Sohum Kaji
November 13th, 2017

Capstone Final Report

Definition

Project Overview

Soccer is a sport that has been played around the world for thousands of years. One would anticipate that given all of the popularity, obsession, and fanaticism involved in the sport there would exist a large amount of data and a huge amount of numerical analyses. Unfortunately, they would be wrong. In The Era Of Advanced Stats, Soccer Still Lags Behind, writes Chadwick Matlin of 538. The key issue, as mentioned by him, is a lack of data sources. Private data services have existed since the 2000 but they have always been priced prohibitively. Until recently there was no place for the armchair data explorer to go.

Enter nowgoal and soccerstats - providing a massive amount of data across a huge number of leagues, all at the low price of $0.00! The new issue is, what subset of the data is valuable? Just browsing over to nowgoal's English Premier League 2016-2017 stats you're presented with tabs upon tabs of links each leading to another page full of descriptive numbers (not to mention this is one of ten leagues in only one particular country).

There is a consensus in the soccer analysis community that the best metric for predicting the outcome of a match is a variation on the theme of expected goals (which also has quite a few haters). Expected goals ignores all but a few features and provides a serviceable prediction metric for the outcome of matches. I do not plan to walk on this well worn path.

Problem Statement

The problem I am looking to solve appears to be ignored by the general academic community on this subject: using an analysis of historical data combined with live match data (up until halftime) to predict the final score of the match. The reason this type of analysis is so unique is that most analyses are based on historical data but do not consider live data. Analyses like expected goals provide a trend that a team generally follows over a long period of time, but is not expected to be accurate in particular instances.

Overall, it is not very useful for the general fan who watches matches and cares about what will occur in a particular match.

To restate: the problem is using previous data from the season as well as what has already occurred in the first half of this match, determining the final state of the game as represented by the score. I have not found any resources that have attempted to do this in this manner.

Metrics

Rather than choose an evaluation function like sum of squared errors, I decided to award points based on the number of true positives. This is complicated slightly by the fact that in this problem there are predictions that can be considered partially "true" and this partialness has degrees as well. For example, if the correct number of goals for either team is predicted that is considered a valuable prediction, however, it is obviously less valuable than predicting the entire score correctly. Another valuable prediction is if the total number of goals between both teams is predicted.

From these three possible valuable predictions I devised a sliding scale of awards: award 2 points for 1 team's correctly predicted number of goals (home or away team), 5 points for the correct number of total goals in the match, and 10 points for the precise score. This is to ensure that the "best" model is evaluated as the one with the most useful predicted scores for the problem rather than the lowest error from the truth (which, for example, in an edge case might have very few actual correct predictions).

Analysis

Data Exploration

The features provided by the fulltime dataset are: possession, fouls, shots, shots on goal, off target, corner kicks, yellow cards, headers, successful headers, saves, blocked shots, tackles, dribbles, throw-ins, total passes, and pass success rate. As a fan of the sport, I'm familiar with the lore of casually tossed around statistics and I wanted to put some of the age-old "common-sense" theories to the test.

The first of these is the idea that possession and shots are the most important stats to judge how good a team is. The common thought is that the number of shots taken and the amount of possession can clearly explain the course of the match (and therefore, would be the most important features in Random Forests). The jupyter notebook for this analysis is called Random_Foresets_fulltime.ipynb . The results were extremely interesting. It looks like the pundits and the armchair coaches were wrong (kind of)! The Random Forest was trained as a classifier with a maximum depth of 9 (slightly more than half of the features). I chose this number to avoid overfitting but also to provide an accurate classifier. Even with a depth of 9 we are only scoring 52.76% on the training data! However, that makes sense as in sport we know that while a team may do better in the stats only 1 stat determines who wins. In soccer, a shot placed 6 inches to the right or left is not an indicator of how well your team played but it is the difference between 1-0 and 0-0. This is what makes prediction in soccer and other low scoring games very challenging.

