CS464 Introduction to Machine Learning

Homework 1 Report

Ece Kahraman

21801879

1. The Online Shopping Case

Given:

$$\mathbf{P}(P) = 0.45 \qquad \mathbf{P}(M) = 0.3 \qquad \mathbf{P}(U) = 0.25$$

$$\mathbf{P}(F_P \mid P) = 0.95 \qquad \mathbf{P}(F_P \mid M) = 0.6 \qquad \mathbf{P}(F_P \mid U) = 0.1$$

1.1.

$$\begin{aligned}\mathbf{P}(F_P) &= \mathbf{P}(F_P \mid P) \times \mathbf{P}(P) + \mathbf{P}(F_P \mid M) \times \mathbf{P}(M) + \mathbf{P}(F_P \mid U) \times \mathbf{P}(U)\\ &= (0.95 \times 0.45) + (0.6 \times 0.3) + (0.1 \times 0.25)\\ &= 0.6325\end{aligned}$$

1.2.

$$\begin{aligned}\mathbf{P}(P \mid F_P) &= \frac{\mathbf{P}(F_P \mid P) \times \mathbf{P}(P)}{\mathbf{P}(F_P)}\\ &= \frac{0.95 \times 0.45}{0.6325}\\ &= 0.6759\end{aligned}$$

1.3.

$$\mathbf{P}(P \mid F_N) = \frac{\mathbf{P}(F_N \mid P) \times \mathbf{P}(P)}{\mathbf{P}(F_N)}$$

$$\begin{aligned}\mathbf{P}(F_N \mid P) &= 1 - \mathbf{P}(F_P \mid P)\\ &= 1 - 0.95\\ &= 0.05\end{aligned} \qquad \begin{aligned}\mathbf{P}(F_N \mid M) &= 1 - \mathbf{P}(F_P \mid M)\\ &= 1 - 0.6\\ &= 0.4\end{aligned} \qquad \begin{aligned}\mathbf{P}(F_N \mid U) &= 1 - \mathbf{P}(F_P \mid U)\\ &= 1 - 0.1\\ &= 0.9\end{aligned}$$

$$\begin{aligned}\mathbf{P}(F_N) &= \mathbf{P}(F_N \mid P) \times \mathbf{P}(P) + \mathbf{P}(F_N \mid M) \times \mathbf{P}(M) + \mathbf{P}(F_N \mid U) \times \mathbf{P}(U)\\ &= (0.05 \times 0.45) + (0.4 \times 0.3) + (0.9 \times 0.25)\\ &= 0.3675\end{aligned}$$

$$\begin{aligned}\mathbf{P}(P \mid F_N) &= \frac{(0.05 \times 0.45)}{0.3675}\\ &= 0.0612\end{aligned}$$

2. Spam Email Detection

2.1.

There are 1183 spam emails out of 4137 emails in `y_train.csv`, so the percentage is 28.6%. So the training set is skewed towards the class of normal emails.

This skewed training set might lead to a bias in my model for the non-spam emails. My model may become inclined to predict data as a non-spam email. Because of this bias, my model may optimize its parameters to better predict non-spam emails so it might perform worse with a set of new and unseen data.

The accuracy of the model may degrade as well. I expect that my model will be inclined to predict data as non-spam, whether it is correct or not.

2.2.

As expected before, the number of false negatives (predicted non-spam but was spam) is higher than false positives (predicted spam but was non-spam) in the confusion matrix. This indicates that my model is indeed biased towards the non-spam class.

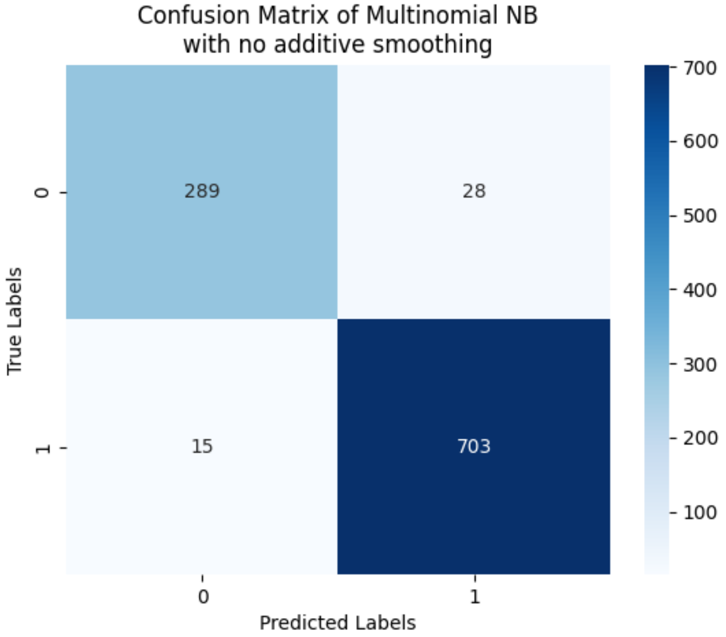
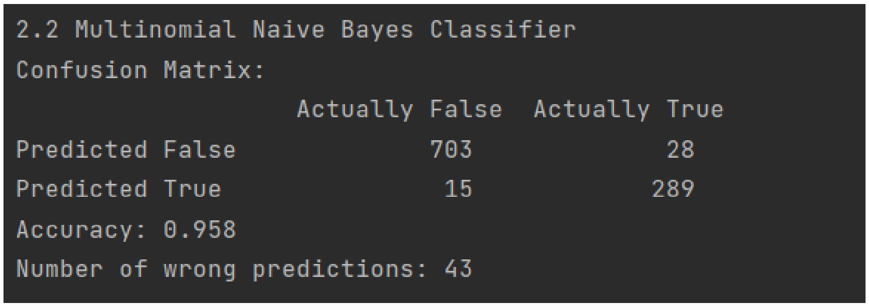


