Random Forest

Gergely A. Marias

In this assignment, I am exploring the application of Random Forest algorithm, to predict the transportation choices individuals make for their daily commutes in Sweden. The aim is to forecast whether an individual will prefer public transport (coded as 0) or a private car (coded as 1), leveraging a range of features related to trip characteristics and personal attributes. The dataset captures the decisions of 5,896 individuals, each journey defined by various crucial parameters. These parameters encompass the duration of the trip for both public transport and car travel, the trip duration ratio, indicators of transfers during public transport trips, walking distance to public transport stops, waiting times during interchanges, public transport service frequency, distance traveled, geographical location (specifically, trips in the Oslo municipality), and demographic information such as income, education level, gender, and age. To construct a predictive model, I divided the dataset into a training set and a test set.

Random Forest: Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees during training. The “random” comes from the fact that each tree is trained on a random subset of the training data, and at each split in the tree, a random subset of predefined number of features is considered. Randomness helps prevent overfitting by introducing diversity among the individual trees. If each tree were trained on the entire dataset with all features, there is a higher risk that the ensemble would memorize the training data, resulting in poor generalization to new, unseen data. If there are highly correlated features in the dataset, training each tree on a random subset of features at each split helps in decorrelating the trees. This is important because correlated features can dominate the decision-making process and lead to an overemphasis on certain aspects of the data. Furthermore, the use of random feature subsets naturally provides estimations of feature importance.

Bagging: Bagging, or Bootstrap Aggregating, is a general ensemble method where multiple models are trained independently on different subsets of the training data, and their predictions are combined. The subsets are created by randomly sampling the data with replacement, meaning that some instances may be repeated while others may not be included at all. The final prediction is often obtained by averaging (regression) or taking a majority vote (classification) across the predictions of the individual models. Each model is trained on a random subset of the original dataset, typically about 63.2% of the data on average. This is because, in bootstrap sampling, each data point has approximately a 63.2% chance of being selected for the training subset. The data points that are not included in the training subset called Out-of-Bag data (~36,8%) which can be use to evaluate the model. Moreover, the final prediction of the ensemble is made by aggregating the predictions of all base models.

The key motivation behind ensemble methods is to improve the overall performance and robustness of a model by leveraging the strengths of multiple weak learners (models that perform slightly better than random chance). By combining these weak learners, ensemble methods aim to reduce overfitting, enhance generalization to unseen data, and provide a more stable and accurate prediction. However, ensembles models can be challenging to interpret. Moreover, the computational complexity and increased training times may limit their practicality, particularly in resource-constrained settings.

Difference between bagging and boosting Bagging and boosting are both ensemble methods, but they differ in their approach to combining multiple models. Bagging involves training each base model independently on a random subset of the training data created through bootstrap sampling. On the other hand, boosting builds a sequence of models sequentially. Each subsequent model in the sequence focuses on correcting the errors made by the previous ones. Data points that are misclassified or have higher errors are given more weight in the training process. Boosting aims to create a strong learner by emphasizing difficult-to-classify instances, resulting in a model with improved predictive performance, especially on challenging data points. In a nutshell, bagging relies on parallel training but boosting takes an iterative approach, emphasizing the correction of errors to sequentially improve model accuracy.

Parameter Tuning I started the process by loading the given dataset. Although it is not necessary, I converted all features, which contain only 0/1, to categorical. As desciptive analysis I examined the distribution of the target variable and observed that the likelihood of correctly predicting a random person's transportation mode as a car is 70%. So I can use this value as a baseline. Afterwards I splited the dataset to training and test sets, with 80% of the data allocated for training. In standard classification Random Forest models, a common parameter selection for the number of splits is based on the square root of the total number of observations. In my case, with 14 features, the recommended number of splits would be 3. However it is a good practice to explore values both below and above this recommendation. Furthermore, I have also investigated the number of trees beside the number of splits to gain a more comprehensive perspective. As the randomForest package already incorporates bagging, there is no necessity to pre-apply boosting on my dataset. Therefore, I can effortlessly examine the out-of-bag (OOB) errors on the training set to assess and compare the results based on varying numbers of trees and splits.

