Predicting European Soccer Matches

Sahil Goyal

Machine Learning Engineer Nanodegree Capstone

Abstract

Numerous tactics, twenty two players on the field, hearts and passions of the people around the world, a billion dollar industry, one beautiful game: Football. Football is by far the most popular and interesting (a personal opinion) that is out there. Through this capstone project, I present and propose my findings to predict the result of a football match, using a dataset I found on Kaggle that contains data for European Football Matches. I also present my results from a mini-project I did to cluster players into the position on the football field they will be most suited for.

Keywords: Football, Machine Learning, Feature Engineering

I. Definition and Project Overview

Football (or soccer) is the biggest global sport and is a fast-growing multi-billion dollar industry; with an estimate of 27 billion dollars in terms of annual revenue for the football clubs [1].

Prediction of soccer matches is a tough problem. Predicting the exact scoreline is a near impossible task. On the field often defies on paper. So much so, in the 2011-12 season of the English Premier League (EPL), it was not known who the champions would be until the last five minutes of a nine month long season [3]. In the 2015-16 season of EPL, at the beginning of the season it was more likely that Kim Kardashian would become the US president, than the eventual team Leicester winning the title [4].

With more and more money pouring into the sport, the betting industry for predicting the outcome of matches is worth a billions pounds every year[2]. The ridiculous amount of money can be summed in the chart below, that shows the amount of money clubs have paid to secure a football player's services (I being a mere mortal find myself counting the number of zeros in these huge sums of money).

The game involves emotional factors such as the passion of the crowd, how pumped the players are that day, players' personal lives; factors which are beyond measure. In this project, I wish to tackle the problem using factors we can indeed measure.

Figure 1: The most expensive player in the world with time

I.I. Personal Motivation

I would divide my motivation to do this project into three reasons:

- **My interest and domain knowledge** in this field. I follow European soccer with passion and interest. I would like to use my domain knowledge to come up with questions interesting to me, and help me solve those questions.
- **Feature selection and generation**, the biggest take away for me from this nanodegree course is the importance of feature selection, and finding correlations between features. The project on customer segments gave me a chance to apply some techniques such as PCA to reduce dimensionality; and I would like to apply them to this problem, in conjunction with my domain knoweldge.
- **Visualizing results and insights**, I wish to draw some insights to the game of soccer through this project, and want to use some visualizations to analyze the results from my analysis.

I.II. Problem Statement

I try to find answers to the following problems through this project:

- The **holy grail** is predicting the outcome of a football match, as mentioned in the kaggle link for the dataset I intend to use [5].
- The **clustering** of players into their positions on the field; such as midfielder, goalkeeper, midfielder and attacker; based on the FIFA skill data given in the database.

Metrics to evaluate

- For the match outcome prediction is the accuracy of predicting results. More details are given in the section II.II on the base naive predictor.
- For the players clustering project V, I pick some famous players that I know the type and their position on the fields of; and analyze if I cluster them properly.

I.III. Datasets and Inputs

The dataset I will be using for this task is the European soccer database over at Kaggle [5]. The database contains data from 25,000 matches; 10,000 players for the years 2008-2016, spread over 11 European countries. The matches are defined well in terms of data, with information about fouls, shots on target, possession etc. The dataset contains players' skills and ratings, as evaluated by the makers of the popular video game *FIFA*.

I.IV. Solution Strategy

The strategy followed to solve the problems above are listed below. Table 1 lists the major steps of the project, along with the time taken for each of them.

- Familiarize with the database and the tables.
- Generate features based on domain knowledge.
- The machine learning techniques I intend to use for match prediction are *supervised learning* machine learning models, including *ensemble methods* such as *adaboost*, models such as *Decision Trees* and *Neural Nets (MLPClassifier)*.

- For the players clustering project V, I intend to use *unsupervised learning* techniques such as *k means*. Also to reduce the dimensionality, I intend to use *Principal Component Analysis*.
  The expected solution is to get a decent accuracy in predicting match results (more details on the benchmark in section II.II). For the players clustering project, the expectation is to see clearly defined player clusters and be able to correctly classify the famous players into the correct categories.

Overview of time spent. Before diving in detail over the solution and the steps taken, I would like to given an overview of the time spent in the various steps of the solution the problem.

Task	**Percentage of total time spent**
Data Loading and Familiarity	10%
Feature Generation using domain knowledge	50%
Model Fitting and Optimization	30%
Results, Visualizations and Discussion	10%

Table 1: Overview of time spent on various steps of the project

II. Analysis

II.I. Reading the data

The data is in the form of a SQLite database. I used python's *sqlite3* module to load and read the database. The database contains the following tables:

- Table with attributes and related data
  - ***Match*** contains the data for the matches from the database with information such as the country it was played in, between the teams it was played, the season and the date it was played, the players belonging to the various teams and of course the result of the match, among other data.
  - ***Player_Attributes*** contains the data for the players; and data from FIFA such as the player's overall rating, players skills such as shot accuracy, passing, defending, tackle skills etc.

