Batu Arda Düzgün

1: Probability:

1.1.)

Let's call the event of getting the (A, heads) pair in the i'th round $M_i$. Then not getting the (A, heads) pair is invers of that and is denoted as $M_i'$. Then let's call the event of getting our first (A, heads) pair in the 8th trial as $N$. To get the first (A, heads) pair in the 8'th trial we must not get (A, heads) pair in the first seven trials, so we can write:

$$N = M_1' \cap M_2' \cap M_3' \cap M_4' \cap M_5' \cap M_6' \cap M_7' \cap M_8$$

$$P(N) = P(M_1' \cap M_2' \cap M_3' \cap M_4' \cap M_5' \cap M_6' \cap M_7' \cap M_8)$$

$M_i$ are i.i.d. events, so we can write:

$$P(N') = P(M_1') + P(M_2') + P(M_3') + P(M_4') + P(M_5') + P(M_6') + P(M_7') + P(M_8)$$

$$P(N') = P(M_i')^7 \cdot P(M_i)$$

We can find $P(M_i)$ and $P(M_i')$ in terms of the given variables. P3 is the probability of selecting A and P1 is the probability of the event of getting head from coin A. Both of these events need to happen so:

$$P(M_i) = P1 \cdot P3$$

$$P(M_i') = 1 - P(M_i) = 1 - P1 \cdot P3$$

Then $P(N)$ is:

$$P(N') = (1 - P1 \cdot P3)^7 \cdot P1 \cdot P3$$

1.2.)

Let's call the event of getting a head in round i, $H_i$. Then let's call the event of selecting coin A in round i as $K$. Then the event of selecting coin B in round i becomes $K'$ as it is the only other option. Then, using total probability theorem we can write:

$$P(H_i) = P(H_i|K_i)P(K_i) + P(H_i|K_i')P(K_i')$$

The probability of these events are defined in the question, implementing these we get:

$$P(H_i) = P1 \cdot P3 + P2 \cdot (1 - P3)$$

Then the question askes $E[H_1 + H_2 + H_3 + H_4 + H_5 + H_6 + H_7 + H_8 + H_9 + H_{10}]$. Once again $H_i$ are i.i.d. events se we can compute this expectation as:

$$E[H_1 + H_2 + H_3 + H_4 + H_5 + H_6 + H_7 + H_8 + H_9 + H_{10}] = 10 \cdot E[H_i]$$

Then we can find $E[H_i]$ from the definition of expectation.

$$E[H_i] = 1 \cdot \big(P1 \cdot P3 + P2 \cdot (1 - P3)\big) + 0 * (1 - \big(P1 \cdot P3 + P2 \cdot (1 - P3)\big))$$

$$E[H_i] = P1 \cdot P3 + P2 \cdot (1 - P3)$$

And thus:

$$E[H_1 + H_2 + H_3 + H_4 + H_5 + H_6 + H_7 + H_8 + H_9 + H_{10}] = 10 \cdot (P1 \cdot P3 + P2 \cdot (1 - P3))$$

1.3.a.)

Oliver's guessing performance could be measured by sampling. Someone could watch him make predictions and note how many of them are correct. Then divide the correct prediction number with all predictions number to find the given rates. It should be noted that his success probability for not heads is 0.99, to get such a sample mean there needs to be at least 100 samples. And Oliver predicts not heads one in every 100 trails, so someone needed to watch him make prediction for around 10000 rounds to get the given success rates.

To validate the given probabilities, we could sample his prediction even more times than calculate the p-value for the null hypothesis of Oliver's prediction success rates. As an example, let $\hat{s}_n$ be the sample new success rate of n sample for Oliver's head predictions, note that $\hat{s}_n$ is not a random variable after the sampling is done. Let $s_G$ be the given success rate for head predations which is not a random variable too. And let the $\hat{S}_n$ be the success rate of n samples for Oliver's head predictions before the sampling so it is a random variable. We compute:

$$P\left(\hat{S}_n \geq |\hat{s}_n - s_G| + s_G \cap \hat{S}_n \leq -|\hat{s}_n - s_G| + s_G \middle| \; s_G\right)$$

This can be seen as the probability of the assumption of Oliver's prediction's success rate holding. If it is small, it means the given probabilities are not valid if it is large (close to one) then it means the given rates are valid. We need to compute this for heads and not heads separately.

