ASSINGMENT 2

Applied Machine Learning

Prepared by:

Ahmed Ahmed

Hadir Hassan

Donia Abdelreheem

Supervised by:

DR/Murat

PART 1

1)

P(C|X) = P(X|C) * P(C) / P(X)

We will compare the probabilities for each class and assign the new instance to the class with the highest probability.

1] Stolen class (Yes):

P (Yes|Blue, SUV, Domestic) =(P(Blue|Yes) * P(SUV|Yes) * P(Domestic|Yes) * P(Yes))/ P(Blue, SUV, Domestic)

P(Yes)= number of stolen cars / total number of cars=6/14=3/7

P(Blue|Yes) = number of blue cars that are stolen / number of stolen cars = 1/6

P(SUV|Yes) = number of SUV cars that are stolen / number of stolen cars = 2/6 = 1/3

P(Domestic|Yes) = number of domestic cars that are stolen / number of stolen cars = 2/6 = 1/3

** For all entries in the dataset, the denominator does not change, it remains static. Therefore, the denominator can be removed.

P(Yes|Blue, SUV, Domestic) = P(Blue|Yes) * P(SUV|Yes) * P(Domestic|Yes) * P(Yes) = (1/6) * (1/3) * (1/3) * (3/7) = 1/126=0.0079

1] Not stolen class (No):

P (No|Blue, SUV, Domestic) = (P(Blue|No) * P(SUV|No) * P(Domestic|No) * P(No))/ P(Blue, SUV, Domestic)

P(No)= number of non-stolen cars / total number of cars = 8/14=4/7

P(Blue|No) = number of blue cars that are not stolen / number of non-stolen cars = 2/8=1/4

P(SUV|No) = number of SUV cars that are not stolen / number of non-stolen cars = 5/8

P(Domestic|No) = number of domestic cars that are not stolen / number of non-stolen cars = 5/8

** For all entries in the dataset, the denominator does not change, it remains static. Therefore, the denominator can be removed.

P (No|Blue, SUV, Domestic) = P(Blue|No) * P(SUV|No) * P(Domestic|No) * P(No) = (1/4) * (5/8) * (5/8) * (4/7) =25/448= 0.055

*Evidence

P (Blue, SUV, Domestic) = P(Blue, SUV, Domestic |Yes)*p(Yes) + P(Blue, SUV, Domestic |No)*p(No)

P (Blue, SUV, Domestic)=[(1/6)*(2/6)*(2/6)*(6/14) +(2/8)*(5/8)*(5/8)*(8/14)] = 0.0637

The probability of the new instance belonging to No is higher than Yes [P(No|Blue, SUV, Domestic)> P(Yes|Blue, SUV, Domestic)], we classify the new instance as (No) not stolen.

2)

$$R(\alpha_i \mid \mathbf{x}) = \sum_{k=1}^{K} \lambda_{ik} P(C_k \mid \mathbf{x})$$

Expected risks of three actions:

R(a1|x) = 0P(C1|x) + 6P(C2|x) = 6*(1- P(C1|x))

R(a2|x) = 3P(C1|x) + 0P(C2|x) = 3P(C1|x)

R(a3|x) = 2P(C1|x) + 2*(1- P(C1|x)) =2

We choose a1 if:

R(a1|x) < 2

6*(1- P(C1|x)) < 2

1- P(C1|x) < 2/6

1- P(C1|x) < 1/3

P(C1|x) >2/3

We choose a2 if:

R(a2|x) < 2

3P(C1|x) < 2

P(C1|x) < 2/3

Reject, if 2/3 < P(C1|x) < 2/3

PAERT 2

	word_freq_make	word_freq_address	word_freq_all	word_freq_3d	word_freq_our	word_freq_over	word_freq_remove	word_freq_internet	word_freq_order	word_freq_mail	...	char_freq_;	char_freq_(	char_freq_[	char_freq_!	char_fr
0	0.00	0.64	0.64	0.0	0.32	0.00	0.00	0.00	0.00	0.00	...	0.000	0.000	0.0	0.778	
1	0.21	0.28	0.50	0.0	0.14	0.28	0.21	0.07	0.00	0.94	...	0.000	0.132	0.0	0.372	
2	0.06	0.00	0.71	0.0	1.23	0.19	0.19	0.12	0.64	0.25	...	0.010	0.143	0.0	0.276	
3	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	...	0.000	0.137	0.0	0.137	
4	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	...	0.000	0.135	0.0	0.135	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4596	0.31	0.00	0.62	0.0	0.00	0.31	0.00	0.00	0.00	0.00	...	0.000	0.232	0.0	0.000	
4597	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.000	0.000	0.0	0.353	
4598	0.30	0.00	0.30	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.102	0.718	0.0	0.000	
4599	0.96	0.00	0.00	0.0	0.32	0.00	0.00	0.00	0.00	0.00	...	0.000	0.057	0.0	0.000	
4600	0.00	0.00	0.65	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.000	0.000	0.0	0.125	

