Name: Aryan Girish Kasliwal
SID: 55972222
EID: akasliwal2

CS3481: Fundamentals of Data Science- Assignment 2

1)

Construct random forest models using different numbers of component trees based on the default training set/test set partition and analyze the resulting change in classification performance. (25%)

While constructing the random forest models, the number of component trees is decided by the parameter 'n_estimators'.

For the first question, the code was looped through different parameter values for n_estimators ranging from 1-120 and random_state ranging from 1-50 in the function RandomForestClassifier. The most optimal prediction value for the random forest was when:

```
clf = RandomForestClassifier(n_estimators=7, random_state=8)
```

When the parameters are set as above, the accuracy of the random forest model is 85.23076923076923 which is the highest observed through the different values. The table below has the top accuracy values for various component tree values.

No. component trees	Accuracy
7	85.23076923076923
10	85.23076923076923
6	84.92307692307692
8	84.92307692307692
9	84.3076923076923
12	84.3076923076923
11	84.0
5	83.6923076923077
14	83.38461538461539
15	83.38461538461539

An observation that can be made is, when the number of component trees range between 5-15, the accuracy of the random tree model is consistently high and then it gradually starts dipping below the average. Further, when component trees range between 1-4, the values are below average. From this we can infer that with less trees, the model isn't trained accurately and hence its accuracy is low, this is called under-fitting. Further, even though the accuracy starts increasing as component trees increase, it flatlines after a limit, this is when the model is most accurately trained. A further observation to this is that accuracy eventually reduces too due to over-fitting.

The table below has the lowest accuracy values for various number of component trees.

No. component trees	**Accuracy**
2	78.46153846153847
1	79.07692307692308
3	80.92307692307692
4	81.53846153846153
40	81.84615384615384
70	81.84615384615384
98	81.84615384615384
99	81.84615384615384
34	82.15384615384616
36	82.15384615384616

Further observations can be made from the different values in the table above. The lowest accuracy was 78.46153846153847 when the parameter n_estimators was set to 2. Amongst other low values are when component tree is set to 1, 3, 4, 98, 99. This supports our inference from before that the random forests generated require a minimum number of trees to classify accurately or they tend to be under-fitted and after a limit they tend to be over-fitted.

In most of the observed cases, the accuracy does not go much above 83% irrespective of how high we keep the n_estimators value. The average accuracy observed when the number of component trees was varied from 1-120 was 82.55167055167041.