1.3.b)

Assuming the question asks the probability of predicting 8 heads in a row in a limitless number of trials the probability of this event happening is 1. Assuming the question is asking the probability of predicting 8 heads in a row in 8 trials we need to do more work. Let $H_i$ be the event of Oliver predicting a head in the i'th trial. The probability of Oliver predicting a not head is given as 0.01 then $P(H_i) = 0.99$. Then let's call the event of getting 8 heads in a row $M$. We can express $M$ as:

$$M = H_1 \cap H_2 \cap H_3 \cap H_4 \cap H_5 \cap H_6 \cap H_7 \cap H_8$$

$$P(M) = P(H_1 \cap H_2 \cap H_3 \cap H_4 \cap H_5 \cap H_6 \cap H_7 \cap H_8)$$

Assuming $H_i$ are independent we can write:

$$P(M) = P(H_1) + P(H_2) + P(H_3) + P(H_4) + P(H_5) + P(H_6) + P(H_7) + P(H_8)$$

$$P(M) = P(H_i)^8$$

$$P(M) = 0.99^8$$

$$P(M) = 0.9227446944$$

1.3.c)

Let's call the event of Oliver predicting head in the i'th trial $H_{P,i}$ and let's call the event of getting a head in the i'th trial $H_{C,i}$. Then we want to find $P(H_{P,i}{}' \mid H_{C,i})$. Using Bayer's rule we can write this as:

$$P(H_{P,i}{}' \mid H_{C,i}) = \frac{P(H_{C,i} \mid H_{P,i}{}')P(H_{P,i}{}')}{P(H_{C,i})}$$

We know $P(H_{C,i} \mid H_{P,i})$ and $P(H_{P,i})$ as they are given in the question as $P(H_{C,i} \mid H_{P,i}{}') = 0.01$ and $P\left(H_{P,i}{}'\right) = 0.01$. Then we can find $P(H_{C,i})$ using the total probability theorem.

$$P\left(H_{C,i}\right) = P(H_{C,i} \mid H_{P,i})P(H_{P,i}) + P(H_{C,i} \mid H_{P,i}{}')P(H_{P,i}{}')$$

$$P\left(H_{C,i}\right) = 0.95 \cdot 0.99 + 0.01 \cdot 0.01$$

$$P\left(H_{C,i}\right) = 0.9406$$

Then we can write $P(H_{P,i} \mid H_{C,i})$ as:

$$(H_{P,i} \mid H_{C,i}) = \frac{0.01 \cdot 0.01}{0.9406}$$

$$(H_{P,i} \mid H_{C,i}) = 0.000106326422$$

2.1)

First I need to talk about how un-even the distributions of the data is, the insulin measurements are between 100 and 300 while Diabetes Pedigree Function has values between 0 and 1. Unless these are normalized the then some futures are incorrectly taken as hundreds of times more important. I could use Mahalanobis distance to solve this problem as it finds distances based on the covariance matrix, however it has a very long computation time and I will be running my code a lot of times during the homework and we also care about the run times of our algorithms. So instead I simply normalized my data by de-meaning it and dividing each future with it's own standard deviation. This is cruder compared to the covariance matrix but it is faster and probably good enough for this case. Then I didn't want to test a lot p hyper parameters for the Minkowski distance and instead just lowered my scope to the two most common cases the l1 and l2 norm (Manhattan distance and Euclidian distance). I was not sure which one would work better because even though l2 norm is known to work better for a low number of futures, l1 norm works better for normalized data. Because of this I tested for both of them and picked the best one. L1 norm gave better results as we will see in the results.