4601 rows × 58 columns

Figure 1 the Dataset shape

A)

Splitting the dataset into two parts as training data and test data. The first 80 percent samples should be selected as training data and last 20 percent samples should be selected as test data.

```
import numpy as np
#Function to Spilt DataSet by rate like (80,20)

def split_data(data,rate):

  last_element=round(len(data)*rate)
  print(last_element)
  #for make suffle in data

  #data= data.reindex(np.random.permutation(data.index))

  #################
  X_train=data.drop(labels='spam',axis=1)[:last_element]
  y_train=data.spam[:last_element]
  X_test=data.drop(labels="spam",axis=1)[last_element:]
  y_test=data.spam[last_element:]
  return X_train, y_train, X_test,y_test
```

Figure 2 function to spilt data by rate.

```
[23] X_first_80_percent_train, Y_first_80_percent_train, X_last_20_percent_test,Y_last_20_percent_test=split_data(df,.8)
```

Figure 3 splitting dataset.

We made this function to split the data set with rate that you choose it and put in it as a parameter.

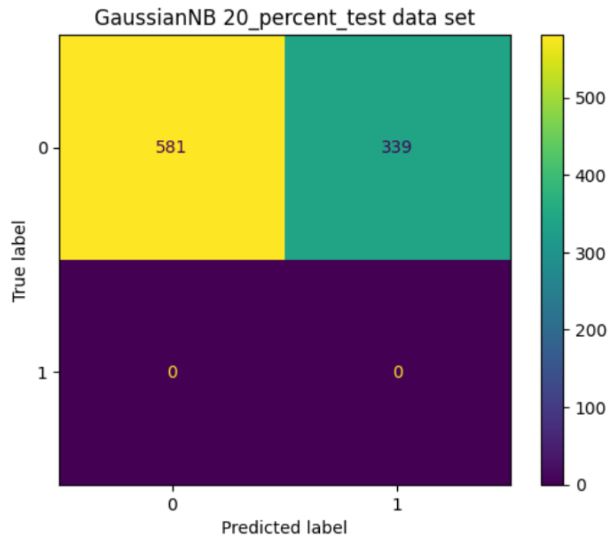


Figure 5 confusion matrix for test set model

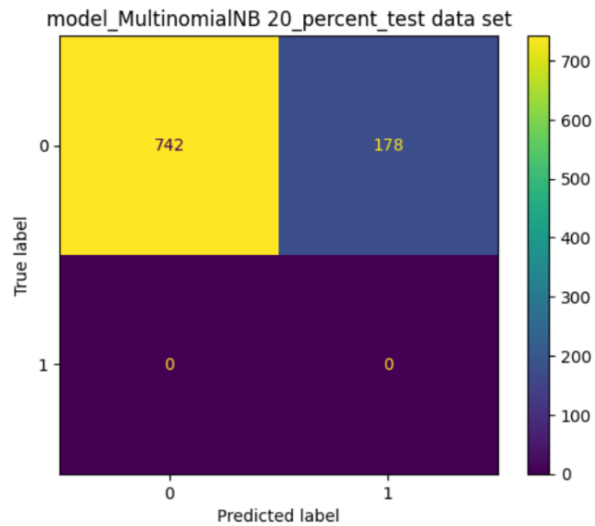


Figure 4 confusion matrix for test set model Multinominal Naive Bayes

This is the result form model Gaussian Naive Bayes, model Multinominal Naive Bayes and show us the model don't get any result on class 1 because the 20 percent test data have the 0 class only and this make the test data is biased to class 0.

B)

train test split function on input and output of the whole data and utilize 80% of samples as train and 20% of samples as test data.

