Predicting Success with Simplyx

Peter Kinder

May 8, 2023

1 Introduction

In the age of big data, analytics has become a critical component in numerous fields. One issue facing industry is the ability to efficiently perform analytics on big data. As a result of this demand, an array of analytical tools have been developed. One tool in particular, Alteryx, is a low-code option that is effectively a graphical user interface (GUI) based on R. Alteryx has over 8,300+ corporate customers, including Coca-Cola, Walmart, Adidas, KPMG, JPMorgan, and many others. Due to this growing market demand and large adoption rate from industry pillars, the ability to use Alteryx has become a valuable skill for employees and soon to be employees such as college students.

At the University of Colorado at Boulder, one class, entitled Business Analytics, incorporates Alteryx into the curriculum. One of the culminating experiences of the class is the Alteryx Designer Core Certification, which assesses an individual's ability to use Alteryx and solve problems using data. However, the percentage of students who were able to obtain the certification was around 25%, leaving a lot to be desired.

The reason why the pass rate was relatively low is debatable, but one theory was the lack of hands-on learning exercises that could be evaluated by an instructor. Alteryx provided a vast amount of documentation, interactive lessons, videos and challenges. However, this either required initiative on the part of the student or was burdensome to assign and evaluate at scale by an instructor. In response, the professor of the course, Kai Larsen, and I developed Simplyx, which is a combination of curriculum and "courseware" built atop Alteryx. As a result, over 55% of students were able to obtain their certification on the first attempt, and over 80% over the course of the semester.

Clearly, from just eyeballing the before and after pass rates, Simplyx has a statistically significant effect on the outcomes of students taking the Alteryx Designer Core Certification. In fact, I proved such in a prior project and report. However, not all students end up obtaining their certification, and based on discussions with Kai as well as the more granular data that has been collected over recent semesters, it was determined that a predictive model should be developed. The main motivation behind the predictive model was two-fold. The first was that a single metric predicting how a student would perform could be used by a professor to target assistance to specific students. Additionally, Kai believed that showing each student their predicted score could serve as motivation to students. Lastly, Kai suggested that the model predict how the student performs on their first attempt because he thought that would be the most helpful as an instructor.

In an effort to create a predictive model, this paper will explore the use of various methods of regression tree ensembles. To do so, the data and its preparation will first be discussed. Then, specific regression tree ensembles and a general procedure will be laid out. Next, the results from the various models will be compared to select the best model. Lastly, some general takeaways and future steps will be discussed.

2 Data

The data for the predictive model was collected in two ways. The first way was simply by the use of Simplyx. Each time that a student submitted an answer to Simplyx, a log of their submission was recorded. The log recorded the problem number, the "belt" color (which is essentially a section of the curriculum), whether the schema and answer were correct, and a timestamp. In total, there are six "belts" labeled as white, yellow, orange, green, purple and blue. Additionally, there are 148 distinct problems spread across the six belts. A sample log file looks as such:

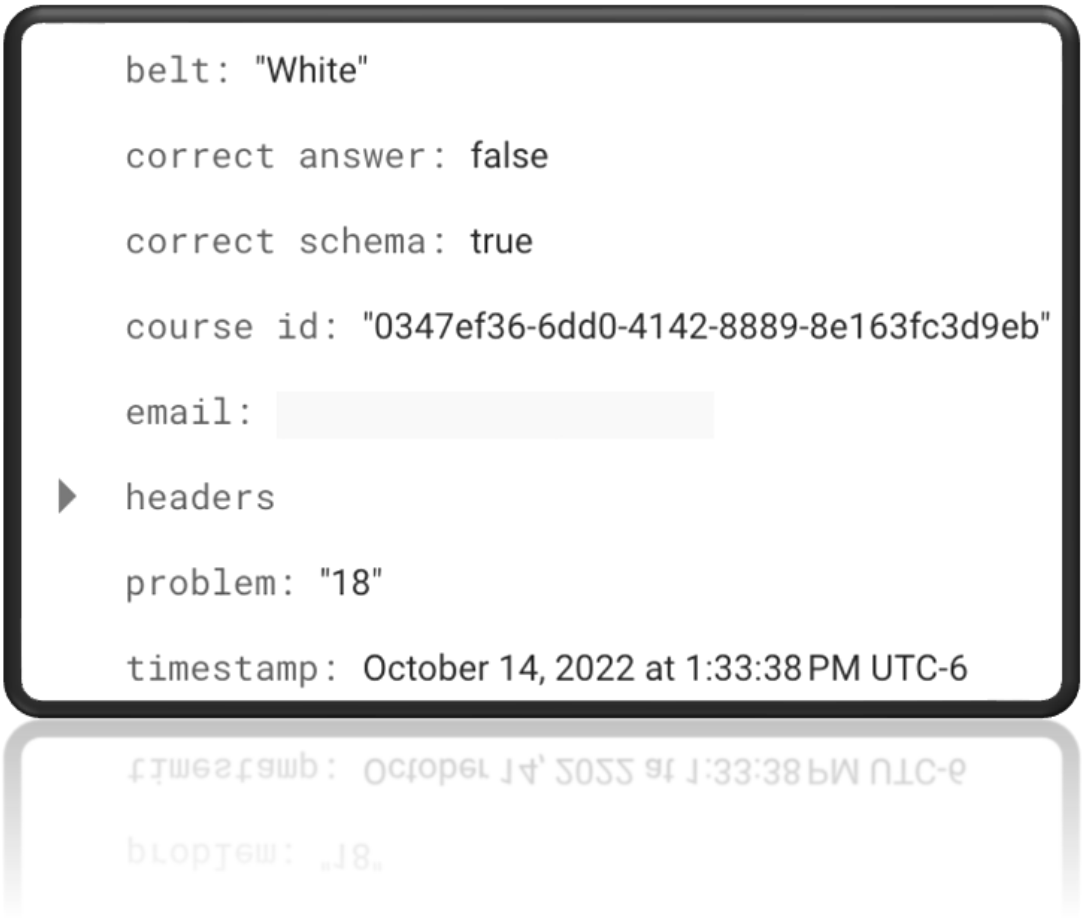

Figure 1: An example log file from Simplyx that was generated by a user submitting an answer to the backend server.

In addition to the log files, students were encouraged to submit how they scored on their certification attempts. Student would forward the email they received from Alteryx after their attempt, which detailed how they scored. Over the Spring 2022, Fall 2022, and Spring 2023 semesters, a total of 206 students submitted their first certification attempt scores. This score was the response variable used in the predictive model.

In total, there were 100K+ log files that corresponded with students who submitted their certification scores. This data was prepped (using Alteryx) so each student had a percent score for each belt, a one hot encoding denoting whether they answered a specific problem correctly, and five usage summary statistics. The usage summary statistics included total attempts, total sessions (with 1-hour gaps signifying an end of session), and average correct answers, attempts, and seconds per session. All combined, there were 159 predictors.

3 Approach

The approach for creating the predictive model will be broken into two parts: the modeling methods used and the general procedure of the approach. In the modeling methods portion, the specific regression tree models will be discussed. In the procedure portion, the the procedure for how the models were applied to the data to obtain robust results will be laid out.

It should be noted that the methods portion will briefly touch on the specifics of the models used. Fully explaining each model used would constitute a paper in itself, and is beyond the scope of this paper. However, for readers who are unfamiliar with the models discussed, I would suggest the Further Reading section at the end of this paper.

3.1 Methods

The general type of predictive models considered were regression tree ensembles. To understand what a regression tree ensemble is, one must first understand what a regression tree is. Seen below is a visualization of a basic regression tree and the partitions of data that represent the splits in the tree:

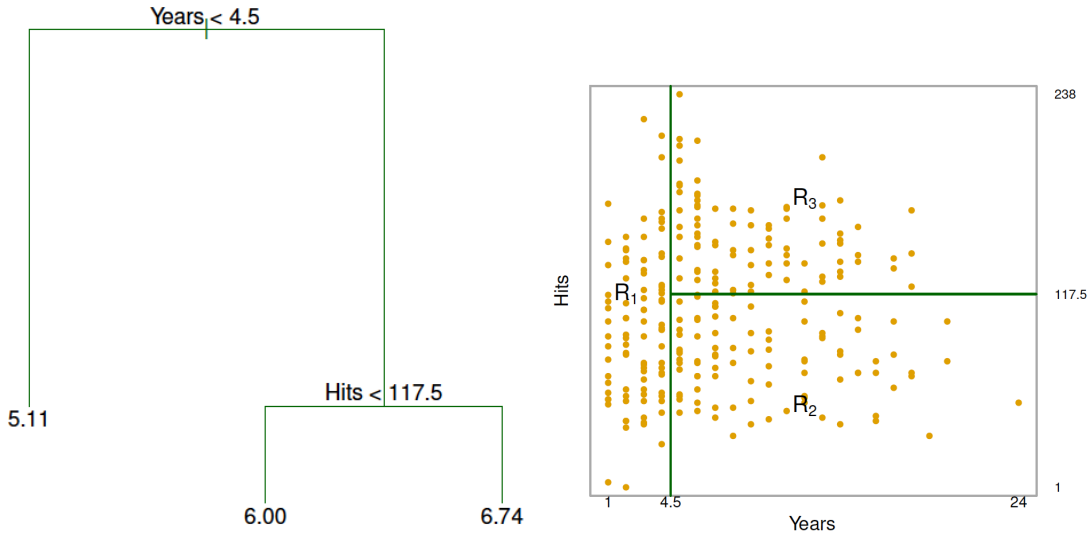

Figure 2: On the left is an visualization of a simple regression tree. On the right is a visualization of the partitions of the data that the regression tree represents. From "An Introduction to Statistical Learning: With Applications in R," by James, Witten, & Tibshirani, 2022. Copyright 2022 by Springer Science+Business Media, LLC.

Tree based methods in general are very useful for modeling non-linear relationships. The model is in Figure 2 is trying to predict a baseball player's salary using hits per year and total years in the league. The vertical partition in the data corresponds to the first split in the tree, and the horizontal partition corresponds to the second split in the tree. Then, the salary response is the average of all the salary data points within that partition. For example, the model would suggests that players with more than 4.5 years in the league and 117.5 hits per year have a 6.74 (obviously transformed, but left for simplicity) average salary. To arrive at this, the model partitioned the data with the goal of minimizing the residual sum of squares (RSS) with each additional partition.

To increase the predictive power of the model, instead of a single regression tree, an ensemble of regression trees is used. An ensemble is "an approach that combines many simple 'building block' models in order to obtain a single and potentially very powerful model. These simple

building block models are sometimes known as weak learners, since they may lead to mediocre predictions on their own" (James, Witten, & Tibshirani, 2022, pg. 340). The types of ensemble methods that are used to create the predictive model in this paper include bagging, random forests, boosting, and Bayesian additive regression trees (BART).

Bagging is a method where a bootstrapped sample of the data is used to create multiple regression trees and the trees are averaged to model the data. The main benefit with bagging is that is "averaging a set of observations reduces the variance" (James, Witten, & Tibshirani, 2022, pg. 340). Additionally, a certain portion of the features are considered for each tree. All of this leads to a set of hyperparameters that need to be selected, which will be mentioned for each ensemble method. For bagging, the hyperparameters include the number of trees, the bootstrapped percent of the sample considered for each tree, and the percentage of features to consider for each tree.

A random forest is a method somewhat similar to bagging that "decorrelates the trees, thereby making the average of the resulting trees less variable and hence more reliable" (James, Witten, & Tibshirani, 2022, pg. 344) and is also an averaging of trees method. To do this, instead of considering a certain subset of features for each tree, the random forest method considers a random subset of features for each split of the tree. Additionally, another consideration that can lead to improved performance with this method is the depth of the tree. By adjusting the depth, a single tree can be prevented from becoming too accurate, which maintains its weak learner characteristic. All in all, the hyperparameters considered for a random forest are number of trees, the percent of sample to consider at each split of a single tree, and the max depth of the trees.

