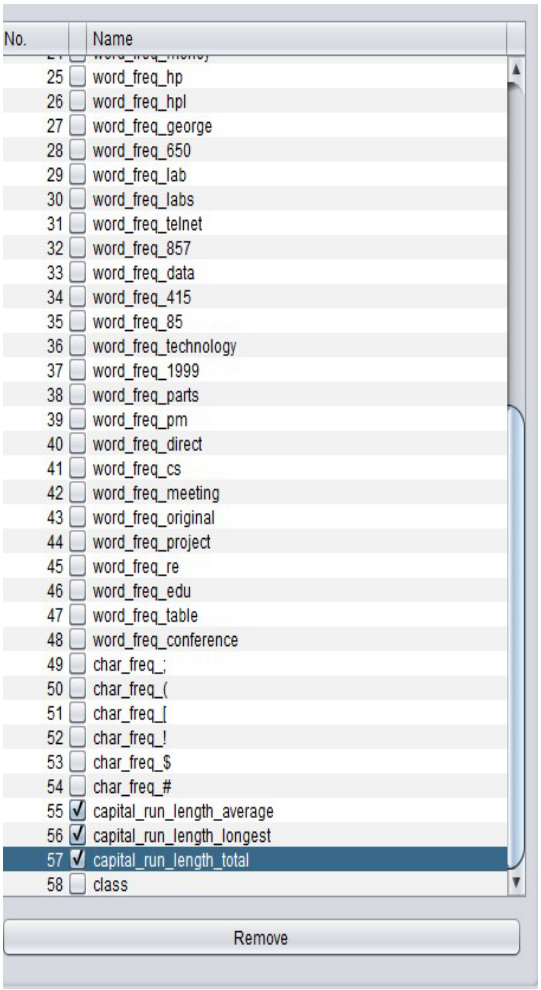
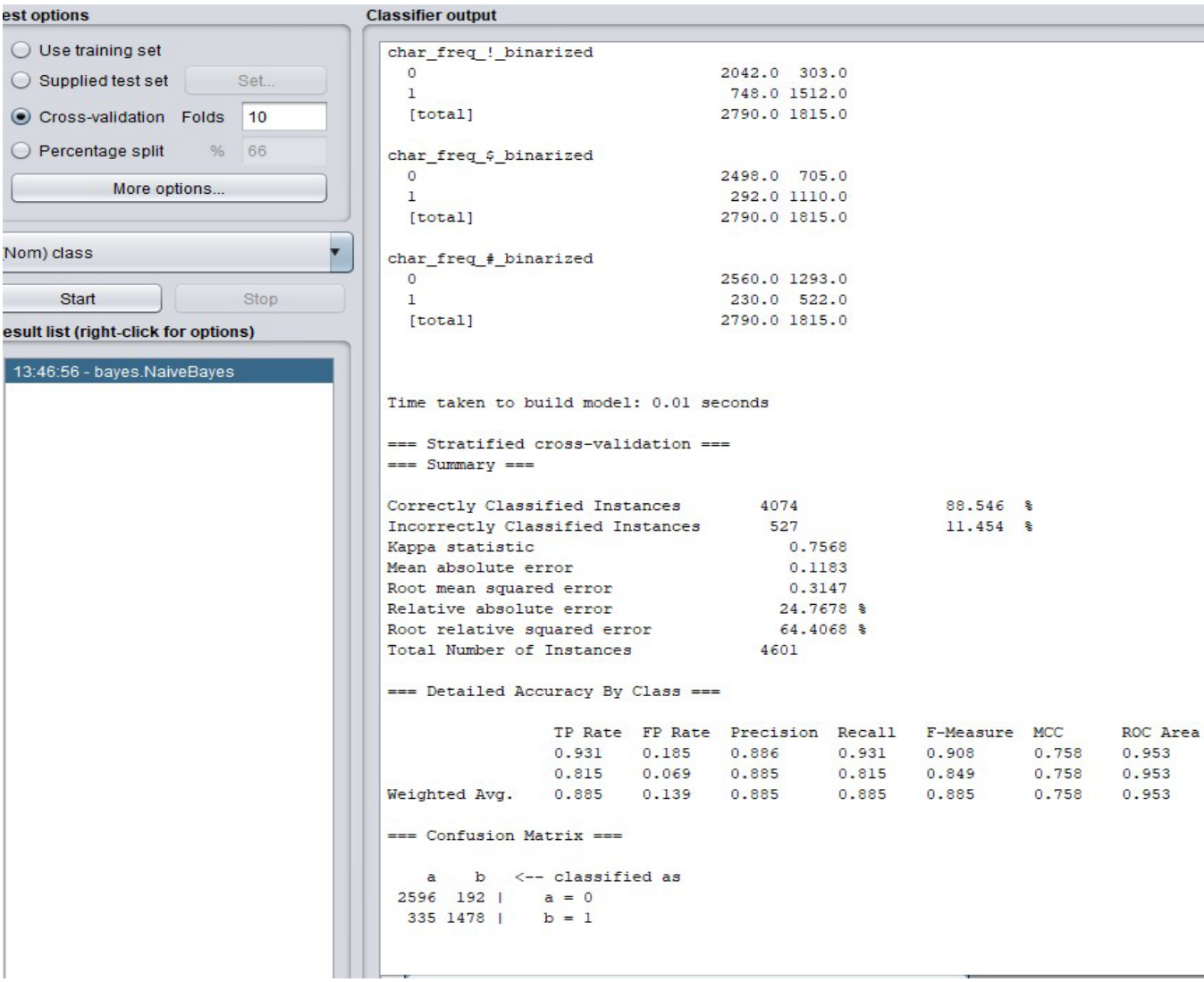
The Screenshots are as Follows:



