Bilkent University

CS 464

Homework 1 Report

Can Kırımca
ID: 21802271

Question 1.1

P(Choosing A) = P3

P(Choosing B) = 1 – P3

P(Getting heads | A was chosen) = P1

P(Getting heads | B was chosen) = P2

P(A, head) = P3 * P1

We can use geometric probability distribution to calculate the probability of getting the first (A, head) pair in the 8th trial. Let T = (Getting the first (A, head) pair in the 8th trial).

$$P(T) = (1 - P(A, head))^7 * P(A, head) = (1 - P1 * P3)^7 * (P1 * P3)$$

Question 1.2

Let H_i = Obtaining heads in the i^{th} toss.

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

Expected value of heads in 8 coin tosses:

$$E(H) = \sum_{i=1}^{8} P(H_i) = 10 * (P1 * P3 + (1 - P3) * P2)$$

Question 1.3.a

Let the events OT, OH, H and T denote:

OT: Oliver predicting tails

$$\boldsymbol{P(OT) = 0.01}$$

OH: Oliver predicting heads

$$\boldsymbol{P(OH) = 0.99}$$

Also,

$$\boldsymbol{P(T|OT) = 0.99}$$

$$\boldsymbol{P(H|OH) = 0.95}$$

Using these probability values, we can calculate the overall probabilities of Oliver's correct guesses.

$$P(T, OT) = P(T|OT) * P(OT)$$

$$P(H, OH) = P(H|OH) * P(OH)$$

Using these probability values, we can get the ratio of Oliver's correct guesses to all of his guesses. This ratio gives the overall accuracy of Oliver's guesses. Also, we can validate the probabilities by observing Oliver's coin tosses and check whether the correct guess ratio is converging to the probabilities mentioned above.

Question 1.3.b

Let H = Oliver predicting Heads

$$P(H) = 0.99$$

Let M = Oliver predicting 8 heads in a row

$$P(M) = (0.99)^8 = 0.9227$$

Question 1.3.c

$$P(OT|H) = \frac{P(OT, H)}{P(H)} = \frac{P(OT, H)}{P(OT, H) + P(OH, H)}$$

We need to calculate P(OT,H) and P(OT,T).

$$P(OT, H) = P(H|OT) * P(OT) = 0.01 * 0.01 = 0.0001$$

$$P(OH, H) = P(H|OH) * P(OH) = 0.95 * 0.99 = 0.9405$$

Plug in these values to the first equation, we get:

$$P(OT|H) = \frac{0.001}{0.0001 + 0.9405} = \frac{0.0001}{0.9406} \sim 0.000106$$

Question 2.1

For this task, I used euclidean distance metric to calculate the distances between the samples. Since the samples can be considered as real valued vectors, using this approach was convenient over other options. Furthermore, since there was not too many features used in this model, using this metric did not cause too much computational complexity.

Question 2.2

We may not want to consider all features in a dataset, because while some features can be irrelevant of the outcome and using those features may cause unnecessary processing time and lower accuracy.

Question 2.3

For this question, I implemented a kNN classifier in Python. I used backward elimination as the feature selection method. Backward elimination increased the accuracy of the model twice. In the first step, the feature “Insulin” and in the second step, the feature “Pregnancies” were eliminated from the feature set. For each step, the confusion matrix and the accuracy is listed below:

Step 0 (With all features present):

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

Accuracy: 0.7207792207792207

Step 1:

	Actual Positive	**Actual Negative**
Predicted Positive	**35**	**18**
Predicted Negative	**20**	**81**

Accuracy: 0.7532467532467533

Step 2:

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

Accuracy: 0.7662337662337663

Question 2.4

For this question, the training and validation times of the kNN classifiers are recorded and analyzed. These durations can be observed in the following table:

	Training time (in seconds)	Validation time (in seconds)
With Insulin eliminated (Step 1)	0.000997781753540039	0.993765115737915
With Pregnancies eliminated (Step 2)	0.000997304916381836	0.8938169479370117

As can be observed in the table, the training time is O(1) and validation time depends on the number of features used in the model. Therefore, the validation time decreases in the second step but the training time is constant.

Question 3.1

In this question, a multinomial Naive Bayes model is implemented using Python. The implemetation is included in the submission file. The confusion matrix and accuracy of the model is the following:

	Actual Positive	**Actual Negative**
Predicted Positive	**99**	**14**
Predicted Negative	**41**	**824**

Accuracy: 0.943762781186094

Question 3.2

We have two parameters for each word that indicate the occurences of that word in ham or spam messages. For that purpose, for a vocabulary with 3458 words, we have 3458*2 = 6916 variables. We also need to estimate the number of spams and hams, which are two extra parameters.

