Hands on with the Kaggle soccer dataset

Arijit Mukherjee(17305t002) , Himadri Sekhar Bandyopahyay(173050004),
Shiva Kulshreshtha(173050023) and Karanveer Singh(173050027)

November 25, 2017

1 The problem at hand

The problem we would like to investigate is predicting the outcome of a soccer game having given , the various factors that may affect the result of the matches in various football leagues(since we are predicting the result of future matches, we donot use any stats of any match). We treat this problem as a multi class classification problem where a match can either be classified as being **Draw, Won,** or **Lost**.

We aim to use machine learning algorithms to predict the outcomes of soccer matches using training data available. The classifier developed would be such that it could be applied to a different sport without much change in underlying model.

2 Dataset and various attibutes of the dataset

The soccer dataset on *Kaggle*[1], provide by Hugo Mathien, consists of data about 25000+ matches,of over 11 European Leagues, with their statistics, players formation, betting information about the matches, and other stuffs.

Also with this we are provided the data about the players involved in the league, with attributes like player ranking, skill, abilities etc.
We are also provided the data about various teams with attributes like buildUpPlaySpeed, buildUpPlaySpeedClass ,buildUpPlayDribbling etc.

3 Where lies the difficulty, and what are the challenges

Soccer is a very unpredictable game, where varoius parameters may affect the game, like a player being injured, weather conditions, where is the match being played, whether it is being played in presence of home crowd or the away crowd. This are just to name a few. Since, it is a three class classification problem (where each dataset belongs to exactly one class), hence the baseline is 33.33%, and in our models we achieved 54 % which is satisfactory. Alternatively we can also treat the baseline to be 45% as the home team wins in 45% cases.

4 Various models used on the dataset

We trained to the dataset on various models so that we can compare each model, and how it performs on each model.We tried to train with various aspects of the data. We used SVM classifier (both linear and kernalised), followed by training the dataset on neural networks. We also implemented AdaBoost Algorithm with base model being decision trees. Finally we tried to approach this problem with a CNN based approach, the details of all these models are presented in subsequent sections.

Linear SVM

Part assigned to: 173050023

Algorithms Applied: Linear SVM.

Data Size: Around 22000 examples, it had 27 features ranging from bets of different bookmakers to weight and height of the players and others.

Accuracy Achieved: Best accuracy of Linear SVM is approximately 51%.

Experiments

Cross Validation was applied to find the best value of 'C' in sklearn's SVC, which is basically the slackness parameter, sklearn's cross_val_score was used for this purpose.

Results

C	Accuracy
1e-6	0.458
0.01	0.510
1	0.511

Confusion matrix

```
[3 2446 321]
[1 4761 260]
[2 2362 823]
```

Motivations

Linear: Linear Kernel worked well with football prediction datasets, as noted by the paper Predicting Soccer Match Results in the English Premier League by *Ulmeretal.*[2]and it is one of the most widely applied classifiers in the industry today, but it suffers from the problem that it is inaccurate when it comes to predicting draws, which is why, we move on to the non-linear SVMs next.

4.1 Support vector Classification -RBF,Sigmoid Kernels and SGDClassifier

Part assigned to: 173050027

Algorithms Applied: SVM with different kernels, SGDClassifier

Data Size: 15000 training examples and 7000 test examples.It had 27 features ranging from bets of different bookmakers to weight and height of the players and others.

Test Data Size: 15000 training examples and 7000 test examples.

Accuracy Achieved: Average accuracy of SVM with RBF kernel lingered around 45%. SGDClassifier also gave similar results

Experiments

A couple of different kernels were applied namely RBF,Sigmoid,Linear. Best results were obtained from RBF kernel. The sigmoid kernel had the tendency to predict the same class for all the test examples leading to erroneous inference.Grid Search was applied to find the optimal values of C and Gamma.

