Predicting the Outcome of ATP Tennis matches

Abstract

My goal is to predict the outcome of ATP Tennis matches. I'll be using the dataset of all ATP matches from 2000 to 2018 with roughly 45,000 matches and 23 variables. By using match and player statistics, the ability to predict match outcomes would be incredibly powerful for determining future match draws, sports betting opportunities, and tournament tactics for players

Design

This project originates from a Kaggle dataset and notebook "Beat the bookmaker with machine learning". The data is provided by Edouard Thom. Each row in the dataset represents one professional ATP match, details about the location and size of the tournament, and details on the match-the winner, the loser, and how many sets each player won or lost. The target variable is "upset", coded as 1 if a lower ranked player beats a higher ranked player (i.e. Player w/ Rank #48 beats Player w/ Rank #4). This was chosen for 2 reasons: First, the intuitive choice for the target variable is win or loss, but since that is a unique string (a player's name), I would have to create two rows for each match, one row for the winner and one row for the loser. Second, for specific sports betting use cases, it is most important to predict for upsets because those occurrences provide the best upside.

Data

The dataset contains roughly 45,000 matches and 23 variables. There are 35% upsets and 65% not upset matches. Some notable variables include:

- ATP = Tournament number (men bracket)
- Data = Date of match
- Series = Name of ATP tennis series (Grand Slam, Masters, International or International Gold)
- Court = Type of court (outdoors or indoors)
- Surface = Type of surface (clay, hard, carpet or grass)
- Round = Round of match
- Best of = Maximum number of sets playable in match
- Winner = Match winner
- Loser = Match loser
- elo_loser= The Elo Model ranking of the loser calculated *before* the match based on the playing history of the two players

- elo_winner= The Elo Model ranking of the winner calculated *before* the match based on the playing history of the two players

Feature Engineering

1. Creating the target variable “upset”
    a. Calculate Rank Delta= winner_rank - loser_rank
    b. 1 if If Rank Delta > 0 , else 0
2. Converting categorical features to binary dummy variables, and dropping one category to avoid multicollinearity
3. Creating elo_winner and elo_loser: The Elo Rating is a well known probability calculation that takes into account player ranking to determine a match outcome.In terms of interpretation, if a player has an Elo rating of 1,800 and his opponent has a rating of 2,000, the probability of the lower Elo rating player winning becomes is 24.1%. Learn more about it here

Modeling

Model selection: Logistic regression, decision tree, random forest, and XGBoost classifiers were used before settling on random forest as the model with strongest cross-validation performance. The logistic regression and decision tree models offered a high level of interpretability to understand the dataset and analyze any initial trends in the data, while random forest and XGBoost would most likely have the strongest performance. All models were evaluated using ROC, as well as a specific breakdown of recall and precision to see the tradeoff between the two metrics for each model

Data Split and Validation: The entire training dataset was split into 80/20 train vs. holdout, and all scores reported below were calculated with 4-fold cross validation on the training portion only. Predictions on the 20% holdout were limited to the very end, so this split was only used and scores seen just once.

Model Tuning: For all models except Logistic Regression, I used the GridSearchCV package to apply 4 fold cross validation to apply selection of hyperparameters specific to each type of model. Because all models are tree based, some parameters were consistent across all models i.e. max_depth, max_leaf_nodes, n_estimators. Below are the tested hyperparameters, the “winning” parameters based on cross validation, and the feature importance scores.

Logistic Regression

The logistic regression model was used 1) to establish a baseline ROC to hopefully outperform using the more complex tree based models 2) have interpretable coefficients of variables to get an initial understanding of the key variables. I scaled the variables using the sklearn StandardScaler package to allow me to use regularization to avoid overfitting. I set the solver='liblinear' because it's a relatively small datatset and penalty='l2' to implement L2 (Ridge) regularization

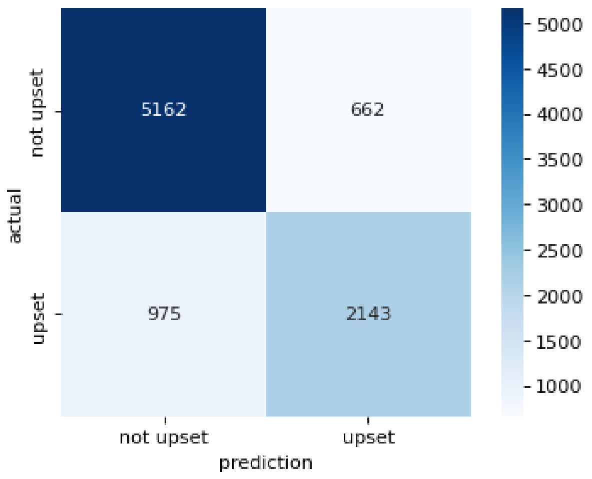
Variable	**Coefficients**
elo_loser	2.261
Tournament_Copa Telmex	0.058
Tournament_Generali Open	0.055
Tournament_U.S. Men's Clay Court Championships	0.053
Tournament_German Open Tennis Championships	0.051
Series_International	-0.065
Court_Outdoor	-0.068
Surface_Hard	-0.158
Surface_Clay	-0.184
elo_winner	-2.865

