Gün Kaynar
22101351

1- Probability

1.1

P(heads|A) = P1 P(tails|A) = 1 – P1
P(heads|B) = P2 P(tails|B) = 1 – P2
P(A) = P3 P(B) = 1 - P3

Getting heads with coin A = P1 x P3
1 – (P1 x P3) is the complementary of getting heads with coin A. Which means we do not get heads with coin A.
$(1- (P1 \times P3)^7$ is not getting heads with A for 7 tossing events.
$(1- (P1 \times P3)^7 \times (P1 \times P3)$ is getting our first heads with coin A in the 8^{th} trial.

1.2

Getting heads with A = (P1 x P3)
Getting heads with B = (P2 x (1-P3)

The expected value of getting heads for one trial is (P1 x P3) + (P2 x (1 – P3))
The expected value of getting heads for 10 trials will be **10 x (P1 x P3) + (P2 x (1 – P3))**

1.3a

P(coin=heads) = 0.4 P(coin=tails) = 0.6
P(coin=heads|Oliver=heads) = 0.95 P(coin=tails|Oliver=heads) = 0.05
P(coin=tails|Oliver=tails) = 0.99 P(coin=heads|Oliver=tails) = 0.01
P(Oliver=tails) = 0.01 P(Oliver=heads) = 0.99

To predict the performance let us see P(Oliver=tails|coin=tails) and P(Oliver=heads|coin=heads)

From the Bayes' Rule on the conditional probabilities, P(Oliver=tails|coin=tails) will be equal to:

$$\frac{P(\text{Oliver} = \text{tails}) \times P(\text{coin} = \text{tails}|\text{Oliver} = \text{tails})}{P(\text{coin} = \text{tails}|\text{Oliver} = \text{tails}) \times P(\text{Oliver} = \text{tails}) + P(\text{coin} = \text{tails}|\text{Oliver} = \text{heads}) \times P(\text{Oliver} = \text{heads})}$$

$$\frac{0.01 \times 0.99}{0.99 \times 0.01 + 0.05 \times 0.99} = 0.167$$

P(Oliver=heads|coin=heads) will be equal to:

$$\frac{P(\text{Oliver} = \text{heads}) \times P(\text{coin} = \text{heads}|\text{Oliver} = \text{heads})}{P(\text{coin} = \text{heads}|\text{Oliver} = \text{heads}) \times P(\text{Oliver} = \text{heads}) + P(\text{coin} = \text{heads}|\text{Oliver} = \text{tails}) \times P(\text{Oliver} = \text{tails})}$$

$$\frac{0.99 \text{ x } 0.95}{0.95 \text{ x } 0.99 + 0.01 \text{ x } 0.01} = 0.99$$

Oliver's performance of guessing tails is 0.167 whereas his performance of guessing heads is 0.99.

1.3b

P(Oliver=heads) = 0.99
$(0.99)^8$ = **0.922**

1.3c

We need to calculate P(Oliver=tails|coin=heads) which is:

$$\frac{\text{P(Oliver} = \text{tails) x P(coin} = \text{heads|Oliver} = \text{tails)}}{\text{P(coin} = \text{heads|Oliver} = \text{tails) x P(Oliver} = \text{tails)} + \text{P(coin} = \text{heads|Oliver} = \text{heads) x P(Oliver} = \text{heads)}}$$

$$\frac{0.01 \text{ x } 0.01}{0.01 \text{ x } 0.01 + 0.95 \text{ x } 0.99} = 0.0001$$

2- kNN Diabetes Classifier

2.1

kNN classifier depends mainly on measuring the distance between the test examples and the training examples. The less the distance, the more similar the test sample to the training example. The distance measure options are Euclidean, Manhattan, Minkowski and Cosine distance metrics. However, in our case, either Euclidean or Manhattan will work properly. For Cosine metric we get values between 0 and 1, and Minkowski is used for real-valued vectors. On the other hand, Manhattan is used when there is high dimensionality, or the features are based on real geographical data. We chose Euclidean distance metric here, because it is the default one for kNN algorithm. It gives the length of the straight line between two points in space.