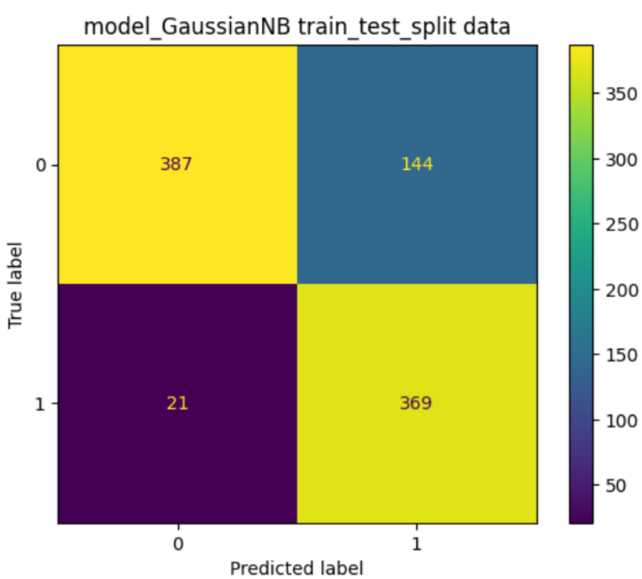


Figure 7 confusion matrix for test set model Gaussian Naive Bayes

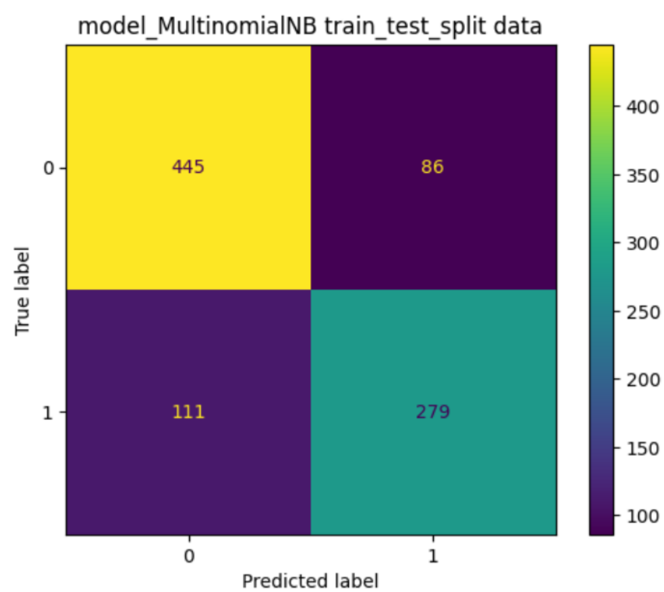


Figure 7 confusion matrix for test set model Multinominal Naive Bayes

This is the result form model Gaussian Naive Bayes, model Multinominal Naive Bayes after using the train test split function from Sklearn and show us the models started to get real a little improvement because the train test split function make shuffle in the data set and remove the block of one class in the same area like the 20 test data set in figure 5 .

C)

Use another Naive Bayes classifier of your choice to check for the improvement in terms of accuracy score of test data in (b) over Gaussian and Multinomial asked in (b) and provide an explanation for the improvement in performance (if any).

Here we choose the 2 naïve Bayes (**Bernoulli Naive Bayes, Complement Naive Bayes**) and train on the data form train test function.

As we see the Bernoulli Naive Bayes had a result better in confusion matrix form all models in figure (6 , 7, 8 , 9)