  - ***Team_Attributes*** contains the data for the teams'; such as speed of the build play, defense pressure, passing in the build up play etc.

- Metadata and lookup tables such as ***Player, Team, Country, League*** which are mostly lookup tables mapping things such as player id, team id, league id, country id into their respective names.

Some of the pre-processing to get the relevant data is mentioned here. The reason for this pre-processing will be clearer in the section on Feature generation III.

- **Extracting relevant columns for Matches** The matches table contains a lot of data, that I considered not to be highly relevant; such as betting data; player formations etc. The columns that were selected were the player ids from both teams, the team ids, the season it was played in, the date of the match etc..

- **Extracting relevant columns for Players** To classify players into their relevant skills, columns corresponding to different skills were selected. I also selected the overall rating of each player.

- **Generating the output labels** The output label (the match result) was generated using the columns *home_team_goal* and *away_team_goal*; which give us the goals scored by the home and the away team respectively. Output label for a match was generated to have one of these three possible values:

  1. *HOME_WIN* The label when the home team has score more goals than the away team.
  2. *DRAW* When both teams have scored the same number of goals.
  3. *AWAY_WIN* When the away team has scored more goals than the home team.

  Therefore, this problem is a **multi class classification** problem.

II.II. Defining the base classifier: The home advantage

Now that we have our labels defined, let us define the baseline classifier. Home advantage is real, as mentioned in the Kaggle link [5],and [6]. I analyze the average home advantage across all matches, and also analyze it by country.

I found that constantly predicting a Home Win will get you right about *46%* of the time. Predicting an Away Win will get you right about *29%* of the time. Figure 2 shows the home advantage across different countries. As evident, every league has a significant advantage for the team that is playing at their home stadium.

name	percentage_home_win
Belgium	46.875000
England	45.723684
France	44.703947
Germany	45.220588
Italy	46.635731
Netherlands	47.834967
Poland	45.312500
Portugal	44.249513
Scotland	41.666667
Spain	48.848684
Switzerland	45.710267

Figure 2: Home advantage across different European leagues

On the Kaggle page, the author of the dataset mentions he gets it right with an accuracy of about *53%*; this is another reference to compare my model's performance against.

II.III. Players' ratings histogram and visualization

A histogram visualization to view player distributions in terms of ratings will be helpful. I will be using this to gain intution for features such as 'players_in_top_X_percent'. Sometimes a single **star player** can win a match for their team [7]. Sometimes, a single mistake can cost a team the match. These moments of brilliance, and the grief of mistakes, is what makes football so unpredictable and beautiful. Figure 3 shows a histogram of player distribution by ratings.

We will be focusing on the players that are at the extreme ends of this histogram; more to be discussed in the section on Features III.

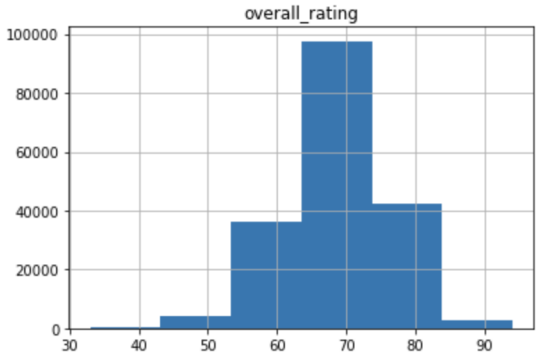

Figure 3: Distribution of players by their ratings

II.IV. Handling Bad Data

The Kaggle page does say that some input values are missing, and indeed it does. I dropped the matches where the values are missing (NaN). For this I use the *dropna* method in pandas [8]. I certainly do not want to blindly check for validity of all columns in the match; a lot of it is information I don't think is necessary for prediction (betting data, number of corners and fouls for example); at least for the problem of determining match results. Therefore I define the columns ***important*** to me. Luckily, even if I drop the data if *any* of the player ratings is absent, I do not lose a lot of data; dropping 4605 matches out of a total of 25000 matches.

III. Methodology

III.I. Feature generation

This is the part of the project which was the most fun, and I got to use my **Domain Knowledge**. Must admit I was a bit nervous about generating features (what if they are not relevant and give me bad results!); more on the results in the section on model fitting III.III. I divide my features into three main categories, and shine light on the rationale and the thinking behind generating these features.