The most important feature was shots on goal with an importance of 18.14%. Given that there were 16 features, the average importance would be 6.25%, so shots on goal was nearly 3 times as important. In a lot of ways this seems obvious, the team that scores the most goals is often the team that takes the most shots. The other features that have an importance of above average are: pass success, corner kicks, and shots in that order.

This seems to make sense because corner kicks and shots are both opportunities to score so it makes sense that they are above average (though barely) related to goals. What is extremely interesting is how high pass success rates (8.12%) while in comparison possession is at 4.91%, while in discourse pass success is almost never discussed. It seem as though these stats are related. It is hard to maintain a lot of possession without a high pass success rate - the other team would just steal the ball from you. Thinking about it differently, however, it seems that how accurate you are with the ball when you do have the ball is more important than how long you have it. Again, this seem like a really reasonable point and I think I will discuss it a lot more in my daily life with friends.

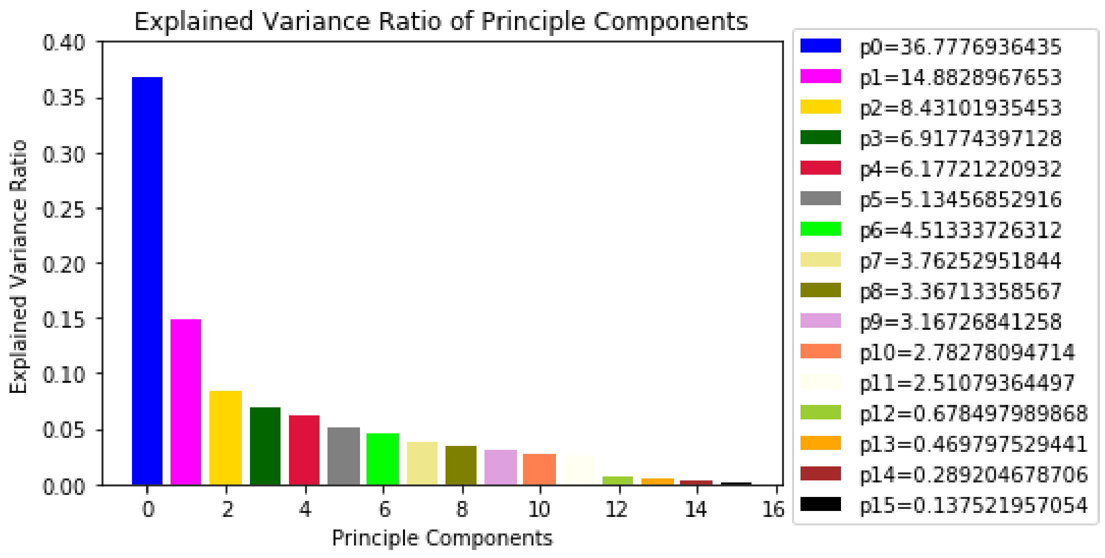
Doing this same exploration but for the halftime data (in the file: Random_Forests_halftime.ipynb, which has the features: ball possession, shots on goal, shots off goal, blocked shots, free kicks, corner kicks, offsides, throw-ins, goalkeeper saves, and fouls. I used the Random Forests Classifier again, except this time there were only 10 features and unless I used 9 or 10 of them I could not get a 50% score for the classifier. To me, this clearly implied overfitting. For this reason, I decided to go for a score of 40% requirement and chose 6 features which gave me a score of 40.56%. It makes sense that it is harder for the halftime data to predict the fulltime result than it is for the fulltime data to do so.

The two most important features were shots on goal and ball possession. Looks like the pundits are on to something! A key fact here is that there is no halftime stat for pass success. Another interesting discovery is that only those 2 features are above the average importance of 10 features (10%). The next highest feature is corner kicks at 5.27%.

