Beating the Odds:
Learning to Predict Soccer Matches Using Historical Data

Michael Painter {mp703}, Soroosh Hemmati {shemmati}, Bardia Beigi {bardia}

Department of Computer Science, Stanford University

Introduction

The primary objective of this project is to learn and predict the outcome of a soccer match based on historical data from various leagues over 8 years.

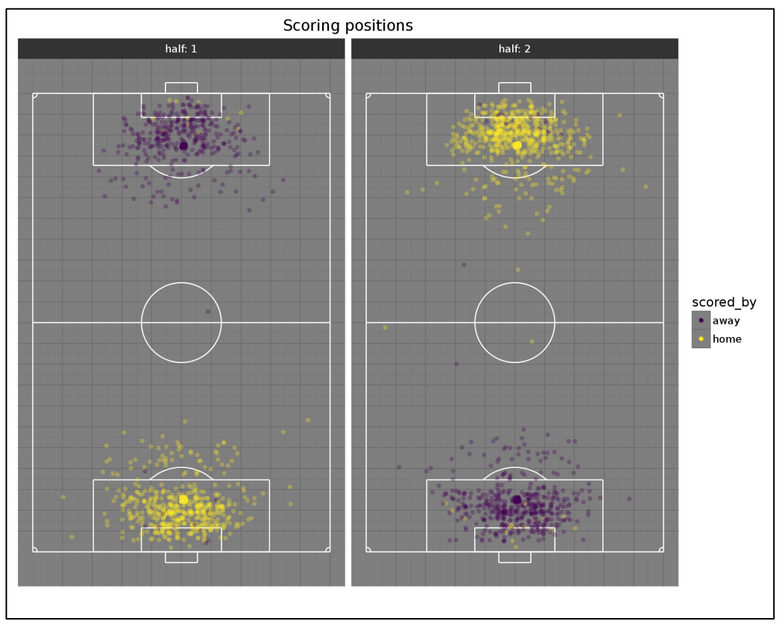

Figure 1: Visualization of Some of the Data

The task is to predict match outcomes, which involves 'classifying' each match as one of: home win, draw, away win.

Data

Our data comes from a comprehensive Kaggle dataset. It contains many different attributes (shots, possession, etc) over 8 seasons of soccer played all over Europe, including stats from the FIFA video games. It contains the ground truth labels (ie. home win/away win/draw) for all matches.

Features

We have 4 classes of features, totalling over 1100, categorized into team, match, player, and betting data features.

- Team Features:
  - Build up speed, formation, passing class, etc.
- Match Features:
  - Team and player names, shots, cards, etc.
- Player Features:
  - Dribbling, passing, running speed, etc.

Future Steps

- Need to investigate further the correlation between accurate match predictions and successful betting on match results.
- Upon selecting and confirming the best features, we can look at derived/correlated features for further optimization.

Models and Algorithms

- **Support Vector Classification**

Extends SVM using "one vs rest" scheme. Training one SVM per output class, and taking the "highest score". We used the Radial basis function:

$$K(x, x') = exp\left(-\frac{\|x - x'\|^2}{2}\right)$$

and linear kernel:

$$K(x, x') = x^T x'$$

- **Softmax Regression**

Another model used was softmax regression, outputting a distribution according to:

$$p(k \mid x; \theta) = \frac{exp(x^T \theta_k)}{\sum_{i=1}^{3} exp(x^T \theta_i)}$$

The θ is learned using stochastic gradient descent, with a cross entropy loss function.

$$L(\theta\ ; x, y) = x^T \theta_y - log\left(\sum_{i=1}^{3} exp(x^T \theta_i)\right)$$

- **Multi-Layer Perceptron**

Using 27 hidden layers each with 20 nodes, it achieved a reasonable test error of 52.6%. Multi-Layer Perceptron uses backpropagation with stochastic gradient descent to find the optimal classification. Each hidden layer applies the arctan activation function to the value of each node.

- **Feature Selection**

Forward search is used for feature selection, to select a small set of features and reduce the dimensionality of input data. Forward selection was used over backward selection because the runtime would be unreasonably slow. Interestingly, "awayTeamLeaguePosition" was almost always selected as the first feature, which is intuitive.

Results

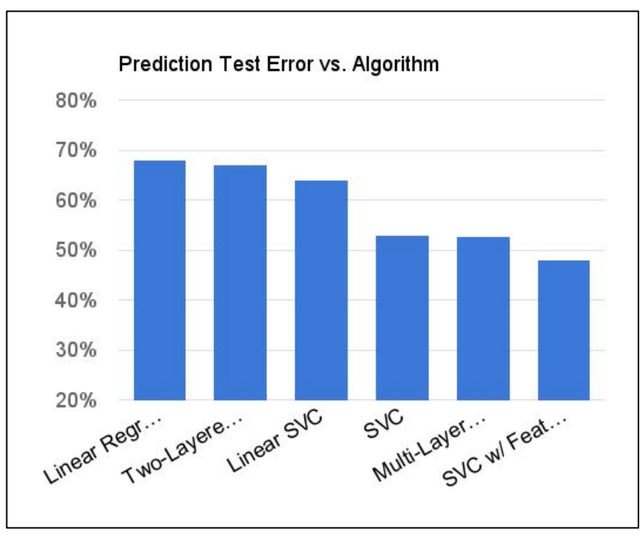

Figure 2: Prediction Test Error vs. Algorithms

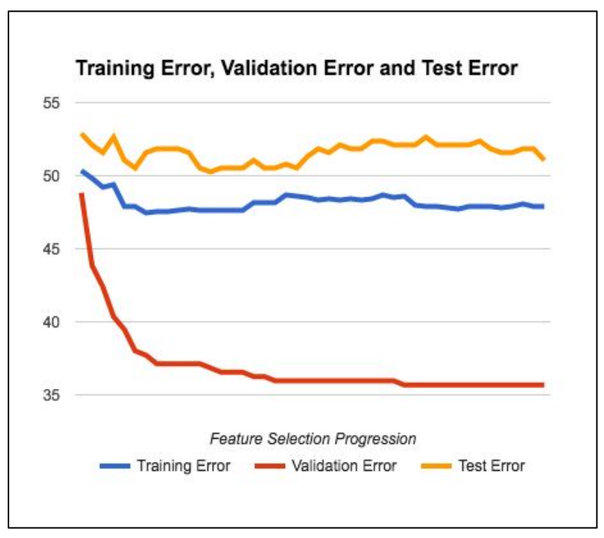

Figure 3: Effects of Running Feature Selection on Various Errors

The best results came out after running feature selection:

- We used forward selection to find the best subset of features that minimizes CV error,
- Those features were then used to find the generalization (test) error.

In the process of running feature selection, the validation error is minimized by adding more features to the optimal feature subset.

In doing so, the validation error is reduced by adding the most informative features.

The test error is then computed on a test set, using the optimal feature subsets generated up until that point.

Discussion

We have seen forward selection pick very interesting features so far, specially in the first few iterations. In addition, we have seen the validation error to be really low, which seems to suggest that using the first little bit of a season along with the training set to be a very valuable idea in predicting the rest of that season. So far, it has been really difficult to get test errors below 50% but with a few extra iterations in the same direction that should be possible. In addition, incorporating player features has so far hindered our path to optimization.