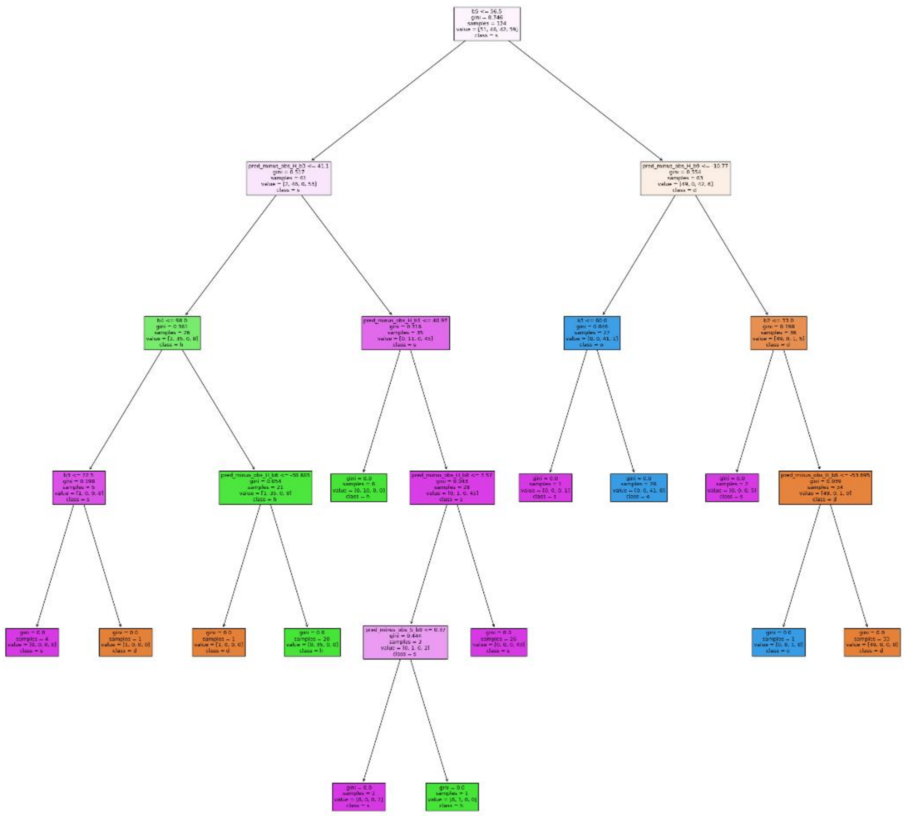
---

2)

For the random forest model corresponding to the best classification performance, select different component decision trees in the model and compare the classification performances of these trees with that of the original random forest model. (25%)

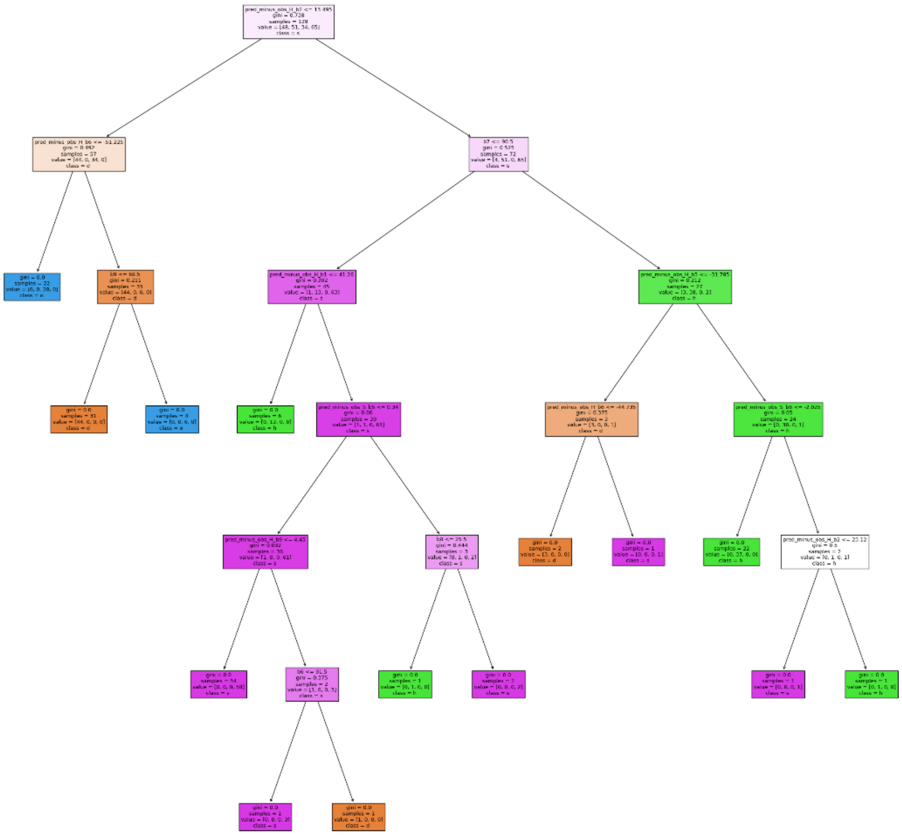
From part (1) we know that the highest accuracy is obtained when n_estimators = 7 and random_state = 7. For the random_forest hence generated, we can obtain all the 7 different decision trees to find out interesting characteristics about each tree in the forest. The following are the 7 different trees and their comparison to the performance of the forest:

Decision tree - 1:



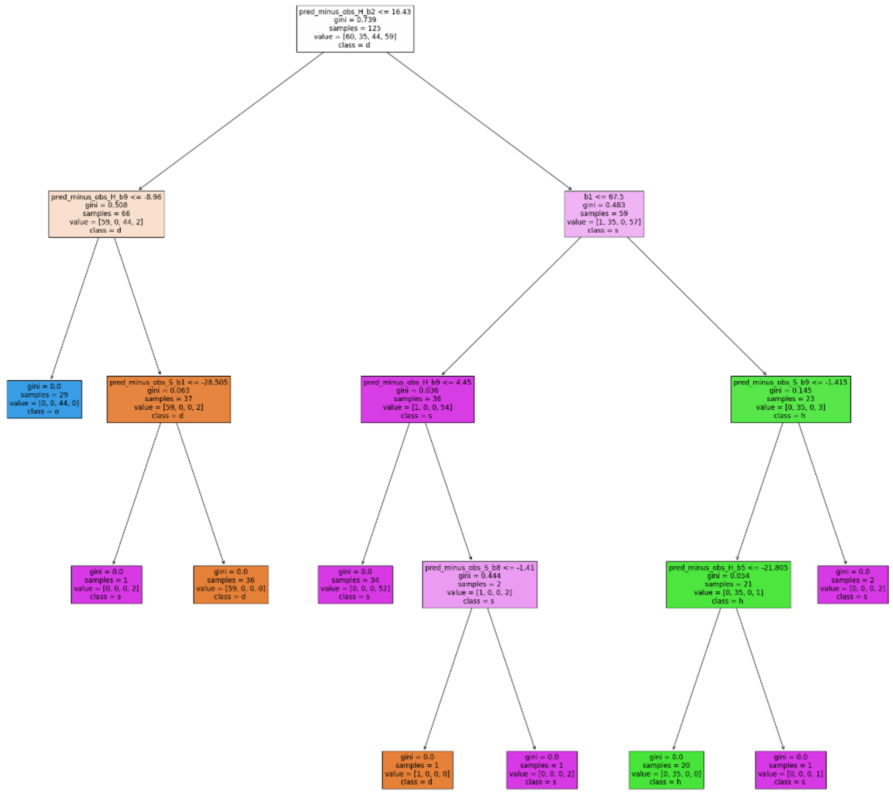
The classification accuracy of this tree is 79.07692307692308, this value is lesser when compared to the accuracy of the random forest to which this belongs.

Decision tree - 2:



The classification accuracy of this tree is 75.38461538461539, this value is lesser when compared to the accuracy of the random forest to which this belongs.

Decision tree - 3:

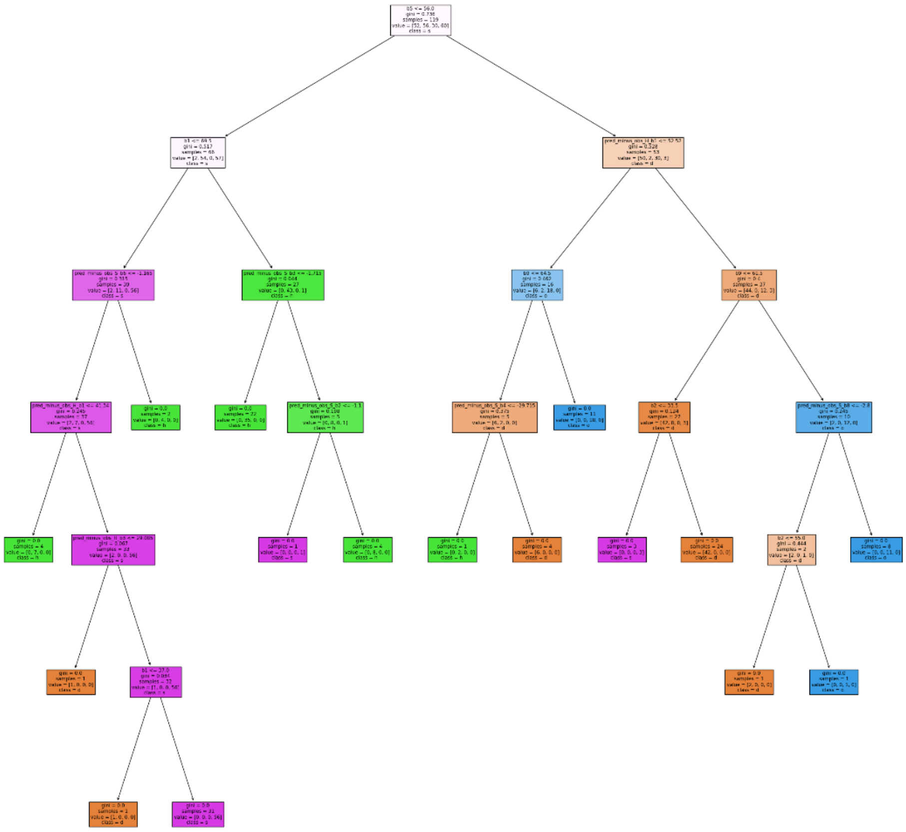


The classification accuracy of this tree is 78.46153846153847, this value is lesser when compared to the accuracy of the random forest to which this belongs.

Decision tree - 4:

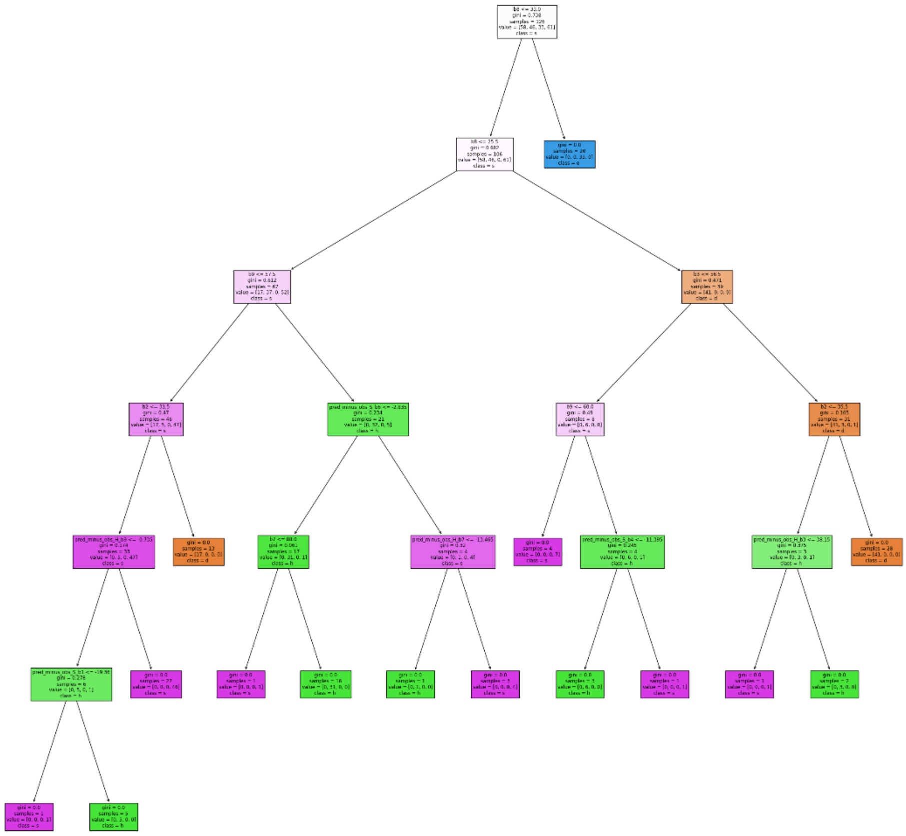
The classification accuracy of this tree is 81.23076923076923, this value is lesser when compared to the accuracy of the random forest to which this belongs.

