Game, Set, Stat: A Comparison of Model Performance Using a Variety of Tennis Metrics

Brady Anderson

January 31, 2020

Introduction

Ever-increasing amounts of data have provided the opportunity to model a plethora of phenomena in practical and meaningful ways, offering new insights into fields of all domains. Utilizing past data to build predictive models is one of the most sought-after applications in statistical learning, particularly in the field of sports analytics. Accurately creating, measuring, and analyzing key metrics that are crucial in the final results can provide competitive advantages to teams and players alike who extract insights about what attributes are most impactful to improving (and determining) future game performances.

While not the forerunner, tennis has adopted widespread use of predictive analytics at all levels of the game. From increased match performance analytics provided in real-time by IBM during matches to the relatively new Universal Tennis Rating player ranking system, tennis continues to innovate in using data to drive both player improvement and sustained growth of the game.

As with all sports, research into collecting better data and uncovering what key variables drive outcomes continues to be explored. This creative component project looks to build predictive models focused on what factors drive winning in professional men's tennis through the usage and creation of more modern tennis metrics.

Goals

Through application of several statistical learning methods, there are four primary goals of this research:

1. Detect which variables are most influential in driving wins, and the strength of each relationship
2. Determine how accurate each of the models are in predicting wins in professional men's tennis
3. Measure which model performs best in predicting professional men's tennis outcomes
4. Ultimately, acquire a greater understanding of the theory behind several foundational predictive statistical learning algorithms while simultaneously building a skillset in execution and analysis of their results using R.

Dataset

There are two main data sources that were used in this project. The primary data source in which most of the explanatory variables were extracted/created is taken from a repository developed by Jeff Sackmann - the Match Charting Project. It consists of user-submitted point-by-point data for professional tennis matches with several other additional variables. Because of the self-selection nature of the dataset leading to mostly 'important' matches with available statistics (those in the later rounds and between the best players), the dataset is not a representative nor a random sample of the population of all men's professional tennis matches. Due to the time-consuming process in charting these detailed statistics for a match by hand, not all matches for every player are documented and thus available for detailed statistics. This limitation does reduce the scope of inference of the study, and will be further discussed later in the analysis.

The second source consists of match results, further summary statistics, and betting lines, gathered from http://www.tennis-data.co.uk/alldata.php. Several variables were used in training and testing models from this source, in which the comparison of model performances is done using betting lines. Due to its greater availability, betting odds taken from Bet365 were used.[1]

Variables

Several match characteristics were created and analyzed to determine which most heavily influence outcomes, as detailed in Table 1 below. In line with standard sport prediction practice, the *differences* between statistics were used for all numeric variables, as exhibited in Sipko and Knottenbelt (2015) as well as Lopez and Mattews (2015).

Variable	Description	Type
Ace.DF.ratio	Ratio of Aces to Double Faults	Numeric
BP.Save.Pct	Break Points Saved (%)	Numeric
Net.Wpct	Points Won at Net (%)	Numeric
Overall.Wpct	Points Won (%)	Numeric
Rank.Pts.Diff	ATP Ranking Points	Integer
Serve1.Pct	1st Serves In (%)	Numeric
Serve1.Wpct	1st Serve Points Won (%)	Numeric
Serve2.Wpct	2nd Serve Points Won (%)	Numeric
Serve.Plus1.Pct	Serves Followed by a Forehand Points Won (%)	Numeric
Surface.Clay	Match Played on Clay Court	Indicator 0-1
Surface.Grass	Match Played on Grass Court	Indicator 0-1
Win.Err.Ratio	Ratio of Winners to Errors	Numeric
Zero.to.Four.Wpct	Points Won between 0-4 Shots (%)	Numeric

Table 1: Summary of Variables

Of the features in Table 1 above, several are commonly used statistics for tennis prediction. These include 1st and 2nd serve win percentages (`Serve1.Wpct`, `Serve2.Wpct`), 1st serve percentage (`Serve1.Pct`), overall points won percentage (`Overall.Wpct`), the Association of Tennis Professionals (ATP) rankings (`Rank.Pts.Diff`), and surface indicators (`Surface.Clay`,`Surface.Grass`). However, I have created several addtional variables that I believe may add explanatory power to modeling match outcomes. The first of these include two 'ratio' variables: `Ace.DF.ratio`, and `Win.Err.Ratio`.[2] Commentators and players alike often comment on the importance of these two factors, suggesting their potential importance in winning a tennis match. Additionally, I added both `BP.Save.Pct` and `Net.Wpct`, to determine if 'clutchness' in preventing breaks of serve as well as effectiveness at net are highly predictive of success.[3]

Finally, the game of men's professional tennis has moved drastically in the direction of short points dominated by the forehand side. Nearly 70% of professional men's tennis points are between zero and four shots, indicating how crucial the first few shots are to success at the top level. Furthermore, in the past 20 years, the number one player in the world has won roughly 55% of his total points played during that season.[4] This slight margin results in the best player in the world winning 90% or more of their matches in any given season

[1] https://www.bet365.com/

[2] In cases in which there were zero double faults or errors the ratio was set to the maximum value of that variable from a player in the dataset

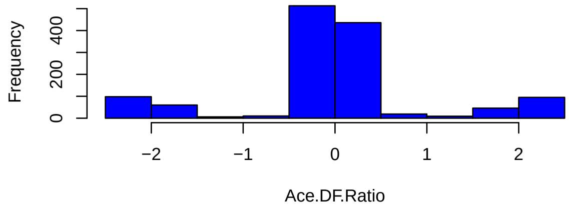
[3] In cases with zero break points faced or net points played the variable was set to be 100%