Relation: spambase-weka.filters.unsupervised.attribute.Remove-R55-57-weka.filters.unsupervised.attribute.NumericToBinary-Rfirst-last

No.	1: word_freq_make_binarized Nominal	2: word_freq_address_binarized Nominal	3: word_freq_all_binarized Nominal	4: word_freq_3d_binarized Nominal	5: word_freq_our_binarized Nominal	6: word_freq_over_bina Nominal
1	0	1	1	0	1	0
2	1	1	1	0	1	1
3	1	0	1	0	1	1
4	0	0	0	0	1	0
5	0	0	0	0	1	0
6	0	0	0	0	1	0
7	0	0	0	0	1	0
8	0	0	0	0	1	0
9	1	0	1	0	1	0
10	1	1	1	0	1	1
11	0	0	0	0	0	0
12	0	0	1	0	1	1
13	0	1	1	0	1	0
14	0	0	0	0	1	0
15	0	0	1	0	1	1
16	0	1	1	0	1	0
17	0	0	0	0	1	0
18	0	0	0	0	0	0
19	0	0	1	0	1	0
20	0	1	0	0	1	1
21	0	0	0	0	0	0
22	1	1	1	0	1	1
23	0	0	0	0	1	0
24	0	0	0	0	1	0
25	0	0	0	0	0	0
26	1	1	1	0	1	1
27	0	0	0	0	0	0
28	0	0	0	0	0	0
29	0	0	0	0	0	0

- The classifier's effective performance can be ascribed to the highly structured nature of the dataset and the validity of the classifier's assumption for this specific dataset. The underlying assumption is that the data exhibits linear separability, indicating that it can be partitioned into two groups using a straight line. In this instance, the assumption holds true due to the earlier application of a filter that converted the data from numeric to binary.

- 

The main practical problems we would face if we were not to make this assumption for this particular dataset are:

Overfitting:

The classifier runs the risk of overfitting to the training data, implying that it might memorize specific patterns within the training set and struggle to generalize effectively to new data. Consequently, this overfitting can result in suboptimal performance when applied to unseen data.

Inaccurate predictions:

In cases where the data lacks linear separability, the classifier is prone to generating inaccurate predictions. This can result in misclassifying instances, posing significant repercussions in real-world applications due to the potential for erroneous classifications.

Complexity:

The classifier may become more complex and computationally expensive if it has to handle non-linearly separable data. This can lead to slower processing times and increased resource usage.

Time taken to build model: 0.01 seconds

The Naive Bayes classifier is known for its scalability and ability to handle large datasets efficiently. It can quickly learn to use high-dimensional features with limited training data, making it suitable for large datasets.

Detailed Accuracy By Class :

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.931	0.185	0.886	0.931	0.908	0.758	0.953	0.967	0
	0.815	0.069	0.885	0.815	0.849	0.758	0.953	0.936	1
Weighted Avg.	0.885	0.139	0.885	0.885	0.885	0.758	0.953	0.955	

So,

The prior probabilities for each class can be found in the "Class" column of the "Detailed Accuracy by Class" section. In this case, the prior probabilities for classes 0 and 1 are 0.931 and 0.815, respectively.

• How does Naïve Bayes compute the probability of an e-mail belonging to a class (spam/not spam)?

In the Naive Bayes approach, the classifier divides the data into training and testing sets. It calculates the probability of each word occurring in spam and non-spam categories. The classifier then multiplies these probabilities together for both classes and selects the class with the highest product as the final classification.

word_freq_3d_binarized		
0	2781.0	1775.0
1	9.0	40.0
[total]	2790.0	1815.0

Using the given counts for the word "3d" in the binarized format, we can compute the conditional probabilities as follows:

P(3d|spam) = (9.0 + 1) / (1775.0 + 2) ≈ 0.0053

P(3d|non-spam) = (40.0 + 1) / (1815.0 + 2) ≈ 0.0222

Utilizing these probabilities, the Naive Bayes classifier can assess the likelihood of an email being categorized as spam or non-spam by considering the occurrence of the word "3d" in its binarized format.