And if we see the f1 score of Bernoulli in figure 10, we got .83 score

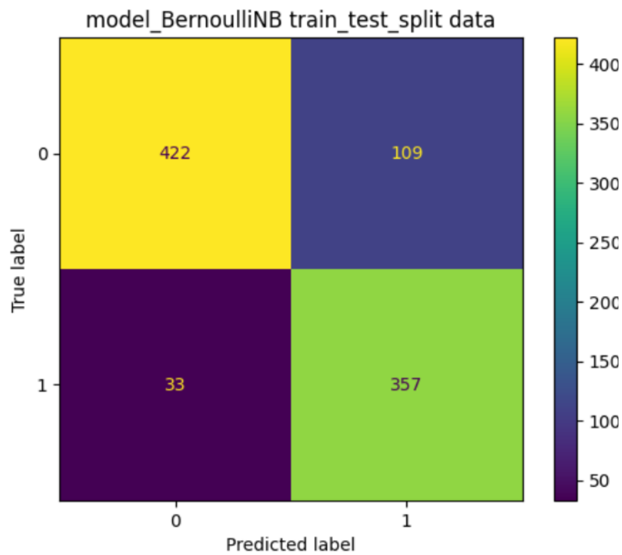


Figure 9 confusion matrix for test set model Bernoulli N**aive Bayes**

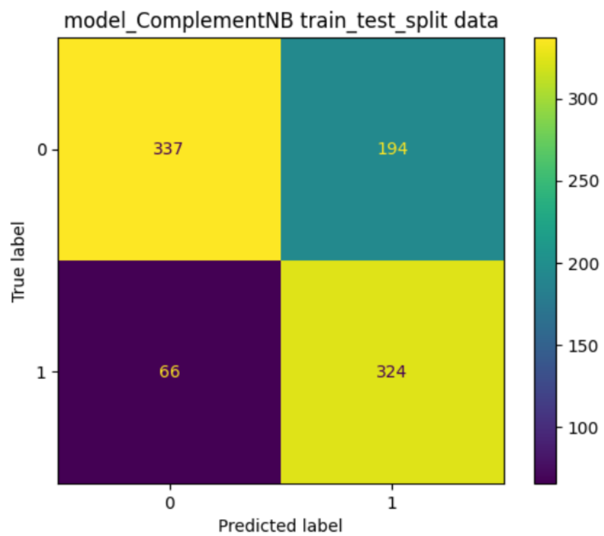


Figure 8 confusion matrix for test set

Difference between Bernoulli, Multinomial and Gaussian Naive Bayes. Multinomial Naïve Bayes considers a feature vector where a given term represents the number of times. On the other hand, Bernoulli is a binary algorithm used when the feature is present or not.

```
model_ComplementNB train_test_split data
              precision    recall  f1-score   support

     class 0       0.81      0.83      0.82       531
     class 1       0.77      0.74      0.75       390

    accuracy                           0.79       921
   macro avg       0.79      0.79      0.79       921
weighted avg       0.79      0.79      0.79       921

model_BernoulliNB train_test_split data
              precision    recall  f1-score   support

     class 0       0.96      0.73      0.83       531
     class 1       0.73      0.95      0.82       390

    accuracy                           0.83       921
   macro avg       0.84      0.84      0.83       921
weighted avg       0.86      0.83      0.83       921
```

Figure 10 the Confusion report for Complement, Bernoulli

and our problem is binary in target and this make the Bernoulli get the highest score form the others models

D)

Take same first 80 percent as asked in (a) training samples and split the data into four equal parts according to order such as the first 25% of training data (subset 1), the second 25% of training data (subset 2), the third 25% of training data (subset 3) and the fourth 25% of training data (subset 4).

```
#function to spilt the .8 of data to 4 subsets
def  split_data_val(dataFrame):
  new_range=round(.25*len(dataFrame))
  return dataFrame[:new_range],dataFrame[new_range:2*new_range],dataFrame[2*new_range:3*new_range],dataFrame[3*new_range:]
```

```
[43] X_subset=split_data_val(X_first_80_percent_train)
     Y_subset=split_data_val(Y_first_80_percent_train)
```

Figure 11 function to split 80 data set to 4 subsets.

In figure 11 we built function to split it and save in list and used to train our selected models get results from them.