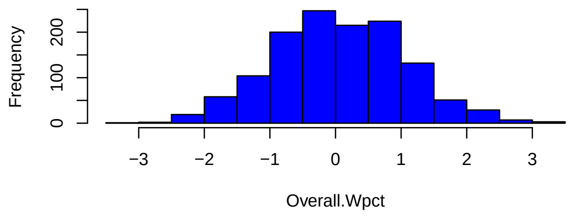
[4] O'Shannessy, 2017

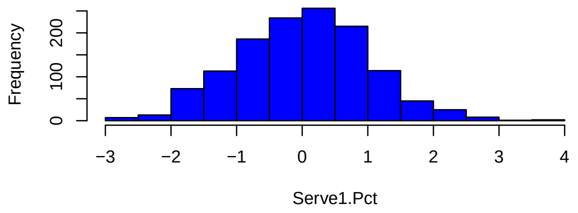
in the past 20 years. With so few points deciding the winner and the majority of these points being short (zero to four shots), I created the `Zero.to.Four.Wpct` and `Serve.Plus1.Pct` variables to try and capture the impact these may drive in today's game.

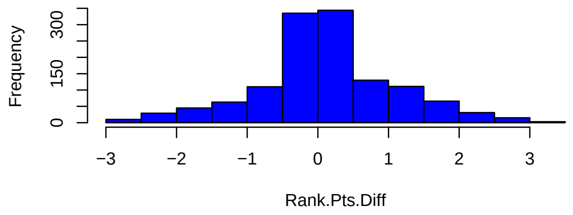
Standardization

The histograms below illustrate the distribution of some of the variables used in predicting match performance. As shown, nearly all are centered at zero and have a symmetrical shape as we would expect due them being the differences between two players performances. Therefore all variables, with the exception of the surface dummy variables, were standardized by dividing by just dividing each value by the variable's respective standard deviations, i.e. $\boldsymbol{X}_{standardized} = \frac{\boldsymbol{X}}{s}$ where $\boldsymbol{X}$ is a variable and s is the sample standard deviation of that variable. This procedure was done because of its usefulness in determining the relative importance of each variable in models in which coefficient weights are available.









Division

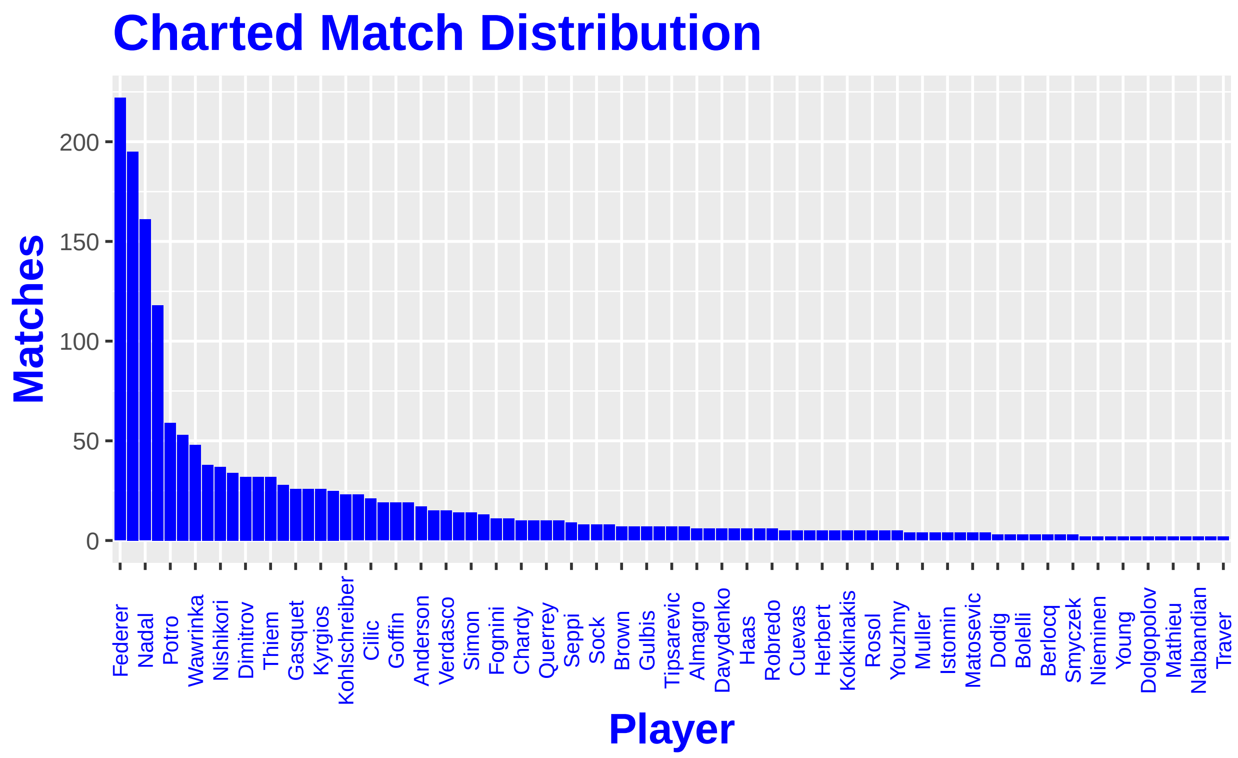
Because predictions are made using past data sequentially, the dataset was partitioned into three sets following a standard "Train-Validate-Test" procedure as follows:

- **Training Set (2009-2015)** - Used to build optimal training model for select methods. The training models use predictor variables that are based on using known results of each variable following each match and thus are the 'true' results to help determine which are most influential and predictive of match results. **565 matches**
- **Validation Set (2016-2017)** - Used for select models to tune hyperparameters that are not optimized via the training set data. For example, in Ridge and Lasso, we test a variety of λ values to determine which yields the lowest test error rate on the validation set. Variables are now averaged from the training set data for each player prior to use in the validation set. **129 matches**
- **Testing Set (2018-2019)** - Once the optimal model has been determined, the training and the validation set data are combined to increase the model training dataset size, with appropriate hyperparameters as determined from the validation process. Test varaiables are again averages of the variables for each player over the past data in the training dataset. **158 matches**

Overall, the data follow a 7-2-2 year split, which was done for continuity purposes as well as to ensure there were enough training data matches to have multiple measures for basing a player's performance to predict on the testing data. For training, validating, and testing individual player outcomes from these matches, only matches in which both players had **at least two** matches prior were used.

