Module 6 Assignment on Trees and Boosting

Jingwen Zhong // Graduate Student

4/1/2021

Module Assignment

You will apply tree, bagging, random forests, and boosting methods to the `Caravan` data set with 5,822 observations on 86 variables with a binary response variable. This is a classification problem.

The data contains 5,822 real customer records. Each record consists of 86 variables, containing socio-demographic data (variables 1-43) and product ownership (variables 44-86). The socio-demographic data is derived from zip codes. All customers living in areas with the same zip code have the same socio-demographic attributes. Variable 86 (Purchase) is the target/response variable, indicating whether the customer purchased a caravan insurance policy. Further information on the individual variables can be obtained at http://www.liacs.nl/~putten/library/cc2000/data.html

Fit the models on the training set (as the split shown at the bottom codes) and to evaluate their performance on the test set. Use the R lab codes. Feel free to use other packs (caret) and k-fold methods if you like.

Q1) (*Modeling*)

a. Create a training set consisting from random 4,000 observations (shuffled and then split) with the seed with `set.seed(99)` and a test set consisting of the remaining observations (see the code at the bottom). Do a brief EDA on the target variable. Overall, describe the data. Do you think a samll number of predictors suffice to get the good results?

```
##
##   No  Yes
## 5474  348
```

```
##
##         No        Yes
## 0.94022673 0.05977327
```

```
##
## training set
```

```
##
##   No  Yes
## 3764  236
```

```
##
##    0    1
## 3764  236

##
## testing set

##
##   No  Yes
## 1710  112

##
##    0    1
## 1710  112

## [1] 4000   87
```

Overall, the data has 86 predictors which is much more than what we used in previous assignments, and I think a small number of predictors among these 86 predictors suffice to get the good results.

---

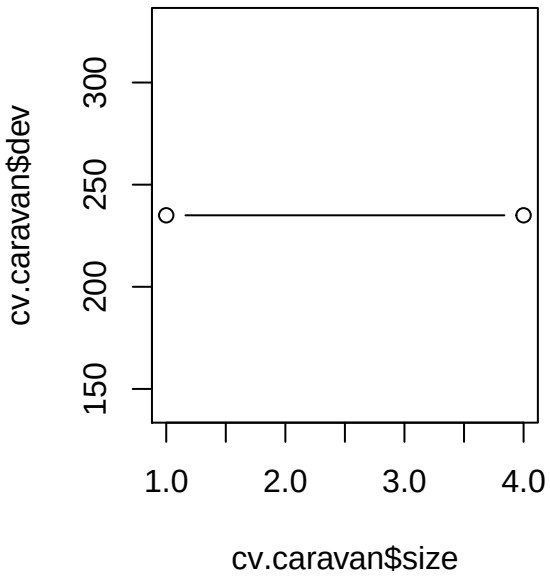
b. Fit a `logistic regression` to the training set with `Purchase` as the response and all the other variables as predictors. Report the *Accuracy* score on the train and test data sets. Discuss if any issues observed.

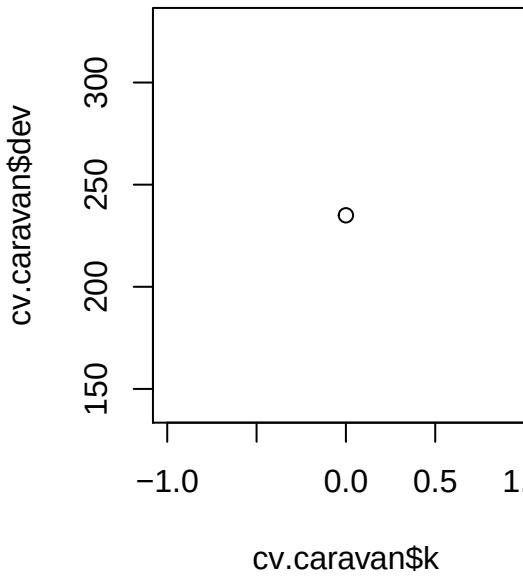
```
## Accuracy score of train data set is 0.94125
```

```
##
## Accuracy score of test data set is 0.9341383
```

---

c. Fit a `classification tree` model to the training set with `Purchase` as the response and all the other variables as predictors. Use cross-validation `cv.tree()` in order to determine the optimal level of tree complexity and prune the tree. Then, report the *Accuracy* score on the train and test data sets. If the R command gives errors, make necessary fixes to run the model. Discuss if