2.2

When there are lots of features, the performance of the classifier drops significantly. Another result of the big number of features is that, because of the unimportant and noisy features the accuracy will decrease. Some features might not bring any information or may be nearly identical to some other features. On the other hand, some features might bring information that we do not desire, i.e., they can corrupt our classifier.

2.3

I have eliminated features on training set and reported the accuracy on test set, then I have eliminated the features on test set, i.e., I used test test as cross validation set, and tried to increase the accuracy of test-set. Both approaches are included in this pdf and in codes as well.

a-) Backward Elimination on Training set

kNN classifier was trained with k = 9. Firstly, all the features were used for the classification and the accuracy of cross validation data was found 0.777, whereas the accuracy of the test set was 0.747.
The time spent on classifying the test set was 0.24 seconds. The confusion matrix was prepared for the first analysis.

	Actual Positive	Actual Negative
Predicted Positive	34	18
Predicted Negative	21	81

The backward elimination method was used. The first eliminated feature has been “Insulin” level. The cross-validation accuracy increased to 0.793, on the other hand the test accuracy increased to 0.766. The algorithm took 0.22 seconds to classify. The confusion matrix can be seen below.

	Actual Positive	Actual Negative
Predicted Positive	31	12
Predicted Negative	24	87

The second feature to get eliminated was “Skin Thickness”. The cross-validation accuracy increased to 0.795 whereas test accuracy dropped to 0.747. The algorithm took 0.21 seconds to classify. The confusion matrix was drawn below.

	Actual Positive	Actual Negative
Predicted Positive	33	17
Predicted Negative	22	82

The third feature which has been eliminated is “Diabetes Pedigree Function”. The accuracy was found 0.795 on cross-validation at this step. The test accuracy stayed the same as 0.747. Therefore, we can ignore this feature and thus the performance of our algorithm increases. The algorithm took 0.20 seconds to classify. The confusion matrix has not changed and can be seen as follows.

	Actual Positive	Actual Negative
Predicted Positive	33	17
Predicted Negative	22	82

The last feature that has been eliminated was “Blood Pressure”. The cross-validation accuracy was found 0.795, and test accuracy has increased to 0.766. The time spent has been 0.16 seconds. The last confusion matrix can be seen below.

	Actual Positive	Actual Negative
Predicted Positive	33	14
Predicted Negative	22	85

b-) Backward Elimination on Test Set

When I used all features in classifying the test set, the accuracy has been 0.721. The time spent was very low (0.03 seconds) because I used numpy and pandas for handling the data and array manipulations. The confusion matrix is as follows.

	Actual Positive	Actual Negative
Predicted Positive	37	25
Predicted Negative	18	74

Then the first feature has been eliminated which was "Insulin" and the accuracy has been found 0.753. 0.03 seconds were spent for classification. The confusion matrix was prepared and can be seen below.

	Actual Positive	Actual Negative
Predicted Positive	35	18
Predicted Negative	21	81

The second feature that was eliminated is "Pregnancies", the accuracy increased to 0.766. The time spent has been 0.03 seconds again. The confusion matrix is as follows.

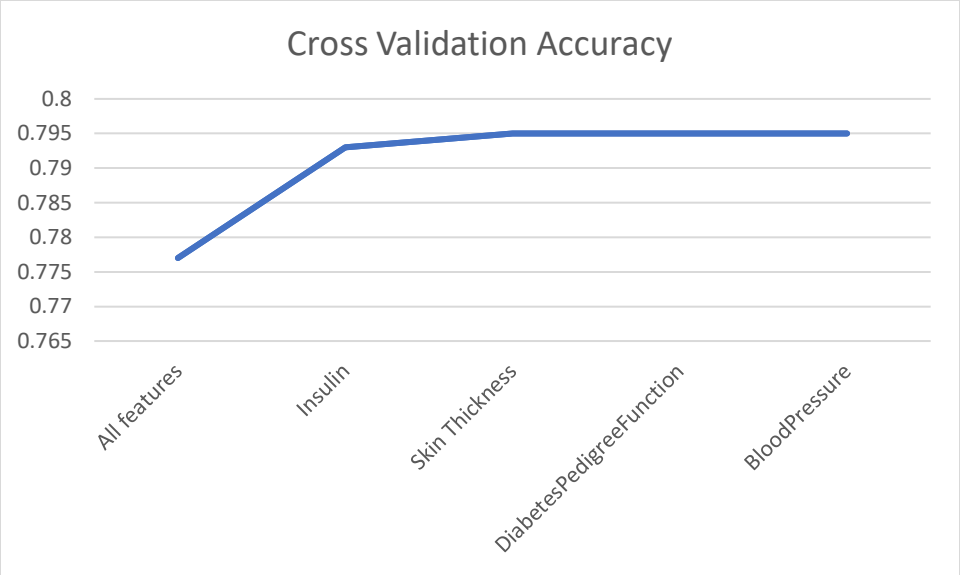
	Actual Positive	Actual Negative
Predicted Positive	35	16
Predicted Negative	20	83