2.2)

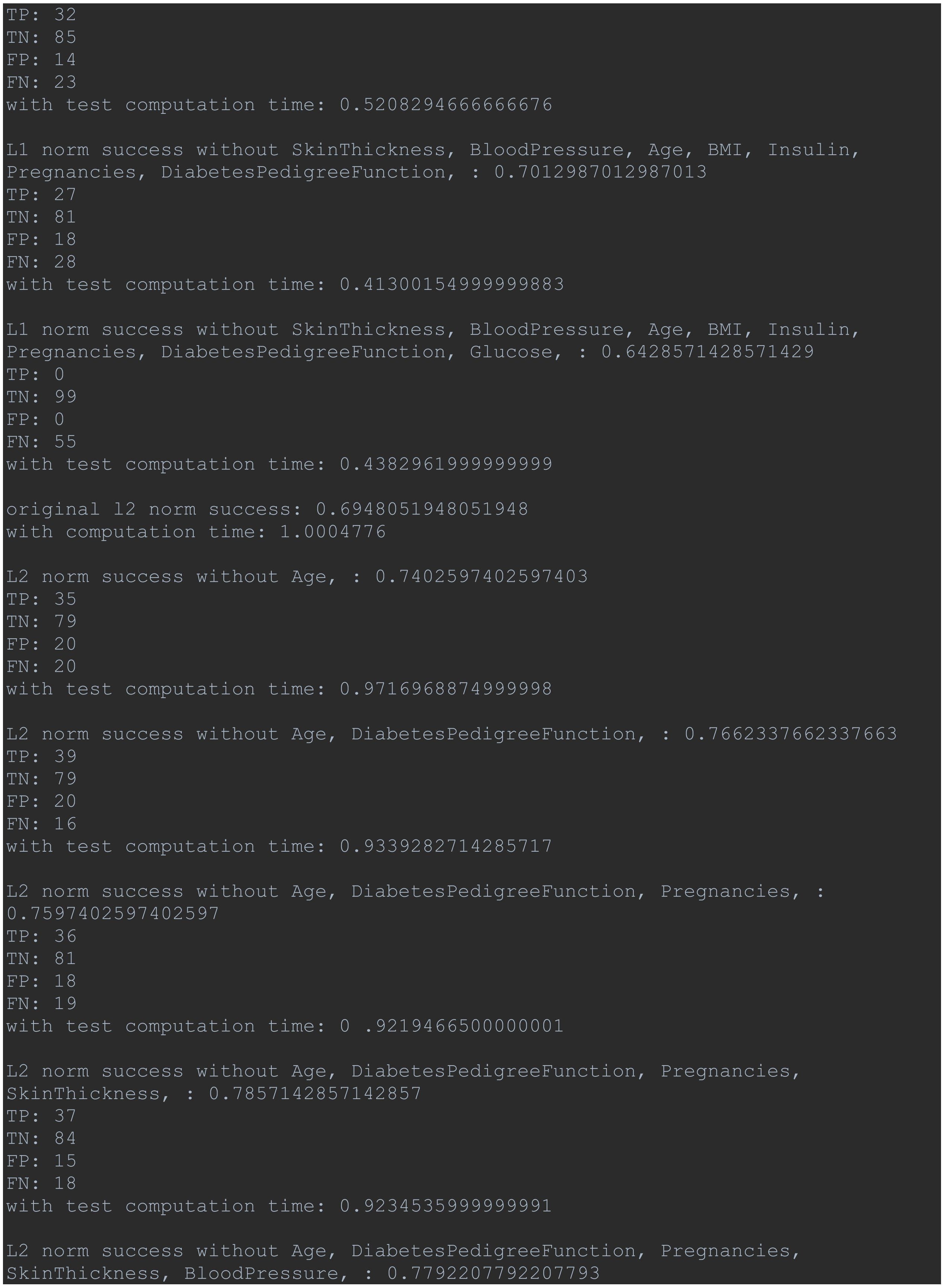
Using less number of futures could be better as some features might be irrelevant and they might cause confusion, so their removal would increase the accuracy. Also, having less number of futures lowers the odds of overfitting the model. Finally, some features could contain the information in other futures and removal of these features could have minimal effect on the accuracy while greatly improving the run time of the algorithm.