Results

Test and Train error:

During the experiments the test error averaged around .55 and train error around .49 for SVM.SGD also showed similar results the train error was at around .54 and test error around .56

Sample Output showcasing the confusion matrix :

Testing phase:

```
SVM RBF Accuracy: 46.6162674196%
[1008  885  900]
[1486 2512 1038]
[ 949  603 1598]

SGD Accuracy: 43.3828217506%
[ 406 1343 1044]
[ 709 2852 1475]
[ 413 1232 1505]
```

Training phase:

```
SVM RBF Accuracy: 64.9866666667%
[2461  685  706]
[1355 4350 1105]
[ 721  680 2937]

SGD Accuracy: 42.72%
[ 755 1458 1639]
[1343 3253 2214]
[ 730 1208 2400]
```

```
For all the above matrices, the row represents the accuracy on ith class
 and columns represents the predictions on jth class
```

Applications

One of the foremost applications of this is that in the betting markets.Given that it performs much better than the random(33%) , it could be applied to that market. Also this can be easily adapted to different games and different leagues.

Motivations

Kernels: RBF Kernel was proven to work for this problem according to the various papers that were consulted during the creation phase namely: Predicting Soccer Match Results in the English Premier League by Ulmer et al. Sigmoid kernel suffered from the problem of always predicting the same class which was mitigated by using different values of gamma but accuracy suffered as a result.
In the SGDClassifier default values of loss(Hinge) and penalty(L2) were taken. In SVM Grid Search was used to tune the hyperparameters.
The performance of both was relatively similar since the decision boundary is not that complex.So an OvA classifier works relatively well.Also due to the simpler boundary of SGDClassifier overfitting was less of an issue with it than with SVM. Similar results were obtained in the above mentioned paper.

4.2 Fully connected Neural Networks

Assigned To 1730T002

Dataset : The dataset for this model used the various bet amounts by various organisation on the match(this data is very important as it gives us data from reliable human source about the match), along with characteristics of the various teams like buildUpPlaySpeed, buildUpPlayPassing etc. Also we feature engineered the dataset to include the performance of both the teams in the last 5 matches, hoping that it would lead to better predictions. Also , so that we could train our model fast and see the results we used the datas only from three leagues English

Premier League, La Liga and German Bundesliga. **This dataset is the same as used in the next section**

Motivation In order to explore non linear classification , we picked up fully connected neural network model. Due to the huge number of features in the model, we decided to shift to neural networks, so that we can get some additional information from the dataset.

Details of the network The neural network had 3 hidden layers, each consisting of 200,100 and 50 neurons. The activation layer at each node was sigmoid, and the last layer was softmax(so as to normalise the probability). The object function to be minimised was softmax cross entropy loss.

Result Let us see the confusion matrix.

	Model Name	Test Accuracy
0	Linear SVM	0.511
1	RBF SVM	0.450
2	SGD Classifier	0.460
3	Fully Connected NN	0.520
4	AdaBoost Decision Tree	0.540
5	CNN	0.450