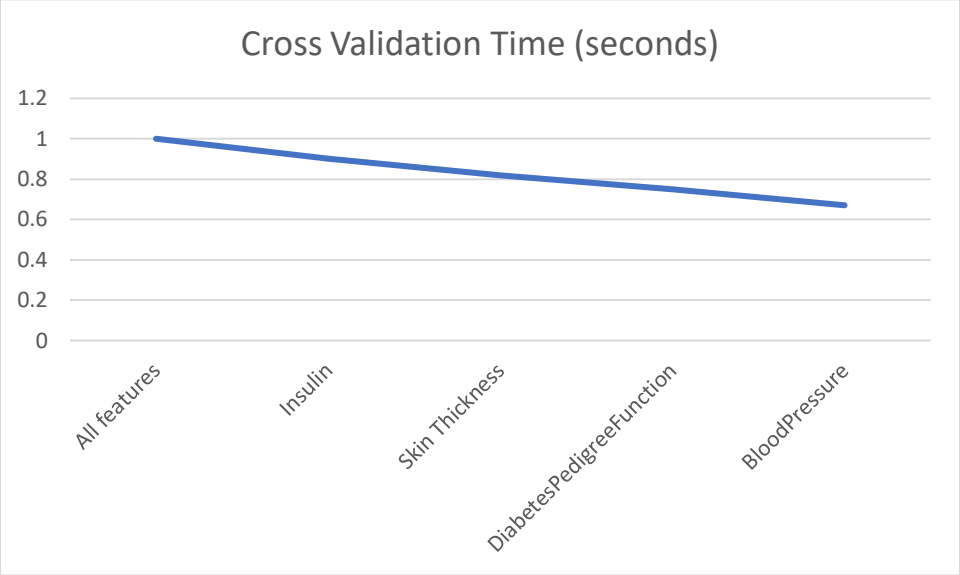
Finally, the last eliminated feature has been "Diabetes Pedigree Function". The accuracy was found 0.766 with the same confusion matrix.

