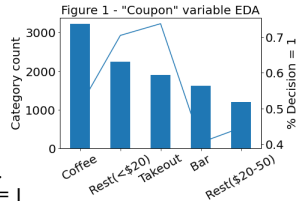
Predicting Coupon Acceptance Using Machine Learning Algorithms

Rob Kravec, Duke University, Statistical Science

Introduction

This project utilized an **in-vehicle coupon recommendation data set**, which was generated via a survey on Amazon Mechanical Turk [1]. In this survey, Turkers with high ratings (95% or above) were offered targeted "20% off" coupons to local businesses. Respondents who said they would drive to the coupon location "right away" or "later before the coupon expires" were marked as "Decision = 1", while users who answered "no, I do not want the coupon" were marked as "Decision = 0." Several user attributes (e.g., gender, age), contextual attributes (e.g., driving destination, time of day), and coupon attributes (e.g., time to expiry) were included in the data set, which contained 10,184 training observations and 2,500 test observations. The challenge, then, was to try to **predict whether a given coupon had been accepted** based on the provided information.

Data preparation and exploratory data analysis



Variable types: All variables were converted to categorical data types. None of the seemingly numeric features were continuous, nor did gaps between the discrete values of the variables have intuitive numerical meaning.
Missing data: Missing values were simply replaced with the string "NA" to create an additional category (relevant for ~5% of training observations).
Feature engineering: "Coupon_category_rating" was created to denote the frequency with which users attended establishments of the type offered in the coupon.
Exploratory analysis focused on (1) counts and (2) observed incidence of Decision = 1 for each category within each variable (see Figure 1 for example). Overall, ~57% of observations indicated an accepted coupon.

Methods

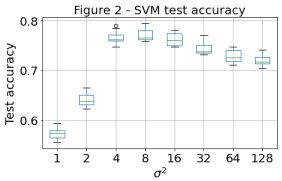
Four different algorithms were considered for this prediction task: Support Vector Machines (SVM), Random Forest, AdaBoost, and XGBoost.

Support Vector Machines (SVM) aim to minimize the penalty (hinge loss) from points being too close to or on the wrong side of the constructed decision boundaries. In this project, a Gaussian kernel was used, which creates decision boundaries by centering a Gaussian distribution on each data point. The σ^2 parameter defines the shape of the Gaussian distribution, where a value of σ^2 that is too small will overfit the data, and a value of σ^2 that is too large will underfit [2,3].

Random Forest: To fit each tree within a Random Forest, a bootstrap sample equivalent in size to the original data set is drawn. In creating a tree, only m out of p of the possible predictors is considered at each split, and the tree is grown until a minimum number of observations is present at each leaf. In the package used for this project, CART trees are used as the base estimator, meaning that Gini index is used as the splitting criteria. Additionally, m is defined as $\sqrt{p}$, and the minimum number of observations in a leaf is 1. Predictions are made based on majority vote, and the number of trees is the main hyperparameter to be tuned [2,4].

AdaBoost: In general, boosting seeks to address the challenge of turning a "weak learning algorithm" into a "strong learning algorithm." At each AdaBoost iteration, the algorithm assigns higher weights to data points that have been classified incorrectly so that these misclassified observations will be emphasized during the next iteration. Additionally, the contribution of each weak classifier to the final prediction is based on its training prediction accuracy. Typically, decision trees are used as the weak classifiers in AdaBoost, and for this project, CART decision trees were used. Both the number of trees and the depth of each tree are hyperparameters that need to be tuned for the AdaBoost algorithm [2,5].

XGBoost is another boosting algorithm that aims to turn a weak learning algorithm into a strong learning algorithm. To highlight a few of the main differences between XGBoost and AdaBoost: (a) The weak classifier in iteration t of XGBoost is built off the residuals of the weak classifier in iteration t-1. (b) XGBoost seeks to minimize the logistic loss, while AdaBoost seeks to minimize the exponential loss. (c) In XGBoost, the learning rate determines how much each new weak classifier contributes to the final model and is typically the same for all weak classifiers. The number of trees, depth of each tree, and learning rate are all hyperparameters that need to be tuned [2,6].



For SVM and Random Forest, a simple 10-fold cross-validation scheme was used to tune a single hyperparameter. Figure 2 shows the results for SVM as an illustration of the process. For AdaBoost, grid search cross-validation was employed to tune two hyperparameters. Given that three different hyperparameters were tuned for XGBoost, a randomized search cross-validation was utilized, testing 500 different combinations.