Boosting is different from bagging and random forest in a number of ways, but one main way is that instead of averaging the various trees, boosting "uses a weighted sum of trees, each of which is constructed by fitting a tree to the residual of the current fit" (James, Witten, & Tibshirani, 2022, pg. 348). This weighting is determined by a hyperparameter often described as a learning rate. In its simplest terms, trees are fit sequentially, and the learning rate determines how much the new tree learns from the last tree, which in turn, translates into the weight of the particular tree. For boosting, the hyperparameters considered are the number of trees, learning rate, max depth of the tree and the percent of features to consider for each tree.

BART is related to all of the previously listed ensemble methods: "each tree is constructed in a random manner as in bagging and random forests, and each tree tries to capture signal not yet accounted for by the current model, as in boosting" (James, Witten, & Tibshirani, 2022, pg. 348). In a bit more detail, "BART can be viewed as a Bayesian nonparametric approach that fits a parameter rich model using a strongly influential prior distribution" (Chipman, George & McCulloch, 2008, pg. 3). For BART, there are three main priors to consider, which are priors for T_j (Chipman, George & McCulloch, 2008, pg. 8), $\mu_{ij}|T_j$ (Chipman, George & McCulloch, 2008, pg. 9) and σ (Chipman, George & McCulloch, 2008, pg. 10). As for the hyperparameters consider with BART, the are the number of trees, α and β (associated with the T_j prior), k (associated with the $\mu_{ij}|T_j$ prior), and degrees of freedom as well as the uantile of variance that the estimate is placed at (associated with the σ prior).

3.2 Procedure

One of the main concerns in the approach was possibility of overfitting. To address this concern, the data was split into a training and test set, with 80% of the data in the training set and the remaining 20% in the test set.

The training set was then used to determine the optimal hyperparameters for each model by conducting a grid search. That is, a combination of values for the hyperparameters were iterated over and used to fit the model. Then, 5-fold cross validation was performed on the training set.

The measures of goodness that were considered are root mean squared error (RMSE) and mean absolute error (MAE). Both were considered because RMSE is more sensitive to outliers by virtue of its construction and MAE is simply the average absolute error. The RMSE and MAE would be averaged over the 5-folds, and the hyperparameters that yielded the lowest RMSE were deemed the optimal hyperparameters.

Once the hyperparameters were identified, the models were fit on the entire training set using those hyperparameters. Then, the models were used to make predictions for the test set, for which RMSE and MAE were calculated. Doing so robustness and generalizability of the optimal hyperparameter models to be evaluated, and the model yielding the lowest RMSE on the test set would be declared the best model.

Lastly, it should be noted that due to the very large number of predictors, principal components analysis (PCA) was used to reduce the dimensionality of the feature set. The number of principal components kept was such that the marginal increase of the total variance explained by an additional component was greater than 0.1. Furthermore, the principal components were normalized before fitting the model.

4 Results

Starting with the bagging method, the optimal hyperparameters produced by cross validation were 12 trees, 100% of the sample considered for each tree, and 90% of the features considered for each tree. Using these hyperparameters to fit a model using the entire training set produced a RMSE of 11.462 and MAE of 9.032 for predictions on the test set. Plotting the actual values against the predictions yielded the following:

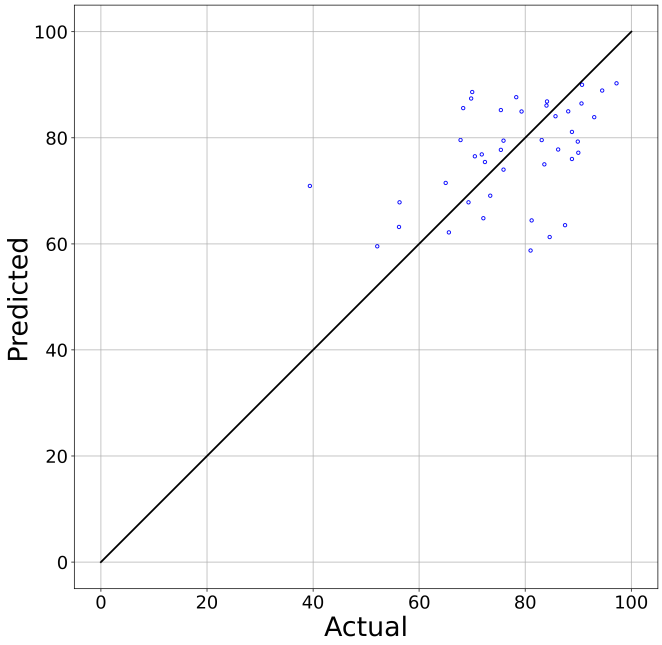

Figure 3: Plot of actual certification scores against predicted certification scores for bagging method on test set. The line is a reference point for a prediction without error.