Players' based features. These are the features based on the players' attributes (skills) and FIFA ratings. Each match contains the information about the line up of both the home and the away team; listing the eleven players that played for each team. These player ids' can then be used to

extract information from the players' attributes table and used to generate relevant features such as:

- Ratings for the teams involved in the match
  On the Kaggle page, I saw that some contributors took the average rating of the players; defined for both home and away[9]. I chose a different approach here; instead of taking average across all players; based on my domain knowledge I split players based on the position in their teams.
  I divided the players into four types- *attacker, defender, midfielder and goalkeeper*; based on a naive algorithm to classify players' based on their skills. The four skills I selected were *'finishing', 'sliding_tackle', 'gk_reflexes', 'short_passing'* from the players' table to classify players into their skills; based on a simple algorithm: if the max skill is *finishing*, then the player is an attacker; if the max skill is *sliding_tackle*; the player is a defender, and so on.
  The validation of this algorithm was done for famous players that I know the type of; and it works pity good. This is also the subject of my analysis for clustering players V.

  Based on this naive *clustering* of players; I got four features for every team: the average ratings for their players in attack, defense, midfielder and goalkeeping: Note that goalkeeping is a single player; albeit very important [10], so I kept this as a separate feature. Note that to get the rating of a player, I get his closest rating to the match date, as every player has yearly entries for his rating.
  Datatype These eight features *(4*2)*; are positive float values in the range 0-100.

- Exceptionally good (and bad players)
  It is often seen that a football match is decided by a single player's moment of brilliance, or a player's costly mistake. I visualized the players' ratings in Figure 3; based on this distribution I defined a top player in a team as any player with a rating greater than *80*; this corresponds to roughly the top *1%* of the players in the histogram. I similarly defined the bottom players as the ones with rating below *50*; this corresponds to roughly the bottom *1%* of the players in the

histogram.
For both teams, the number of top and bottom players was calculated.
Datatype These four features *(2*2)*; are positive integer values in the range 0-11.

Form Stats features. A team's current form in this season, its historic home record, its away record, head to head with the team its playing against affect players' confidence; and have a significance in that it tells us the trend of a *stronger club between the two.* A club that wins against another *90%* of the time is likely to win again. Using the data given, I generated the following features in terms of percentages:

- Home Team
  - All time home record - The percentage of home matches that the home team wins historically.
  - This season home record - The percentage of home matches that the home team has won this season.
- Away Team
  - All time away record - The percentage of away matches that the away team wins historically.
  - This season away record - The percentage of away matches that the away team has won this season.
  - Away record at this ground- This is a feature I thought of through following football. If historically a club has won at a particular ground many times in the past, it's fans and players are pumped up; causing *upsets.* These features tells us the percentage of away matches that the away team has won at this particular home team.
- Head To Head
  - Head to head draws- The percentage of head to head matches between the two teams that have resulted in draws.
  - Head to head home wins- The percentage of head to head matches between the two teams that have resulted in the home team winning.

– Head to head home loss- The percentage of head to head matches between the two teams that have resulted in the home team losing/away team winning.
An example for the home team win rate with time for *FC Barcelona*, one of the world's most famous clubs is given in Figure 4.

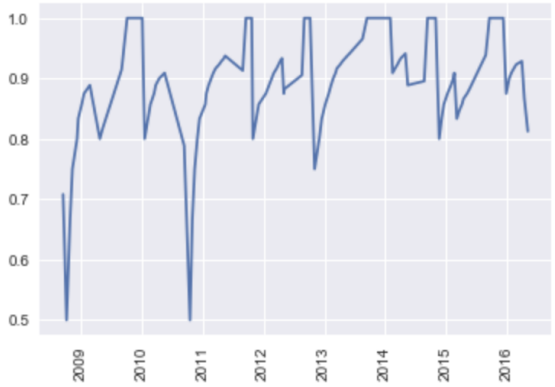

Figure 4: FC Barcelona's home win rate with time (2008-2016)

Datatype These eight features are float percentages in the range 0-100.

Form Guide Features. The idea for a form guide comes to me from years of looking at football league tables; every table has a form guide of the team's performance in the last few matches.
I defined the form guide for a team as an aggregation of its last five results. So if a team has won the last two matches, lost the two before that and won the one before those; its form guide will be ***WWLLD***. A win corresponds to the letter W, loss to the letter L and draw to the letter D.
Note that there are a total of 3^5 possible values for this variable. I calculated this for both teams for the current season.
Datatype These two features are categorial variables.

