NBA Game Outcome Prediction using ELO

Filip Jevtic
MSU NetID: jevticfi

Final Report

1. Introduction

Modeling a sport to gain a better understanding of game outcomes is very challenging. Many different organizations and people have attempted to model the game of basketball and in doing so, metrics have been developed to measure teams' performance. One such metric is ELO, a rating system based only on the final score of each game. It's also zero-sum, meaning that the winning teams will increase in rating as much as the losing team decreases in rating. This project aims to develop a model to predict a team winning a regular season game based on ELO ratings and 4 other predictors using a range of optimized methods that includes k-Nearest Neighbors, Support Vector Machines, Decision Trees, Random Forest Classification, QDA, and Logistic Regression.

2. Related Work

Previous work has shown that Elo ratings can be correlated with other statistics to predict whether a team will win a game. In a towardsdatascience.com article (which is based on 2013 paper Renato Torres from the University of Wisconsin-Madison), Josh Weiner and his partners were able to generate a prediction accuracy of 74.1% accuracy for playoff games. To generate this accuracy, Weiner used 5 features and grid search cross validation to optimize a random forest classifier.

This project takes the work done in previous studies a step further. I use 5 features, too, but tried prediction on regular season games as opposed to playoff, and created more models to discover the best one.

3. Dataset

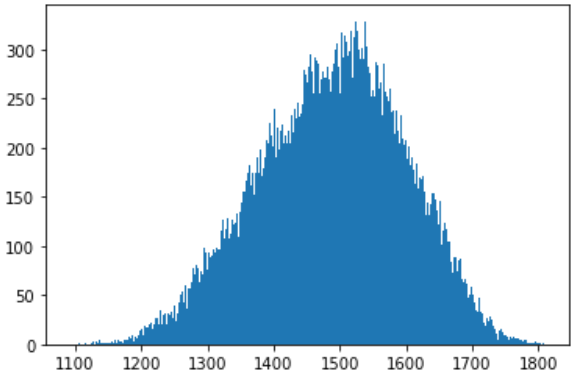
The data used is the same as the relevant works', sourced from FiveThirtyEight.com. Each row in the data represents a game for a team (meaning there are 2 rows for each game, 1 for losing team and 1 for winning).

For the initial preprocessing, features that weren't directly relevant to the statistics directly before a game begins were removed. This left about 8 features. The variables that contained results of the game were removed (end game ELO ratings, number of wins that season, points scored in the game). This left 5 features:

year_id: The year in which the game occurs. I include this to reduce bias of games that occurred many years ago.
Seasongame: The game in the team's season. I include this to model predictions based on when in the season the game occurs
Opp_elo: Opponent team's ELO rating before the game
Elo: Team's ELO rating before the game
Game_location: Binary variable (1: Home, 0: Away)
Forecast: percentage representing the win chances based on FiveThirtyEight.com's Calculation.

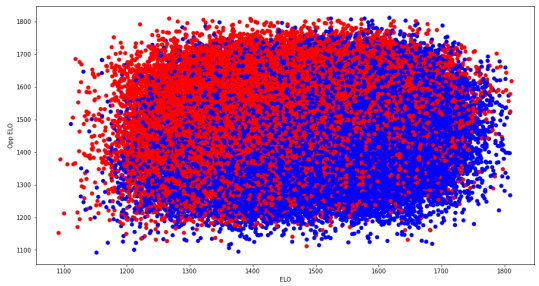
After cleaning the data, subsetting it to regular season games only, and adding a binary variable representing whether the team won the game, the data is in the shape: (118210, 7)

The ELO ratings are distributed as follows:

