STAT 5650 Statistical Learning and Data Mining 1 Spring 2020

Name: Michael Huber

Submission Date: 04/06/2020

Homework 4 (03/18/2020)

120 Points — Due Monday 04/06/2020 (via Canvas by 11:59pm)

(i) **Question 1:** In a typical bootstrap sample approximately 37% of the observations that are in the original dataset do not occur in the bootstrap sample. Here's a derivation of that result. Consider a dataset with n observations which we may label 1, 2, $\cdots$, k-1, k, k+1, $\cdots$, n-1, n.

(a) Suppose I select n observations from the original dataset with replacement. What is the chance that observation k is not among the selected observations? (Another way to think about this question is the following. Suppose I have a fair n–sided die with sides labelled 1, 2, $\cdots$, k, $\cdots$, n. I roll the die n times—and the rolls are all independent. What is the chance that the side labelled k does not occur in the n rolls?)

Answer: To figure the chance that observation k is not among the selected observations we take the product of the probabilities that each bootstrap observation is not the jth observation from the original sample. e.g. $(1 - 1/n)^n$. Definition taken from the links below.

References:

- https://rpubs.com/ppaquay/65561 Look at Question 2.
- https://blogs.sas.com/content/iml/2017/06/28/average-bootstrap-sample-omits-data.html

(b) Evaluate the expression you obtained in part a) for an increasing sequence of values of n.

```
> n = 10
> (1-1/n)^n

[1] 0.3486784

> n = 100
> (1-1/n)^n

[1] 0.3660323

> n = 1000
> (1-1/n)^n

[1] 0.3676954
```

(c) Do you recognize the limit as n -> ∞ of the expression in part a)? If so, identify and evaluate it. If not, compute the expression in part a) for some very large values of n.

```
> n = 100000000
> (1-1/n)^n

[1] 0.3678794

> n = 500000000000
> (1-1/n)^n

[1] 0.3678876
```

Answer: Computing two very large values for n we see that the values both equal 0.368.

We know from calculus that the *limit* as $n \to \infty$ $(1 + x/n)^n = e^x$. We also know that the limit a $n \to \infty$ of $(1 - 1/n)^n$ is $1/e$. Which shows us that as n approaches infinity the value will always equal 0.368. Which matches what we have shown above for very large numbers of n. We can write this equation as $1/e \approx 0.368$ as $n \to \infty$

References:

- https://rpubs.com/ppaquay/65561 Look at Question 2h
- https://blogs.sas.com/content/iml/2017/06/28/average-bootstrap-sample-omits-data.html

(d) (Quite hard). What is the standard error of the observed number and proportion of observations in the original sample that are not in a bootstrap sample?

Answer: The standard error of the observed number of observations can be figured out based on what we have calculated above. So we know that $X = np$ then $P = X/n$ so $E[p] = E[X/n] = E[X]/n = n\pi/n = \pi \approx 0.37$ and $SE(p) = SE(X/n) = 1/nSE(X) = \sqrt{\pi(1-\pi)/n} \approx \sqrt{0.37(1-0.37)/n}$

So for the formula above where n = 1000 we get the following output

```
> n = 1000
> sqrt((0.37*(1-0.37))/n)

[1] 0.01526761
```

(ii) **Question 2:** This problem continues the analysis of the Forensic Glass data.

(a) Apply random forests to the data and obtain the out-of-bag confusion matrix. How well can we classify this data, and where are the major misclassifications? How do your results compare to the classification tree you fitted in Homework 3.