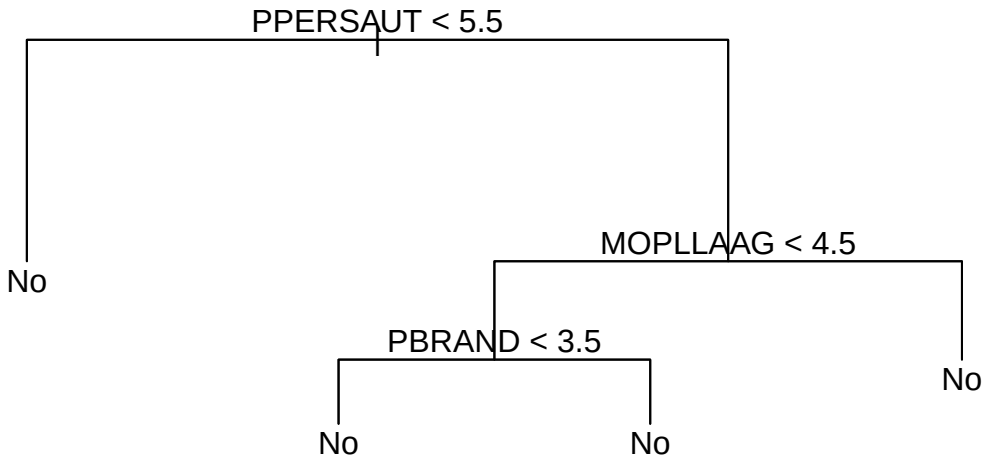




any issues observed.

```
## Accuracy score of train data set is 0.941
```

```
##
## Accuracy score of test data set is 0.9385291
```



One issues I observed is that in the tree graph, there is no prediction node is YES and my model predict everything as No, and the k is 1, so that means it only have 1 class, and it does not really come out with a tree. I think it's because the data is imbalanced, the percentage of NO of this data is so high (around 0.94), and also even if everything is predict as NO, accuracy still can be high.

---

d. Use the `bagging approach` on the classification trees model to the training set with `Purchase` as the response and all the other variables as predictors. Report the *Accuracy* score on the train and test data sets. Discuss if any issues observed.

```
## Accuracy score of train data set is 0.9925
```

```
##
## Accuracy score of test data set is 0.9242591
```

---

e. Use the `random forests` on the classification trees model to the training set with `Purchase` as the response and all the other variables as predictors. Find the optimal `mtry` and `ntree` with a sophisticated choice (no mandatory to make cross-validation, just try some) and report these. Report the *Accuracy* score on the train and test data sets. Discuss if any issues observed.

Table 1: Some of my tries

ntree	mtry	train_accuracy	test_accuracy	train_precision	test_precision
50	3	0.95400	0.9374314	0.2245763	0.0000000
50	8	0.97725	0.9335895	0.6355932	0.0357143
50	9	0.98100	0.9330406	0.6864407	0.0446429
50	10	0.98000	0.9319429	0.6694915	0.0357143
50	45	0.99050	0.9253568	0.8559322	0.0625000
50	80	0.99075	0.9253568	0.8601695	0.0625000
250	3	0.95425	0.9363337	0.2288136	0.0089286
250	8	0.98000	0.9319429	0.6694915	0.0357143
250	9	0.98150	0.9319429	0.6949153	0.0446429
250	10	0.98250	0.9308452	0.7118644	0.0357143
250	45	0.99225	0.9281010	0.8813559	0.0625000
250	80	0.99250	0.9253568	0.8855932	0.0625000
500	3	0.95400	0.9379802	0.2245763	0.0000000
500	8	0.97975	0.9308452	0.6610169	0.0178571
500	9	0.98200	0.9319429	0.7033898	0.0446429
500	10	0.98250	0.9324918	0.7118644	0.0446429
500	45	0.99225	0.9275521	0.8771186	0.0625000
500	80	0.99250	0.9248079	0.8813559	0.0625000

Table 2: optimal mtry and ntree by test accuracy

	x
ntree	500.0000000
mtry	3.0000000
train_accuracy	0.9540000
test_accuracy	0.9379802
train_precision	0.2245763
test_precision	0.0000000

Table 3: optimal mtry and ntree by test precision

	x
ntree	50.0000000
mtry	45.0000000
train_accuracy	0.9905000
test_accuracy	0.9253568
train_precision	0.8559322
test_precision	0.0625000

---

f. Perform `boosting` on the training set with `Purchase` as the response and all the other variables as predictors. Find the optimal `shrinkage` value and `ntree` with a sophisticated choice (no mandatory to make cross-validation, just try some) and report these. Report the *Accuracy* score on the train and test data sets. Discuss if any issues observed.