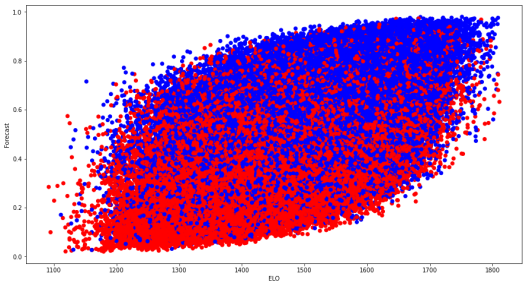


Red = Loss, Blue = Win

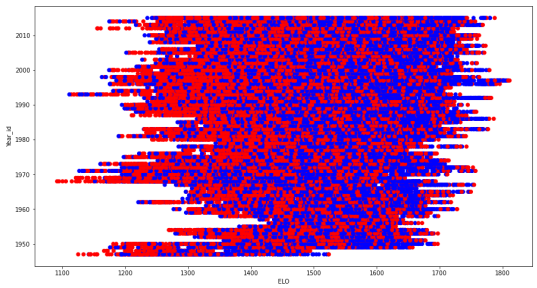
ELO vs Opp ELO



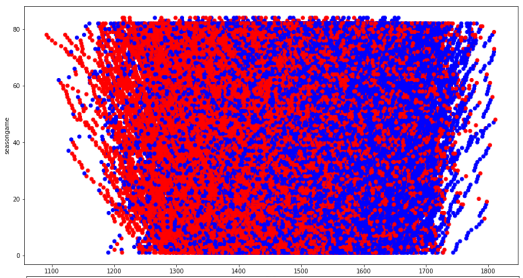
ELO vs Forecast



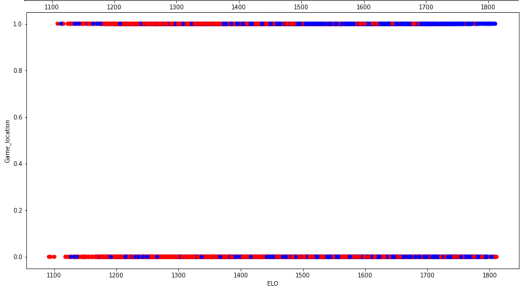
ELO vs Year



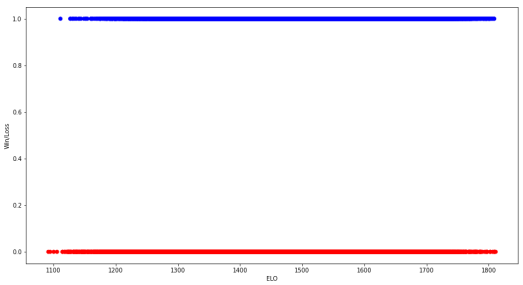
ELO vs Seasongame



ELO vs Game_location



ELO vs Win



4. Methods

The following models were used to determine the best predictive model using the sci-kit learn library, with input parameters optimized through grid search cross validation. The best combination of parameters was selected for each model, and each model was compared against one another using 4 different metrics on the testing data (Log Loss,MSE,Score,ROC Curve) to determine the best one.The data was split with a 60/40 train/test split.

KNeighborsClassifier:
Instance-based learning where classification is determined by a single majority vote of the nearest k neighbors for each point in the data, with k being specified in the function call. Since it is a non-parametric method, it is often very successful at classifying irregular decision boundaries.

Support Vector Classification:
Supervised learning method for classification that is effective in high dimensional spaces. It uses a subset of the training data to from a decision function (also called support vectors) which is based on the kernel function used. I test both linear and polynomial kernels.

Decision Tree Classifier:
Non-parametric supervised learning method that is used for classification. It forms a model that makes predictions based on learning simple decision rules that are formed from the features. Adding depth to the tree adds complexity and fits the data tighter to the training set.

Random Forest Classifier:
An estimator that fits many decision trees on bootstrap samples over the training data, which is supposed to improve accuracy and limit overfitting. The size of the bootstrap subsamples and number of decision tree estimators can be changed to optimize the model.

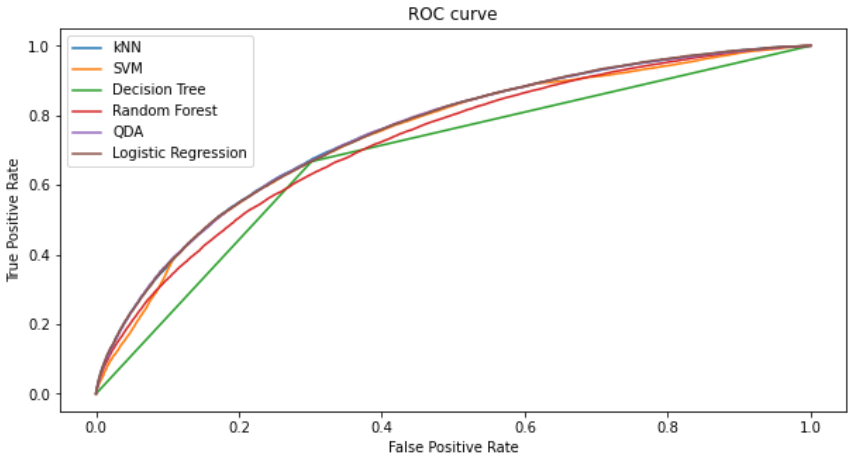
Quadrant Discriminant Analysis (QDA):
This classifier fits training data in such a way that it forms a quadratic decision boundary. It is a supervised dimensionality reduction method used to project input data into a quadratic subspace that maximizes the separation between classes.

Logistic Regression:
A linear model used for classification, also known as logit regression. Probabilities to describe the outcome of an input. The model can be optimized using either L1 or L2 regularization

5. Experiments and Discussion

Mean squared error, R^2, and Log Loss were used to evaluate the trained models. The best performing parameters for each model were selected through a grid search cross validation. Using the GridSearchCV function is sklearn, ranges of parameters were inputted for numerical parameters and the best model was automatically selected. Results for the best models are provided below.