Figure 1: Confusion Matrix of Multinomial Naive Bayes Classifier



```
2.2 Multinomial Naive Bayes Classifier
Confusion Matrix:
                   Actually False  Actually True
Predicted False              703             28
Predicted True                15            289
Accuracy: 0.958
Number of wrong predictions: 43
```

Figure 2: Confusion Matrix, accuracy, and the number of wrong predictions of Multinomial Naive Bayes Classifier

2.3.

This time, a constant alpha is added to the Multinomial Naive Bayes model to implement Dirichlet distribution. Dirichlet distribution is used to prevent overfitting in multinomial naive bayes classifiers. It is seen on the confusion matrix that the number of false negatives is now less than the number of false positives, the model is now less biased towards the non-spam class. I expected the accuracy to be higher but this model was curiously less accurate than the previous model.

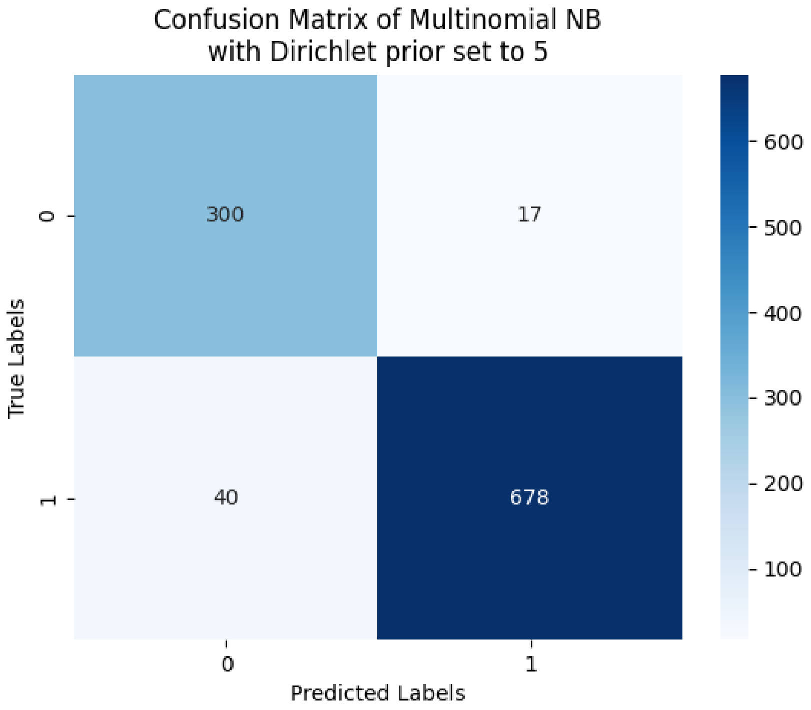
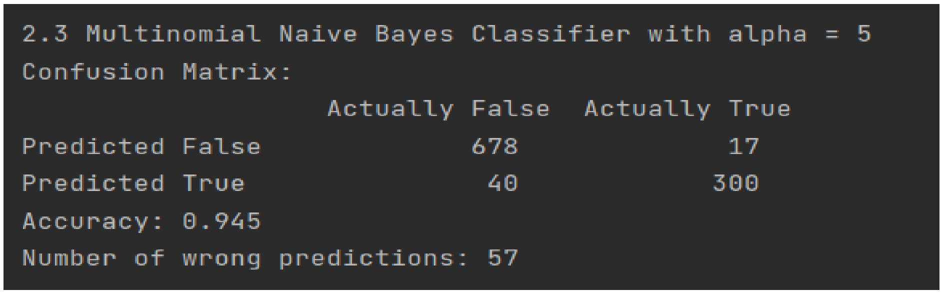


Figure 3: Confusion Matrix of Multinomial Naive Bayes Classifier with alpha set to 5



```
2.3 Multinomial Naive Bayes Classifier with alpha = 5
Confusion Matrix:
                   Actually False  Actually True
Predicted False               678             17
Predicted True                 40            300
Accuracy: 0.945
Number of wrong predictions: 57
```

Figure 4: Confusion Matrix, accuracy, and the number of wrong predictions of Multinomial Naive Bayes Classifier with alpha set to 5

2.4.