Table 1: Comparing all models based on accuracy on test set

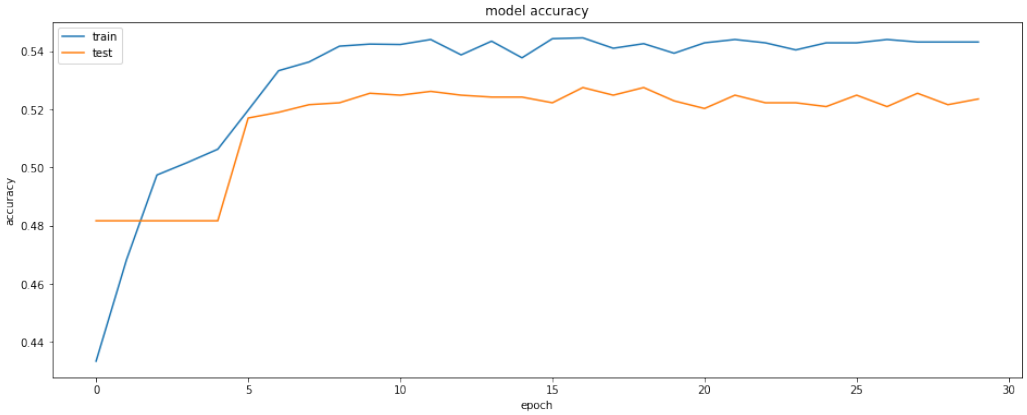

Figure 1: Error vs epochs on test and train datset

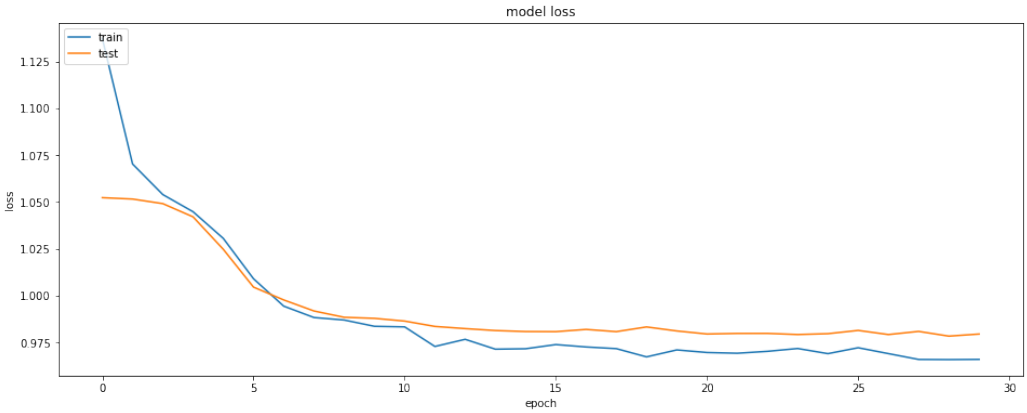

Figure 2: Accuracy vs epochs on test and train datset

Inferences The neural network was trained with a dropout of 0.4, which was obtained after doing K-fold cross validation with K=5. The accuracy on the test set was 52.4% which is comparable to the Boosting method discussed in the later section.

4.3 AdaBoost Algorithm with decision tree as the base model

Assigned To 173050004

Dataset :The dataset for this model used the various bet amounts by various organisation on the match(this data is very important as it gives us data from reliable human source about the match), along with characteristics of the various teams like buildUpPlaySpeed, buildUpPlayPassing etc. Also we feature engineered the dataset to include the performance of both the teams in the last 5 matches, hoping that it would lead to better predictions. Also , so that we could train our model fast and see the results we used the datas only from three leagues English Premier League, La Liga and German Bundesliga.
paragraphThe model:Adaboost algorithm was implemented with the base model as Decision trees. The motivation for this model can be thought of as overfitting of the neural networks, which leads to an overfit trained model, which donot perform well on the test data. However on training this model , we see that this improves slightly better than the neural network model.

The model Adaboost was implemented with its base model begin decision tree, where the hyperparameters were

1. the impurity measure was selected as entropy for each split.
2. max depth of the tree was set to 12.
3. the number of minimum elements that must be present for a node to get spilt was 2.
4. Random set of features was candidate for the split at each node, and out of those random features, the best split was found out.

Result : In this section we have tried to represent various results obtained from training the data on the model. The data was splitted into **train and test test** randomly, and we also found the accuracy after **K-fold cross validation**. The results are discussed in the subsequent section.