As discussed previously, all ATP matches were not available for the analysis, and the Match Charting Project dataset is a crowdsourced dataset that has been developed by several contributors. Therefore, the matches that are available tend to be from the higher ranked players and from the more important matches, notably from the four Grand Slam tournaments (Australian Open, French Open, Wimbledon, and the US Open) as well as in the later rounds of tournaments (Quarterfinals, Semifinals and Finals).

The diagram below highlights which players are most prominant in the final dataset used in the analysis.[5] Of note is the exponential shape of this dsitribution, as the recent world's best and most beloved ATP players (Federer, Djokovic, Nadal) are massively over-represented compared to their lower-ranked counterparts. Since a majority of all the matches played by these elite players are charted, there is much more confidence in the predictions made for their matches. For lower-ranked players, there may be bias in the sense that they are more likely to be included in the dataset if they are competing against one of these top players. This may systematically underestimate their true performance in the key variables measured when averaging them using the past available data. This concern does reduce generalizability of the models predictions across all players, however there is more confidence in predictions for those players with more matches. This limitation will be further considered in the conclusion portion of this analysis when making final inferences of the robustness of competing models.

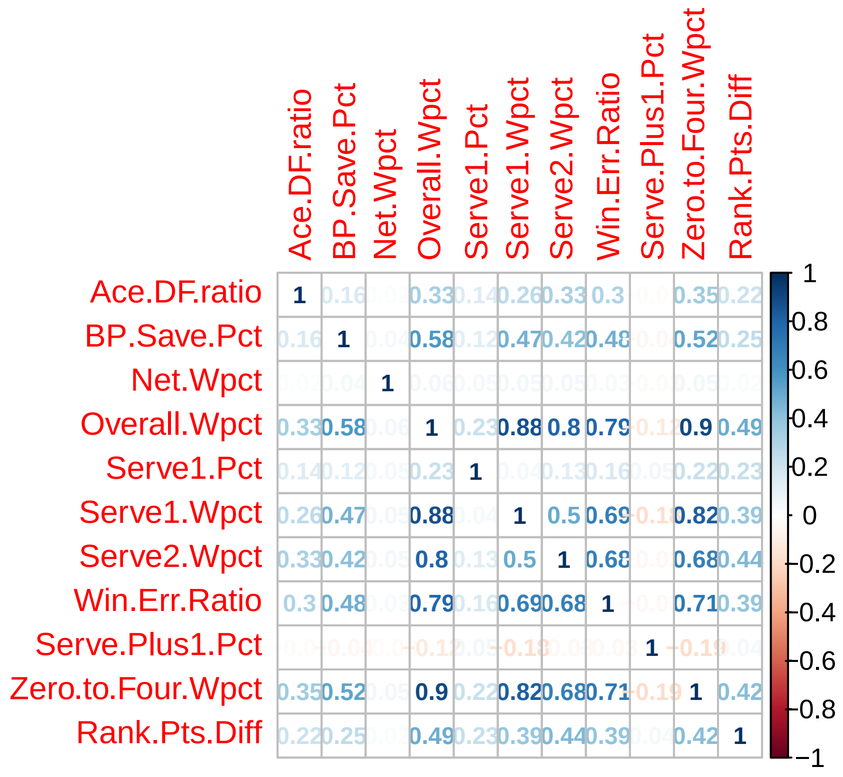


Diagnostics

Examining the correlation between our predictors, we found some evidence in the figure below of multicollinearity which may be problematic in our modeling procedures. This is particularly of note with the `Overall.Wpct` variable and several of the other variables for win percentage, such as `Serve1.Wpct` and

[5]For readability, the bar chart is ordered from high to low with the x-axis only showing the label for every other player

`Serve2.Wpct` as we would expect. However, most of the correlations between predictors are well below 0.9 and multicollinearity in the variables appears to not be too drastically violated. Nevertheless, all models were optimized based on whichever combination of predictors and parameters minimize the test error rate, as discussed below.



Model Optimization

There were two main objectives of this tennis predictive modeling procedure:

1. Minimize the incorrect number of classifications, or the **Test Error Rate**
2. Maximize the Return on Investment, or the **ROI**

Both are complex optimization procedures that do not necessarily align with one another depending on one's betting strategy and the model deployed. This study first sought to determine which model was most *accurate* in classifying match outcomes using a training, validation, and testing split as discussed previously. Subsequently, determining which of the 'best' models of each method then provided the greatest *ROI* using a few select strategies was tested to determine which was most profitable.