Filling missing values
You might have observed it, but some matches might not have enough data to calculate the features I suggested. For example: how should I calculate the head to head for the two teams if they have never played before? How should I get the form guide for a team, if this is only the 2nd match of the

season? (I need five matches to define the form guide)
I thought about using sklearn's *Imputer* function; however then decided to implement my own code to do so; in order to fill those *NaN* values more in accordance with my domain knowledge. For example:

- Head to head wins/draw/loss : Take all head to head matches, and replace np.nan with the mean of all non nan values. It is possible that some teams have NEVER played before, in which case I will just randomly assign a value of 0.33 to each of the three events (equal probability of happening).

- Away team's win rate at this ground: Find all samples where this team played away at this home team's ground; and take average. If this team has never played here; just take the average of its all time away record. Note that this is rare, but possible.

- Form guide: Replace the NaN values with the most common form guide for that team during that season.

III.II. An overview of the techniques used

For the problem, I tried out a bunch of supervised learning models, here I list out the top three performing models and the main hyperparameters associated with them.

- AdaBoost Classifier: AdaBoost is an *ensemble* classifier; i.e. it tries to iteratively adjust the weights of incorrectly classified examples and that focuses on the more difficult cases.
  Note that since *AdaBoost* gave me the best performance, I tuned the above hyperparameters in the section on model tuning III.III.
  Hyperparameters include:

  - The **base_estimator** is the base estimator used; I used a *Decision Tree classifier*
  - **n_estimator** is the number of estimators at which boosting terminates; I initially started with 500.
  - **algorithm** corresponds to the boosting algorithm used.
  - **learning_rate** corresponds to the rate by which the contribution of each classifier is shrunk.

- Quadratic Discriminant Analysis [15] is related to *Linear Discriminant Analysis*, albeit it fits the model with a quadratic decision surface using *Bayes' rule*. As mentioned in [15], this classifier is attractive because it inherently works on multi-class solutions (as is the case here); works well in practice and has no hyperparameters that need tuning.

- MLP Classifier is a neural net classifier provided by sklearn. In the case of neural nets, the main concern is *overfitting* to the training data by building a *deeper* network. The hyperparameters for this classifier include

  - **hidden_layer_sizes**, this is specified in the form of separating the number of neurons in each of the hidden layers in terms of comma separated values. This governs the depth of the neural network and the number of units in each layer.
  - **activation** is the activation function used in the hidden layer. I used the default layer, *ReLu* because it has proven recently in neural net research to work better than others.
  - **solver** is the algorithm used for weight optimization. I used the *lbfgs* solver, because the sklearn page says in works well on smaller datasets and converges faster.

III.III. Model fitting and optimization

Weehee! I made it this far. Now to trying out our classifiers and *fingers crossed* that my labour in generating features comes to fruition. But wait.. **I must one hot encode** the result labels and also the form guide variables defined in section III.I.
To do this, I used sklearn's *LabelEncoder* [12]. And yes, now we are ready to classify.

Notes on the model training. I mostly followed what was done in the *Charity ML* project in this course.

- A train-test split of 80-20.

- Using a *F1 score* to gauge performance; although contrary to the *Charity ML* project, this is a multi class problem (has three labels). Therefore for the *F1 score's* averaging parameter values, I chose *micro*.

- I used the same procedure used in *Charity ML* to run a bunch of classifiers and compare their performance for the dataset.

Initial Data Results and Analyzing Feature Importance. After training the classifiers on the training set and predicting the testing set, I found the three best classifiers to be: *QuadraticDiscriminantAnalysis, MLPClassifier and AdaBoostClassifier*. The results for the three of them are shown in Figure 5. The best performing classifier is the *AdaBoostClassifier*; which gives an

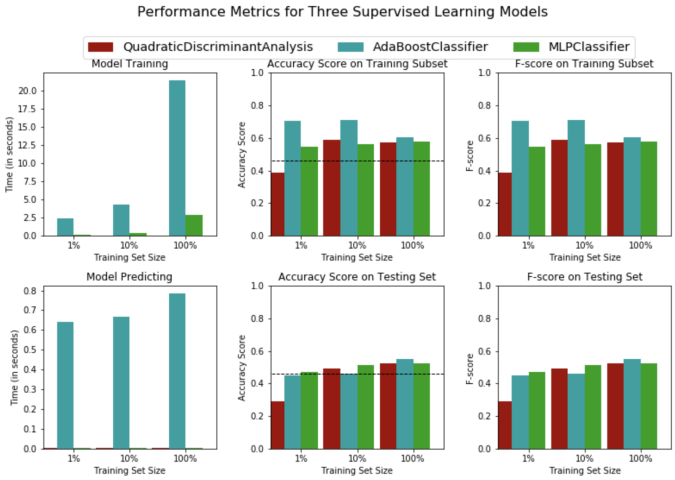

Figure 5: Initial results on the best three classifiers

accuracy of *54.75%*; which is pity good, considering the Kaggle page thus far has mentioned achieving an accuracy of around *53%* and the base classifier is *46 %*.

