its unique values (v1, v2... vk)
- **fit(X, y):** This function takes features X and labeled output of training data as y as inputs, trains the model, and finds conditional probabilities for each value of each feature given the value of labeled output.
- **predict_example(X, y):** This method takes a data record X of testing data and labels of training data y. Using the precomputed probabilities in the fit(X, y) method and labels of training output Y, we predict the label of the testing data record.

6.b.2 Comparison with scikit-learn

After fitting the train data sets and predicting on the test sets, we can see that our implementation with the accuracy of 72.37% is in par with the sklearn's implementation.



(a) Confusion Matrix 1



(b) Confusion Matrix 2

Figure 7: Comparison of Own Implementation VS sklearn

6.c Decision Tree

Using the Decision tree codes from the Homework and Assignments, we have computed the model for the depths in the range of [1, 20]. It is clearly evident that the scikit-learn's model performs significantly better than our own implementation. Both of the test accuracies eventually try to converge as we increase the depths but we have computed accuracies till depth 20 to avoid over-computations. We also tried bagging decision trees, but till bags = 10, depth = 20 we could not see any significant improvement.

7 World Cup Simulation

7.a Implementation

We have used Logistic Regression (the best-performing algorithm) as our base model for making the schedule-wise predictions. Using scikit-learn's **predict_proba(X)** method that returns the class probabilities for each match, we have made a prediction on which teams advance to the next round. The schedule of the FIFA matches is in the order (with the format $\text{Stage}_{\#\text{teams}}$),

$$Group\_stage_{32} \rightarrow Round\_of\_16_{16} \rightarrow Quarter\_finals_8 \rightarrow Semi\_finals_4 \rightarrow Final_2 \rightarrow Winner_1$$

We have implemented a function **who_wins(Team A, Team B)** using the **predict_proba(X)** to predict with what probability a team wins over another. The function returns the Winner with the probability by which it will win a certain match.

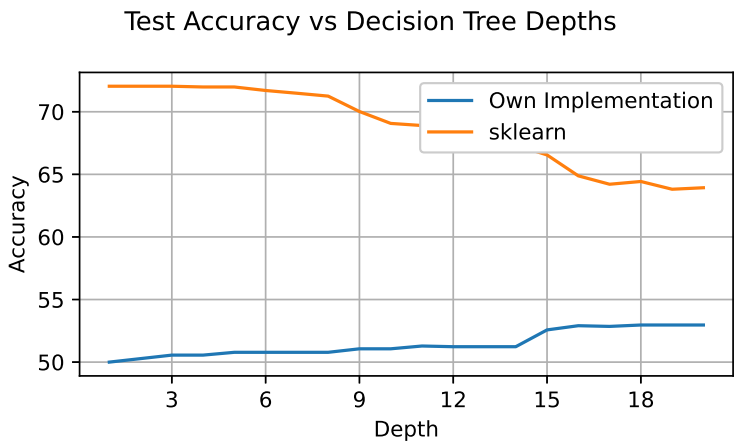


Figure 8: Decision Tree (till 20 depths) comparison

For example,
Our match correctly predicted the outcome of the match between Costa Rica and Spain.

$$who\_wins(Costa\ Rica,\ Spain) \rightarrow Spain\ wins\ with\ 67.49\%$$

However, we also got some wrong predictions (as some underrated teams played really well in this World Cup!),

$$who\_wins(Spain,\ Morocco) \rightarrow Spain\ wins\ with\ 57.20\%$$

7.b Results

Following are the results for the final 2 stages. Each box contains the team and the probability with which it qualified from the previous round after winning against the other team.

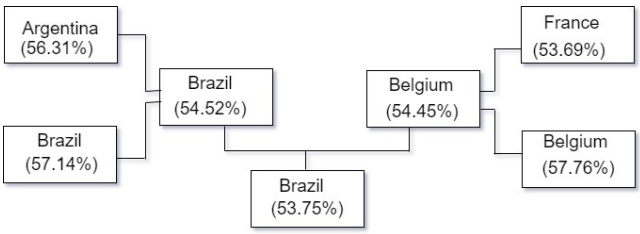


Figure 9: Last 2 stages of the World Cup

Our model started really well on the predictions (compared to the real events) in the initial stages of the matches. But as the World Cup advanced and low-ranked teams like Japan and Morocco started qualifying for further rounds, the probability of teams actually present in the given stage started to decrease. For example, our model accurately predicted 68.75% and 75% of the teams in the Group Stage and Round of 16 respectively, but only 50% (Argentina and France) of the teams in the Quarterfinals.

8 Conclusion

We have compared different Machine Learning Models over various metrics. Being a categorical binary classification problem, Logistic Regression and Naive Bayes have a better performance than other algorithms. We have also created our own implementation for these 2 algorithms and compared those with scikit-learn using a few metrics.

9 What did we learn

Making predictions solely based on historical data has its limitations. There are a lot of other factors (including human intervention) that determine the outcome of real-life events.

10 Scope of Improvement

- Feed the outcome of the prediction back to the model and include it for all future predictions.
- Better handling of the missing data. For example, a lot of old data from the 1990s is not captured in the data set.
- Many other factors can be included that would help in accurately capturing the overall environment. For example, if a significant goalkeeper from team has been injured, we can lower the defense score / overall points of that team.