CS B551: Elements of Artificial Intelligence

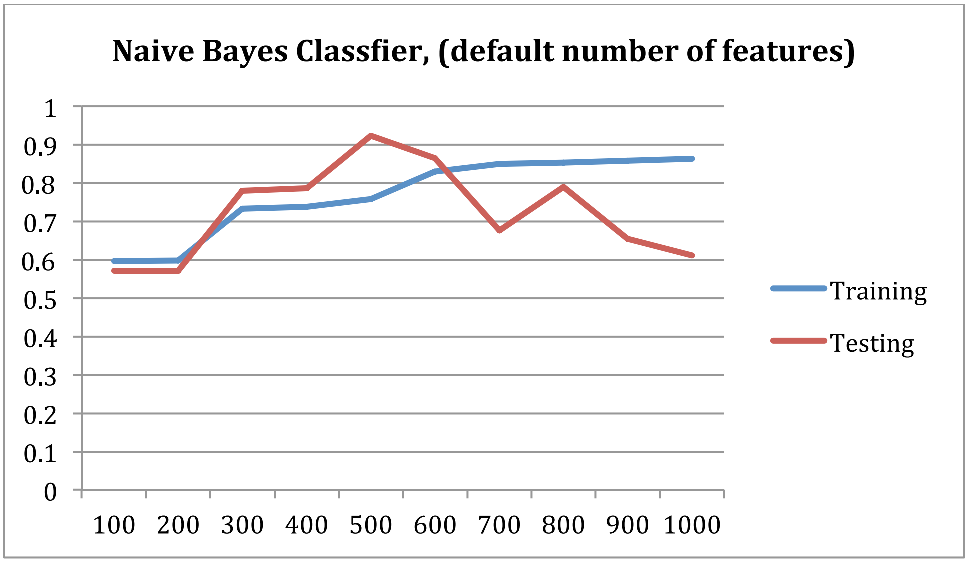
Enrique Areyan

Homework 6

Answers to Written Questions:

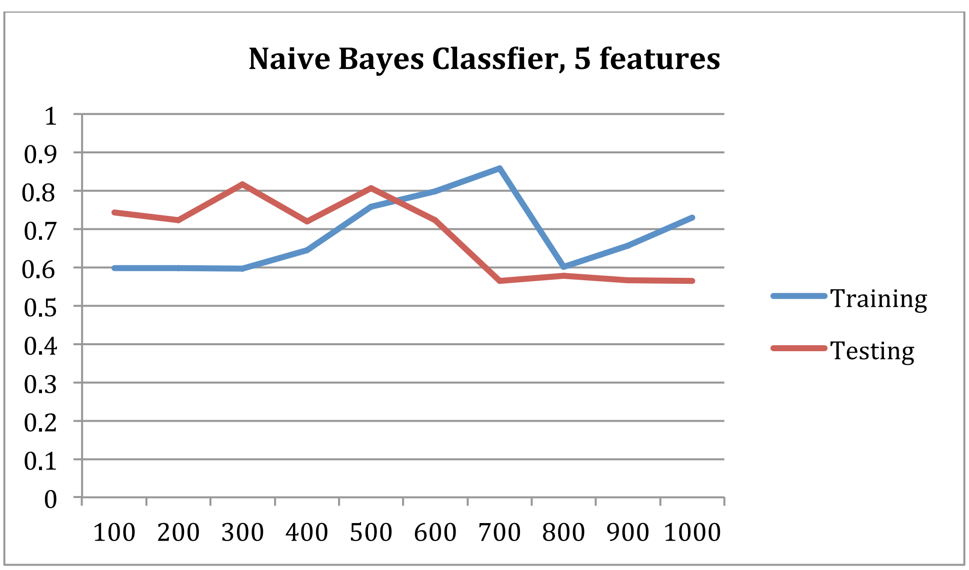
I. Naive Bayes Classifier.