Methods

Following is a run-through of the models developed, generally beginning with those that are more parametric and ending with those that are less parametric. For all methods, a cutoff of $q = 0.5$ is used, where model predictions in which $p(\boldsymbol{X}) > 0.5$ are classified as a 1 or 'Win' for the player and a 0 or 'Loss' otherwise. This is very much in line with standard procedure and our testing dataset, as there are 76 cases out of the 158 that are classified as 1's or 'Wins' ($\approx 48.1\%$)

Logistic Regression

Logistic regression is a *generalized* linear model that seeks to assign a set of covariates ($\boldsymbol{X}$) to one of several disjoint classes. Let $Y = 0$ represent a 'Loss' and $Y = 1$ represent a 'Win' for a given player. Furthermore, let $p(\boldsymbol{X}) = P(Y = 1|\boldsymbol{X})$ be the probability that a player wins given $\mathbf{X}$, the set of covariates described in the Dataset discussion. To ensure that our model's range of predictions for Y stay within 0 and 1, we let $p(\boldsymbol{X}) = \frac{e^{\boldsymbol{\beta X}}}{1+e^{\boldsymbol{\beta X}}}$, where $\boldsymbol{\beta} = (\beta_0, \beta_1 ...)^T$ is the vector set of coefficient parameters for each covariate. After manipulation, one yields the logit link function used for the regression as:

$$log\left(\frac{p(\boldsymbol{X})}{1 - p(\boldsymbol{X})}\right) = \boldsymbol{X\beta}$$

The above regression model is now linear in $\boldsymbol{X}$, whose $\boldsymbol{\beta}$ parameters are estimated using maximum likelihood, also found by minimizing the negative log likelihood, which is formulated as:

$$L = -log\left(\prod_{i:Y_i=1} p(X_i) \prod_{j:Y_j=0} (1 - p(X_j))\right)$$

where $p(X_i)$ will be closer to 1 for the set of $i : Y_i = 1$ and $p(X_j)$ closer to 0 for the set of $j : Y_j = 0$ if the model fits well.

Because logistic regression is a generalized linear model, it does not have quite as many assumptions as a typical linear model, specifically in that it does not require variables to be linearly related to the response nor does it require homoscedasticity and normality of the error terms. However, logistic regression does assume no high multicollinearity between the predictors, a linear relationship between the predictor variables and the logit response, and that observations are independent. The first of these assumptions is certainly violated as shown in our correlation matrix previously. The latter two are less clearly violated, and for the purposes of this analysis, will not be further explored in depth.

Full Model

Running a logistic regression model containing all of the variables resulted in three statistically significant variables at the $\alpha = 0.1$ level, and just one at the $\alpha = 0.05$ level. The most impactful variable in determining win percentage was the `Overall.Wpct` variable unsurprisingly, with an extremely small p-value of < 0.001. `Serve1.Wpct` and `Rank.Pts.Diff` were the other two that were statistically significant at the 0.1 level. However, as mentioned previously, many of these variables are correlated and the assumptions of logistic regression are not completely met, discrediting their validity.

The model correctly classifies 87 of the 158 matches in the testing dataset.

Regularization and Dimension Reduction: Ridge, Lasso, and Principal Components Regression (PCR)

Ridge Regression

Ridge regression is a *shrinkage method* that we perform with all of our predictor variables as detailed earlier. Ridge regression seeks to solve the Lagrange optimization problem as follows:

$$\beta_\lambda^R = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} L + \lambda \sum_{j=1}^{p} \beta_j^2$$

where λ is the shrinkage penalizing term for our β estimators obtained via the logistic regression, and $\hat{y}$ is our predicted class for a given set of p variables in $\boldsymbol{X}$. Thus, the Ridge optimization problem above trades off bias and variance in the typical machine learning lingo, as we seek to select a model with predictor variables that fits the outcome well (has low bias) but does not dramatically change when seeing new data with large parameter estimates (high variance). Fitting a model through intentional introudction of some bias but a larger reduction in variance can often yield more accurate predictions, and a lower Test Error Rate.

The optimal value of our shrinkage parameter $\lambda_R = 0.5$, which correctly classifies 86 of the 158 test matches.

Lasso Regression

Lasso regression is extremely similar to Ridge Regression. Like Ridge, Lasso is a *shrinkage method* that seeks to solve a Lagrange optimization similar to that of Ridge, detailed in the following problem:

$$\beta_\lambda^L = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} L + \lambda \sum_{j=1}^{p} |\beta_j|$$

where λ is the shrinkage penalizing term for our β estimators obtained via logistic regression. As with Ridge, the equation given above trades off bias and variance, where it allows a small amount of bias into the model in exchange for a reduction in the model's variance. This hopefully provides an overall reduction in the Test Error Rate. Unlike Ridge, Lasso can often act as a variable selection technique, shrinking the more inconsequential variables all the way to 0 for their parameter estimates.

The optimal value of our shrinkage parameter $\lambda = 0.25$, which correctly classifies 87 of the 158 test matches.

PCR

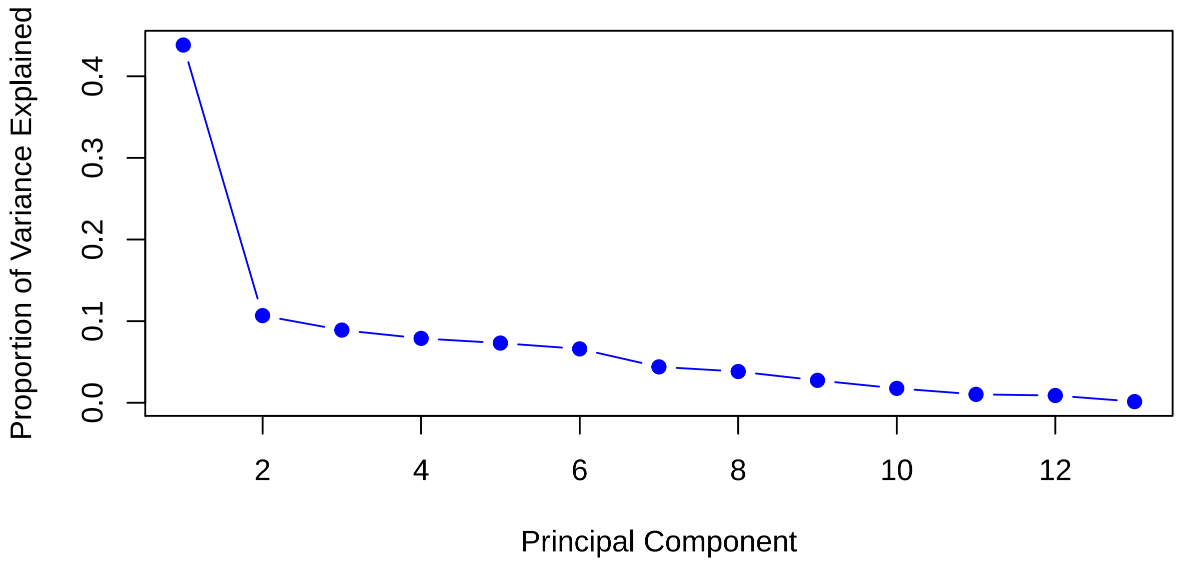
PCR is a dimension reduction method that seeks to transform the predictor variables $X_1, X_2, ..., X_p$ to a new set of predictor variables $Z_1, Z_2, ..., Z_M$ such that $M < p$. The method represents each Z_m as a linear combination of our p predictors in the model in the following manner:

$$Z_m = \sum_{j=1}^{p} \phi_{jm} X_j$$

The new predictor variables $Z_1, ..., Z_M$ are then used to fit a logistic regression model to our response Y using maximum likelihood estimation. Crucially, PCR assumes that the response variable Y does not play *any role* in determining our new set of predictor variables. Therefore, we first perform Principal Component Analysis (PCA) on our original data and get corresponding Principal Component (PC) loading vectors $\phi_j, j \in \{1, ..., 13\}$ which successively explain less and less of the total variance in our predictor variable data. Secondly we assess the performance of various PC sizes on the testing dataset to determine the appropriate number of PC's to use in the final full training dataset. PCA iteratively chooses each ϕ_j by solving the following optimization problem:

$$\begin{aligned} \phi_1 &= \max_{\phi} Var(\phi^T X) \\ \text{s.t. } \phi^T \phi &= 1 \\ \phi_2 &= \max_{\phi} Var(\phi^T X) \\ \text{s.t. } \phi^T \phi &= 1, \phi_1 * \phi_2 = 0 \end{aligned} \tag{1}$$

Where each successive ϕ_j PC vector orthogonal to the rest. Due to the relatively highly correlated tennis data, the goal in PCR is to reduce the dimension of the data such that the decrease in variance in our estimation outweights the increase in bias from information compression.



As shown in the scree plot above, there appears to be a dramatic dropoff after just the first principal component, which explains nearly 50% of the variation of our variables in our $\boldsymbol{X}$ matrix.

Not unexpectedly, PCR with just one PC variable produces the lowest Test Error Rate at approximately 43.7%, correctly classifying 89 of the 158 matches in the testing dataset.

Optimal Model

After fitting logistic regression using the full set of predictor variables and testing out various regularization and dimension reduction techniques, we find that **PCR** yields the lowest testing error rate, followed by the full logistic regression and the Ridge regression counterpart, and lastly Lasso regression.

Linear Discriminant Analysis (LDA)

LDA uses Bayes' theorem in order to indirectly estimate probabilities, which states that:

$$P(Y = k|\boldsymbol{X = x}) = \frac{\pi_k f_k(\boldsymbol{x})}{\sum_{i=1}^{K} \pi_i f_i(\boldsymbol{x})}$$

where $\pi_k = P(Y = k), k = \{0, 1\}$ where k represents the class of Y, $f_k(\boldsymbol{x}) = P(\boldsymbol{X = x})|Y = k)$ and $\sum_{i=1}^{K} \pi_i f_i(\boldsymbol{x}) = P(\boldsymbol{X} = \boldsymbol{x})$. LDA estimates the left-hand side of the equation using estimates from the data itself. In multivariate LDA, we typically assume that $f_k(\boldsymbol{x}) \sim N(\boldsymbol{\mu_k}, \Sigma)$ where each class k has its own mean vector and all share the same covariance matrix Σ. After some algebra we obtain the *discriminant function*:

$$\delta_k(\boldsymbol{x}) = \boldsymbol{x}^T \Sigma^{-1} \boldsymbol{\mu_k} - \frac{1}{2} \boldsymbol{\mu_k}^T \Sigma^{-1} \boldsymbol{\mu_k} + log(\pi_k)$$

where we assign an observation $\boldsymbol{X} = \boldsymbol{x}$ to whichever class yields the largest value.

We first perform LDA on the full set of predictors, and then perform LDA on the PCA-transformed first predictor which explained nearly half of the variation in our data $\boldsymbol{X}$ to compare its performance to logistic regression.

Optimal Model

Just as with logistic regression, performing LDA with the first PC vector yields the lowest test error rate, correctly classifying 88 of the 158 matches in the testing dataset. LDA performed on the full set of predictors correctly classified 85 of the 158 matches.

Generalized Additive Models (GAMs)

GAMs provide greater flexibility in fitting our qualitative win-loss response by allowing *non-linear* f_j fits to each X_j variable. Therefore, GAMs tend to reduce the bias of predictions while increasing the variance when compared to standard logistic regression. While there is a wide class of functions that can be used for the f_j, we use cubic smoothing splines which solve the following optimization problem:

$$\hat{f}_\lambda = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \, L + \lambda \int_{\mathcal{X}} \{f''\{x\}\}^2 dx$$

where $\mathcal{F}$ is a class of basis functions. The λ parameter controls the amount of shrinkage of our cubic smoothing spline functions for each variable, reducing the variance and 'wigglyness' of the functions to fit the model which lowers the effective degrees of freedom in the model as λ increases.