Moving on to the random forest method, the optimal hyperparameters produced by cross validation were 6 trees, 70% of features considered at each split, and a max depth of 5. Using these hyperparameters to fit a model using the entire training set produced a RMSE of 10.454 and MAE of 7.750 for predictions on the test set. Plotting the actual values against the predictions yielded the following:

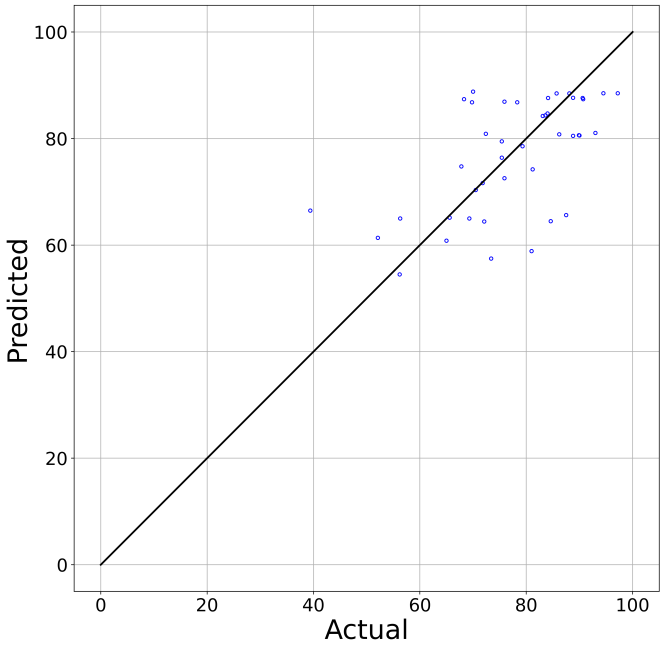

Figure 4: Plot of actual certification scores against predicted certification scores for random forest method on test set. The line is a reference point for a prediction without error.

For the boosting method, it should be noted that the particular type of boosting method used was AdaBoost. For specifics on AdaBoost, please refer to the further reading section. The optimal hyperparameters produced by cross validation were 16 trees, a 1.0 learning rate, a max depth of 15 and 30% of features considered for each tree. Using these hyperparameters to fit a model using the entire training set produced a RMSE of 10.874 and MAE of 8.007 for predictions on the test set. Plotting the actual values against the predictions yielded the following:

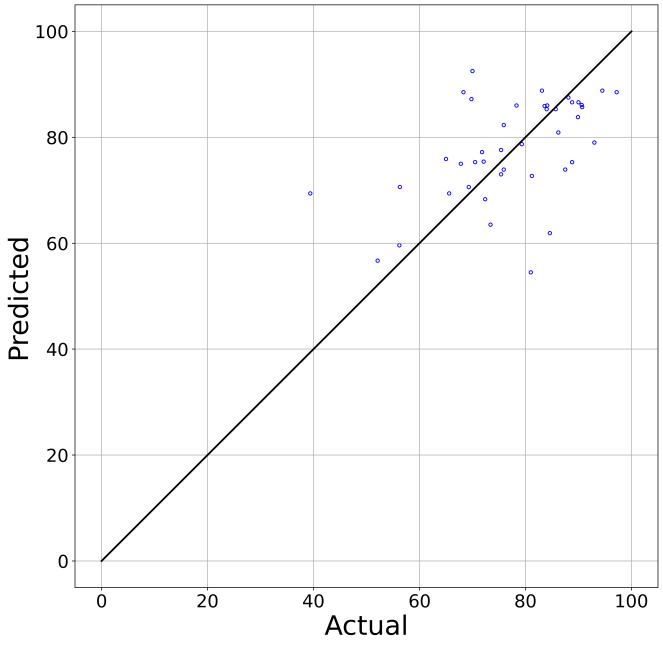

Figure 5: Plot of actual certification scores against predicted certification scores for boosting method on test set. The line is a reference point for a prediction without error.