2.3)

Instead of doing backwards elimination while only a slight loss in accuracy or improvement in accuracy is seen in a step, I did it all the way (until all futures are eliminated) to see the effect of number of futures on the run time. I did this for both the version with l1 norm and the l2 norm. I should also note that there are 0 values in places where there should not be such as insulin, for these places I placed NaN and did not take them in to the distance computation.

Their results are as seen below:

```
The training (opening the files an normalizing them) computation time:
0.014804199999999999

original L1 norm success: 0.7337662337662337
with computation time: 0.6051527999999999

L1 norm success without SkinThickness, : 0.7662337662337663
TP: 34
TN: 84
FP: 15
FN: 21
with test computation time: 0.617642275

L1 norm success without SkinThickness, BloodPressure, : 0.7597402597402597
TP: 34
TN: 83
FP: 16
FN: 21
with test computation time: 0.5836634

L1 norm success without SkinThickness, BloodPressure, Age, :
0.7727272727272727
TP: 38
TN: 81
FP: 18
FN: 17
with test computation time: 0.5744975666666668

L1 norm success without SkinThickness, BloodPressure, Age, BMI, :
0.7857142857142857
TP: 36
TN: 85
FP: 14
FN: 19
with test computation time: 0.56078176

L1 norm success without SkinThickness, BloodPressure, Age, BMI, Insulin, :
0.8051948051948052
TP: 36
TN: 88
FP: 11
FN: 19
with test computation time: 0.5452718750000001

L1 norm success without SkinThickness, BloodPressure, Age, BMI, Insulin,
Pregnancies, : 0.7597402597402597
```



```
TP: 32
TN: 85
FP: 14
FN: 23
with test computation time: 0.5208294666666676

L1 norm success without SkinThickness, BloodPressure, Age, BMI, Insulin,
Pregnancies, DiabetesPedigreeFunction, : 0.7012987012987013
TP: 27
TN: 81
FP: 18
FN: 28
with test computation time: 0.41300154999999883

L1 norm success without SkinThickness, BloodPressure, Age, BMI, Insulin,
Pregnancies, DiabetesPedigreeFunction, Glucose, : 0.6428571428571429
TP: 0
TN: 99
FP: 0
FN: 55
with test computation time: 0.4382961999999999

original l2 norm success: 0.6948051948051948
with computation time: 1.0004776

L2 norm success without Age, : 0.7402597402597403
TP: 35
TN: 79
FP: 20
FN: 20
with test computation time: 0.9716968874999998

L2 norm success without Age, DiabetesPedigreeFunction, : 0.7662337662337663
TP: 39
TN: 79
FP: 20
FN: 16
with test computation time: 0.9339282714285717

L2 norm success without Age, DiabetesPedigreeFunction, Pregnancies, :
0.7597402597402597
TP: 36
TN: 81
FP: 18
FN: 19
with test computation time: 0 .9219466500000001

L2 norm success without Age, DiabetesPedigreeFunction, Pregnancies,
SkinThickness, : 0.7857142857142857
TP: 37
TN: 84
FP: 15
FN: 18
with test computation time: 0.9234535999999991

L2 norm success without Age, DiabetesPedigreeFunction, Pregnancies,
SkinThickness, BloodPressure, : 0.7792207792207793
```

```
TP: 34
TN: 86
FP: 13
FN: 21
with test computation time: 0.8994639249999992

L2 norm success without Age, DiabetesPedigreeFunction, Pregnancies,
SkinThickness, BloodPressure, Insulin, : 0.7792207792207793
TP: 32
TN: 88
FP: 11
FN: 23
with test computation time: 0.8586079000000006

L2 norm success without Age, DiabetesPedigreeFunction, Pregnancies,
SkinThickness, BloodPressure, Insulin, BMI, : 0.7012987012987013
TP: 27
TN: 81
FP: 18
FN: 28
with test computation time: 0.822071949999998

L2 norm success without Age, DiabetesPedigreeFunction, Pregnancies,
SkinThickness, BloodPressure, Insulin, BMI, Glucose, : 0.6428571428571429
TP: 0
TN: 99
FP: 0
FN: 55
with test computation time: 0.7382346000000055
```