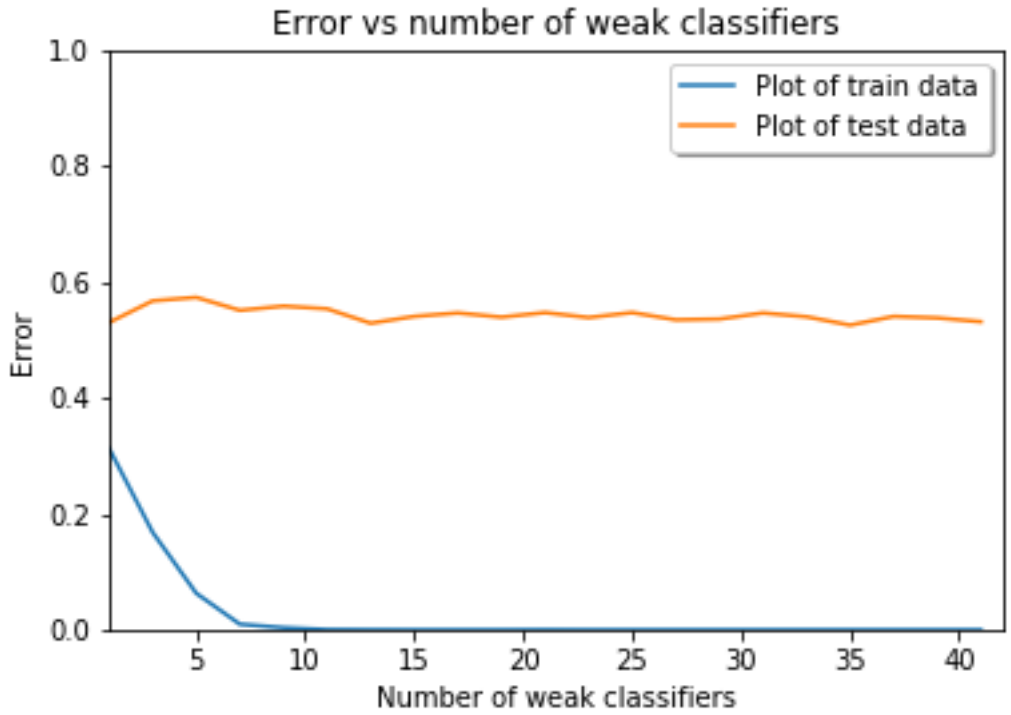

Figure 3: Error vs number of weak learners in the decision tree

	Maximum_depth_of_the_tree	Number of weak_learners	Cross_Validation _accuracy
0	8	1	0.523144
1	8	10	0.461569
2	8	15	0.444997
3	8	25	0.440569
4	10	1	0.505716
5	10	10	0.441714
6	10	15	0.451717
7	10	25	0.455002
8	12	1	0.489432
9	12	10	0.454861
10	12	15	0.455284
11	12	25	0.474002
12	15	1	0.448428
13	15	10	0.450858
14	15	15	0.466286
15	15	25	0.477711

Table 2: Cross validated error on changing the maximum dpeth of tree and number of weak learners

	Unnamed: 0	Predicted 0	Predicted 1	Predicted 2
0	Actual0	65.0	57.0	239.0
1	Actual1	60.0	122.0	207.0
2	Actual2	131.0	125.0	522.0

Table 3: Confusion Matix

Conclusion from the model We got an accuracy of 54% using this model. Though we were hopeful about getting better results from the ensemble model, owing to decrease in variance, but the gain was slight and insignificant. The core problem lies in unpredictability of the dataset, and this is the reason why this dataset is claimed to be one of the hardest one. However it was interesting to observe how the change in the number of base estimators affect the test and train accuracy.

5 A Convolutional Neural Network based approach

Assigned To 17305T002

Approach: Why CNN ? , we wanted to see the problem from all possible aspects , one feature we found really intersting was the X,Y coordinates of players from different team. But including that feature meaningfully to the feature vector was challenging. So other than an image can encode the the X,Y coordinates better ? Next we tried to encode the features inside an image. We encoded the player skills , betting odds and the X,Y coordinates of a player into an image for each match.