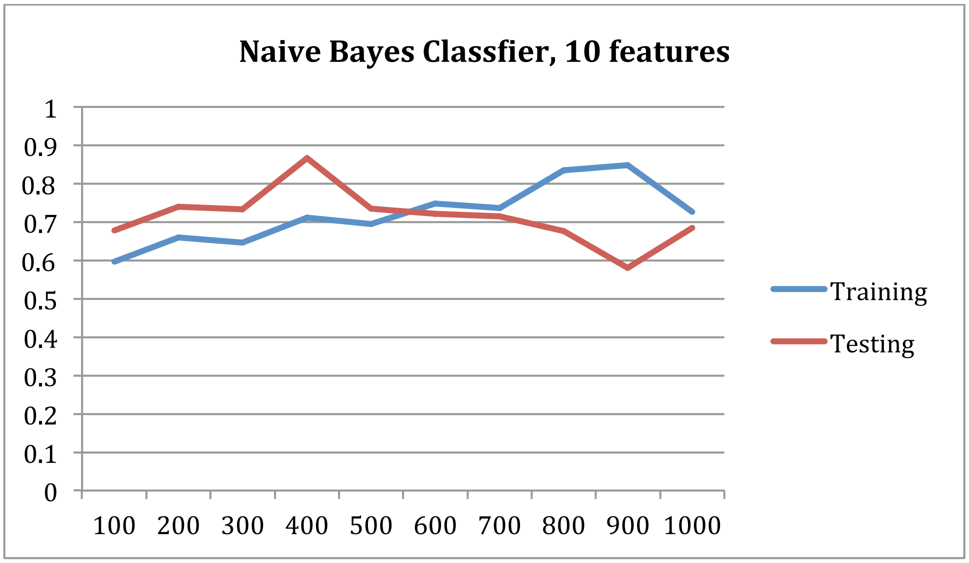
I.2

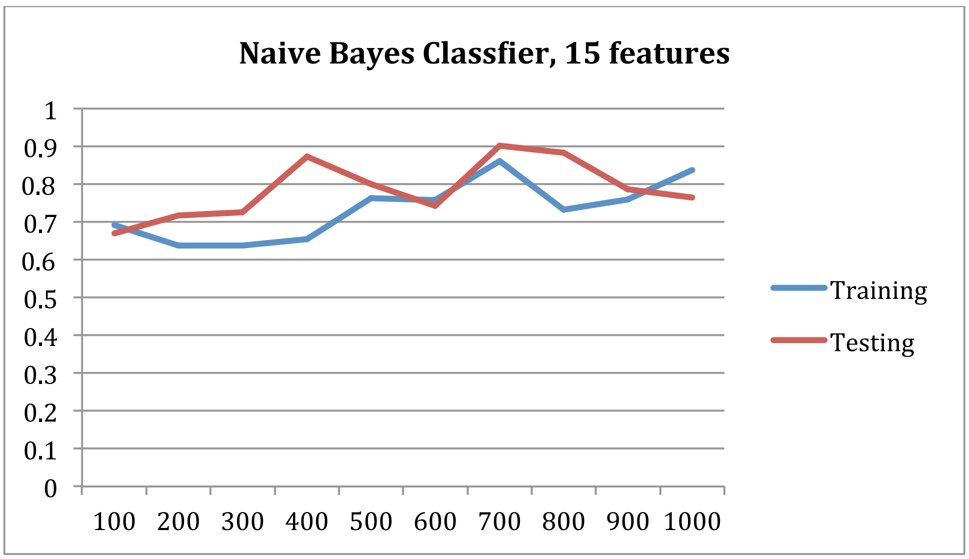


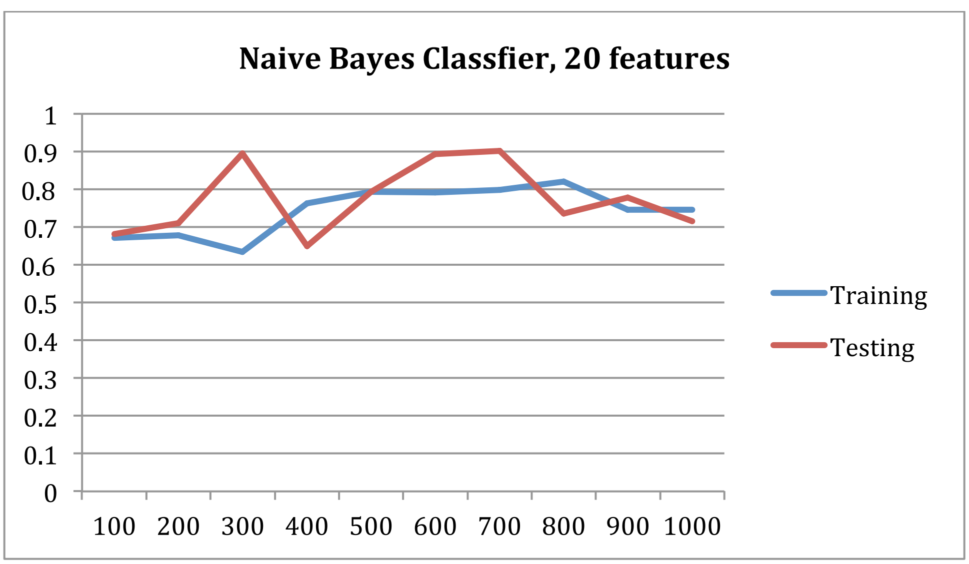
On the one hand, there is a general trend towards improvement of the training accuracy (blue line) as the number of training examples increases. The accuracy improves faster at the beginning and then reaches an upper bound around 0.9.