For BART, the optimal hyperparameters produced by cross validation were 200 trees, 0.75 for α, 1 for β, 1 for k, 3 for the degrees of freedom and 0.7 for the quantile of variance that the estimate is placed at. Using these hyperparameters to fit a model using the entire training set produced a RMSE of 9.832 and MAE of 7.518 for predictions on the test set. Plotting the actual values against the predictions yielded the following:

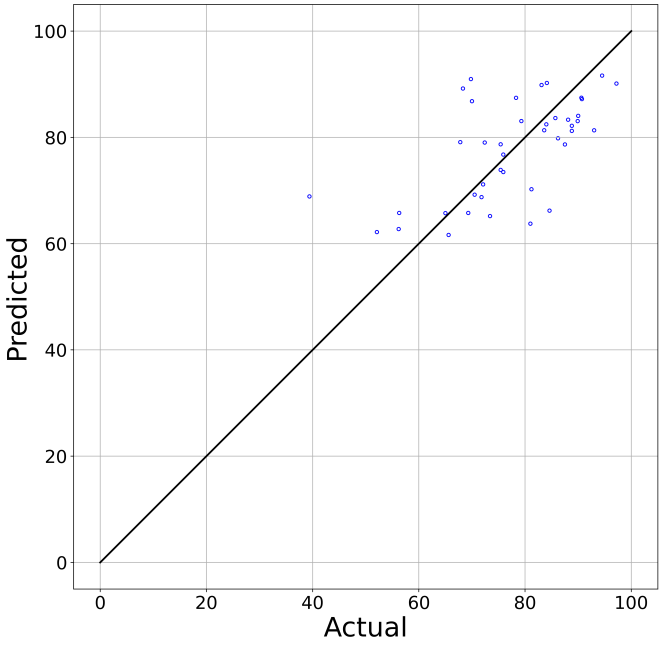

Figure 6: Plot of actual certification scores against predicted certification scores for BART on test set. The line is a reference point for a prediction without error.

Lastly, for comparison of how the models performed on the test set:

Method	**RMSE**	**MAE**
BART	9.832	7.518
Random Forest	10.454	7.750
Boosting	10.874	8.007
Bagging	11.462	9.032

5 Conclusion

In conclusion, BART was the model the produced the lowest RMSE and MSE on the test set, followed by random forest, boosting, and bagging. I honestly expected that random forest would perform the best, followed by bagging, BART and boosting. The reason I thought so was the way that boost and bart were constructed would lead to overfitting when fit to the whole training set. However, it appears that BART overcame this doubt and produced the most accurate and robust model. Additionally, apparently BART is "remarkably robust to hyperparameter specification, and remain[s] effective when ... buried in ever higher dimensional spaces" (Chipman, George & McCulloch, 2008, pg. 40). I think one of the main takeaways from this project will be to consider BART when exploring tree based methods in the future.

As for the specific predictive model that was created, I believe it will be helpful for instructors and students in the future. I plan to implement this model on the website of Simplyx so that instructors can see how their students are predicted to perform on the certification. Additionally,

I plan to allow students to see how they themselves are predicted to perform, with the hopes that it might motivate them by being able to see a predicted value for themselves. All in all, this was a very enjoyable project that opened my eyes to new modeling methods while working on a project that was meaningful to me.

References

[1] Chipman, H. A., George, E. I., & McCulloch, R. E. (2008). Bart: Bayesian Additive Regression Trees. *The Annals of Applied Statistics*, 4(1). https://doi.org/10.1214/09-aoas285

[2] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2022). *An Introduction to Statistical Learning: With Applications in R*. Springer.

Further Reading

[1] AdaBoost. scikit. (n.d.). https://scikit-learn.org/stable/modules/ensemble.html#adaboost

[2] Chipman, H. A., George, E. I., & McCulloch, R. E. (2008). Bart: Bayesian Additive Regression Trees. *The Annals of Applied Statistics*, 4(1). https://doi.org/10.1214/09-aoas285

[3] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2022). *An Introduction to Statistical Learning: With Applications in R*. Springer.