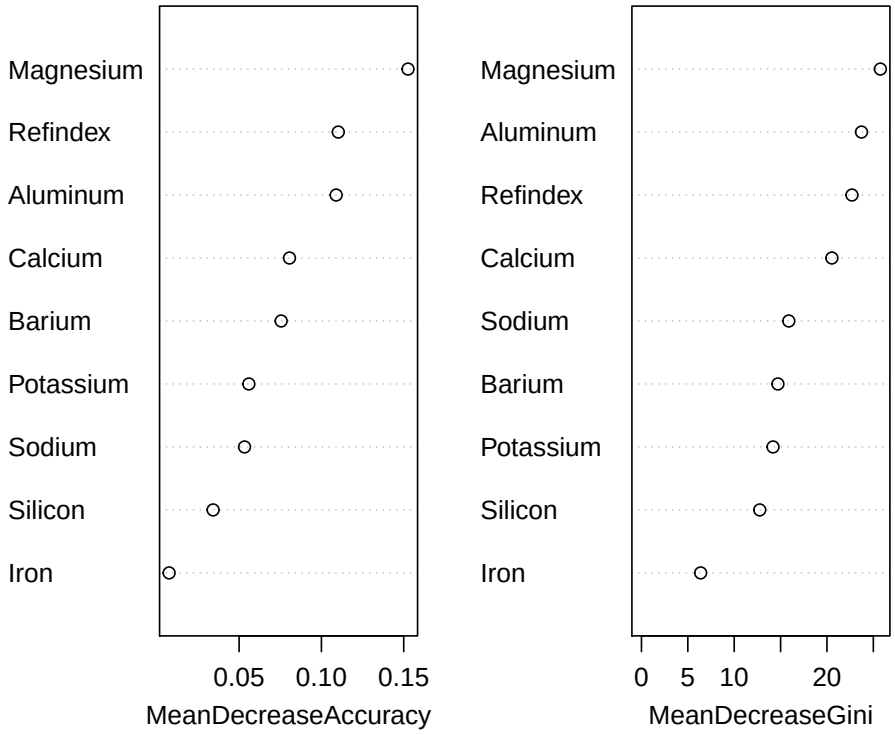
```
   1  2 3  4 5  6 class.error
1 61  6 3  0 0  0   0.1285714
2 10 61 1  1 2  1   0.1973684
3  7  4 6  0 0  0   0.6470588
4  0  2 0 10 0  1   0.2307692
5  0  2 0  0 7  0   0.2222222
6  1  2 0  0 0 26   0.1034483
```

```
[1] "Accuracy:"
```

```
[1] 79.33951
```

Answer: Right out of the box without any changes to the data we are getting an accuracy rate of 79.34% using random forests.
The top 3 missclassifications are GlassType 3, 4, and 5 respectively. 3 has an error rate of 0.647. While 4 as an error rate of 0.231 and 5 has an error rate of 0.222.
Comparing our accuracy rate of 79.34% to the accuracy rate of 67.29% that we saw in homework 3 when we fit a classification tree to the data. We can see that even without any variable selection or changes to the data random forests are already out performing the classification tree on this data. Improving our accuracy by around 13%.

(b) Use random forests to select a subset of the variables (which may be all the variables!) Refit random forests with only the important variables and obtain the out-of-bag confusion matrix. Did you observe any change in predictive accuracy?



Variable Selection: based on the chart above I first chose Magnesium, Refindex, Aluminum, Calcium, and Barium as my subset of variables to see if I could improve the accuracy. But using that subwset dropped the accuracy to around 75%. So I kept adding additional variables back into the subset of data until my accuracy surpassed what I had above. In the end I ended up using all of the variables to get a better accuracy, I think the accuracy came in slightly higher because subsetting the data just reordered the data frame.

```
   1  2 3 4 5  6 class.error
1 62  6 2 0 0  0   0.1142857
2  8 63 1 1 2  1   0.1710526
3  6  3 8 0 0  0   0.5294118
4  0  3 0 9 0  1   0.3076923
5  0  2 0 0 7  0   0.2222222
6  1  2 0 0 0 26   0.1034483
[1] "Accuracy:"
```

```
[1] 81.22605
```

Summary:
In the end I got an accuracy of 81.226% Which is just 1 or 2 percent higher than the accuracy I got in question a. So it is very close to what I was able to calculate above. Slightly better but from time to time as I run the Random Forests the accuracy changes slightly.

(c) Summarize your results for your analyses of the forensic glass data using classification trees and random forests.

Summary Answer: After comparing the random forests models on the glass data to the classification tree models we computed in homework 3 using the glass data. We can see in this instance that random forests outperformed the classification trees by increasing our predictive accuracy over 10%. It also is a better choice because for random forests you do not need to run the data through cross-validation, which improved the performance of the code along with decreasing the number of lines of code that you need to write.

(iii) This problem continues your analyses of the Uintah Mountains cavity nesting birds' data.

(a) Apply random forests to all the data with Species as the response variable and obtain the out-of-bag confusion matrix. How well can we classify these data, and where are the major misclassifications? How do your results compare to the classification tree you fitted in Homework #3.