The elo_loser variable is most positively correlated with an upset. We can interpret this to mean the higher Elo score the loser has, the more likely an upset will occur. This is logical because higher ranked players have higher Elo scores, and the loser in an upset is the higher ranked player. The same explanation applies to why the elo_winner variable is most negatively correlated with an upset. Note that because the data has been scaled, we cannot interpret these coefficients in absolute terms, only relative to each other. For example, whether the match is played at Telmex, Generali Open, U.S. Men's Clay Court Championships, German Open Tennis Championships all have similar magnitude of impact on the match outcome. However, elo_winner and elo_loser but have much higher magnitude of impact on the match outcome compared any tournament related variables

Model Results

```
             precision    recall   f1-score   support
         0      0.84       0.89      0.86      5824
         1      0.76       0.69      0.72       3118
  accuracy                           0.82      8942
 macro avg       0.80      0.79      0.79      8942
weighted avg     0.81      0.82      0.81      8942
```

ROC: 0.787



Baseline Metrics for Tree Models

Once identifying a baseline model and ROC using the Logistic Regression, I wanted to tuned tree based models to optimize for model performance. I used a DecisionTreeClassifier, RandomForestClassifier, and XGBoostClassifier model from the sklearn package. First, I generated baselines for each model-all parameters set to default.

	Decision Tree	**Random Forest**	**XGB**
accuracy	0.769	0.810	0.822
precision	0.668	0.756	0.766
recall	0.671	0.671	0.702
ROC	0.746	0.778	0.794

Cross Validated Decision Tree Model

The DecisionTreeClassifier ran very quickly but produced a ROC that was only slightly higher than baseline (0.766). The only two importance features were elo_winner and elo_loser. The decision tree graphic shows exactly how the 2 variables were used to classify an upset vs. not an upset

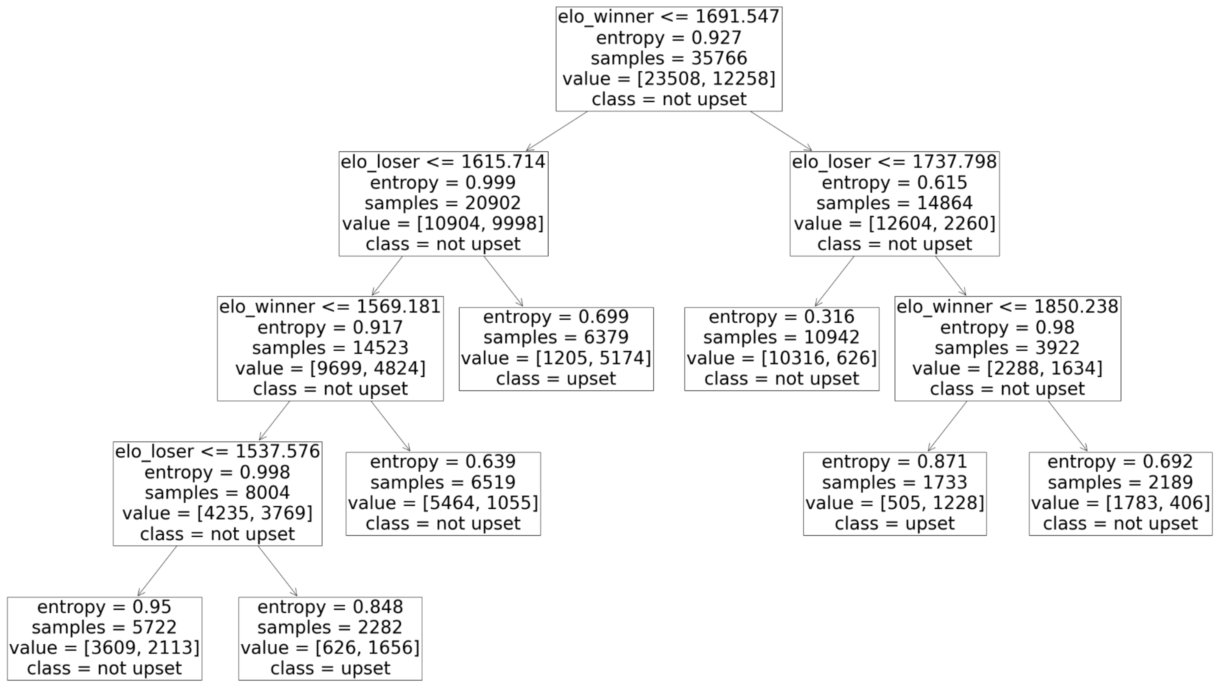
Tuned Hyperparameters:

- criterion='entropy'
- max_depth=50
- max_leaf_nodes=1000
- min_impurity_decrease=0.01
- random_state=65

Feature Importance Scores

Variable	**Feature Importance**
elo_winner	0.519
elo_loser	0.418

Decision Tree Breakdown



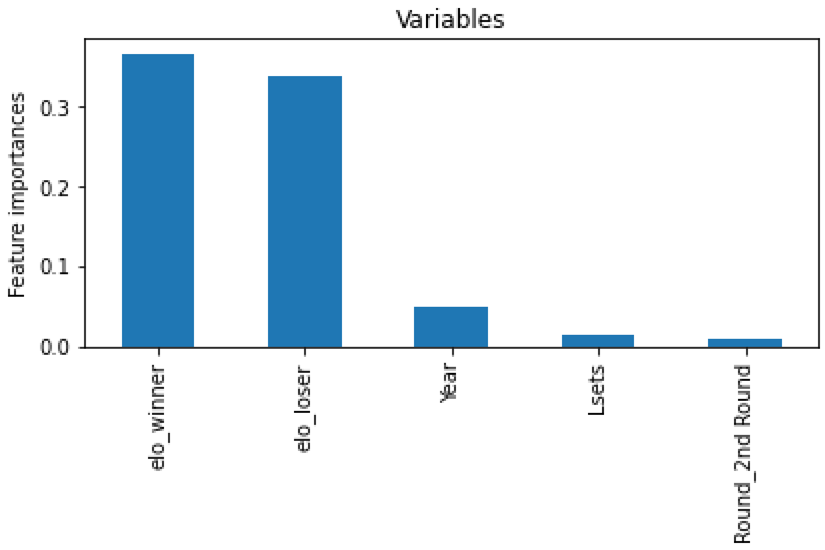
Cross Validated Random Forest Model

As expected, elo_winner and elo_loser were the most important features. However, compared to the Logistic Regression and DecisionTree model, the Random Forest Classifier found additional signals in year of the match, number of sets won by loser (Lsets) and whether the match was the 2nd round of a tournament (Round_2nd Round)

Tuned Hyperparameters:

- class_weight= {0: 1, 1: 2}
- max_features= None
- n_estimators= 900

Variable	**Feature Importance**
elo_winner	0.366
elo_loser	0.337
Year	0.048
Lsets	0.015
Round_2nd Round	0.010



Cross Validated XGBoost Model

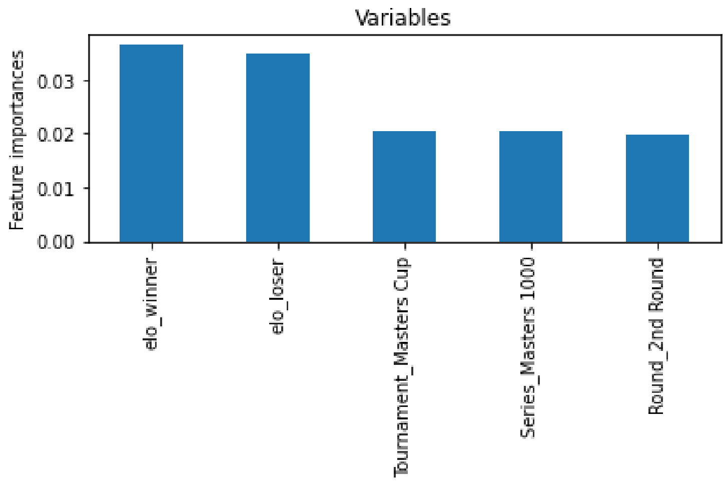
The XGBoost Model also found elo_winner and elo_loser to be most important, but also identified tournament level to be an important feature. We were not able to tune as many parameters given the complexity of the XGBoost Model, and found that performance was not better than RandomForest

Tuned Hyperparameters

- max_depth=10
- n_estimators= 1000
- subsample= 1

Model Results

Variable	**Feature Importance**
elo_winner	0.036
elo_loser	0.035
Tournament_Masters Cup	0.020
Series_Masters 1000	0.020
Round_2nd Round	0.020



Tuned Model Performance

Based on ROC alone, the RandomForest Classifier does the best with an ROC of 0.795. Specifically, if we were to apply this model to the use case of betting on potential upsets, it's more important to have the best recall possible and the RandomForest Classifier also achieves that with 0.73.

	Decision Tree	**Random Forest**	**XGB**
accuracy	0.804	0.815	0.815
precision	0.758	0.737	0.753
recall	0.642	0.729	0.699
ROC	**0.766**	**0.795**	**0.788**

Future Exploration

In future iterations of this project, the goal would be to gather more recent data to more accurately predict future results (2018-present), include WTA matches, and implement a betting strategy to determine the cost benefit analysis of lower upside on more predictable matches vs. higher upside on less likely match outcomes