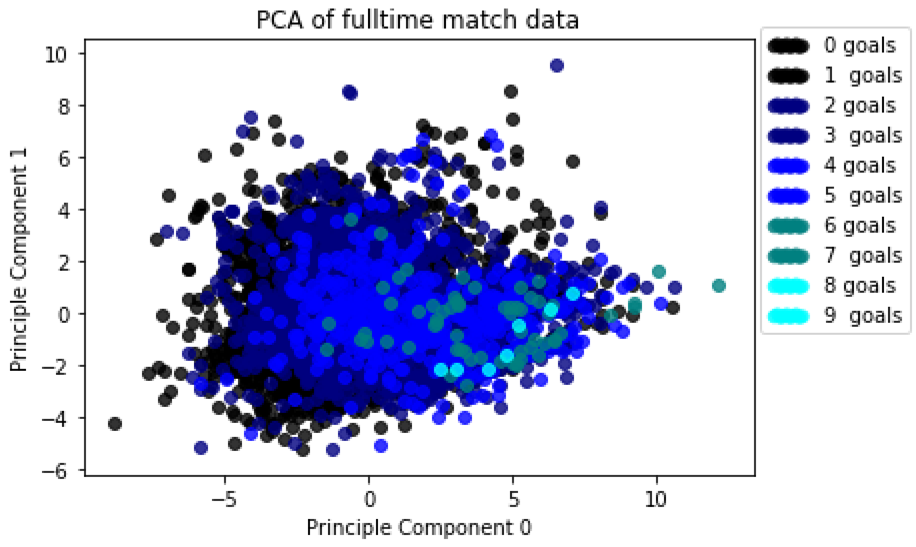
Either way it seems that shots on goal is the best predictor of goals even versus total shots and possession. It seems so obvious now but I only realized this after exploring the data.

Exploratory Visualization

The Random Forest Classifier was nice to see which of the features was most important. Continuing our exploration of the data, I wanted to see how the features in the data related to each other. This was done through PCA. PCA calculates combinations of the features called principal components which can represent the data, often in a smaller dimensionality than the original features. I used PCA on the fulltime data and here is a description of the explained variance ratio of each new component:

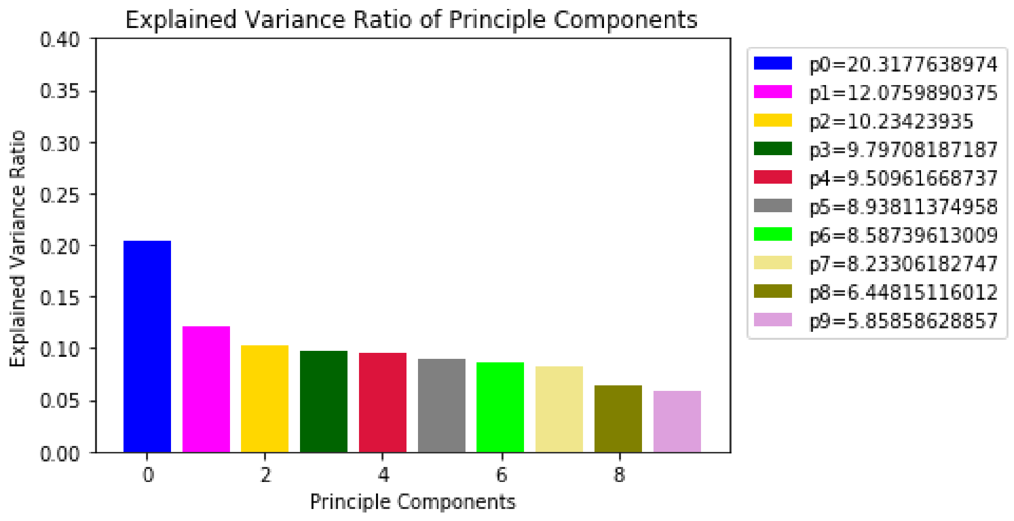


The explained variance ratio shows how much of the variance of the original data each principal component can encode in comparison to how the remaining principal components can encode. Therefore, they always sum up to 1. In the graph above, I've converted them from ratios into percents for easier reading. The main note of interest from this graph is that there are 2 components P0 and P1 that can explain over 50% of the variance in the data. I wanted to examine these two components better, so I created this scatterplot:



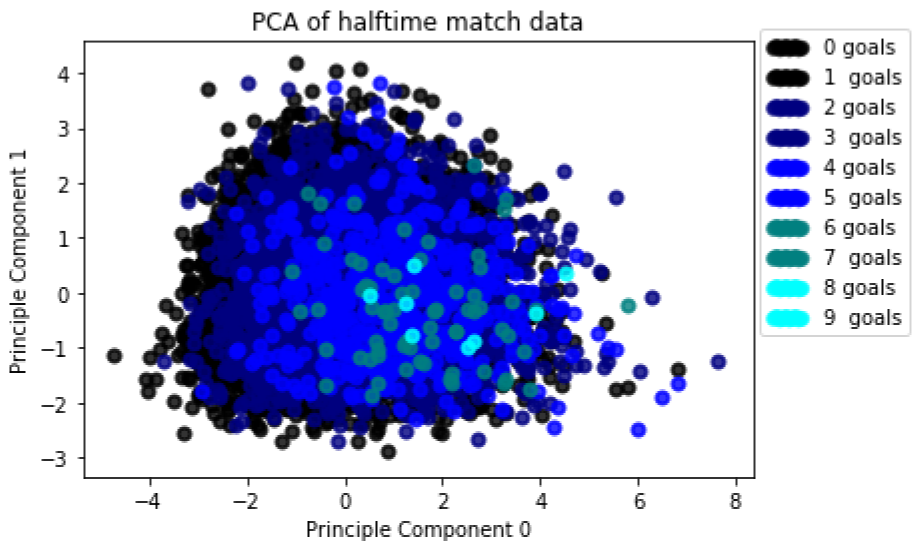
This graph shows where the different labels lie when graphed using only principal components 0 and 1. I've grouped each of the labels into groups of two as the original graph of 10 distinct colors was not legible at all (the points were not clustered distinctly enough). The labels in this case are goals scored by the team in the match described by the feature vector. As we can see a pattern emerges and as the number of goals increases the datapoints appears to move in the positive direction for principal component 0 and towards the 0 from component 1.

I did the same analysis for the halftime data, it can be found in PCA_halftime.ipynb:



For the halftime data, most of the features have an explained variance ratio of just under the average of 10% (for 10 features). The two principal components that have the largest explained variance ratio only explain 32.39% of the variation in the data.

Here is a graph of those two components:



It is harder to tell but the datapoints appear to move in the positive direction of principal component 0 and the negative direction of the principal component 1 as the number of goals increases.

Algorithms and Techniques

The two main techniques used for this problem will be a Stochastic Gradient Descent Classifier and a Neural Network.

The SGDClassifier iteratively attempts to minimize an objective function. It is suited well for our data as it performs best on large datasets and it outperforms other similar methods on noisy data (like the data in this project). Due to the stochastic optimizations in SGD it outperforms other gradient descent methods on large, redundant data sets (also like the data in this project). The datasets used in this project are taken from actual games over the past 3 years and therefore have many redundancies.

The SGD classifier was trained on the historical full match data which has the 16 features that describe the home and away team each at the completion of the match played between the two. These 16 dimensional vectors were fed into the

SGD classifier and the labels were vectorized versions of the final score. Upon completion we could test the SGD classifier versus the benchmark model (even prior to using the NN) using our custom scoring function. As mentioned above, the function gives 2 points for matching 1 team's score, 5 points for matching the total number of goals, and 10 points for a perfect match. 0 point are given for the remaining possibilities.

After the SGD classifier was complete we could implement a Neural Network which takes the halftime data for both teams as an input as well as a final score prediction from the SGD classifier. The halftime data has 10 features in this case and will also be fed into the input layer of the NN.