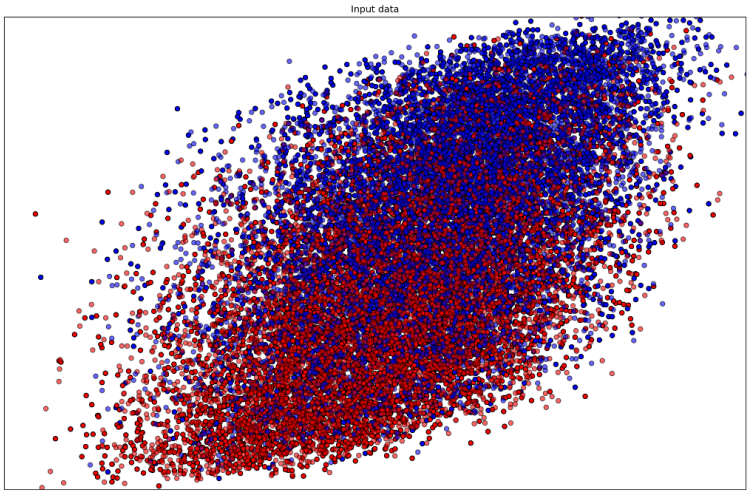
	Best Parameters	Mean Train R^2	Mean Test R^2	Test MSE	Test Log Loss
kNN	**K = 1000**	0.682	0.68169642	**0.31391168**	**0.59272884**
SVM	**C = 1 Kernel = "poly"**	0.680920	0.68065308	**0.31702056**	**N/A**
Decision Tree	**Max_depth = 1**	0.681562	0.67968023	**0.31704171**	**0.62455243**
Random Forest	**N_estimators = 1000**	**0.999971**	0.66142176	**0.33465866**	**0.61737806**
QDA	**Reg_param = 0**	0.681893	**0.681922**	**0.31606886**	**N/A**
Logistic Regression	**Penalty = 'l2'**	0.680801	0.68070947	**0.31735894**	**N/A**

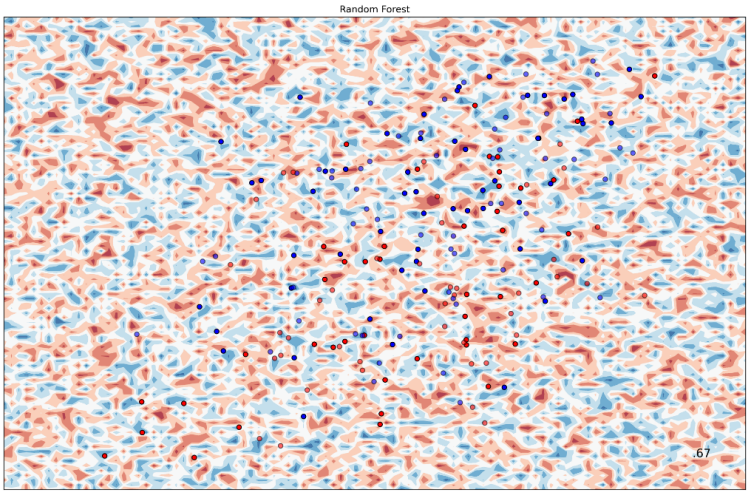


It can be seen that most of the models performed similarly, with one glaring difference. Random Forest classification seems to have a significantly better training R^2 value than any other, which means that 99.9% of the variance in the training data can be explained by the model. The other models all cap out at around 68%. The fact that the training data can be fit much better than the other models suggests that Random Forest is the best performing model.

In addition, each method was plotted into a 2 dimensional space with x as the test ELO and y as the test Forecast using a meshgrid to plot the decision boundaries:

The Random Forest Classifier had the following decision boundaries for the test data, the first plot is the training data with testing data plotted underneath with a softer hue. The decision boundary is not clearly clustered into 2 spaces, however there are concentrated areas in this 2 dimensional space that have higher correlations to a win/loss. The plot shows that no visual/linear relationship exists between ELO and Forecast in the model, yet the model was able to make predictions with about 67% accuracy.





6. Conclusions and Future Work

This project is an attempt to find the best model for predicting the outcome of a regular season NBA game. Using supervised machine learning classifiers (kNN,SVM,Decision Trees,Random Forest, QDA, and Logistic Regression) and grid search cross validation, metrics (MSE,Log Loss,R^2) show that Random Forest fits and predicts data best in this case, which verifies the results of the previous work mentioned in 2. However, the prediction was not as high performing as the previous work. The reasoning behind this is the feature selection. The features selected in the previous work were more statistical in nature and accounted for different measures of team performance as well as ELO, which I believe was the biggest factor in establishing the high accuracy. A deficiency of ELO is it is represented as a single value, which can induce bias potentially if it is not combined with other metrics. The features used in this project performed relatively well, still, since average accuracy for most nba game outcome models is about 70%. However, these features were not all performance related, which could have an impact on the variance in the data (hence why many models had lower R^2 value). A change to make in the

future is adding/subtracting features that are more/less relevant to the performance of a team and its players, as well as trying different models/hyperparameters. It would be interesting to fine tune the Random Forest in an effort to maximize its test set performance.

7. References

Codebase for meshgrid/classification script:
https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html
Grid Search CV:
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV.predict

Previous work:

- https://towardsdatascience.com/predicting-the-outcome-of-nba-games-with-machine-learning-a810bb768f20
- https://projects.fivethirtyeight.com/complete-history-of-the-nba/#lakers
- https://homepages.cae.wisc.edu/~ece539/fall13/project/AmorimTorres_rpt.pdf

Algorithm Descriptions: https://scikit-learn.org/stable/