Decision tree - 5:



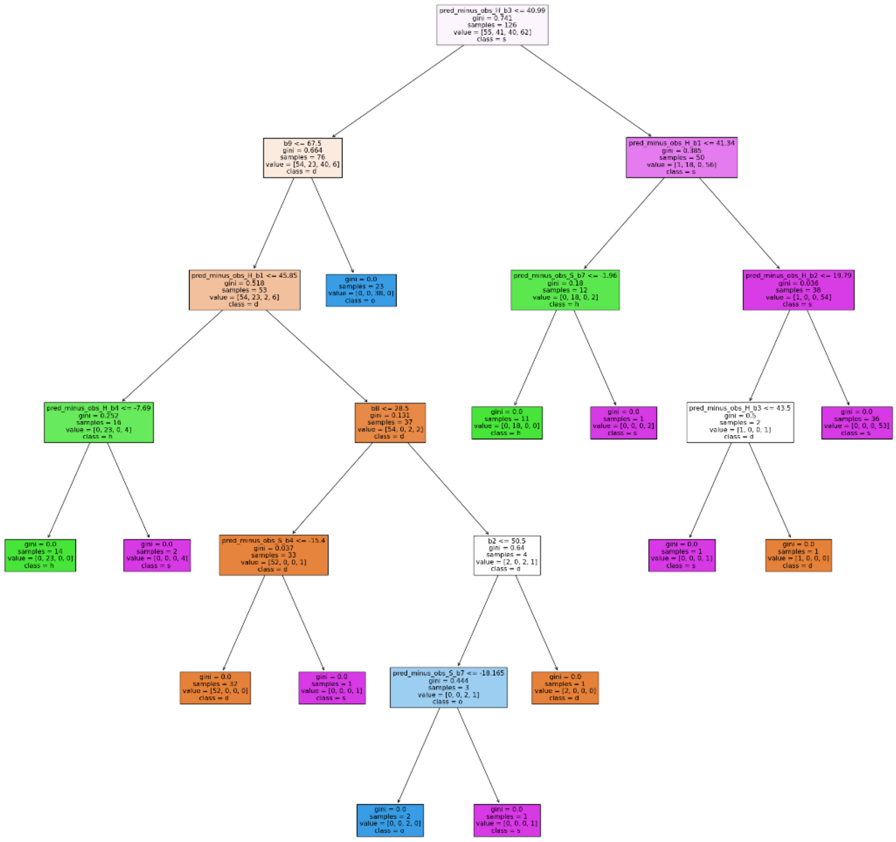
The classification accuracy of this tree is 81.84615384615384, this value is lesser when compared to the accuracy of the random forest to which this belongs.

Decision tree - 6:



The classification accuracy of this tree is 79.6923076923077, this value is lesser when compared to the accuracy of the random forest to which this belongs.

Decision tree - 7:



The classification accuracy of this tree is 73.846153846153884, this value is lesser when compared to the accuracy of the random forest to which this belongs.

Classifier	Accuracy
Random forest	85.23076923076923
Decision tree 1	79.07692307692308
Decision tree 2	75.38461538461539
Decision tree 3	78.46153846153847
Decision tree 4	81.23076923076923
Decision tree 5	81.84615384615384
Decision tree 6	79.6923076923077
Decision tree 7	73.84615384615384

The table above lists the accuracies of the random forest and all its 7 decision trees. It can be consistently observed that all the trees have an accuracy lower than the random forest. This is because the random forest classifies any data point based on maximum votes and these votes are cast based on the results of these decision trees. For example, if a data point is classified as forest type ‘d’ but decision tree 1, ‘s’ by tree 2, ‘d’ by tree 3, ‘s’ by tree 4, ‘o’ by tree 5, ‘h’ by tree 6, ‘d’ by tree 7, the random forest then aggregates these results and classifies the data point as ‘d’ because majority of the decision trees classify the point as type ‘d’. By this aggregation, even if some trees predict wrong values, the overall prediction of the forest type can still be more accurate than any individual tree at once. Hence the random forest generated by the seven trees above is more accurate than the trees themselves.