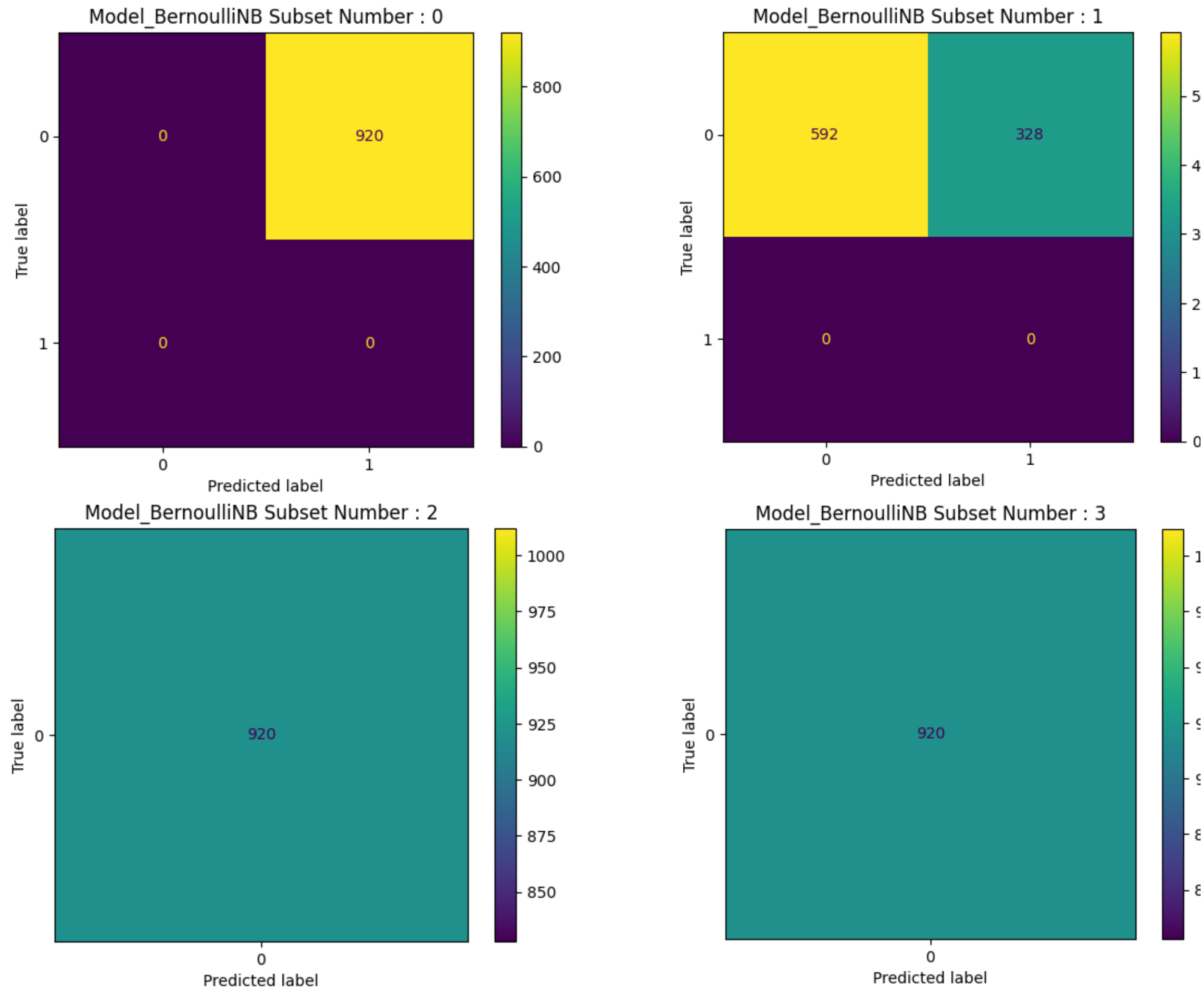


Figure 12 confusion matrix model Bernoulli Naive Bayes **for 4 subsets.**

In subset 0 our model Bernoulli Naive Bayes didn't learn anything with this data because All target data in training in class 1 in figure 13 and all test data in class 0, this made the model confused.

```
pd.DataFrame(Y_subset[0]).value_counts()

spam
1       920
dtype: int64
```

Figure 13 subset 0 of target

In subset 1 our model Bernoulli Naive Bayes learned by this data because this subset had two classes 0,1 nd the test on 0 class and this
make this test data baised on class 0.

```
pd.DataFrame(Y_subset[1]).value_counts()

spam
1       893
0        27
dtype: int64
```

Figure 14 subset 1 of target

In subset 2,3 in figure 15 the result was like each other because All data in target train data was on class 0 and the target test data in class 0 and this make the accuracy 100 if we input anything to the model, we would get class 0.

```
[87] pd.DataFrame(Y_subset[2]).value_counts()

     spam
     0       920
     dtype: int64
```

```
[88] pd.DataFrame(Y_subset[3]).value_counts()

     spam
     0       921
     dtype: int64
```