The neural network has an input layer, 2 hidden layers, and an output layer. The final layer of the Neural Network will use the softmax function. The optimization algorithm used in the Neural Network will be the Adagrad algorithm. Adagrad adapts the learning rate to the parameters. The benefit for our data is that Adagrad performs smaller updates for frequent parameters and larger updates for infrequent parameters. This is important for our data as many of our points will be matches that have 0-4 total goals. The smaller updates for these types of matches and the larger updates for matches with more goals (the max goals in our data is 11!) allow for better predictions for the less likely matches.

Benchmark

The benchmark for this model was created by predicting a final score based on the halftime score for each match:
gA = goals scored by Team A in the first half
gB = goals scored by Team B in the first half

[2*gA, 2*gB]

So, if a match was 1-1 at halftime the benchmark model would predict:

[2, 2]

The above describes two outputs of: double the home team first half score and double the away team first half score. This brings about a very rudimentary but baseline prediction that: the same result as occurred in the first half will occur in the second half. This benchmark was chosen as, to markedly outperform this type of prediction, the project would have to some insight the second half.

Methodology

Data Preprocessing

Collecting and cleaning the data was a large part of this project. Multiple sources were used, the scheduled matches with halftime and fulltime score were provided by worldfootball.net. The flashscore archive was used for the halftime data. The historical fulltime data was provided by nowgoal. It was important to use each of these sources as they each had particular information that the other's did not. The method of collecting data was using Selenium scripts which have been included in the submission folder as: DC_schedule.py, DC_halftime.py, DC_fulltime.py. After scraping this data (it is made for public use) I had to clean it and explore it.

A large component of the cleaning was dedicated to the cleaning and formatting of the scraped web data. As mentioned in the first paragraph the scripts DC_schedule.py, DC_halftime.py, and DC_fulltime.py are all customized to read in data from the particular websites that store the data they need. From that point it needs to be categorized, converted, and formatted into the standard format we need for the SGD classifier and the NN. They were eventually saved in .csv files for easy manipulation for the data exploration, SGD classifier, and NN.

Next I wrote functions to take care of matching up the names between teams as there seems to be some variability in the naming conventions on different websites and in different countries and dealing with dates, missing values, and some sporting anomalies.

In order to deal with the varied naming conventions, I wrote two functions: remove_spaces() and name_fix(). remove_spaces() is included in both the halftime and full match data collection and, as you guessed removes the spaces in names using regular expressions. In addition to that, it has some hardcoded conversions between team names as well.

The name_fix() function is used in both the SGD classifier and the NN to create a quick heuristic for determining which teams in the schedule of matches correspond to the halftime and fulltime match data. This is important as due to special characters, founding dates, and the inclusion/exclusion of city names there was no pattern in why the names were not identical. The heuristic is based on taking a sum of the total matching characters between the name in the schedule and the list of all names in the data. The highest match is the correct name, except in one scenario (Barcelona FC and Espanyol Barcelona needed to be hard coded).

The conv_date() function converts all of the match dates into the number of days since the year 2000 that the match occurred upon. This makes it so that an integer can refer to a match date and also so that I can perform date subtraction and date comparison more easily than using Datetime objects.

Finally, for the SGD classifier and the NN we need reorganize the halftime and fulltime data so that we can pair the teams that need to be paired for each match in the schedule. Further, the scores need to be vectorized and then reshaped so that they can be used for the SGD classifier and NN.

Implementation

The implementation of the SGD classifier was done in the SGDClassifier.ipynb jupyter notebook. The main process involved was two nested for loop which were able to take in a list of match schedules and for all of the matches on a particular day to find the corresponding data for the matches that occurred on each day. This involved reading in particular lines from a dataframe. After doing that the features were appended in order to the X_train variable while the labels were appended to the y_train. A train/test split of 80/20 was used.

From there I ran into my first major issues. I learned that I could not use the match scores as I had stored them (tuples), nor could I use the SGDClassifier as I had learned in the course so far. The solution to these problems is discussed in the Refinement section.

The implementation of the neural network was a bit more complicated. First I began by collecting and organizing the data in same processes as in the SGDClassifier. It was more challenging because we needed to account for both the SGD data and the NN data. It was especially important as the testing data for both models had to be paired for our predictions to be accurate.