---

3)

For a random forest classifier (or one of its component trees), the relative importance of the attributes can be measured through the `feature_importances_` field of the classifier. For selected component trees in (b), compare their associated lists of relative attribute importance values. (25%)

The relative importance of different features for all the 7 trees in section (2) are below:

Decision tree - 1:

Feature name	**Feature importance**
b5	0.2830185757680727
pred_minus_obs_H_b9	0.27670444764470664
pred_minus_obs_H_b3	0.12359109424577443
pred_minus_obs_H_b1	0.10645529849221387
b4	0.08493947062444432
b2	0.06034881335066
pred_minus_obs_H_b6	0.02643731618904317
b3	0.02525721320605391
pred_minus_obs_S_b8	0.009028110252376719
pred_minus_obs_H_b8	0.004219660226654325

The features are listed in the table above in descending order. From the table we can infer that feature ‘b5’ has the highest importance. All other features not mentioned in the tables have zero importance.

Decision tree - 2:

Feature name	Feature importance
pred_minus_obs_H_b2	0.3129615836822421
b7	0.2096555640545302
pred_minus_obs_H_b6	0.2006085110684322
pred_minus_obs_H_b1	0.13241351635431448
b9	0.07230375544643476

Feature name	Feature importance
pred_minus_obs_H_b5	0.03880548799943981
b6	0.010270419807732209
b8	0.009129262051317519
pred_minus_obs_S_b6	0.006495821074975917
pred_minus_obs_S_b5	0.004153474434513834
pred_minus_obs_H_b9	0.0032026040260670343

The features are listed in the table above in descending order. From the table we can infer that feature 'pred_minus_obs_H_b2' has the highest importance. All other features not mentioned in the tables have zero importance.

Decision tree - 3:

Feature name	Feature importance
pred_minus_obs_H_b9	0.34273565604093675
pred_minus_obs_H_b2	0.32828798072287324
b1	0.25563048542384853
pred_minus_obs_S_b1	0.026449581758346996

Feature name	Feature importance
pred_minus_obs_S_b9	0.024487622619839437
pred_minus_obs_H_b5	0.013293280850769978
pred_minus_obs_S_b8	0.009115392583385125

The features are listed in the table above in descending order. From the table we can infer that feature 'pred_minus_obs_H_b9' has the highest importance. All other features not mentioned in the tables have zero importance.

Decision tree - 4:

Feature name	Feature importance
pred_minus_obs_H_b9	0.4867317991627098
b2	0.3650483469896758
b4	0.05896483230265763
pred_minus_obs_H_b1	0.026648235718000856

Feature name	Feature importance
pred_minus_obs_H_b4	0.02331720625325076
b9	0.02296753519642943
pred_minus_obs_S_b3	0.016322044377275535

The features are listed in the table above in descending order. From the table we can infer that feature 'pred_minus_obs_H_b9' has the highest importance. All other features not mentioned in the tables have zero importance.

Decision tree - 5:

Feature name	Feature importance
b5	0.2917017940351189
b1	0.2516043020857641
b9	0.16164774405007973
pred_minus_obs_H_b1	0.14632018255900617
b2	0.04756097560975611

Feature name	Feature importance
pred_minus_obs_S_b6	0.039791173831470766
pred_minus_obs_S_b4	0.02057926829268293
pred_minus_obs_S_b8	0.014372822299651575
pred_minus_obs_H_b3	0.013014032136691582
pred_minus_obs_S_b2	0.012195121951219513
pred_minus_obs_S_b3	0.0012125831485587566

The features are listed in the table above in descending order. From the table we can infer that feature 'b5' has the highest importance. All other features not mentioned in the tables have zero importance.