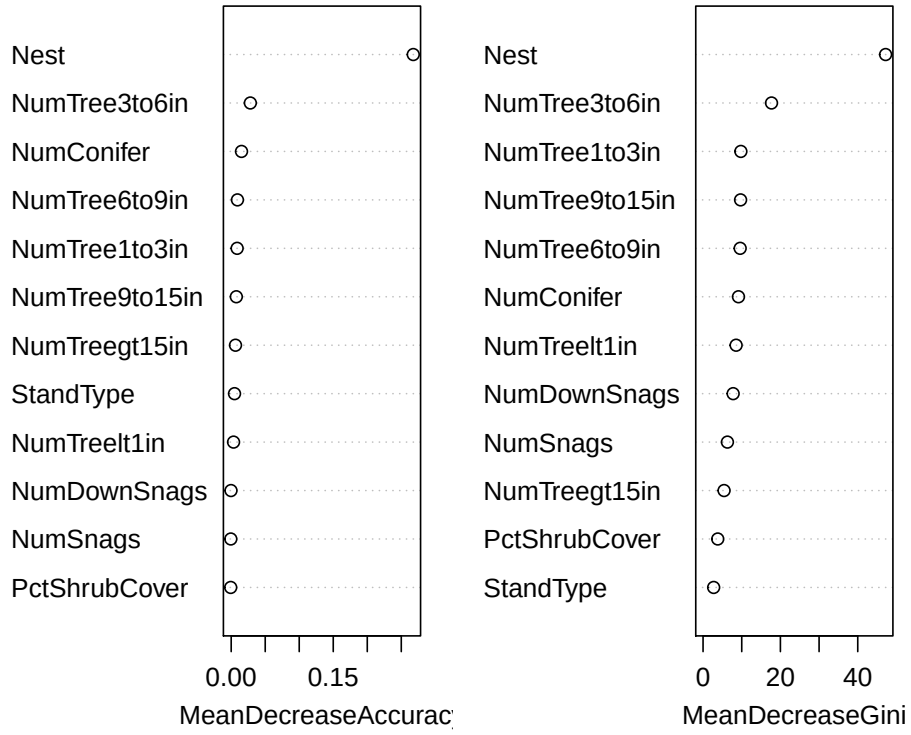
	Chickadee	Flicker	Non-nest	Sapsucker	class.error
Chickadee	23	1	0	18	0.4523810
Flicker	5	6	0	12	0.7391304
Non-nest	0	0	106	0	0.0000000
Sapsucker	14	8	0	20	0.5238095

```
[1] "Accuracy:"
```

```
[1] 72.18861
```

Answer: Right out of the box we obtain an accuracy of 72.19 which is an okay prediction rate but when we look at the confustion matrix it looks like we aren't doing very well classifying the different species. The model predicts Non-nest perfectly. But flicker has an error rate of 0.74, while Sapsucker has an error rate of 0.52 and Chickadee has an error rate of 0.45. So we are not able to classify the specific species very well.
When I ran this data through the classification tree in homework 3 I got an accuracy of 54.46%. So getting an accuracy rate of 72.19 is a huge improvement over the classification tree.

(b)) Use random forests to select a subset of the variables (which may be all the variables!) Refit random forests with only the important variables and obtain the out-of-bag confusion matrix. Did you observe any change in predictive accuracy?



Variable Selection: Based on the above plot I chose to use the Nest and NumTree3to6in variables. But after I ran this code I saw that the accuracy got worse. In the end I used Nest, NumTree3to6in, NumConifer and NumTree6to9in which ended up improving

```
          Chickadee Flicker Non-nest Sapsucker class.error
Chickadee        27       2        0        13   0.3571429
Flicker           6       8        0         9   0.6521739
Non-nest          0       0      106         0   0.0000000
Sapsucker        14       8        0        20   0.5238095
```

```
[1] "Accuracy:"
```

```
[1] 75.04669
```

Answer: Substting the data to only use Nest, NumTree3to6in, NumConifer and NumTree6to9in I was able to get the accuracy up to 75.05%. Which is an improvement from 72.19%

(c) Summarize your results for your analyses of the birds' nest data using classification trees and random forests.