Many more functions and lines of code were dedicated to formatting and manipulating the data into a format that was functional for both the SGD and the NN. After all of that, we needed to define the neural network. I originally chose to use a network with 1 hidden layer but I discovered that the complexity was too low for the data. I ended up using 2 hidden layers, the first with 1024 nodes and the second with 512. From there I created the required weights, biases, and propagation function and implemented softmax for the output layer and Adagrad for the optimization function. To view the progress made in epoch, I also implemented a calculation of the average cost for each epoch.

Refinement

Refining my match scores methodology required learning about the Multi-label Binarizer class in Sci-kit learn. The number of goals scored in the match needed to be represented as a binary vector for ease of use with the planned SGDClassifier and the Neural Net.

The main issue with the SGDClassifier was that I needed to submit two vectors for input to it and I wanted to receive a prediction of two labels in return. This was a problem for me because it is not how stochastic gradient descent works in the lectures that I had seen so far. I did some research and I found out that the problem I had was that I wanted to use multi-label learning. My solution was to use the OneVsRestClassifier provided in sklearn.multiclass. It fits a 2-d matrix where one dimension is samples and the other is labels. A 1 indicates that the sample has that label and a 0 indicates that it does not.

Next, the NN posed a number of difficult questions that required refining over time, the first being how many hidden layers and how many nodes in those layers? I started out with 1 hidden layer and 256 nodes in each as a starting point. After doing some research I learned that 1 hidden layer is useful for a continuous mapping from one space to another. From playing with may data I realized that this was not the situation I was in and decided to switch to 2 hidden layers. The key benefit being that 2 layers allowed for the network to represent an arbitrary decision boundary - which seemed to be more relatable to my data. The number was chosen through experimenting with the data, the cost function, and the speed that each epoch was processed. In a compromise to benefit all three of those metrics I ended up with a first layer of 1024 and a second layer of 512. I also needed to choose an optimization function. I did not necessarily even need to choose a cost function but I realized how that was more valuable for looking at my data than the metrics of precision or accuracy (in the business of guessing goals in soccer games, we expect to be wrong more often than right).

The result was using the Adagrad gradient-based optimization. The biggest benefit is that Adagrad adapts its learning rate to the parameters. This mean that it uses larger updates for uncommon data and smaller updates for common data. As mentioned above, this type of modulation is integral for the type of data we are looking at.

Finally, the most imposing and interesting refinement: “how do you incorporate the SGDClassifier predictions into the Neural Net”. The obvious choice was to take the output for the SGDClassifier and append it to the input for the Neural Net. Since the SGDClassifier was a sparse matrix of width 15, it filled my data with a lot more zeroes. Overall, it did not provide as much of a gain as I had hoped. I ended up also using the mean of the NN and SGD outputs as part of the post-processing for the Neural Network outputs.

Results

Model Evaluation & Validation

The result of all of the tinkering described above is a Neural Network that takes in halftime data and SGD Classifier data and outputs an insight into the final score of the match. I used a custom validation technique and the benchmark model to quantify the level of this “insight”.

As mentioned above, the custom validation technique is to give 2 points for 1 matching score, 5 points for matching total goals, and 10 points for a matching precise score. 0 points are given otherwise.

Unfortunately, this scoring method weighted the comparably worst category of the NN + SGDC model the highest (and rightfully so). Due to this the benchmark outperformed not only the SGDClassifier but the combination of the two as well. The benchmark was based on doubling halftime scores and received a total of 4,553 points over 1,479 predictions. The NeuralNet and SGDClassifier combination received 3,788 points and the SGDClassifier by itself received 3044.