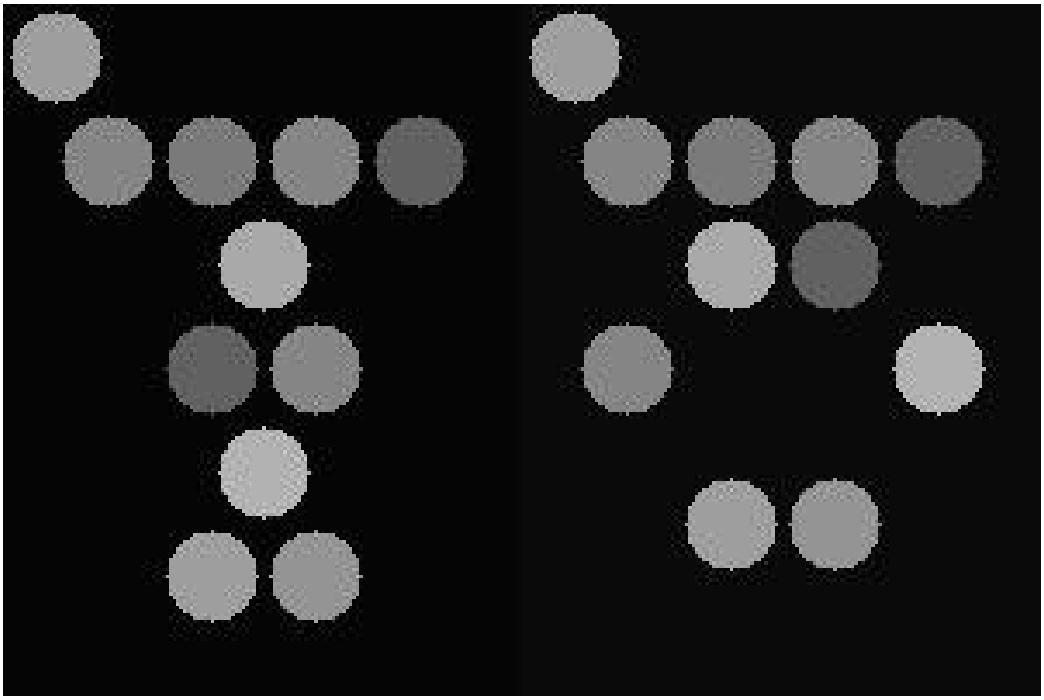

Figure 4: Feature Image of a match

Here each circle is the location of the player , the shade of gray of the circle is the overall skill of the players , the level of gray in each side is proportional to the betting odds of that team winning the match .
We created 25000+ such images for all the matches, also the image is made 3D and other skills are encoded along the depth of the Image in the same way.
We used the VGG16 model and trained it on 3 classes .

Results Though at first this approach looked promising but there was no significant gain in Accuracy. The accuracy was around .45 . From our thinking this may be due the fact that we were not able to logically represent the relation between the player nodes and also the only good representation for CNN perspective was only the X,Y of the players . May be that does not have much significant role in the outcome of the match .

6 Comparing all models

Here we intend to compare all models based on their accuracy.

	Model Name	Test Accuracy
0	Linear SVM	0.511
1	RBF SVM	0.450
2	SGD Classifier	0.460
3	Fully Connected NN	0.520
4	AdaBoost Decision Tree	0.540
5	CNN	0.450

Table 4: Comparing all models based on accuracy on test set

7 Conclusion

The baseline for this task is .45 accuracy , because .45 of the time the Home team wins , so only predicting Home wins will give you .45 accuracy. So we can say that SVM with RBF kernel , SGD Classifier and our experimental CNN failed in this task , while Adaboost with Decision Tree and Fully Connected Neural Network and Linear SVM performed somewhat better . None of the models could make a significant accurate prediction , this is due to the fact the nature of the game is so un predictable, thats why with this much amount of features we were still not able to genrelize the data beyond a certain limit .

8 References

1. Kaggle soccer dataset https://www.kaggle.com/hugomathien/soccer
2. Predicting Soccer Match Results in the English Premier League by Ulmer et al.