Summary of prediction results

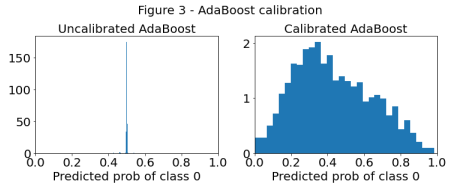
Table 1 provides a summary of tuned hyperparameters, training error, and test error for the four algorithms discussed in the Methods section. **Of note, the SVM model achieved the highest training (77.13%) and test (76.68%) accuracy**, followed closely by XGBoost (76.53%, 76.28%). All algorithms produced models that generated accuracy values within a range of 2.5 percentage points on both training (74.81% - 77.13%) and test (74.24% - 76.68%) sets.

Table 1 – Tuned hyperparameters and model accuracy

Algorithm	Hyperparameters	Training error	Test error
SVM	$\sigma^2 = 8$	77.13%	76.68%
XGBoost	Number of trees = 189 Max depth = 6 Learning rate = 0.2388	76.53%	76.28%
Random Forest	Number of trees = 175	76.35%	75.56%
AdaBoost	Number of trees = 250 Max depth = 2	74.81%	74.24%

To further improve predictive accuracy, ensembles of multiple models were considered. First, predicted values were compared between algorithms to determine whether there existed a significant enough difference in predictions to expect a performance improvement from aggregating predictions. For each pair of algorithms considered, 200-300 out of 2,500 test set predictions were different, suggesting a moderate opportunity for different algorithms to correct each others' faults.

The best models for each algorithm were used to predict probabilities of test observations belonging to each class, as opposed to simply predicting a 0 or 1 label. It was observed that nearly all predicted probabilities for AdaBoost were within 0.01 of 0.5 (see Figure 3). As a result, predicted probabilities for all models were calibrated using the CalibratedClassifierCV function within scikit-learn [2].



While numerous combinations of algorithms were considered, a **simple average of calibrated predicted probabilities between the SVM and XGBoost models produced the best observed test accuracy of 77.92%**, a slight improvement over the accuracies of any of the individual algorithms.

Model reliance

In addition to computing model accuracy, it was of interest to determine **which variables were most "important"** for the best-performing models from each algorithm. Given that one-hot encoded data sets were used to train all models, a custom model reliance function was written to group relevant columns together (e.g., the "Bar" feature had become 5 one-hot encoded variables). These groups of columns were then permuted in the test set, and test errors were calculated. Out of convenience (i.e., readily available scikit-learn functions), hinge loss was used for the SVM model reliance calculation, while logistic loss was used for all other algorithms. The five most important features for each algorithm are shown in Table 2. Note that **coupon category rating and coupon type appear in the top three for all algorithms.**

Table 2 – Most important variables based on model reliance

Rank	SVM	Random Forest	AdaBoost	XGBoost
1	Coupon category rating	Coupon category rating	Coupon category rating	Coupon category rating
2	Coupon validity	Coupon type	Coupon type	Coupon type
3	Coupon type	Coupon validity	Occupation	Coupon validity
4	Income	Occupation	Income	Occupation
5	Occupation	Frequency of visiting inexpensive restaurants	Driving destination	Marital status

References and acknowledgements

This portfolio project was based on an end-of-semester Kaggle competition in Professor Cynthia Rudin's STA 671 course (Theory and Algorithms for Machine Learning, Fall 2021) and was advised by Professor Merlise Clyde.

1. Wang, Tong, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille. 'A bayesian framework for learning rule sets for interpretable classification.' The Journal of Machine Learning Research 18, no. 1 (2017): 2357-2393.
2. Pedregosa, F. et al., 2011. Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), pp.2825–2830.
3. Vapnik, V.N., Chervonenkis, A.Y. "On a class of algorithms of learning pattern recognition." Avtomat. i Telemekh. 25.6 (1964): 937.
4. Ho, T.K. (1995) Random Decision Forest. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, 14-16 August 1995, 278-282.
5. Freund, Y, Schapire, R. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," Journal of Computer and System Sciences. 1997. 119-13
6. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794). New York, NY, USA: ACM. https://doi.org/10.1145/2939672.2939785