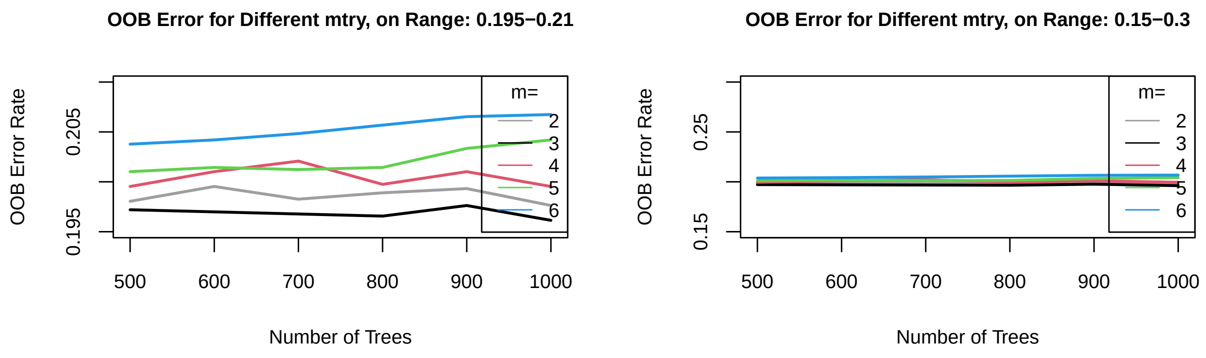

Figure 1: OOB Error Rates for Different mtry Values, Providing Guidance for Model Parameter Selection

Figure 1 highlights the superior performance of the configuration with three splits, denoted by the black line, showcasing the lowest error. However, upon a broader examination, particularly on the right side of the plot, it becomes evident that selection of the number of trees may not significantly impact the overall performance. In conclusion, it can be confidently stated that training the model with default parameter (with the number of trees set to 500) is acceptable. However, a critical refinement lies in setting the number of splits to 3 for optimal results.

From Figure 2 from Appendix A, the three most crucial features, according to the variable importance plot, are time_ratio (ratio of trip durations), time_car (car duration) and time_pt (public transport duration). These emphasize the significant influence of temporal aspects on predicting transportation modes. Time spent in public transport and cars, along with their relative duration, holds key insights for the model.

By applying predictions from my model trained with 3 splits, I've significantly improved the chances of correctly forecasting the transportation mode by nearly 10%, surpassing random guessing, thanks to the understanding of the target variable distribution. The current accuracy stands at 79.9%, as outlined in Table 1 within Appendix A.

References:

Molnar, C. (2019). Interpretable Machine Learning: A Guide for Making Black Box Models Explainable (pp. 79-124).

Chapman, C., & Feit, E. M. (2015). R for Marketing Research and Analytics (pp. 327-336).

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning (pp. 303-321).

Table 1: Evaluation metrics

	Accuracy	Precision	Recall	F1_score
Percentage	0.7991525	0.71875	0.5272206	0.6082645

Appendix A

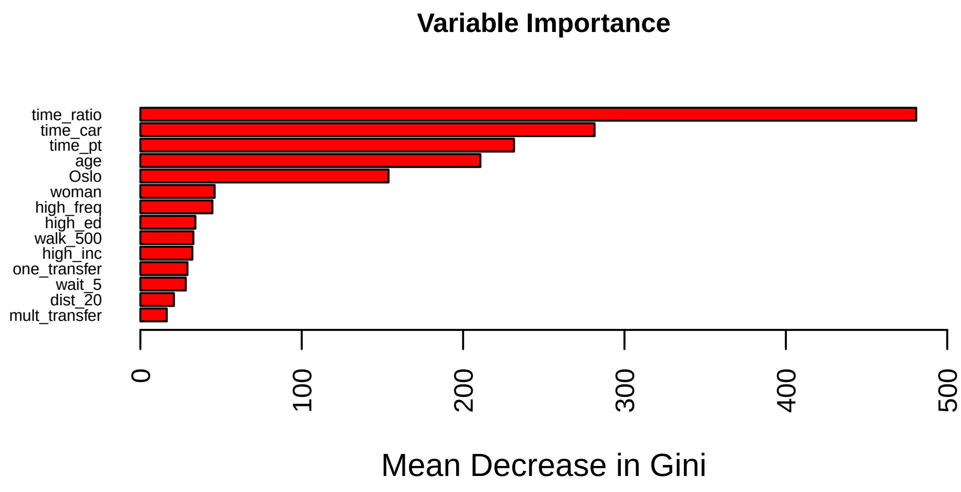

Figure 2: Barplot Illustrating the Mean Decrease in Gini for Each Variable, Reflecting Their Importance in the Random Forest Model