Using Machine Learning to Predict and Bet on Professional Tennis Matches

Andre Cornman, Grant Spellman, Daniel Wright

Objectives

Our project had two main components:

- Use historical data from tennis matches to develop a model to predict match outcomes.
- Use this model and historical betting data to develop a betting strategy.

Data

- Tennis match data was retrieved from an open source data set available on GitHub. It includes all match results from the Open Era (1968) to September of this year. Only the more recent matches have statistics associated with them.
- Betting data was retrieved from `http://tennis-data.co.uk/`, which has odds from various betting services from 2001 on.
- These two datasets had to be merged. We were able to merge about 93% of the data. The merged dataset has 46,114 matches.
- This was split into a training set of size 41,324 and a test set of 4,790. (Roughly a 90-10 split.)

Features

- We constructed features from match statistics that we thought would be relevant for match prediction.
- Careful not to have any "look-ahead"... features one could not know in advance of the match being played
- In total we had 85 features. Each feature was of the form, for symmetry:

$$\text{FEATURE}_i = \text{STAT}_{i,\text{player1}} - \text{STAT}_{\text{i,player2}}$$

Some of the player statistics used were:

- Rank, ranking points
- Head-to-head wins
- Head-to-head wins on the given surface
- Match wins in the last $\{5, 10, 20\}$ matches
- Match statistic averages for last $\{1, 2, 3, 4, 5, 10, 20\}$ matches (such as aces, double faults, etc.)

Prediction Models

We tried a variety of models and evaluated them using **5-fold cross validation**.

Model	5-Fold CV Accuracy
Random Forest	69.7
Neural Network	65.2
(1 HL, 300 nodes, logit.)	
SVM	
→ Linear Kernel	69.9
→ RBF Kernel	51.0
→ Polynomial Degree 3	54.0*
Logistic Regression w/L1 Reg.	69.9
Logistic Regression w/L2 Reg	69.7

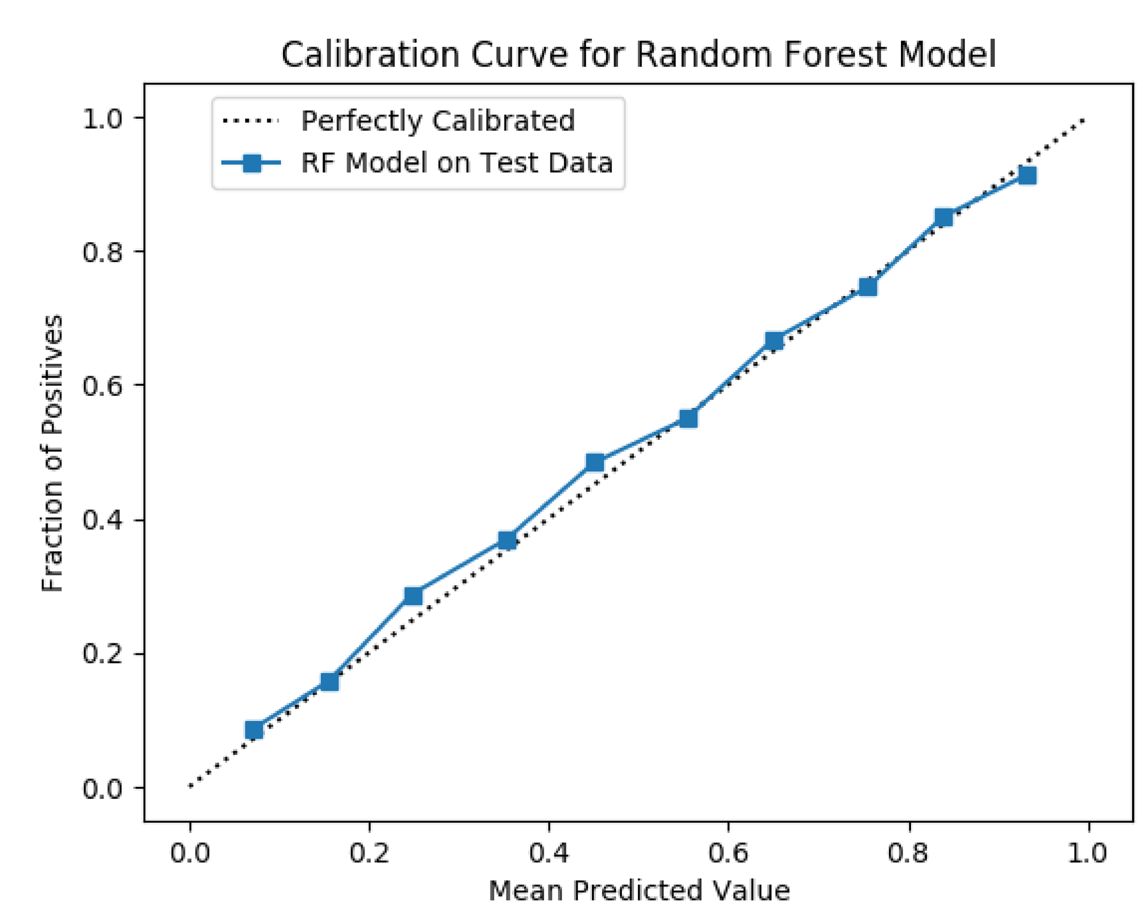
** Training the polynomial kernel SVM was too slow to cross validate*

SVM Models

- Non-Linear Kernels
  - Slow to train, difficult to iterate hyperparameters, and prone to overfitting
- Linear Kernel
  - Faster fitting, model matches data complexity, however it is difficult to obtain probability estimtates from model

Ultimately we chose to use the **Random Forest** model because:

- CV accuracy comparable to the best of any other model (better than NN)
- Fast to train (logistic regression was too slow)
- Outputs well-calibrated probability (unlike SVM)