Figure 15 subset 2, 3 of target

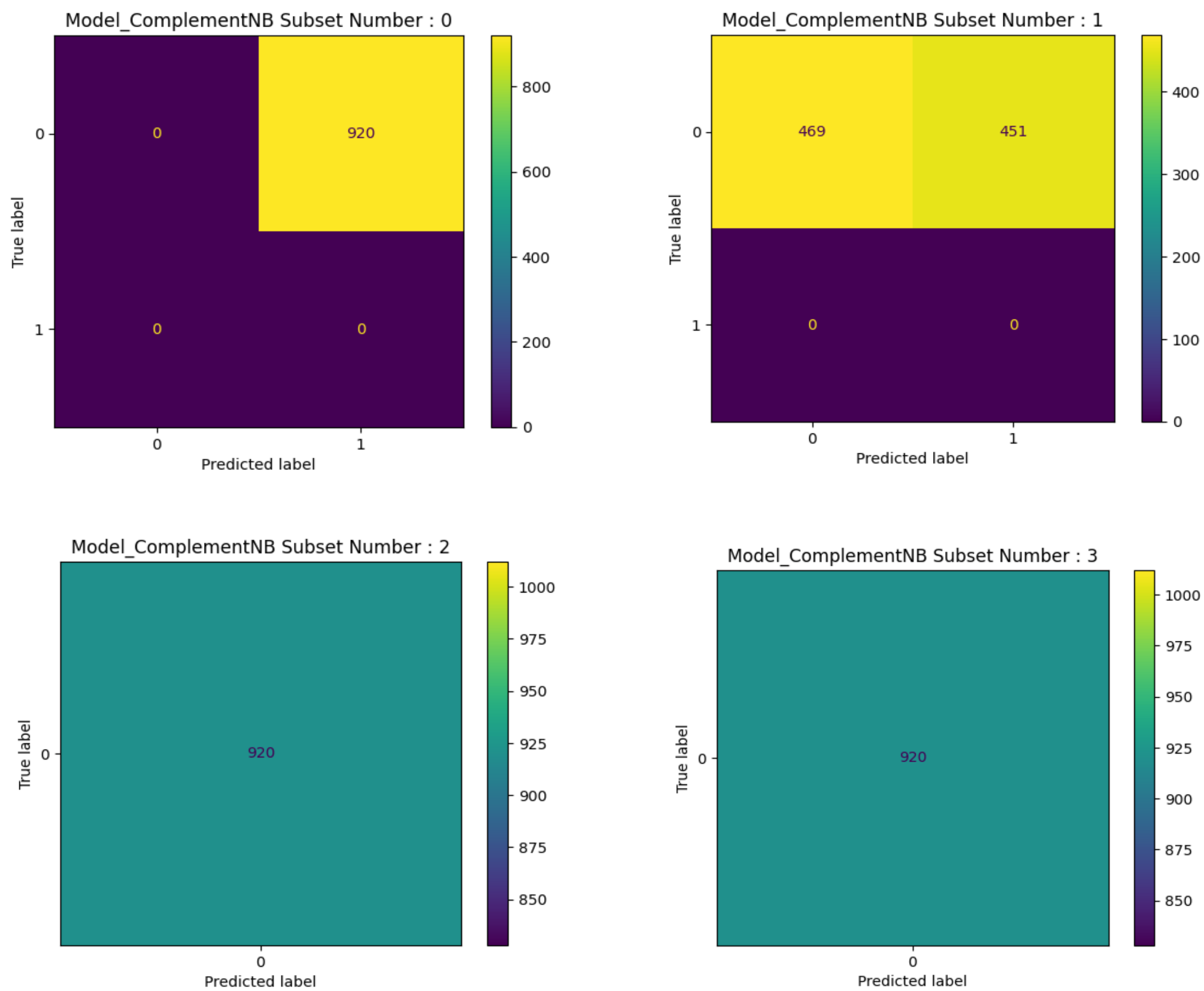
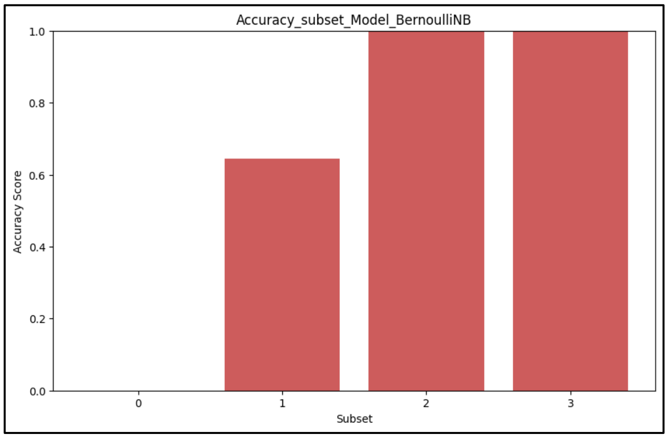


Figure 16 confusion matrix model Complement Naive B**ayes for 4 subsets.**

In subset 0 our model Complement Naive didn't differ from the Bernoulli Naive Baye and didn't learn anything with this data because All target data in training in class 1 in figure 13 and all test data in class 0, this made the model confused.