Table 4: Some of my tries

ntree	shrinkage	train_accuracy	test_accuracy	train_precision	test_precision
4000	0.001	0.94100	0.9368825	0.0042373	0.0000000
4000	0.020	0.95800	0.9319429	0.3050847	0.0357143
4000	0.050	0.97275	0.9324918	0.5508475	0.0714286
4000	0.100	0.98225	0.9281010	0.7161017	0.0625000
4000	0.200	0.98725	0.9242591	0.7966102	0.0714286
4000	0.500	0.99025	0.9165752	0.8559322	0.1071429
5000	0.001	0.94125	0.9363337	0.0084746	0.0000000
5000	0.020	0.96100	0.9335895	0.3559322	0.0535714
5000	0.050	0.97550	0.9302964	0.5932203	0.0535714
5000	0.100	0.98475	0.9264544	0.7542373	0.0892857
5000	0.200	0.98850	0.9209660	0.8177966	0.0625000
5000	0.500	0.96650	0.9072448	0.7627119	0.0803571
6000	0.001	0.94175	0.9363337	0.0169492	0.0000000
6000	0.020	0.96300	0.9330406	0.3898305	0.0625000
6000	0.050	0.98075	0.9324918	0.6864407	0.0714286
6000	0.100	0.98675	0.9248079	0.7838983	0.0714286
6000	0.200	0.98925	0.9231614	0.8389831	0.0714286
6000	0.500	0.93050	0.8918771	0.4915254	0.1339286

Table 5: optimal shrinkage and ntree by test accuracy

	x
ntree	4000.0000000
shrinkage	0.0010000
train_accuracy	0.9410000
test_accuracy	0.9368825
train_precision	0.0042373
test_precision	0.0000000

Table 6: optimal mtry and ntree by test precision

	x
ntree	6000.0000000
shrinkage	0.5000000
train_accuracy	0.9305000
test_accuracy	0.8918771
train_precision	0.4915254
test_precision	0.1339286

Q2) (*Discussion and Evaluation*)

a. Overall, compare the five models (parts b-f) in Question#1. Which one is the best in terms of *Accuracy*? Also, what fraction of the people predicted to make a purchase do in fact make one for on each model (use test data, what is called this score?)? Accuracy or this score: which one do you prefer to evaluate models?

In terms of *Accuracy* of test data set, classification tree is the best:

Table 7: Comparison of 5 models

	Train data accuracy	Test data accuracy
Logistic regression	0.94125	0.9341383
Classification tree	0.94100	0.9385291
Bagging	0.99250	0.9242591
Random forest	0.95400	0.9379802
Boosting	0.94100	0.9368825

Fraction of the people predicted to make a purchase do in fact make one for each model is called precision:$precision = \frac{TP}{Predicted'YES'}$ and I prefer precision score to evaluate models.

In terms of precision, boosting gets the best result.

Table 8: Comparison of 5 models by precision

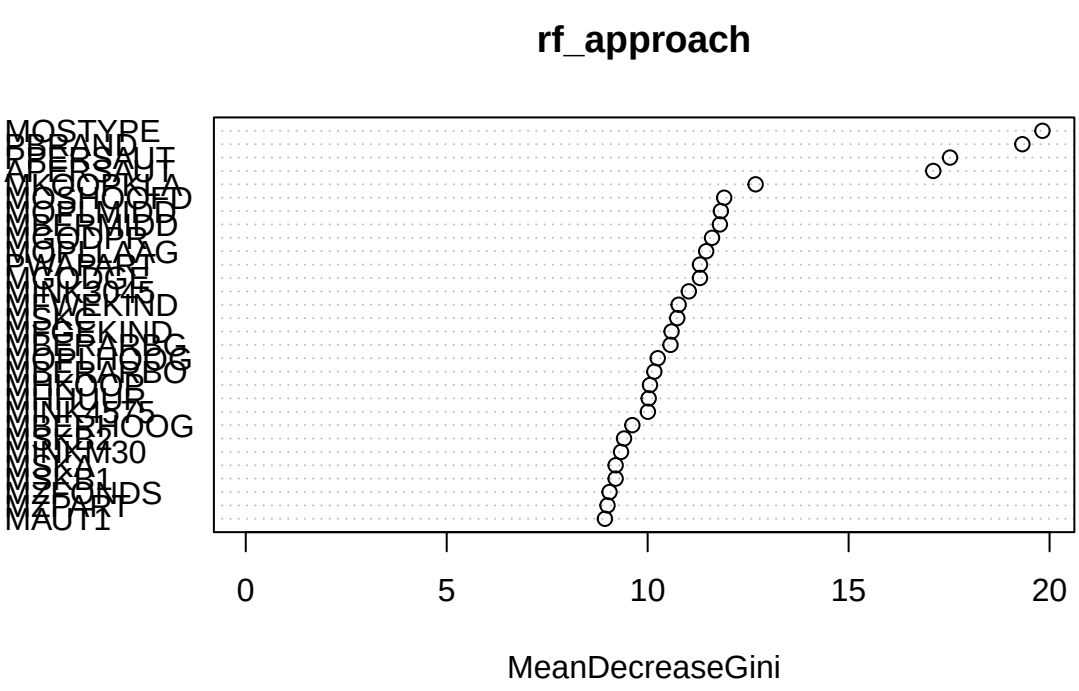
	Train data accuracy	Test data accuracy	Train data Precision	Test data Precision
Logistic regression	0.94125	0.9341383	0.0254237	0.0089286
Classification tree	0.94100	0.9385291	0.0000000	0.0000000
Bagging	0.99250	0.9242591	0.8813559	0.0625000
Random forest	0.99050	0.9253568	0.8559322	0.0625000
Boosting	0.93050	0.8918771	0.4915254	0.1339286