On the other hand, the test accuracy (red line) improves into a good classifier when using around 500 examples, but then drops its performance. There is a trend towards diminish the accuracy of the classifier on the test sample as the sample size increases. This is a good example of over fitting.









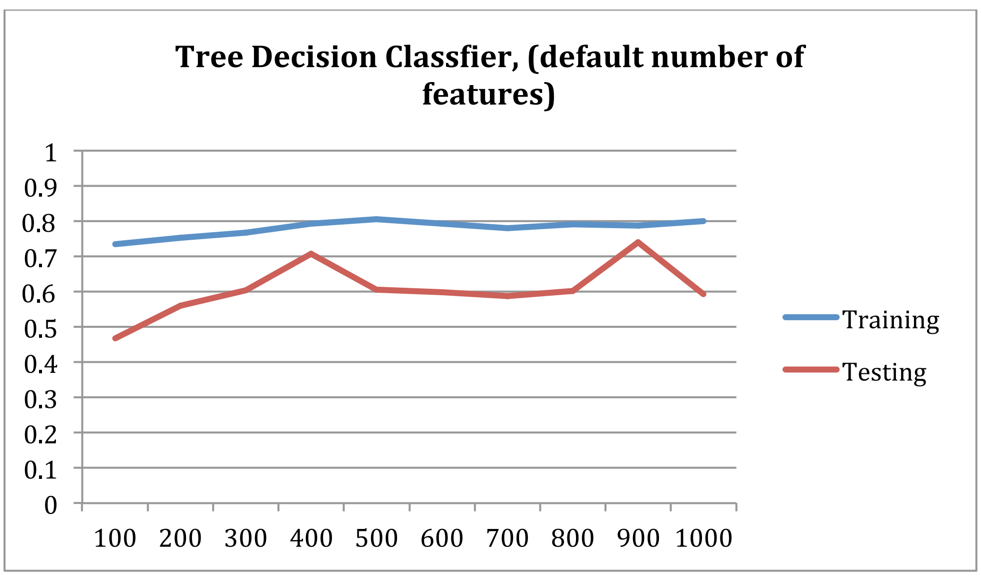
I.3 All the graphs show an upward trend on the training set relatively to the number of training examples, except for some random noise. For the NB classifier, there seems to be an improvement when using more features, with the highest peak obtained when using the default number of features (100). Still, the overall tendency with the testing accuracy is an improvement at first, up to a peak point, and then a decline. Interestingly, in most graphs both lines seem to cross at a point in which the test accuracy diminishes and the training improves.

The following data was produced with the classifiers and used in the above graphs.