As with logistic regression, these non-linear functions are assumed to be additive in their impact on the log-odds, represented as:

$$log\left(\frac{p(\boldsymbol{X})}{1-p(\boldsymbol{X})}\right) = \beta_0 + f_1(X_1) + ... + f_p(X_p)$$

As done previously, we first build a GAM using the full model with all standardized variables as with logistic regression. Then we build a GAM using solely the first PC-transformed variable, represented as:

$$log\left(\frac{p(\boldsymbol{X})}{1-p(\boldsymbol{X})}\right) = f_{PC1}(Z_1)$$

and compare their performances on the Test Error Rates.

Optimal Model

The full set of predictor variables corrrectly classifies 86 of the 158 matches in the testing dataset using a full GAM with cubic smoothing splines. Once again, using the first PC-transformed variable as the lone predictor, we find a slightly improved model in reducing prediction error. This variant of the GAM correcty classifies 88 of the 158 matches in the testing set.

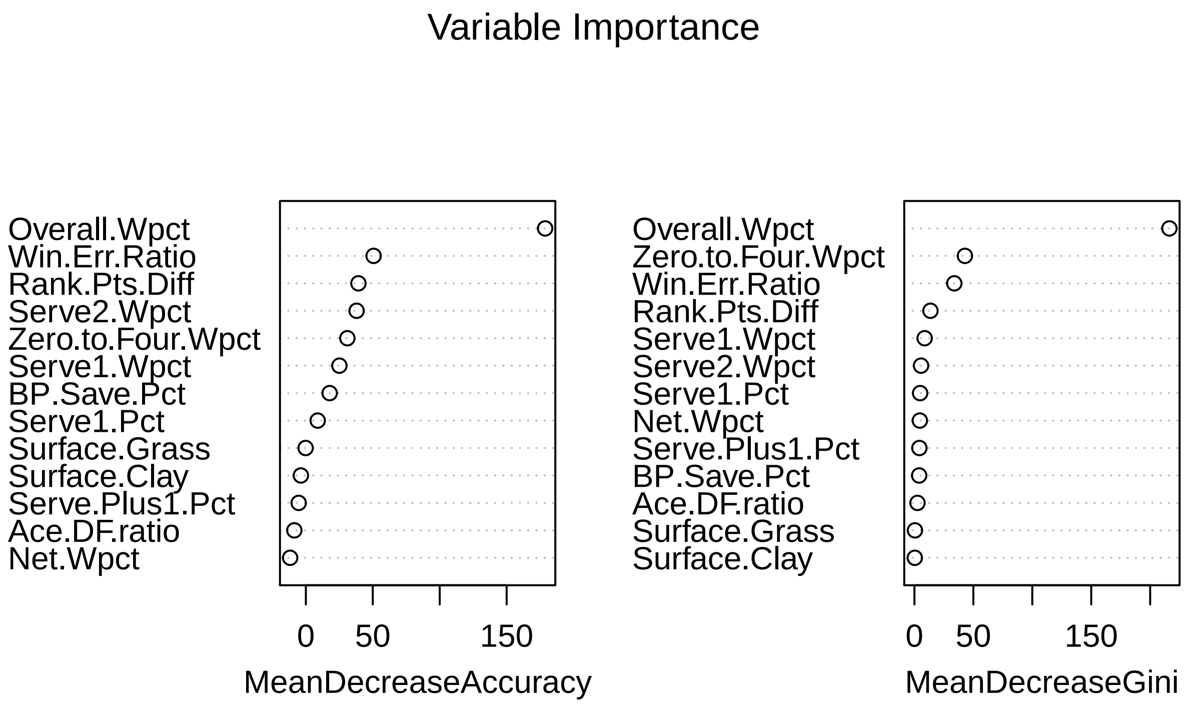
Random Forest

The random forest algorithm for classification problems is a tree-based method that seeks to assign observations to classes by splitting the predictor variable space ($\boldsymbol{X}$) into distinct regions. Random forests are composed of decision trees, which are generated from recursive binary splitting of variables chosen by minimizing a diagnostic statistic, typically the Test Error Rate, $E = 1 - \max_k \hat{p}_{mk}$, where $\hat{p}_{mk}$ is the proportion of observations in the m^{th} region of $\boldsymbol{X}$ that belong to class k. For each decision tree, a new observation is assigned the class that has the majority vote of the training observations in that same region of $\boldsymbol{X}$. In practice, the Gini index is more commonly used to grow decision trees, defined by

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

where scores determine node or leaf 'purity' of a final class assessment for a decision tree. Lower Gini index scores indicate similarity in class assignment for all observation in that specific leaf. If the Gini score or the classification error rate are decreased enough (by some pre-determined threshold) at a corresponding split, the decision tree will add that additional split in its construction.

Random forests use the averages (bagging) of the classifications assigned by many decision trees to help reduce variance in the prediction estimates. For our binary class problem of win/loss, a new obsevation X^* is given the probability of winning based on $p(X^*) = \frac{1}{B}\sum_{b=1}^{B} I(Y^* = 1)$, where B is the number of bootstrap datasets constructed, each building a decision tree. Furthermore, random forests use a random *subset* of variables at each split in a decision tree to help decorrelate the trees, reducing the variance in predictions with the goal of improved accuracy.



Optimal Model

We find that the random forest model using `mtry` = 9 or a 9 random variable split leads to the lowest Test Error Rate, correctly classifying 87 out of the 158 matches. Furthermore, the variable importance plot above shows us that, when using out-of-bag (OOB) observations in determining prediction accuracy for each tree with and without certain variables, the Mean Decrease in Accuracy without the `Overall.Wpct` is by far the most significant variable, followed by the `Win.Error.pct` ratio statistic. The `Zero.to.Four.pct` statistic also makes the top five in most influential in reducing the error rates for the random forest algorithm model, suggesting that two of the top five variables in prediction accuracy were from the newly created ones for this dataset.