---

b. Determine which four features/predictors are the most important in the `random forests` and `boosting` models fitted. Include graphs and comments. Are they same features? Why?

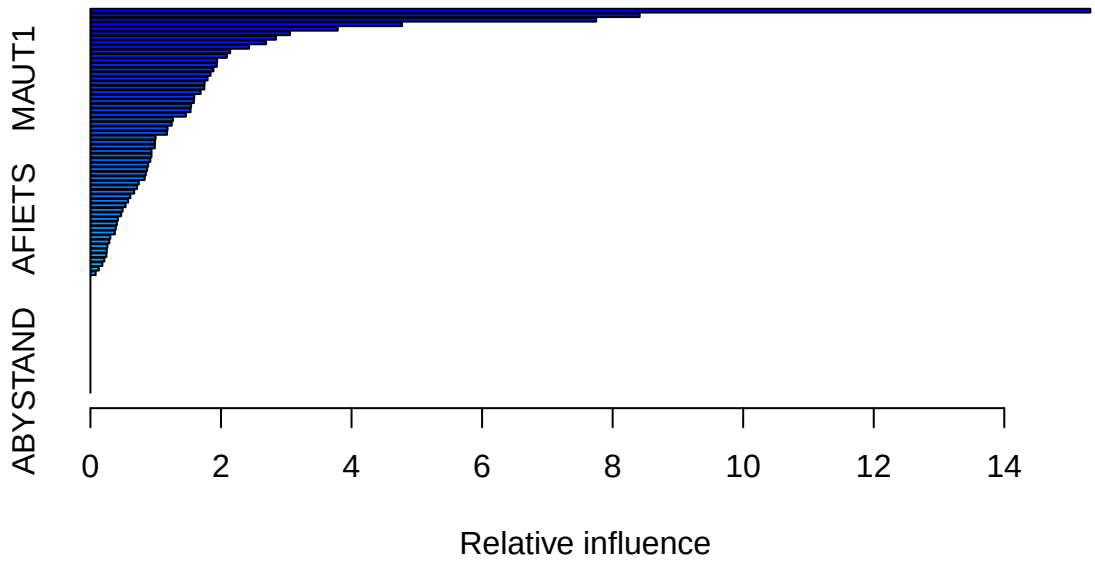
```
## The importance of predictors in the random forests:

##           MeanDecreaseGini
## MOSTYPE          19.82387
## PBRAND           19.32216
## PPERSAUT         17.52345
## APERSAUT         17.10597
```




```
## The importance of predictors in the boosting:

##                var   rel.inf
## PPERSAUT PPERSAUT 15.318542
## PBRAND     PBRAND  8.414359
## PPLEZIER PPLEZIER  7.747525
## MOSTYPE   MOSTYPE  4.774143
```



They have 2 same one. They are different because they use different loss function. Random forest uses Gini index to determine importance. Boosting: The function Boosting method uses is in Generalized Boosted Models: A guide to the gbm package page 10. And in both method they calculate the importance by randomly permuting each predictor variable at a time and computes the associated reduction in predictive performance..

---

c. Joe claimed that his model accuracy on the prediction for the same problem is 94%. Do you think this is a good model? Explain. No, because as I said in question 1, the data is imbalanced, the percentage of NO of this data is around 0.94.$Accuracy = \frac{TP+TN}{total}$.Therefore, even if everything is predict as NO, the accuracy still can be around 0.94.

---

d. (BONUS) How to deal with `imbalanced data` in modeling? Include your solution and one of model's test result to handle this issue. Did it improve?