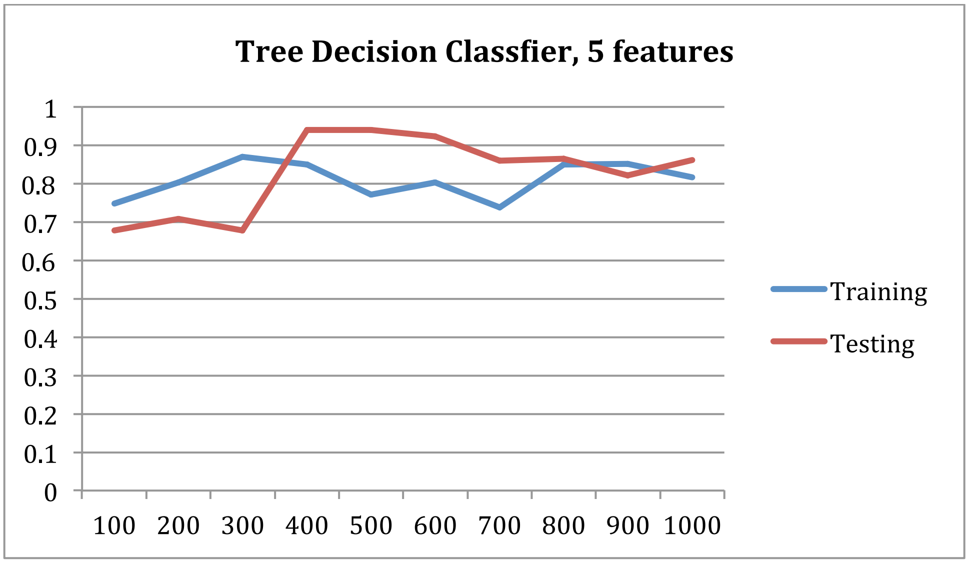
Naive Bayes Classifier

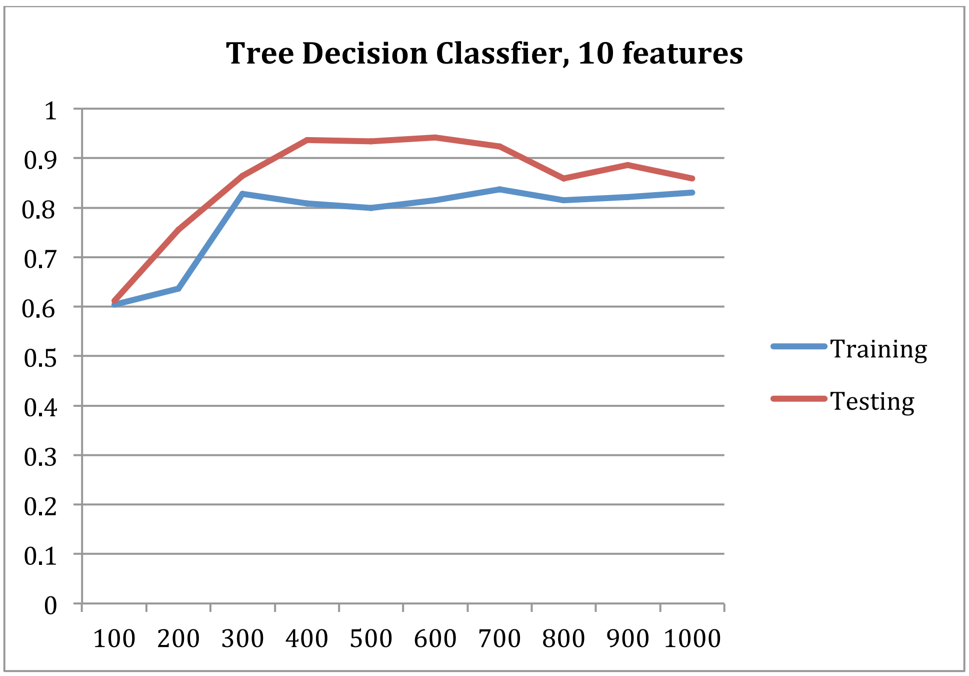
Training Set	Features Default Number									Average
100	200	300	400	500	600	700	800	900	1000	
0.596639	0.597689	0.732353	0.738445	0.757878	0.830462	0.85021	0.853992	0.857773	0.862605	0.7678046
Test Set	Features Default Number									
100	200	300	400	500	600	700	800	900	1000	
0.571057	0.572098	0.779282	0.786049	0.922436	0.865695	0.67569	0.789172	0.654867	0.611661	0.7228007
Training Set		Features = 5								
100	200	300	400	500	600	700	800	900	1000	
0.597689	0.598109	0.596639	0.643908	0.758613	0.798319	0.859034	0.601681	0.656723	0.730042	0.6840757
Test Set		Features = 5								
100	200	300	400	500	600	700	800	900	1000	
0.743883	0.72254	0.816762	0.719938	0.806351	0.723581	0.564289	0.577303	0.566372	0.564289	0.6805308
Training Set		Features = 10								
100	200	300	400	500	600	700	800	900	1000	
0.596639	0.659454	0.645588	0.711345	0.695378	0.747689	0.737185	0.834454	0.848739	0.726261	0.7202732
Test Set		Features = 10								
100	200	300	400	500	600	700	800	900	1000	
0.678293	0.739198	0.733472	0.866736	0.735554	0.720979	0.715252	0.67621	0.580427	0.68506	0.7131181
Training Set		Features = 15								
100	200	300	400	500	600	700	800	900	1000	
0.691176	0.636765	0.637815	0.654832	0.763655	0.757773	0.861345	0.732563	0.759664	0.838235	0.7333823
Test Set		Features = 15								
100	200	300	400	500	600	700	800	900	1000	
0.669443	0.717855	0.725664	0.872462	0.800104	0.742322	0.901093	0.883394	0.786569	0.764185	0.7863091
Training Set		Features = 20								
100	200	300	400	500	600	700	800	900	1000	
0.671008	0.678782	0.634244	0.762815	0.793277	0.791387	0.798319	0.820588	0.745168	0.745168	0.7440756
Test Set		Features = 20								
100	200	300	400	500	600	700	800	900	1000	
0.681936	0.711088	0.895367	0.649662	0.793337	0.893805	0.901093	0.736596	0.77824	0.714732	0.7755856