Summary: Looking at the different results between classification trees and random forest on the nest data we are able to see that random forests help to improve our accuracy by 17.73% right out of the box without performing any variable selection. After variable selection in our random forest model we were able to increase the accuracy from 54.46% in our classification tree model up to 75.05%. Which ended up being an increase of 2.86% over our first random forest model that didn't have any variable selection.

(iv) This problem also continues your analyses of the Uintah Mountains cavity nesting birds' data.

(a) Apply random forests to all the data with nest as the response variable and obtain the out-of-bag confusion matrix. How well can we classify these data, and where are the major misclassifications? How do your results compare to the classification tree you fitted in Homework 3.

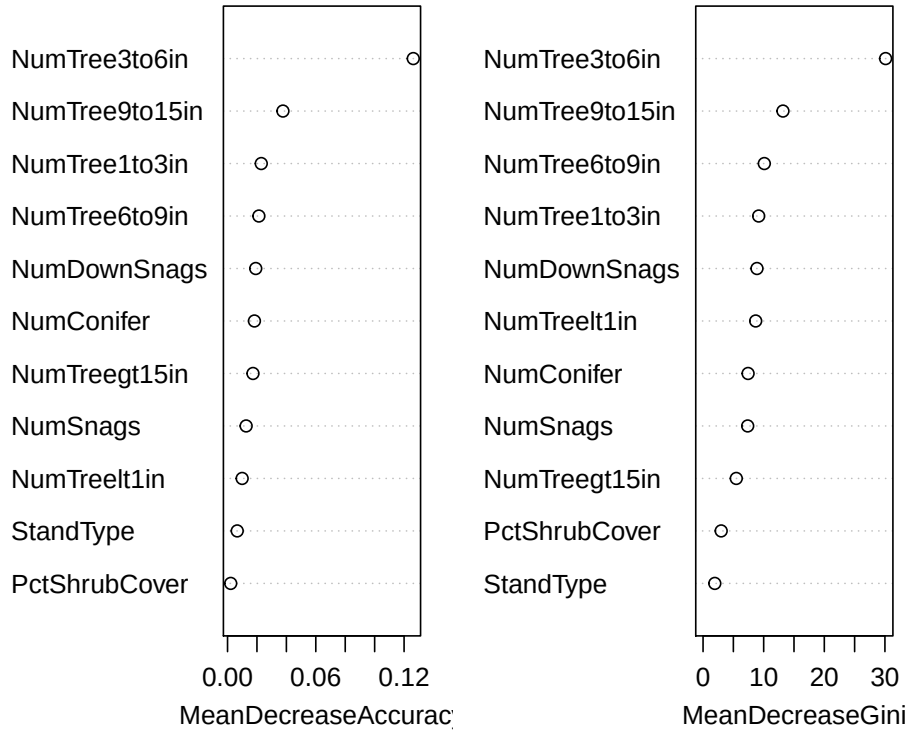
```
   0  1 class.error
0 87 19   0.1792453
1 16 91   0.1495327
```

```
[1] "Accuracy:"
```

```
[1] 83.43928
```

Answer: Once I removed the Species variable from the data I am able to classify this data with an 83.44% accuracy rate. The area where we are having the largest missclassification rate is where Nest equals 0, which has an error rate of 0.179. The values where Nest equal 1 perform a little bit better only having an error rate of 0.15.
Comparing our accuracy rate to that of the classification tree in Homework 3, the classification tree had an accuracy rate of 80.28%. While the random forest model had an accuracy of 83.44%. So in this instance the classification tree does perform much better than those in the past but random forest does still perform slightly better than the classification tree. With an improvement of 3.16%.

(b) Use random forests to select a subset of the variables (which may be all the variables!) Refit random forests with only the important variables and obtain the out-of-bag confusion matrix. Did you observe any change in predictive accuracy?




```
   0  1 class.error
0 93 13   0.1226415
1 17 90   0.1588785
```

```
[1] "Accuracy:"
```

```
[1] 85.80209
```

Variable Selection and Summary: After reviewing the Variable importance plot above and playing around with different combinations of variables. I settled on using the following variables for the nest model. NumTree3to6in, NumTree9to15in, NumTree1to3in, NumTree6to9in, NumDownSnags, NumConifer, NumTreegt15in, NumSnags, and NumTreelt1in. Essentially I used all of the variables except StandType and PctShrubCover. Using these variables I was able to increase the accuracy of the mdoel from 83.44% up to 85.8%. An increase of 2.36%.