In subset 1 our model Bernoulli Naive Bayes learned by this data because this subset had two classes 0,1 nd the test on 0 class and this make this test data baised on class 0.

In subset 2,3 in figure 15 the result was like makes other because All data in target train data was on class 0 and the target test data in class 0 and this make the accuracy 100 if we input anything to the model, we would get class 0.



from the confusion matrix in figure 17 we can guess the accuracy directly in subset 0 and 2,3 we can say the subset 1 is considered to be the Truely model that you depend on it.

Figure 17 Bernoulli Naive **Bayes Accuracy graph for 4 subsets.**

from the confusion matrix in figure 19 like Bernoulli Naive Bayes accuracy graph we can guess the accuracy directly in subset 0 and 2,3 we can say the subset 1 is considered to be the Truely model that you depend on it.

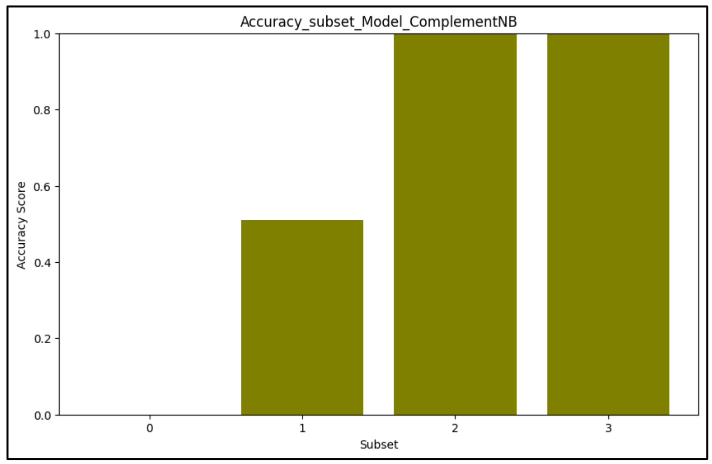


Figure 18 Complement Naive **Bayes accuracy graph for 4 subsets.**

CONCLUSION

We had problem in the data without shuffle it and if we want to solve to avoid this problem by shuffling the dataset and activate this heighted line in figure 20.

```
import numpy as np
#Function to Spilt DataSet by rate like (80,20)

def split_data(data,rate):

  last_element=round(len(data)*rate)
  print(last_element)
  #for make suffle in data

  #data= data.reindex(np.random.permutation(data.index))

  #################
  X_train=data.drop(labels='spam',axis=1)[:last_element]
  y_train=data.spam[:last_element]
  X_test=data.drop(labels="spam",axis=1)[last_element:]
  y_test=data.spam[last_element:]
```

Figure 19 spilt data function.

After shuffling the dataset, the result from model Bernoulli Naive Bayes will be in figures 21, 22.