Phew! My feature generation work has paid off. Now lets dive into checking the *feature importances*. A bar chart showing feature importances is plotted in Figure 6. **Some notes about the feature importances**

- The highest bar in the figure corresponds to the away win rate at this ground (accounts for about *10.8 %* of the feature importances).
- The next five important features are the home team's rating average for it's midfielders and defenders, the away team's historic away win rate and the home team's historic home win rate.
- It was good to see the form guides having decent amount of importance (*5.5 %* for the home team and *4.5%* for the away team).

- Alas, the features on the number of top/bottom players in each team had negligible importances; seems these features are not relevant, as I had hoped they would be.

The *adaboost classifier* is an ensemble classifier, which means it combines a group of *weak learners* to predict the output label. In ensemble we ask a series of simple questions to the data, and it does seem that the away team's historic form at the current match's ground is a simple question that gives us a definite answer.

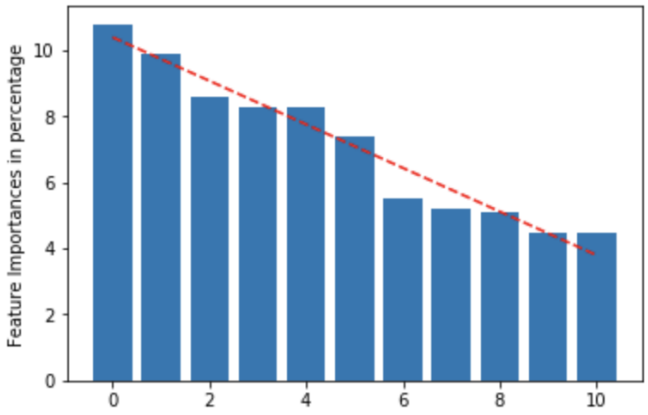

Figure 6: Importances of features in percentages to the best model

Bad Performance on draws and response to it. I had an *epiphany* upon plotting the confusion matrix for the classifier, that gives an idea of the true/false positives/negatives. (Figure 7); I realized I completely ignored the features relevant to the when the matches end in a draw. The classifier predicts home wins with an accuracy of *80%*, away wins with an accuracy of *50%*; and draws with a very bad accuracy of just *12%*; labeling draws as home/away wins instead most of the time.

I generated statistics for home/away team's win rate; but never accounted for their rate of drawing matches. So akin to the feature generation for form stats in III.I; I generated features for the away/home team's draw rate this season; their all time draw rate and away team's draw rate at this ground historically. The feature generation process was similar as before, and so was the process to fill missing (NaN) data.

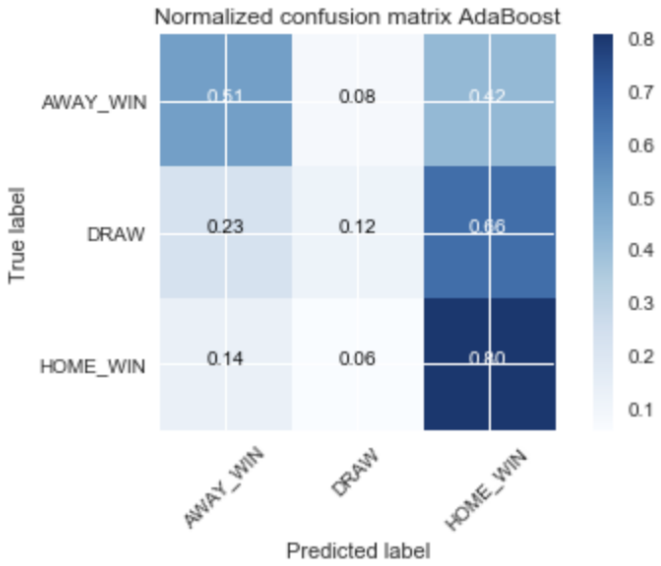

Figure 7: Confusion matrix for the initial adaBoost Matrix

After running the dataset with the newly added features, I ran the process of classification again; and the accuracy remained more or less the same, around *54.5 %*
The results did not improve much in terms of mis-predicting draws, and I obtained a similar confusion matrix as in Figure 6 (with very minor changes). Seems like drawn matches is really my *Achilles Heel* in this project. My guess would be that predicting draws is harder, and would probably need a different set of features.

Optimizing the AdaBoost Classifier. I chose the entire set of generated features, including the ones corresponding to historic draw rates generated in the discussion above to optimize further.
Similar to the *Charity ML* project, I then proceeded to run *Grid Search Cross Validation* [14] to find the best set of hyperparameters for the *adaboost classifier*. The results are discussed in the section IV.