Betting Strategy

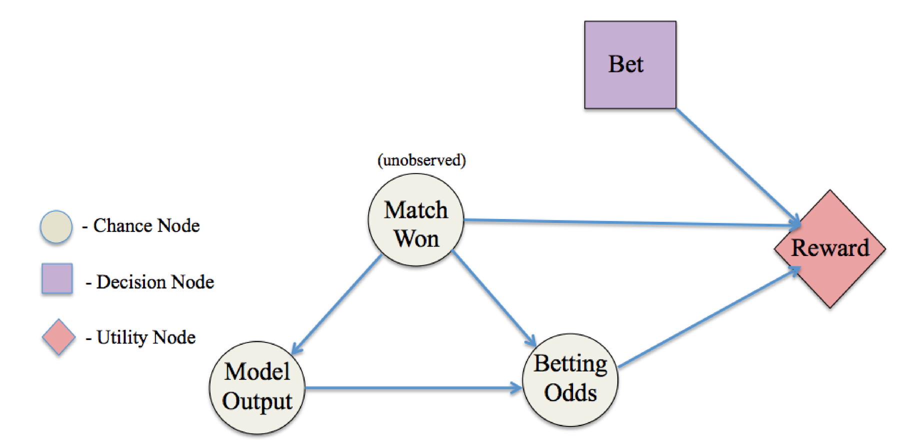


Figure 1: Decision network for single-shot betting strategy.

Model for Choosing the Bet

$$b^* = \underset{b\in\{0,+1,-1\}}{\arg\max}\; E[U(win, odds, b)]$$

The betting strategy is modeled as a single-shot decision problem, where we choose the bet that maximizes the expected returns. Possible bets include betting on player 1 ($b = 1$), betting on player 2 ($b = -1$), or not betting ($b = 0$).

Results

The random forest model **accuracy on the test set was 69.6%**. Given the cross-validation accuracy was 69.7% and the training accuracy was 73.5%, this indicates good generalization (low variance).

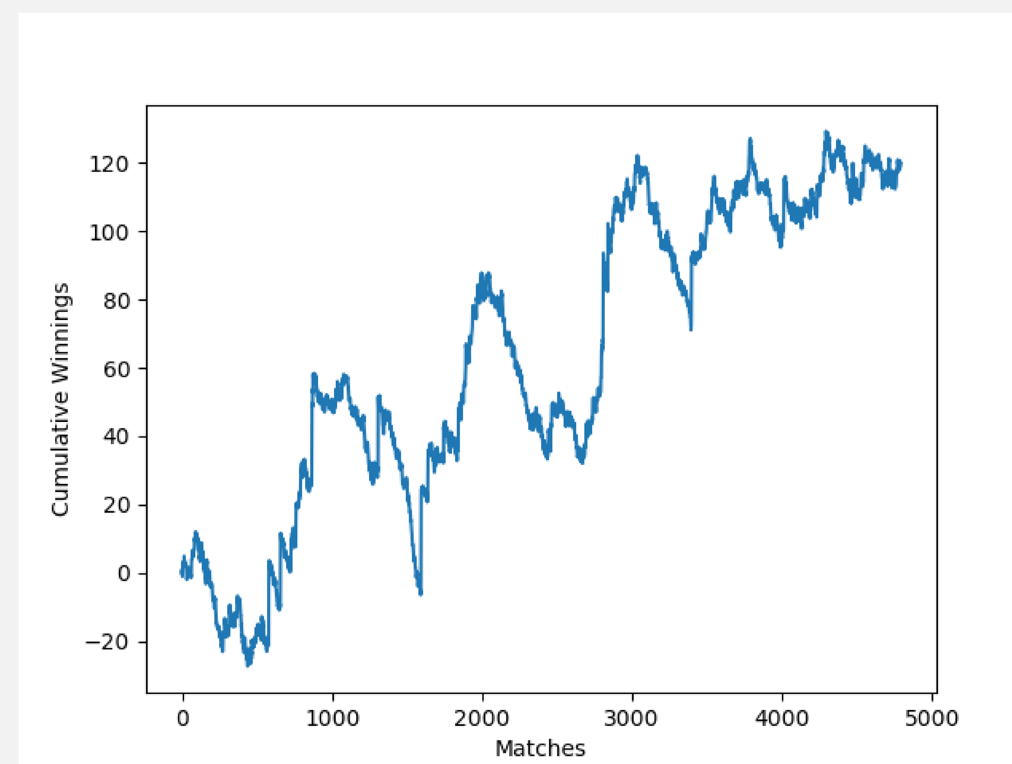


Figure 2: Cumulative winnings on the test set for simple betting strategy.

Error Analysis

Two of the important features were the ranking and the odds.

Predicted	Total #	Correct	Pct.
Lower Ranked Player	814	486	59.7
Higher Ranked Player	3976	2850	71.7

Predicted	Total #	Correct	Pct.
Favored Player	4669	3279	70.2
Disfavored Player	121	57	47.1

- Most of the time the model predicts the higher ranked player and the favored player to win.
- Model is less than 50% accurate when predicting the disfavored player: area for improvement of the model, but betting strategy still does well because of model calibration
- Currently, model has little insight into why/when a lower ranked or disfavored player wins (richer features, more data could help)

Further Work

- More sophisticated betting strategies
- Trying additional models
  - Some of our ideas required more computational resources than we had
  - Deeper neural networks, tune hyper-parameters
  - Including polynomial order features
- Richer data such as on injuries, expert predictions, weather conditions, anything else that could affect match outcome.
- Further research into SVM models
  - Speeding up training on non-linear kernels
  - Best strategies to yield probabilties from SVM model predictions
- Understanding why our betting strategy is so streaky... we have longs periods of winning and losing