KNN Classification

KNN (K-Nearest Neighbors) is completely non-parametric in that it simply assigns each new observation $\boldsymbol{x_0}$ to the class with the most frequent outcome among its K nearest neighbors, where nearest is defined in terms of Euclidean distance $\sqrt{\sum_{j=1}^{p}(x_{0j} - x_{ij})^2}$, where p is the number of variables in $\boldsymbol{x_0}$. More formally, KNN estimates the conditional probability for class j as the fraction of points in the neighborhood of $\boldsymbol{x_0}$ ($\mathcal{N}_\prime$) whose value are in the same class j:

$$P(\widehat{Y = j|\boldsymbol{X}} = \boldsymbol{x_0}) = \frac{1}{K}\sum_{i \in \mathcal{N}_\prime} I(y_i = j)$$

The class j with the highest estimated probability is the class $\boldsymbol{x_0}$ is assigned.

On the full dataset, setting $K = 60$ seems to provide the best trade-off in bias and variance in regards to yielding the lowest test error rate in its predictions, as it correctly classifies 90 of the 158 test matches.

When using just the first PC vector as the sole predictor variable, the KNN model which minimizes the Test Error Rate occurs when $K = 10$, correctly classifying 89 of the 158 test matches.

Optimal Model

Interestingly in the KNN case unlike the other models, we see better performance using the entire set of predictor variables rather than the most explanatory PC-transformed variable. Thus, KNN does not appear to be suffering from the 'curse of dimensionality' problem on our dataset, as the 13 predictor variables used offer a slightly lower Test Error Rate than when using the sole PC variable that is a linear combination of all the predictors and explained nearly half of the variance in the dataset.

Model Results Summary

As shown in Table 2, **KNN** performed the best in minimizing the Test Error Rate among all models tested, with an error rate of 43.0% in its best model with all the predictor variables. While all models achieved similar Test Error rates, we note that the Ridge and Lasso regularized models were worse than the logistic regression model with no regularization performed on its parameters. However, dimension reduction using PCA tended to decrease the error rate for all models in which it was applied.

Model	Type	Error Rate
Logistic	Full	44.9%
Logistic	Ridge Regularized	45.6
Logistic	Lasso Regularized	44.9%
Logistic	PCA Compressed	43.7%
LDA	Full	46.2%
LDA	PC Compressed	44.3%
GAM	Full	45.6%
GAM	PC Compressed	44.3%
Random Forest	Full	44.9%
KNN	Full	43.0%
KNN	PC Compressed	43.7%

Table 2: Test Error Rates

Evaluation: Betting Performance

Implied Probability

Our dataset gives odds from Bet365 in Decimal odds d_i for each player i of a given match, which means for every 1 unit wagered that is successful, the bettor gets d_i units in return. We convert these Decimal odds into an implied probability that the oddsmakers give a player of winning the match as $p_i^{implied} = \frac{1}{d_i}$. For example, if a player is given odds 2.00 to win a match, his implied probability is $p_i^{implied} = \frac{1}{2} = 0.5$, and if one bets \$1 and is succesful in the wager, they will get \$2 in return (a net profit of \$1 in this case).

Strategies

There are a wide variety of betting strategies that could be employed for any given model. In this analysis, we consider three different strategies for two classes of matches, resulting in a total of six different strategies. Let r_i = the amount (in \$) risked on player i, p_i^{model} = the predicted probability of player i winning under the given model of consideration, and b_i = the net odds, or the multiplier of one's r_i denoting the amount of profit for a successful bet on player i, given as $d_i - 1$ for decimal odds.

The two classes are:

(1) Both competing players have $\mathbf{5^+}$ previous matches in the training dataset, and

(2) Both competing players do not have $\mathbf{5^+}$ previous matches in the training dataset.

For each class, the following three strategies will be evaulated:

1. **Betting on the predicted winner**

Under this strategy, we place a fixed bet c on the player i that the model predicts to win, i.e.

$$r_i = \begin{cases} c, & \text{if } p_i^{model} > 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

2. **Betting the predicted winner when at better odds**

Under this strategy, we place a fixed bet c on a player i when the predicted probability of winning under the model is greater than the implied probability from the betting odds by *at least* 5% (to overcome the 'juice'). Therefore, this strategy only places bets when the model perceives an 'edge' to be gained that warrants taking the risk in placing the wager.

$$r_i = \begin{cases} c, & \text{if } p_i^{model} > p_i^{implied} + 0.05 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

3. **Betting the predicted winner using the Kelly criterion**

Under this final strategy, we use the Kelly criterion which is widely regarded as an optimal betting/investing formulation that calculates how much one should risk on player i that is based on how large the perceived edge is from the model over the implied probabilities from the betting odds. We place a maximum bet size of c on the winner if there is a perceived edge such that:

$$r_i = \begin{cases} c * \frac{p_i^{model} b_i - (1 - p_i^{model})}{b_i}, & \text{if } p_i^{model} > p_i^{implied} + 0.05 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

The above three strategies get progressively more complex and selective in determining one's wager amount r_i, with the final two not placing a bet on every match. For the Kelly criterion, often the size c of the maximum bet evolves dynamically with one's bankroll, however in this analysis we keep it fixed and equal with the other strategies for comparative ROI purposes.

Results

The table below highlights the ROI on each model under each respective class and betting strategy. ROI is calculated as ROI $= \frac{\text{Return - Investment}}{\text{Investment}} * 100\%$

The three best and worst models by overall ROI across each class and strategy are denoted in red and green highlighting in Table 3 below. Despite having the lowest test error rate, KNN does not appear to be the best models in providing the highest ROI across the three betting strategies and two different classes. Additionally, it appears that the "Edge" strategy (2) yielded the best results almost completely across all model and class types. Nearly every model resulted in a positive ROI under strategy (2) and Class (2). Intriguingly the Random Forest and Logistic Regression models appear the best when examined on the whole, and the Logistic Regression model under the "Edge" strategy for Class (2) returned the highest ROI at 14%. Astonishingly, the "Kelly" strategy performed much worse than expected across nearly all models. Finally, there was a nearly unanimous improvement across all models and strategies when only placing bets on players with at least **5+** matches in the training dataset, i.e. Class (2). While not completely surprising, the consistency and noticeable improvement regardless of model given the fairly moderately sized test dataset is of particular note.