Deal with imbalanced data with oversampling

```
##
##   No  Yes
## 5474 5474
```

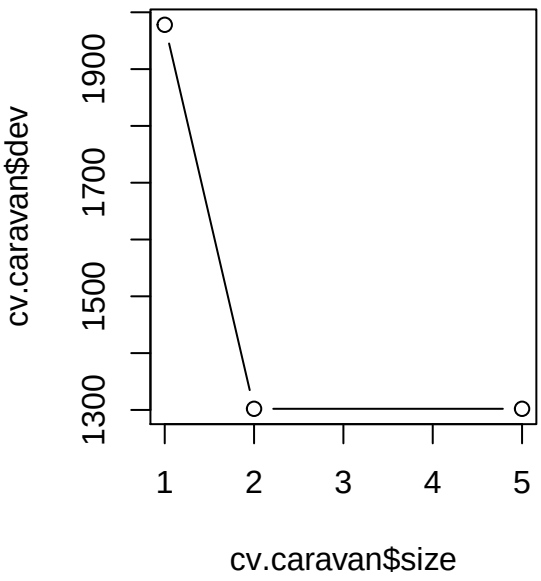
```
##
##  No Yes
## 0.5 0.5
```

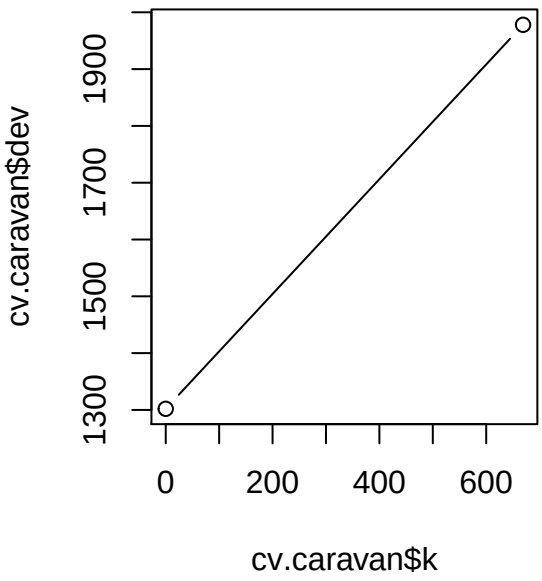
```
##
## training set

##
##   No  Yes
## 1978 2022

##
## testing set

##
##   No  Yes
## 3496 3452
```



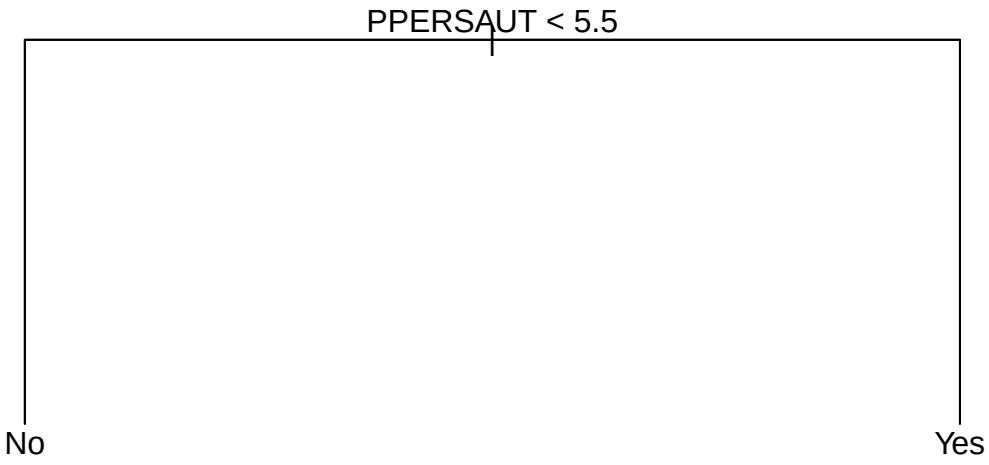



```
## Accuracy score of train data set is 0.67275

##
## Precision score of train data set is 0.7339268

##
## Accuracy score of test data set is 0.6888313

##
## Precision score of test data set is 0.7589803
```



The result improves a ot, first, it doesn't classfy everything as NO, so it has ok precision score, and although the accuracy is not that high, our model finally seems useful(The previous model is just a junk).

---

e. (BONUS) What happens to the results if you scale the features? Discuss.
Nothing will happen to Classification tree

Write comments, questions: ...

I hereby write and submit my solutions without violating the academic honesty and integrity. If not, I accept the consequences.

List the fiends you worked with (name, last name): ...

Disclose the resources or persons if you get any help: ...

How long did the assignment solutions take?: 10

References

...