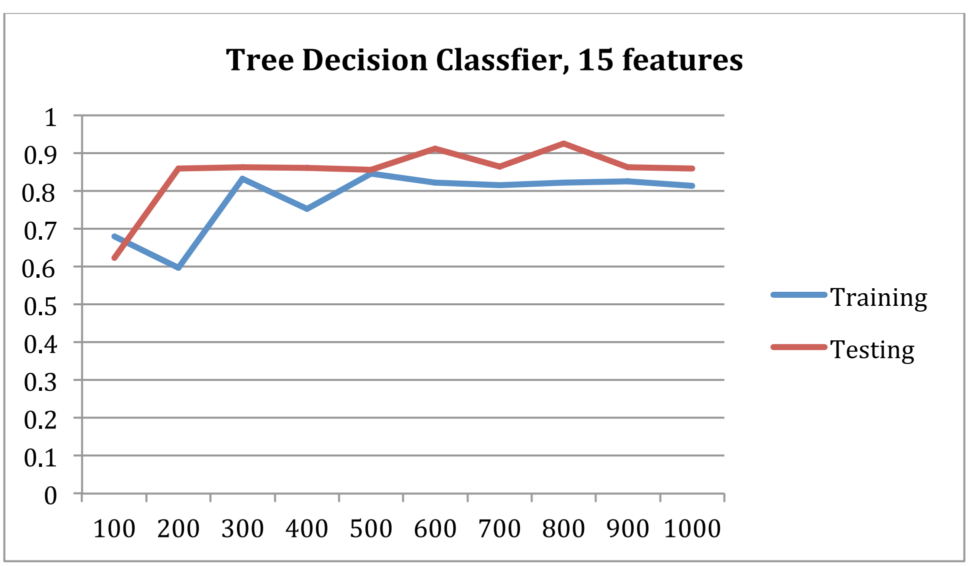
II. Decision Tree Classifier.

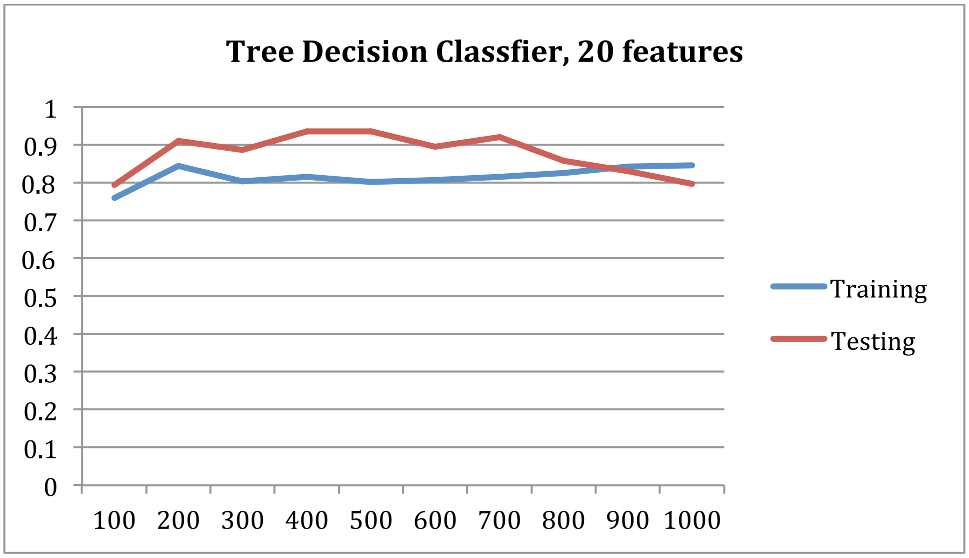


II.2. There is a similar situation as that found in I.2, however, it is less obvious to see it. The training accuracy improves with the size of the training samples. The testing accuracy slowly improves and ends with a reduced trend. One needs to keep in mind that the default number of features is 100, which could account for the fact that the tree may be over fitting the data. As we will see, trees with fewer features tend to be more accurate above a certain number of features.