The run times are random so a more precise measure could be achieved by running the code multiple times and taking its average. The table of these results are shown below:

| L1 norm | L1 norm success | L1 norm new eliminated | TP | TN | FP | FN |
|---|---|---|---|---|---|---|
| 8 features | 73.38% | | 32 | 81 | 18 | 23 |
| 7 features | 76,62% | SkinThickness | 34 | 84 | 15 | 21 |
| 6 features | 75,97% | BloodPressure | 34 | 83 | 16 | 21 |
| 5 features | 77,27% | Age | 38 | 81 | 18 | 17 |
| 4 features | 78,57% | BMI | 36 | 85 | 14 | 19 |
| 3 features | 80.52% | Insulin | 36 | 88 | 11 | 19 |
| 2 features | 75.79% | Pregnancies | 32 | 85 | 14 | 23 |
| 1 features | 70.13% | DiabetesPedigreeFunction | 27 | 81 | 18 | 28 |
| 0 features | 64.29% | Glucose | 0 | 99 | 0 | 55 |

| L2 norm | L2 norm success | L2 norm new eliminated | TP | TN | FP | FN |
|---|---|---|---|---|---|---|
| 8 features | 69.48% | | 34 | 73 | 26 | 21 |
| 7 features | 74,03% | Age | 35 | 79 | 20 | 20 |
| 6 features | 76.62% | DiabetesPedigreeFunction | 39 | 79 | 20 | 16 |
| 5 features | 75.79% | Pregnancies | 36 | 81 | 18 | 19 |
| 4 features | 78,57% | SkinThickness | 37 | 84 | 15 | 18 |
| 3 features | 77.92% | BloodPressure | 34 | 86 | 13 | 21 |
| 2 features | 77.92% | Insulin | 32 | 88 | 11 | 23 |
| 1 features | 70.13% | BMI | 27 | 81 | 18 | 28 |
| 0 features | 64.29% | Glucose | 0 | 99 | 0 | 55 |

The highest L2 norm can reach is 78.57% with four features while the L1 norm archives 80.52% success with three features. So L1 norm seem like the better distances matric for this setting.

2.4)

The KNN algorithm has no training part as the only training is opening the files. How ever if we want to see how long it takes to do this I measured the time it takes to open the file and normalize it.

Training time: 0.0148s

This is not significant compared to the testing time so I did not create different csv files with different futures to time them. As it is mentioned this being so small is expected as almost nothing is made with the training set other than looking at points in it during the testing phase which is part of the time spend testing.

The testing times of different algorithms are given in the table below:

| Test times | L1 norm | L2 norm |
|---|---|---|
| 8 features | 0.7761s | 1.0004s |
| 7 features | 0.6176s | 0.9717s |
| 6 features | 0.5837s | 0.9339s |
| 5 features | 0.5745s | 0.9219s |
| 4 features | 0.5608s | 0.9235s |
| 3 features | 0.5453s | 0.8995s |
| 2 features | 0.5208s | 0.8586s |
| 1 features | 0.4130s | 0.8221s |
| 0 features | 0.4383s | 0.7382s |

First of all, we can see L2 norms testing takes longer for all number of futures. That is because for L2 norm we are computing square root of sum of squares of the differences which is quite more computationally difficult than just summing absolute values of the differences. Also, even though the times are random there is a clear downwards trend for the number of features used. That is because the less number of futures there are les number of variables are used to compute the distances. It should also be noted a significant time is spent ordering the distances of the point in the training set to choose the smallest 9 in them.