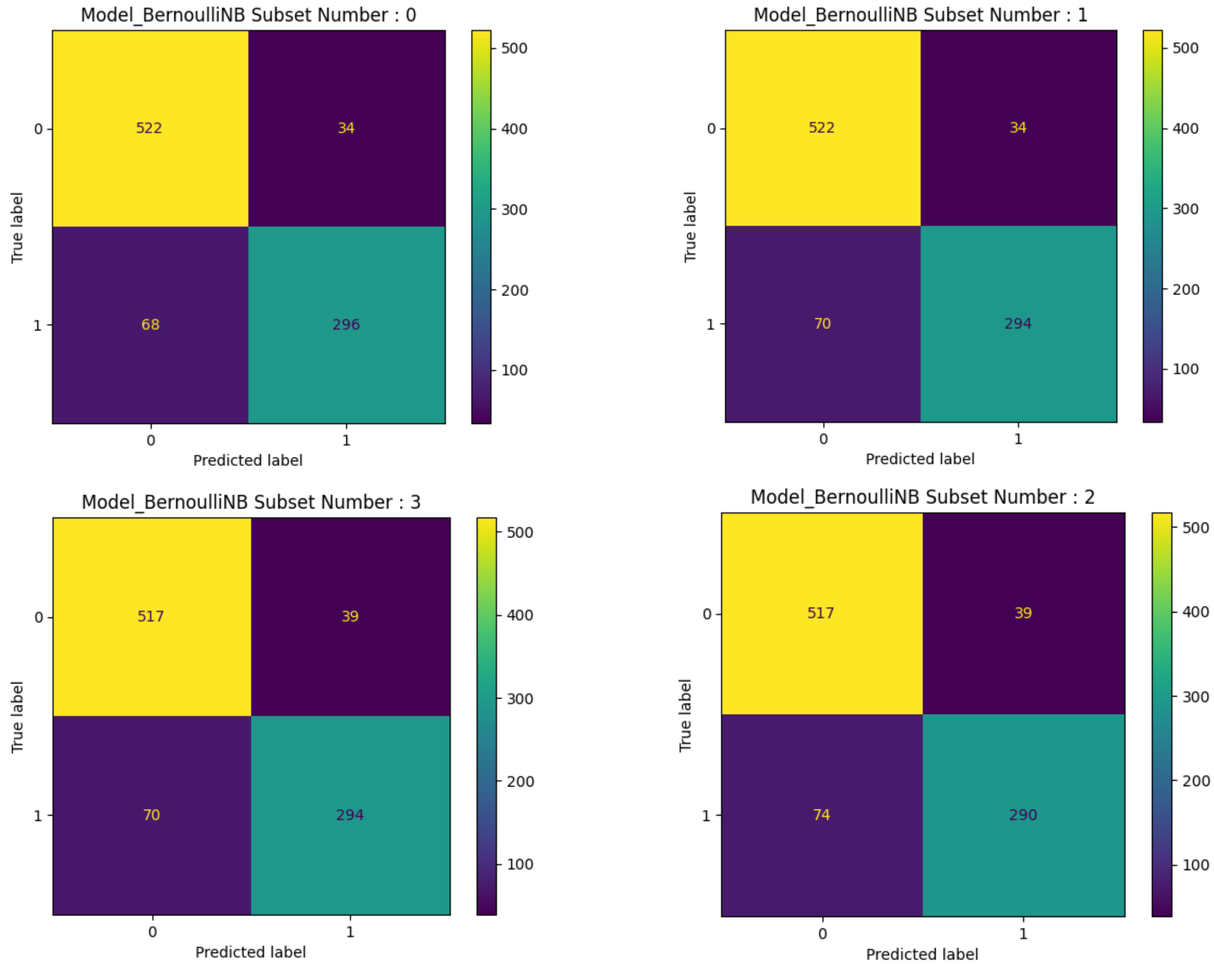


Figure 20 confusion matrix model Bernoulli N**aive** B**ayes for 4 subsets after shuffling the All dataset.**

```
Accuracy_subset_Model_BernoulliNB [0.8891304347826087, 0.8869565217391304, 0.8771739130434782, 0.8815217391304347]
```

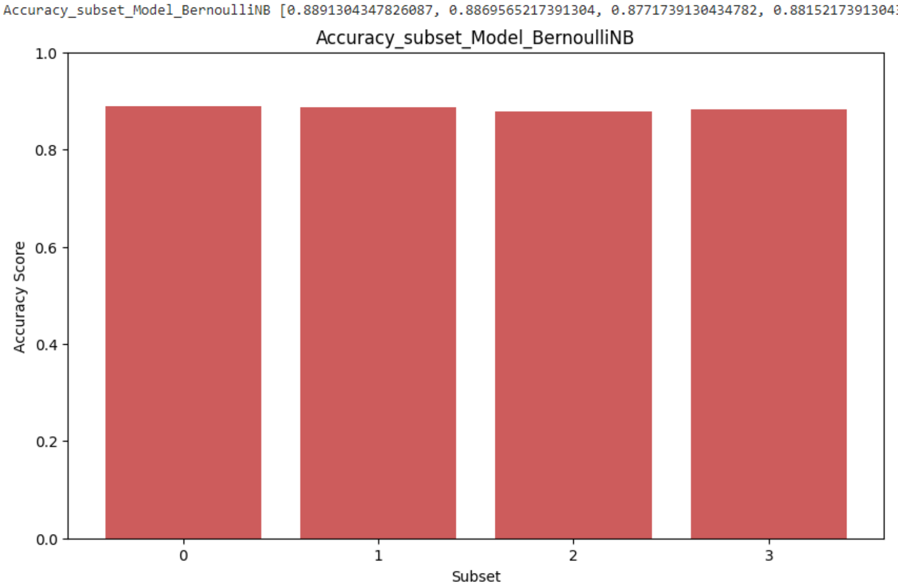


Figure 21 Bernoulli Naive B**ayes Accuracy graph for 4 subsets.**