III.IV. Coding processes followed in the steps above

In this section, I give some details about how I implemented the algorithms:

- **Pre-processing and feature generation:** In terms of generating the features, the dataset is huge (25000 matches); I educated myself on the use of *pandas' lambda and partial* functions to process the dataset match by match. This was a fun exercise, and as a programmer, I loved being able to do this after an initial steep learning curve. I also learnt the importance of saving work to disk, in order to being able to quickly pick it up the next day.

- **Model training and prediction:** I followed the same procedure as followed in the *Charity ML* code, by defining a training and predicting pipeline where in I could quickly evaluate the results of my classifiers. I used the same visuals used there, and used to compare the F1-scores, training time and accuracy of various classifiers on the dataset.
  I used various sklearn libraries for the models used.

- **Model optimization** For tuning the *adaboost classifier*, I used the *make_scorer* function in sklearn to define the F-score; and defined the set of hyperparameters I wanted to tune in the form of a dictionary. I used *grid search cross validation* to find the set of hyperparameters with the best score.

- **Visualizing results** For visualizing results, I believed that comparing accuracies across various countries is a nice way to see how the model is performing, and is it adapting to the styles of the matches played in each country.
  I also realized the importance of analyzing the performance on each of the three cases (home win, away win and draw) separately to analyze the model's performance better.

IV. Results

The best classifier found was one with 1000 estimators and a decision tree classifier with *max_depth* param set to 2.
I also ran the classifiers without any draw data; as discussed above, predicting draws is hard and I wanted to evaluate the performance of my model on matches that did not end in draw. The results for the best classifier are summarized below and compared to the base classifier.

Classifier	Accuracy
Naive Classifier(Always predict a home win)	46%
Kaggle's best (Based on page info)	53%
My optimized AdaBoost on entire data	55%
Initial unoptimized model on data without any draws	73%

Table 2: Comparing classification results

Note that I did not optimize in the case of data without draws (as that is not the problem I am trying to solve); but it is evident that predicting draws is the hardest part, at least as per my feature generation and understanding.

IV.I. Discussion

- **Home Wins** are predicted with an accuracy of *80%*; based on statistics, home wins occur *46%* of the time. So this is really good.
- **Away Wins** are predicted with an accuracy of *50%*; based on statistics, away wins occur *29%* of the time. So this is really good.
- **Draws** are predicted with an accuracy of *12%*; they occur statistically about *25%* of the time, so this model is really bad with draws.

IV.II. Accuracy across various countries

The dataset contains data from eleven countries; I took my best model and retrained data on only a particular country's data and evaluated the performance; there is not a marked difference in the accuracy achieved across different countries. **This suggests that my data could benefit from more features that are country specific, for example football matches in England are more aggressive, football matches in Italy are more defensive etc.**. The bar chart comparing accuracy across various countries is shown in Figure 8.

V. Mini Project: Players Clustering

As mentioned before in the section on feature generation III, I did a mini project to evaluate players clustering into their positions on the field. In the feature generation here, I only selected a subset of the players' skills and tried to cluster them using a simple algorithm. Here I try to cluster them using all the skill features provided. The project follows the steps and the structure of the *Customer Segments* project done in this course.

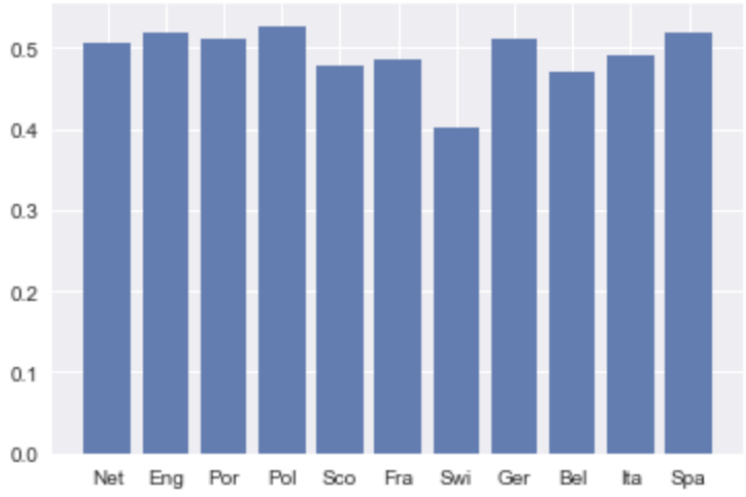

Figure 8: Model's test accuracy per country

V.I. An overview of the techniques used

I briefly list out the techniques used in this mini project

- PCA: PCA or *Principal Component Analysis* is a dimensionality reduction technique used to project data onto a lower dimension space, and tries to capture the *variance* in the data in lower dimensions.