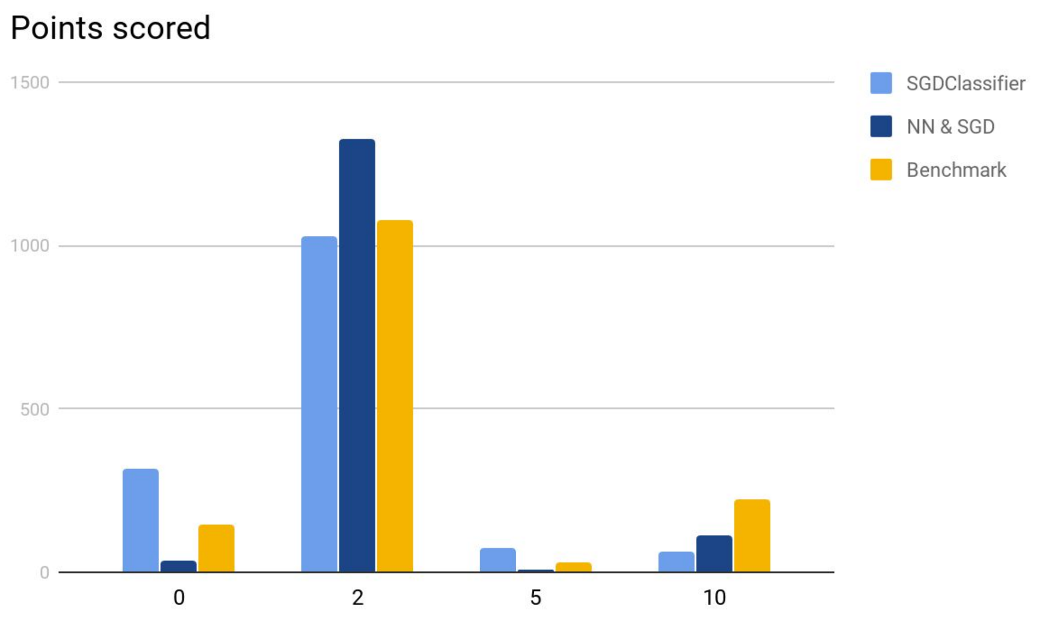
Justification

The final model and solution is not good enough to have adequately solved the problem of predicting the scores of soccer games. However, a portion of this project was to attempt to use publicly available, free data to do what no one else has ever done before: make a model that can successfully choose the correct score of a match. Soccer data packages generally charge thousands of dollar per month and currently there appears to be no model that can successfully select the score of a match from halftime. Part of this project was to see exactly why this is the case and I think that I successfully learned how.

Conclusion

Free-Form Visualization

So, it is clear the neural network helped but it is still worse than the benchmark. As mentioned above, points are collected through 3 means: match the entire score = 10 points, match the total number of goals = 5 points, match 1 of the team's goals scored: 2 points, otherwise you get 0 points. Here is a graph of the breakdown for the 3 models:



As you can see the NN & SGD model is great at choosing 1 of 2 scores correctly but the most valuable category is dominated by the Benchmark.

Reflection

The final model and solution is not significant enough to have adequately solved the problem of predicting the scores of soccer games. However, a portion of this project was to attempt to use publicly available, free data to do what no one else has ever done before: make a model that can successfully choose the correct score of a match. Soccer data packages generally charge thousands of dollar per month and currently there appears to be no model that can successfully select the score of a match from halftime. Essentially, the community would propose that this was an extremely daunting task. For me, part of this project was to see exactly why this is the case and how far we could go and I think that I successfully learned that.

Improvement

The two easiest improvements for this project would be to have a better source for the data as well as more data and incorporating a doubling of the halftime score into the main model. A new benchmark would have to be used but as we discovered through this project: over the last 3 years, in the top 5 leagues of Europe, doubling the halftime score is an accurate predictor of the final score about 15% of the time. I would never have guessed that prior to starting this project.

Beyond these two technical improvements I think this project could have been improved by asking a more focused question, such as: “how many total goals will be scored in the game” - essentially the question associated with receiving 5 points in our model. I feel this question has a lot of untapped potential to be answered since our model was more concerned with getting the precise scores it was the least scored category in all tests. I believe it has the most potential to be “solved” of our 3 evaluation criterion. Personally, I find it very interesting and I plan to continue work on it in the future.