(c) Now apply adaboost to the data and add the classification accuracies to those

you have previously obtained for classification trees and random forests.

```
    0  1
0 88 18
1 19 88
```

```
[,1]                              [,2]
"Percent Correctly Classified = " "82.63"
"Specificity = "                  "83.02"
"Sensitivity = "                  "82.24"
"Kappa ="                         "0.6526"
"AUC= "                           "0.873"
```

Summary:

- Ada Boost: Accuracy = 82.63%
- Random Forest (without Variable Selection): Accuracy = 83.44%
- Random Forest(With Variable Selection): Accuracy = 85.8%
- Classification Trees: Accuracy = 80.28%

Looking at the three methods they all seem to perform similarly in accuracy rates. Random Forest still performs the best of the three methods with and without variable selection. But they are all within a few percent of each other.

(d) Fit untuned and tuned gradient boosting machines to the data and compare the results to those previously obtained.

Untuned Gradient Boosting Machine:

```
    0  1
0 78 28
1 27 80
```

```
[,1]                              [,2]
"Percent Correctly Classified = " "74.18"
"Specificity = "                  "73.58"
"Sensitivity = "                  "74.77"
"Kappa ="                         "0.4835"
"AUC= "                           "0.8273"
```

Tuned Gradient Boosting Machine:

Tuning Parameters: After running the data through the tuning functions we got the following results to plug into the Gradient Booting Model.

- interaction.depth = 14
- n.trees = 75
- shrinkage = 0.1
- n.minobsinnode = 10

```
   0  1
0 90 16
1 17 90
```

```
[,1]                                [,2]
"Percent Correctly Classified = " "84.51"
"Specificity = "                  "84.91"
"Sensitivity = "                  "84.11"
"Kappa ="                         "0.6901"
"AUC= "                           "0.896"
```

Summary:

- GBM without Tuning: Accuracy = 74.18%
- GBM with Tuning: Accuracy = 84.51%
- Ada Boost: Accuracy = 82.63%
- Random Forest (without Variable Selection): Accuracy = 83.44%
- Random Forest(With Variable Selection): Accuracy = 85.8%
- Classification Trees: Accuracy = 80.28%

Adding the gradient boosting machines, Random Forest wtih variable selection still performs the best in this instance. But the tuned Gradient Boosting machine does move into second place, replacing Random Forests without any variable selection. Looking at GBM though without tuning it performs the worst out of all the models.

(e) Fit untuned and tuned support vector machines to the data and compare the results to those previously obtained.

Untuned Support Vector Machines

```
   0  1
0 90 16
1 13 94
```

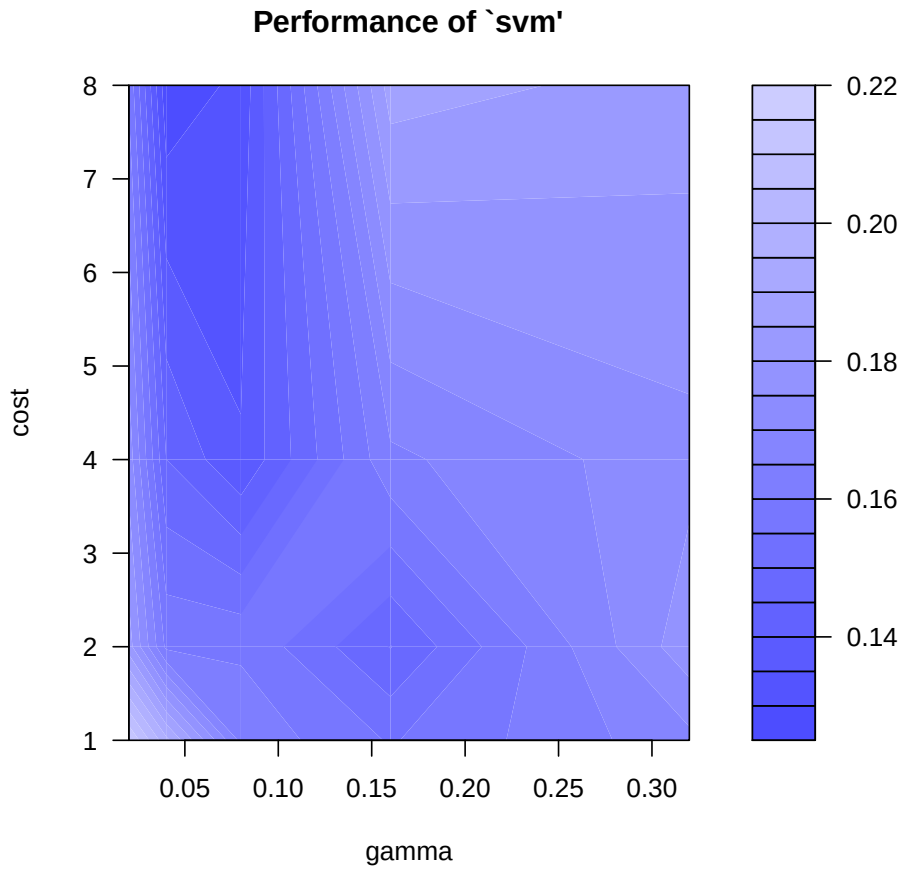
```
[,1]                                [,2]
"Percent Correctly Classified = " "86.38"
```

```
 "Specificity = "                  "84.91"
 "Sensitivity = "                  "87.85"
 "Kappa ="                         "0.7277"
 "AUC= "                           "0.9035"
```