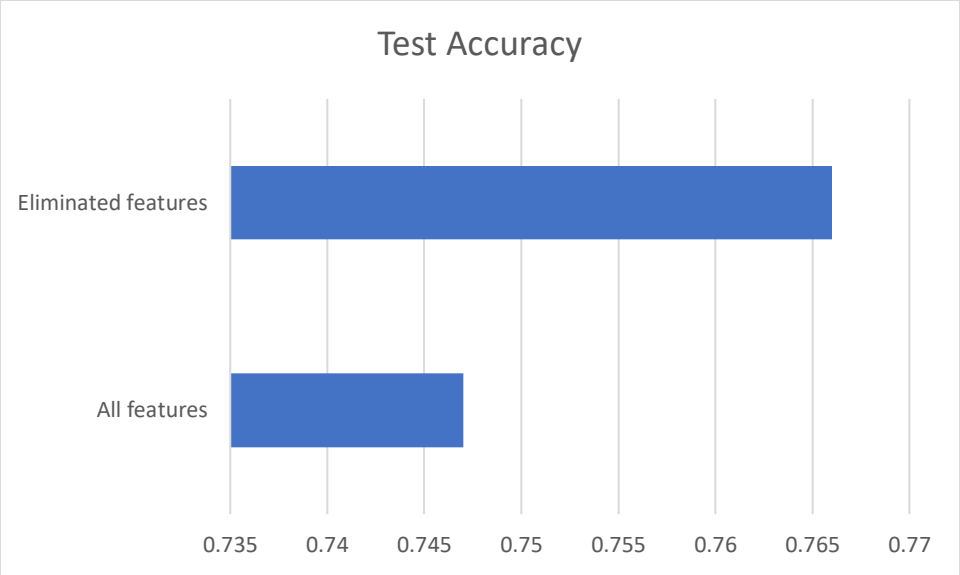
2.4

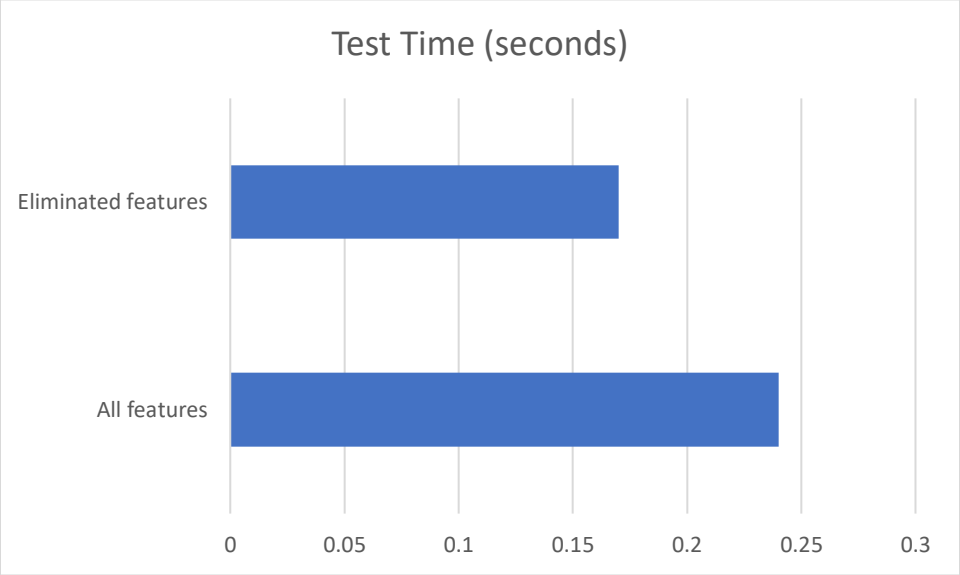
a-) Backward Elimination on Training set

I eliminated features on the training set and the eliminated features were "Insulin", "Skin Thickness", "Diabetes Pedigree Function", "Blood Pressure". The time gained on cross-validation has been 0.328 seconds. And the time gained on test has been 0.075 seconds.