Decision tree - 6:

Feature name	Feature importance
b8	0.36580444916996213
b2	0.20077026948559015
b9	0.19796221667284758
b3	0.09233252623083132
pred_minus_obs_H_b9	0.05048126805752868

Feature name	Feature importance
pred_minus_obs_S_b6	0.03500060290628872
b7	0.013267794148163519
pred_minus_obs_S_b4	0.011739246435241458
pred_minus_obs_S_b1	0.011413156256484747
pred_minus_obs_H_b7	0.010956630006225355
pred_minus_obs_H_b3	0.010271840630836273

The features are listed in the table above in descending order. From the table we can infer that feature 'b8' has the highest importance. All other features not mentioned in the tables have zero importance.

Decision tree - 7:

Feature name	Feature importance
pred_minus_obs_H_b1	0.360430842608571
b9	0.2568606480447678
pred_minus_obs_H_b3	0.2534663293077482
pred_minus_obs_H_b4	0.046442256946834626
pred_minus_obs_S_b7	0.03362015557238246

Feature name	Feature importance
b8	0.016518918690385128
pred_minus_obs_S_b4	0.013372626897734973
b2	0.012721139946306876
pred_minus_obs_H_b2	0.006567081985268804

The features are listed in the table above in descending order. From the table we can infer that feature 'pred_minus_obs_H_b1' has the highest importance. All other features not mentioned in the tables have zero importance.

The aggregate of all feature importance for various component decision trees are listed in descending order in the table below:

Feature name	Feature importance
pred_minus_obs_H_b9	1.1598557749319491
pred_minus_obs_H_b1	0.7722680757321063
b9	0.7117418994105592
b2	0.686449545381989
pred_minus_obs_H_b2	0.6478166463903842
b5	0.5747203698031916
b1	0.5072347875096126
pred_minus_obs_H_b3	0.4003432963210504
b8	0.39145262991166474
pred_minus_obs_H_b6	0.22704582725747538
b7	0.2229233582026937
b4	0.14390430292710193
b3	0.11758973943688524

Feature name	Feature importance
pred_minus_obs_S_b6	0.08128759781273541
pred_minus_obs_H_b4	0.06975946320008539
pred_minus_obs_H_b5	0.052098768850209795
pred_minus_obs_S_b4	0.04569114162565936
pred_minus_obs_S_b1	0.03786273801483175
pred_minus_obs_S_b7	0.03362015557238245
pred_minus_obs_S_b8	0.032516325135413415
pred_minus_obs_S_b9	0.02448762261983944
pred_minus_obs_S_b3	0.01753462752583429
pred_minus_obs_S_b2	0.012195121951219511
pred_minus_obs_H_b7	0.010956630006225355
b6	0.010270419807732209
pred_minus_obs_H_b8	0.004219660226654325
pred_minus_obs_S_b5	0.004153474434513834

The table was made by just adding up the feature importance of all the component trees. And from this table we can infer that pred_minus_obs_H_b9 has the highest feature importance in all tables combined.

4)

Construct a naïve Bayes classifier model based on our data set and compare the classification performance with that of the random forest model. (25%)

```
nb = GaussianNB()
train_Y = training_data["class"]
train_X = training_data.drop(["class"], axis=1)
test_X = testing_data.drop(["class"], axis=1)
nb.fit(train_X, train_Y)
predicted_Y = nb.predict(test_X)
```

The code above was used to construct a naïve Bayes classifier. It resulted in an accuracy of 80.3076923076923, which is lesser when compared to the random forest classifier. However, an interesting comparison of accuracy is that the average accuracy of the component trees of the random forest is 78.5054571 which is lesser than that of the naïve Bayes classifier. This is interesting because this shows that the strongest component tree is alone still not accurate enough, instead a voting system based on these trees leads to a higher accuracy when compared to naïve Bayes classifier or the individual component trees.