3.1)

In this part I did not do add-one or Laplace smoothing because the zeros did not create any error. So I estimated all the parameters as they are given in the training data.

```
The training computation time for Multiinomial Naive Bayes:
0.26041159999999985

The success for Multiinomial Naive Bayes: 0.9314928425357873
TP: 76
TN: 835
FP: 3
FN: 64
with test computation time: 8.9637885
```

The accuracy of this algorithm is 93.15% and the confusion matrix is given below:

| Confusion matrix | | Actual | |
|---|---|---|---|
| | | + | - |
| Classifier | + | 76 | 3 |
| | - | 64 | 835 |

Even though it misses 64 spam messages which is close to half of them, it only marks 3 normal mails as spam which is quite low. This is very important for us as we wouldn't want our important mail to be marked as spam, because if they are marked we might miss them.

3.2)

The formula for number of estimated parameters for naïve multinomial Bayes is $2n + 1$. $n$ is the number of features. In our setting there are 3458 features so we estimated 6917 parameters.

We need to find the probability of a word existing in a message given the mail is spam and given the mail is normal for every word. This is the $2n$ part of the formula. Then we also need to find the probability of a mail being spam and probability of a mail being normal. However, the probability of mail being normal is 1 minus probability of mail being spam, so we only need to compute only one of them. This is the $+1$ part of the formula.

3.3.a)

In this part I did add-one / Laplace smoothing because without it the Mutual Information calculation gives an error of division by zero. The results are given below:

```
The training computation time for Bernoulli Naive Bayes: 0.1962843000000003

Bernoulli Naive Bayes success for all futures: 0.9631901840490797
TP: 111
TN: 831
FP: 7
FN: 29
with test computation time: 2.7599652

Bernoulli Naive Bayes Success for 100 futures: 0.8486707566462167
TP: 2
TN: 828
FP: 10
FN: 138
with test computation time: 0.05813239999999986

Bernoulli Naive Bayes Success for 200 futures: 0.8333333333333334
TP: 2
TN: 813
FP: 25
FN: 138
with test computation time: 0.11639480000000013

Bernoulli Naive Bayes Success for 300 futures: 0.8302658486707567
TP: 5
TN: 807
FP: 31
FN: 135
with test computation time: 0.23415229999999987

Bernoulli Naive Bayes Success for 400 futures: 0.8312883435582822
TP: 6
TN: 807
FP: 31
FN: 134
with test computation time: 0.2671328000000006

Bernoulli Naive Bayes Success for 500 futures: 0.8425357873210634
TP: 8
TN: 816
FP: 22
FN: 132
with test computation time: 0.2958438999999995

Bernoulli Naive Bayes Success for 600 futures: 0.8404907975460123
TP: 5
TN: 817
FP: 21
FN: 135
with test computation time: 0.3552861000000007
```

The table for of these results are given below:

| Bernoulli Naive Bayes | Success | TP | TN | FP | FN |
|---|---|---|---|---|---|
| 3458 features | 96.32% | 111 | 831 | 7 | 29 |
| 100 features | 83.33% | 2 | 813 | 25 | 138 |
| 200 features | 83.03% | 5 | 807 | 31 | 135 |

I did not write the rest as they are probably not correct.

3.3.b)

As expected the computation time of the testing increases almost linearly with the number of futures. As we are doing probability analysis for each future this behavior is quite expected.  This is even correct if I couldn't select the correct features.

The testing step takes quite shorter amount of time so I did not do it for different number of features but if we did do it would have a computation time proportional to the number of features.

3.4)

Bernoulli Bayes classifier has 96.32% success rate compared to the 93.15% success rate of the Multinomial Bayes classifier. However, it marked 7 normal mails as spam while Multinomial Bayes classifier only marked 3 of them. Depending on the desires of the user of these algorithms either one could be taken as the better one. Personally I would Bernoulli Bayes classifier as I don't get any crucial mails. And I don't want to be bogged down by spam mails.