Tree Decision Classfier, 5 features
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100 200 300 400 500 600 700 800 900 1000
Training
Testing


Tree Decision Classfier, 10 features
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100 200 300 400 500 600 700 800 900 1000
Training
Testing


Tree Decision Classfier, 15 features
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100 200 300 400 500 600 700 800 900 1000
Training
Testing



II.3 Again, as the training accuracy increases the testing accuracy diminishes, and both cross at some point in which the classifier can be thought of as over fitting the data. Unlike the NB classifier, the Tree classifier reaches its best accuracy with a limited number of features, between 10 and 20. One can see that if the classifier is given too many features, then over fitting occurs more rapidly and one cannot obtain as good results as with fewer features.

Data use for the Tree Classifier.

Tree Decision Classifier

Training Set	Features Default Number									Average
100	200	300	400	500	600	700	800	900	1000	
0.733824	0.753571	0.767857	0.792437	0.806303	0.792017	0.779412	0.790756	0.787605	0.79937	0.7803152
Test Set	Features Default Number									
100	200	300	400	500	600	700	800	900	1000	
0.466944	0.560125	0.603332	0.707444	0.605934	0.598126	0.587715	0.60229	0.74076	0.59292	0.606559
Training Set		Features = 5								
100	200	300	400	500	600	700	800	900	1000	
0.748529	0.802941	0.869958	0.84958	0.771429	0.803782	0.737395	0.84958	0.851261	0.815966	0.8100421
Test Set		Features = 5								
100	200	300	400	500	600	700	800	900	1000	
0.678293	0.708485	0.677251	0.940135	0.940135	0.922957	0.859448	0.864654	0.821968	0.862051	0.8275377
Training Set		Features = 10								
100	200	300	400	500	600	700	800	900	1000	
0.603992	0.635924	0.828361	0.809034	0.798739	0.814286	0.837185	0.815126	0.821218	0.830882	0.7794747
Test Set		Features = 10								
100	200	300	400	500	600	700	800	900	1000	
0.611661	0.755336	0.864654	0.937012	0.933368	0.941697	0.923998	0.859448	0.885997	0.859448	0.8572619
Training Set		Features = 15								
100	200	300	400	500	600	700	800	900	1000	
0.679202	0.596218	0.832353	0.752731	0.845798	0.822059	0.815966	0.821639	0.82605	0.814286	0.7806302
Test Set		Features = 15								
100	200	300	400	500	600	700	800	900	1000	
0.623113	0.859448	0.863092	0.860489	0.856325	0.912025	0.864654	0.92556	0.863613	0.859448	0.8487767
Training Set		Features = 20								
100	200	300	400	500	600	700	800	900	1000	
0.759244	0.844958	0.803151	0.815966	0.801261	0.806303	0.816176	0.82563	0.841807	0.845168	0.8159664
Test Set		Features = 20								
100	200	300	400	500	600	700	800	900	1000	
0.793857	0.909943	0.887038	0.93545	0.936491	0.894326	0.920354	0.858407	0.830297	0.797501	0.8763664

III. Describe your new technique and your rationale for choosing it.

My classifier uses both the NB and DT Classifier previously constructed. First, I train both classifiers in the same way I trained them before. Then, I test each one separately. The prediction of my classifier follows simple rules: If the prediction of both NB and DT agree, return that prediction. If not, break the tie randomly and proportionally to the performance of NB and DT so far.

The proportionality is obtained by drawing a number from the uniform distribution between 0 and 1. At the beginning of the testing, both classifiers have the same chance of being selected to break a tie. However, if a classifier is selected and it predicts incorrectly, then it is penalized by lowering (by some fix percentage) the probability of being selected for future tiebreakers.

The rationale for using this classifier is that if both NB and DT agree on the classification, then maybe there is a good chance that the classification is correct. If they disagree, then we do not have more options than to choose one at random. However, my classifier will incrementally penalize that classifier which is giving bad results, and thus indirectly reward the other classifier. In this way, my classifier can “adapt” in real time and rely more on the classifier that is giving the best results.

Note: for my classifier I filter the feature set used in the tree classifier. I only use those features that are more than 2 characters long, avoiding features such as EXISTS_In or EXISTS_1, which I believe present little information for the classification task. I also truncate the amount of features to be used by the tree classifier to a maximum of 20 (each of which is more than 2 characters long).