Tuned Support Vector Machines

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
  0.04    8

- best performance: 0.1264069

- Detailed performance results:
   gamma cost     error dispersion
1   0.02    1 0.2151515 0.10539444
2   0.04    1 0.1965368 0.10267034
3   0.08    1 0.1636364 0.09896507
4   0.16    1 0.1545455 0.07650085
5   0.32    1 0.1686147 0.05844111
6   0.02    2 0.1822511 0.09350493
7   0.04    2 0.1588745 0.11276297
8   0.08    2 0.1590909 0.08594797
9   0.16    2 0.1448052 0.07323463
10  0.32    2 0.1781385 0.06474278
11  0.02    4 0.1729437 0.11026670
12  0.04    4 0.1450216 0.09760281
13  0.08    4 0.1354978 0.08029885
14  0.16    4 0.1638528 0.06579557
15  0.32    4 0.1733766 0.07615473
16  0.02    8 0.1590909 0.09433329
17  0.04    8 0.1264069 0.06947569
18  0.08    8 0.1313853 0.06208303
```

```
19   0.16     8 0.1874459 0.06512079
20   0.32     8 0.1826840 0.07408095
```



I have used the chart and information above to help me tune the Suport Vector Machine.

```
    0  1
 0 92 14
 1 11 96
```

```
[,1]                                [,2]
"Percent Correctly Classified = " "88.26"
"Specificity = "                  "86.79"
"Sensitivity = "                  "89.72"
"Kappa ="                         "0.7652"
"AUC= "                           "0.9049"
```

Summary:

- SVM without Tuning: Accuracy = 86.38%
- SVM with Tuning: Accuracy = 88.26%
- GBM without Tuning: Accuracy = 74.18%
- GBM with Tuning: Accuracy = 84.51%
- Ada Boost: Accuracy = 82.63%
- Random Forest (without Variable Selection): Accuracy = 83.44%
- Random Forest(With Variable Selection): Accuracy = 85.8%
- Classification Trees: Accuracy = 80.28%

In this instance the Support Vector Machine (SVM) performs best in classifying the nest data. It has a better accuracy than the Random Forest with variable selection, before and after tuning the model.

(f) Briefly discuss the results of all your analyses. Which method(s) did best on these data?

Summary:

- SVM without Tuning: Accuracy = 86.38%
- SVM with Tuning: Accuracy = 88.26%
- GBM without Tuning: Accuracy = 74.18%
- GBM with Tuning: Accuracy = 84.51%
- Ada Boost: Accuracy = 82.63%
- Random Forest (without Variable Selection): Accuracy = 83.44%
- Random Forest(With Variable Selection): Accuracy = 85.8%
- Classification Trees: Accuracy = 80.28%

Looking over all of the models they all perform in the 80% range after being tuned. Some of them are very close to one another in performance that under different conditions one may perform slightly better than the other. But it does appear in this instance that Support Vector Machines perform the best out of all the models. Before tuning and especialy after tuning the model. Out of the tuned models, Classification trees perform the worst out of all the models at 80.28% but SVM at its best is at 88.26%. so it is a significant improvement but it does not go above a 10% increase in the accuracy rate.