- kmeans: is a clustering algorithm used to separate unlabel data into groups. My reason to choosing k means is both the fact that the number of groups corresponding the different types of players in football, is already defined to be four; and that k means to me is the most intuitive and simple clustering algorithm.

V.II. Choosing the players for validation

To validate the clustering, I choose some famous players that I know the position of. To name a few, The players include attackers Lionel Messi and Cristiano Ronaldo; defenders John Terry and Gary Neville; goalkeepers Petr Cech and Iker Casillas; Midfielders Frank Lampard and Xavi Hernandez. The players were chosen using my domain knowledge so that I can validate if the clustering makes sense.

V.III. Exploring correlation between features

I took all the features and plotted a correlation heatmap (there are a lot of features); shown in Figure 9.

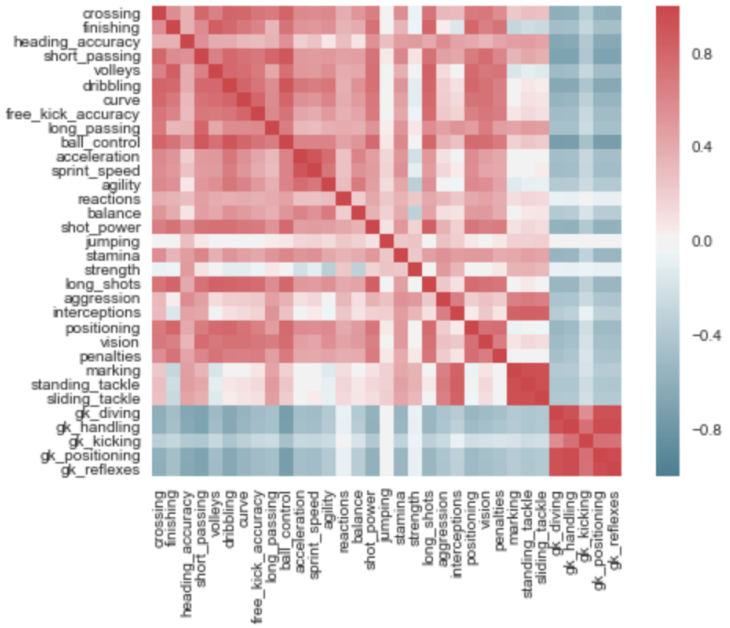

Figure 9: Correlation heat-map of the players' skills features

Look at the rightmost/bottommost features on the X/Y axis and you can clearly see that the goalkeeper features are highly uncorrelated with other features and register low values on the heat-map, thus being blue with all the other features. This makes a lot of sense; as a goalkeeper depends on reflexes, and not a lot of stamina (as he is not running as much as other players).

V.IV. PCA and validating results

There are a lot of features; therefore I used used PCA to boil them down to two *principal components*. The reason for choosing two is to be able to visualize the clusters in 2-D and validate the clustering. The two components had *explained variance ratio* of *46.5 %* and *22.1 %*.

V.V. Visualizing clusters and overlaying chosen players

I then fitted the entire data to these two principal components, and used *k-means clustering* to cluster them into four clusters. Note that these problem is easier than Customer Segments in the sense that we *already know the number of clusters we want*. The results are shown in Figure 10.

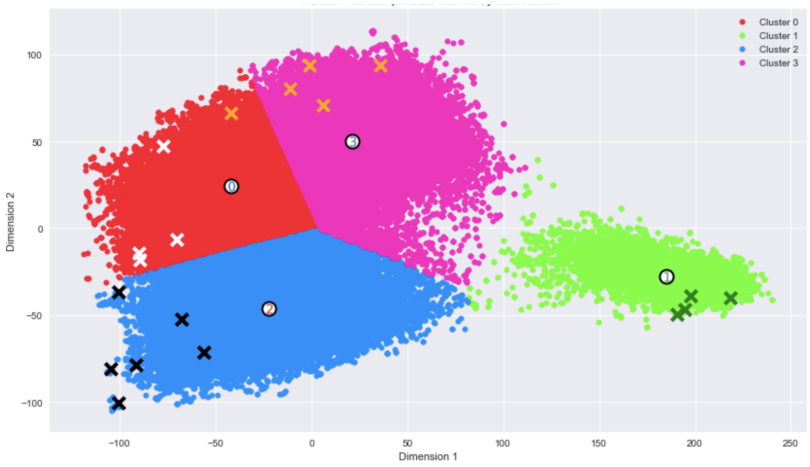

Figure 10: Players clustered into four categories. The markers correspond to the initially chosen player samples, which each player type marked with a different color

V.VI. Discussion

In Figure 10, the chosen sample attackers are marked with black crosses, chosen defenders are marked with orange, midfielders with white and goalkeepers with green.