This model utilizes Bernoulli distribution, it checks if the spam words exist in the mails or not, rather than checking for the frequencies of those words. As a consequence of this, a non-spam can be mispredicted for containing some spam words. Number of wrong predictions doubles the previous ones, and the number of false negatives is much larger because the model is inclined to predict spam once it finds a spam word.

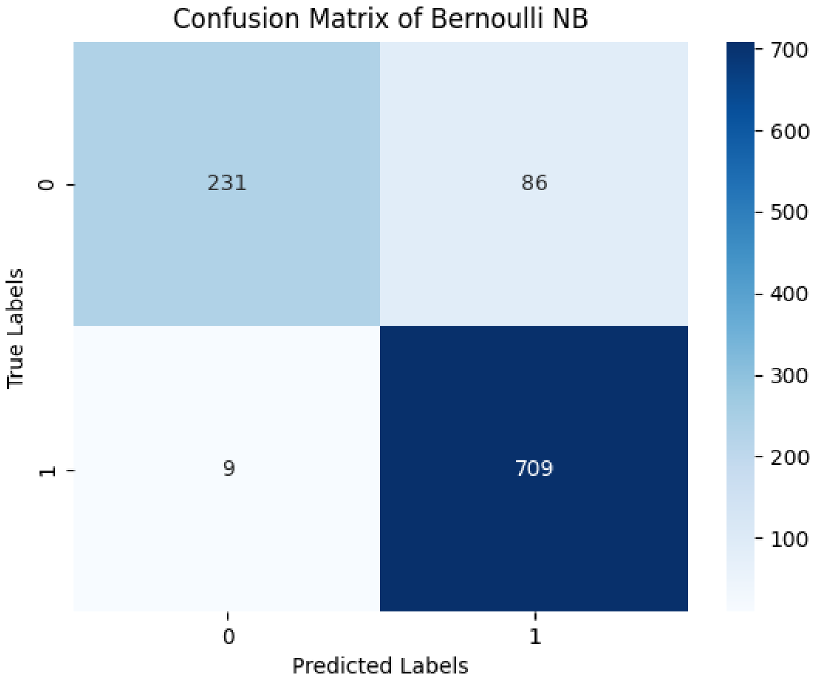
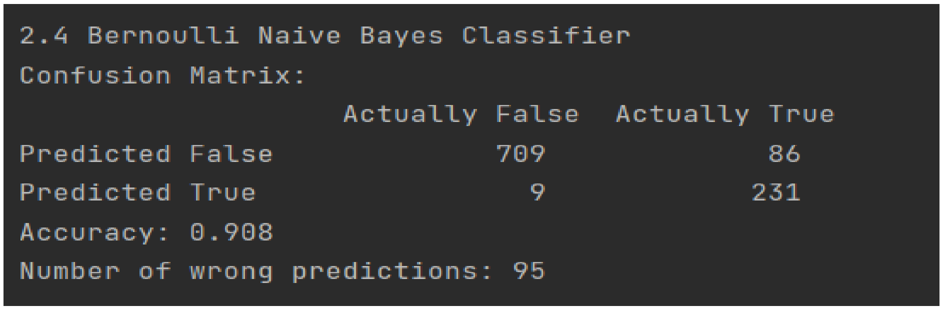


Figure 5: Confusion Matrix of Bernoulli Naive Bayes Classifier



```
2.4 Bernoulli Naive Bayes Classifier
Confusion Matrix:
                   Actually False  Actually True
Predicted False             709             86
Predicted True                9            231
Accuracy: 0.908
Number of wrong predictions: 95
```

Figure 6: Confusion Matrix, accuracy, and the number of wrong predictions of Bernoulli Naive Bayes Classifier

2.5.

In the Multinomial NB Classifier, it is observed that the false negatives are on the higher side while the accuracy is high. The accuracy might not be a reliable performance metric because we already knew that the datasets are skewed towards the non-spam class. It might perform worse when a truly balanced data set is given.

When the Dirichlet distribution is introduced to the same classifier, we observe the previous bias is mostly gone, the number of false negatives are lower. Its accuracy is a bit low but I think it is acceptable because the model performs without a bias on a skewed data set.

With the Bernoulli NB Classifier, we implemented a totally different approach. With Multinomial NB Classifiers, we calculated the likelihoods according to the occurrences of the spam words. In the Bernoulli NB Classifier, we are calculating the likelihoods according to the existence of the spam words. For this reason, I believe, this model performed the worst. The number of false negatives on its confusion matrix is quite higher than the others, it predicted falsely when a non-spam email contained a spam word.

In a real world setting, I think the Multinomial NB Classifier would work the best, if the model is given a more balanced training data set. The Bernoulli NB Classifier could work on a stricter binary goal, like completely having a feature or absolutely not.