Table 3: Betting Returns Summary

Model	Class	Strategy	ROI
Logistic	(1)	All	-3.38
		Edge	5.28
		Kelly	-5.42
	(2)	All	-0.29
		Edge	14.00
		Kelly	4.55
Ridge	(1)	All	-9.08
		Edge	-9.51
		Kelly	-7.82
	(2)	All	-8.03
		Edge	-9.05
		Kelly	-5.00
Lasso	(1)	All	-11.50
		Edge	-8.12
		Kelly	-9.34
	(2)	All	-10.17
		Edge	-10.28
		Kelly	-11.27
Logistic PCA	(1)	All	-7.36
		Edge	1.35
		Kelly	-13.78
	(2)	All	-1.37
		Edge	7.92
		Kelly	-6.48
LDA Full	(1)	All	-7.51
		Edge	1.34
		Kelly	-7.82
	(2)	All	-5.77
		Edge	8.78
		Kelly	1.03
LDA PCA	(1)	All	-9.16
		Edge	-6.54
		Kelly	-16.29
	(2)	All	-3.76
		Edge	-1.88
		Kelly	-10.22

Model	Class	Strategy	ROI
GAM Full	(1)	All	-3.00
		Edge	4.01
		Kelly	-5.01
	(2)	All	0.22
		Edge	10.64
		Kelly	-0.67
GAM PCA	(1)	All	-9.13
		Edge	0.93
		Kelly	-13.38
	(2)	All	-5.04
		Edge	6.21
		Kelly	-9.01
Random Forest	(1)	All	-2.41
		Edge	4.21
		Kelly	-4.87
	(2)	All	-0.81
		Edge	10.26
		Kelly	0.21
KNN Full	(1)	All	-7.49
		Edge	-1.80
		Kelly	-7.44
	(2)	All	-7.83
		Edge	1.41
		Kelly	-1.40
KNN PCA	(1)	All	-12.50
		Edge	0.74
		Kelly	-0.81
	(2)	All	-10.39
		Edge	9.29
		Kelly	-4.89

Concluding Remarks

Professional men's tennis continues to evolve in its focus on analytics. Through the creation of some more modern tennis metrics, this project found that, among the data available, a player's past overall point winning percentage still tended to be more predictive than most of these variables when measuring their respective influence on match outcomes.

When looking at which model best predicted match outcomes, we found that the non-parametric model, KNN, produced the lowest Test Error Rate. The more assumption-heavy models, like Logistic Regression, tended to perform the worst in terms of minimizing this classification rate (although not drastically so). Interestingly, KNN was not the profitable model in ROI among those investigated. When looking for the best overall model in terms of both minimizing the classification test error rate while simultaneously maximizing ROI, the Random Forest method proved best in this analysis.

Limitations

As discussed in the introduction, there were a number of limitations in this analysis. For starters, the dataset did not include every single match for each player. Since many of the matches are from bigger tournaments and later rounds (and composed of the best players), this factor may bias a lower ranked player's abilities in that there is an underrepresentation of their true average performance across the many variables studied.

However, given that the test dataset is a similar ‘set’ of biased matches, this should not be as detrimental for prediction purposes as it would were the testing dataset entirely representative (thus incongruent) of professional men’s tennis matches.

Additionally, this analysis, due in part to the irregularity in the spacing of matches among the players, did not incorporate any time discounting factor in building models and variables in the dataset. Rather, each match, regardless of when, was weighted equally in divising a player’s performance for each statsitic. Placing more influence on recent performance with complete data using moving averages (MA models) or other methods may provide a more accurate representation of true player abilities and offer more predictive insights.

Lastly, the data from the repository is self-charted by a variety of contributors including myself. This may result in minor inconsistencies in shot classification, slightly altering the accuracy of certain variable statistics. However this effect is expected to be minimal, as errors should likely be minor or possibly even cancel each other out.

Further Study

Moving forward, acquiring access to a dataset with available statistics across every professional match from each year would be ideal to see how, if at all, the results change across different modeling procedures. This would also help mitigate the biased sample issue discussed above while also providing an opportunity to incorporate a time discounting factor to more accurately project player performance. Pivotally, complete and representative full match data for all professional players would expand the scope of inference regarding the results of various predictive models developed.

Finally, the same type of analysis could be applied to the Women’s Tennis Association (WTA) to compare both variable importance and predictive power between the men’s and women’s game.

References

1. James, G., Witten, D., Trevor, H., & Tibshirani, R. (2013). *An Introduction to Statistical Learning.* New York : Springer.
2. J. Kelly. "A new interpretation of information rate". *IRE Transactions on Information Theory*, 2(3):917–926, 1956.
3. Lopez, Michael J., and Gregory J. Matthews. "Building an NCAA men's basketball predictive model and quantifying its success." *Journal of Quantitative Analysis in Sports* 11.1 (2015): 5-12.
4. O'Shannessy, Craig. "The First 4 Shots." (2017).
5. Sackmann, Jeff. "Tennis Abstract: Match Charting Project Metadata." tennisabstract.com, (2018), www.tennisabstract.com/charting/meta.html.
6. Sipko, Michal, and William Knottenbelt. "Machine learning for the prediction of professional tennis matches." *MEng computing-final year project, Imperial College London* (2015).
7. Tennis Betting | Tennis Results | Tennis Odds. http://www.tennis-data.co.uk/alldata.php (2019).