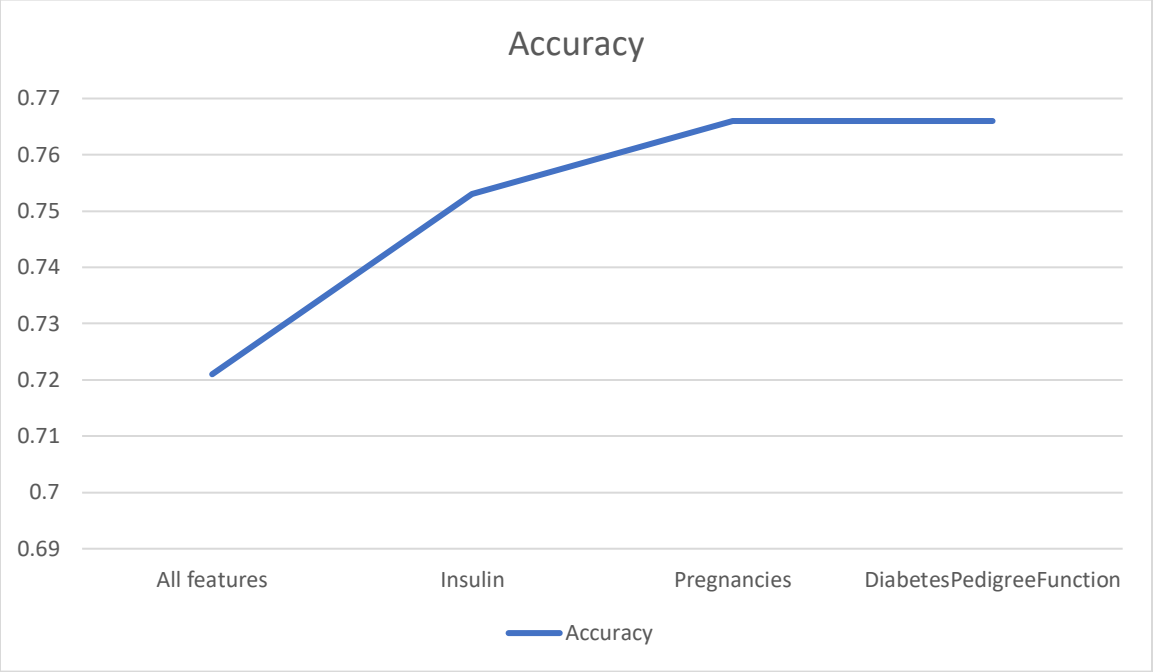

Cross Validation Accuracy
0.8
0.795
0.79
0.785
0.78
0.775
0.77
0.765
All features
Insulin
Skin Thickness
DiabetesPedigreeFunction
BloodPressure


Cross Validation Time (seconds)
1.2
1
0.8
0.6
0.4
0.2
0
All features
Insulin
Skin Thickness
DiabetesPedigreeFunction
BloodPressure


Test Accuracy
Eliminated features
All features
0.735
0.74
0.745
0.75
0.755
0.76
0.765
0.77


Test Time (seconds)
Eliminated features
All features
0
0.05
0.1
0.15
0.2
0.25
0.3

b-) Backward Elimination on Test Set



As the time spent has not changed, I will not include the time difference plot here. The reason is that our test set is small, and the algorithm is very efficient.

3- Spam SMS Detection

3.1

The model was trained by training set and tested on test set. The accuracy of the Multinomial Bayes Classifier was found 0.948. The confusion matrix can be seen below.

	Actual Positive	Actual Negative
Predicted Positive	93	4
Predicted Negative	47	834