- The green cluster on the far right represents the goalkeepers; this makes a lot of sense as a goalkeeper has excellent skills such as reflexes, but probably does not have the stamina of other outfield players. Goalkeepers have a well demarcated cluster from the rest of the players.

- The **outliers** are outliers in the true sense of the word. The black crosses to the bottom far left correspond to Lionel Messi and Cristiano Ronaldo; who are by far the two best players in the world.

- There is a case of misprediction, Gerard Pique is identified as a midfielder; he generally plays as a defender, however in more of an attacking

role. I think his ability as a good passer of the ball and also goal scoring ability has classified him as a midfielder.

- All except one players are clustered into the right group.

VI. Conclusion

VI.I. End to end discussion

My work is a solid first effort towards predicting the outcome of football matches. Here is an overview of the steps taken:

- **Preprocessing and feature generation**; I generated all the features using my domain knowledge, and found them relevant to the problem. Through the course of this project, I realized how important domain knowledge is, and how one must *learn the domain* first to do well on a machine learning problem.

- **Model fitting**; after generating the features, I proceeded to try out various models. Luckily I was helped my the code in the *Charity ML* project, however the challenging part was defining the *F1_score* and results in the context of a multi-class classification. I found a *confusion matrix* useful to visualize the results in a multi-class classification.

- **Model tuning** the best performing model *AdaBoost* was then tuned; the challenge here was identifying the important set of hyperparameters and the values to try out.

- **Results and visualization** The accuracy results were analyzed in terms of accuracy per country for the match prediction problem. For the players' clustering problem, the model's performance was evaluated using handpicked samples of famous players.

VI.II. Future Work

A LOT can be done. This is a highly interesting problem, and an interesting database. This problem is also relatively new in the field of machine learning. To give a few ideas,

- Capture country per country variations
  As mentioned in when discussing the model's performance per country; the model might benefit from adapting to the style of each country.

- Find a players' worth
  How much is a player worth to a club? A player who is good at one club, might not suit another club.

- Cluster matches with excitement
  Using the data in the dataset, use the number of fouls, goals, possession stats to gauge how exciting/boring a match is. This is a highly *subjective* perception, so clustering might be interesting to see if there are any patterns pertaining to a match's excitement.

VI.III. Personal Musings and thank you Udacity

Here are some of my personal learnings through the pursuit of this Nanodegree:

- Machine learning is more than just **clf.fit()**. A lot more. Being a programmer who had only some academic machine learning experience, I used to naively think that machine learning is throwing the proverbial *kitchen's sink* of classifiers at the problem. That is really not the case; unless features are chosen carefully, no classifier can do good. This is the biggest takeaway of this course for me, the importance of knowing and researching your problem's domain.

- Defining the problem is almost as important as solving it. Through the projects on Udacity and my capstone project, I have realized the importance of being able to define and express your problem. Working in *iPython notebooks* was something I did for the first time through this course; and I can see why it is a format machine learning engineers prefer; plain code without any context is not very useful in machine learning.

  At last, I would like to give a huge thanks to all the instructors at Udacity, and all the reviewers; you guys are awesome and have a life long fan in me. It had been a while that I had looked forward to studying for something technical with zeal and vigour, and loved it.

  I will be taking the Artifical Intelligence Nanodegree program this coming December, and hope to continue my education there!
  If I had to choose one line from the course, it would be Luis's *Killing a fly with a bazooka vs killing godzilla with a flyswatter.*

VII. Links to code and notebooks

1. IPython notebook for match outcome prediction analysis
2. IPython notebook for clustering mini project
3. Link to code used, please see the ReadMe for details

VIII. Acknowledgment

I would like to acknowledge the following sources:

- I used the code corresponding to the *visuals.py* file in both of the *Charity ML* and *Customer Segments* project.
- Code from the above mentioned projects to run classifiers, clustering and draw results.
- Some code from sklearn, for example the *the confusion matrix* code available online on sklearn to draw the confusion matrices.

IX. References

1. Call for Papers for a Special Issue in the Springer Journal
2. Football betting - the global gambling industry worth billions
3. City beat QPR in the last five minutes
4. Things likelier than Leicester winning the league
5. European Soccer Database
6. How much does home advantage matter in soccer
7. Messi single-handedly guides Argentina to the World Cup
8. Pandas Drop NaN
9. Match outcome prediction in football
10. David De Gea wins the match by himself
11. sklearn preprocessing Imputer
12. sklearn preprocessing LabelEncoder
13. sklearn metrics f1_score
14. Grid Search Cross Validation
15. Linear and Quadratic Discrimant Analysis