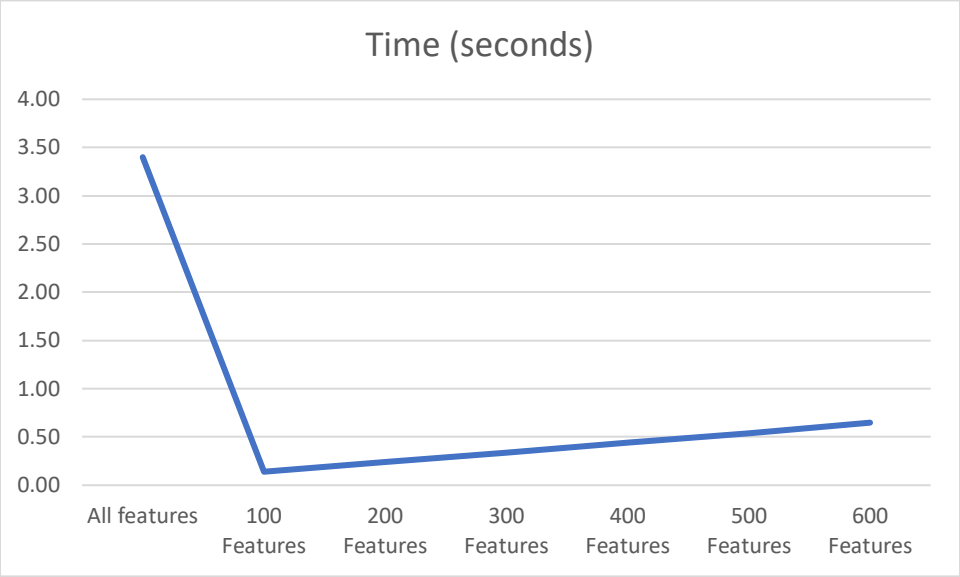
3.2

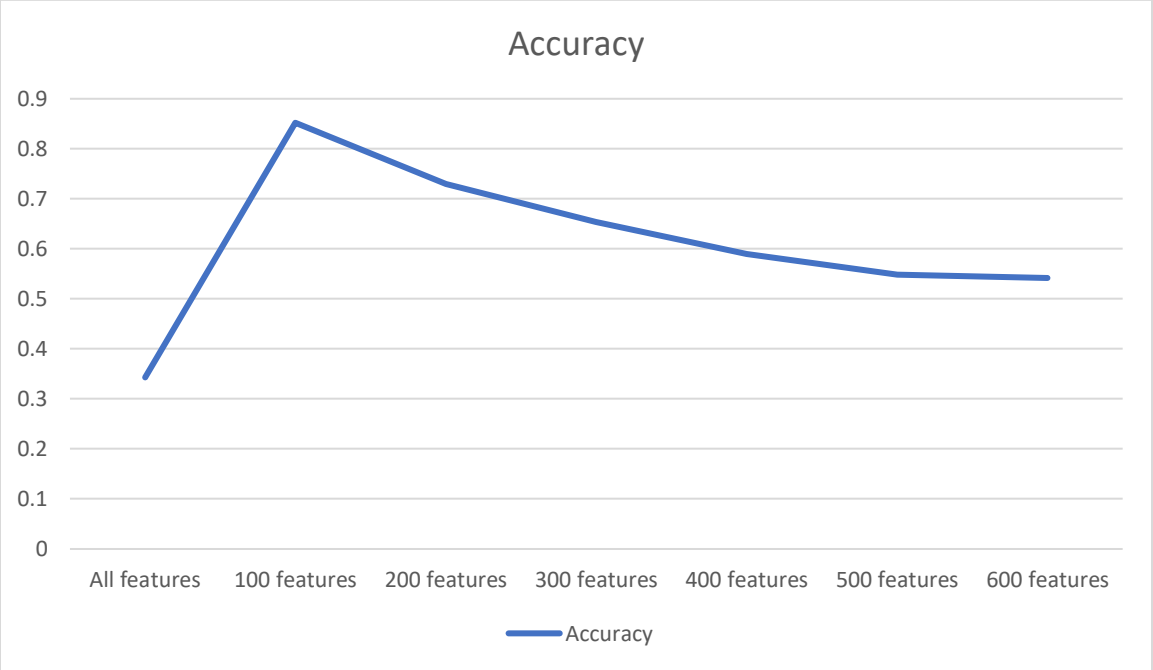
I needed to estimate either P(spam) or P(spam) which are complements of each other. Additionally, 3458 parameters were estimated for $P(x_j|y=spam)$ and $P(x_j|y=ham)$.
So, in total I estimated 3459 parameters for this model.

3.3a

Bernoulli Bayes Classifier was trained on training set and tested on test set. Firstly, I used all the features and accuracy was significantly low compared to Naïve Bayes Classifier. The accuracy was 0.343. Then Mutual Information method was used to eliminate the unnecessary or noisy

features. When first 100 features were used, the accuracy increased significantly to 0.852. The accuracy and the change in time can be seen below when more features are added to the model.





3.3b

As more features are used for the model, more time is spent on the classification. Because our time complexity is O(n) for each SMS where n is the number of words. Adding more feature increases the time needed for each SMS linearly. The time complexity of the whole program is close to $O(n^2)$ if we assume that the number of SMSs is equal to the number of words in each SMS.

3.4

The accuracy of Multinomial Naïve Bayes Classifier model was 0.948. This model is good when we do not apply a feature selection method. Whereas the Bernoulli Naïve Bayes Classifier model's accuracy was 0.343 when we used all the features. It can be said that if Bernoulli will be used, feature selection is a must.

On the other hand, Multinomial model was trained much faster than the Bernoulli. The difference in time needed for both algorithms was much higher before feature selection by mutual information on Bernoulli. After the feature selection was done, the time difference has become negligible. The advantage of the Bernoulli model is that it can do what Multinomial does by using only 100 features.

I use Multinomial when the frequency of words matters. In this case, the number of words in an SMS will be very important to predict whether the SMS is spam, because if you consider spam SMSs, you will see that they contain repeats of words like "dollar", "money" and "promotion".