Question 3.3.a

For this question, a Bernoulli Naive Bayes classifier was implemented in Python. This classifier used mutual information as the feature selection method. In each step, the training time and accuracy was recorded and analyzed. The mutual information method was implemented as follows: The mutual information of each feature was calculated at the beginning and was put into an array. Then, this array was sorted in descending order so that the features with the greatest information will appear in the beginning. Then, train and test functions use only the 100, 200, ..., 600 features with the greatest mutual information.

For the implementation of the train function, I designed two algorithms one of which is commented out in the code. One of them uses a double for loop and the other one uses a single for loop. In the submission, the one with the double for loop is commented out since it works relatively slower. However, runtime analysis was performed for both implementations. The accuracy was the same for both implementations.

ABOUT THE WARNINGS: While executing the code of this question, Python gives RuntimeWarning about the values in the logarithm (divide by zero etc.). These are handled in the code, however, these warnings still appear in the output. To ignore them, one should execute the code with the command given in the README.txt file.

Analysis of double for loop implementation:

	Accuracy	**Training time (in seconds)**
Step 1	0.9601226993865031	2.5234
Step 2	0.9693251533742331	2.9765
Step 3	0.9611451942740287	3.4265
Step 4	0.9591002044989775	4.7608
Step 5	0.9621676891615542	5.2317
Step 6	0.9621676891615542	5.8418

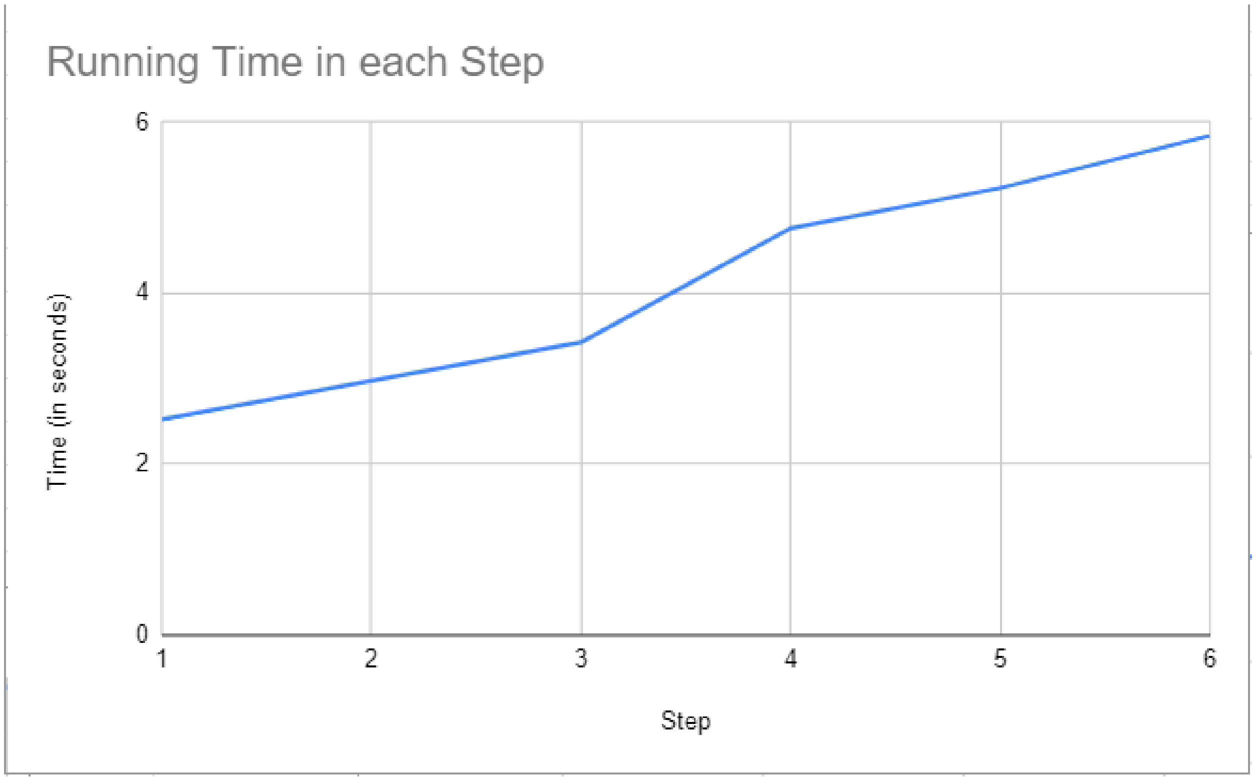

Fig. 1. The running time of the double for loop implementation

As can be seen in Fig. 1, the running time increases in a linear way as the number of features increase.

Analysis of single for loop implementation:

	Accuracy	Training time (in seconds)
Step 1	0.9601226993865031	0.4912
Step 2	0.9693251533742331	0.5099
Step 3	0.9611451942740287	0.5235
Step 4	0.9591002044989775	0.5705
Step 5	0.9621676891615542	0.5571
Step 6	0.9621676891615542	0.5926

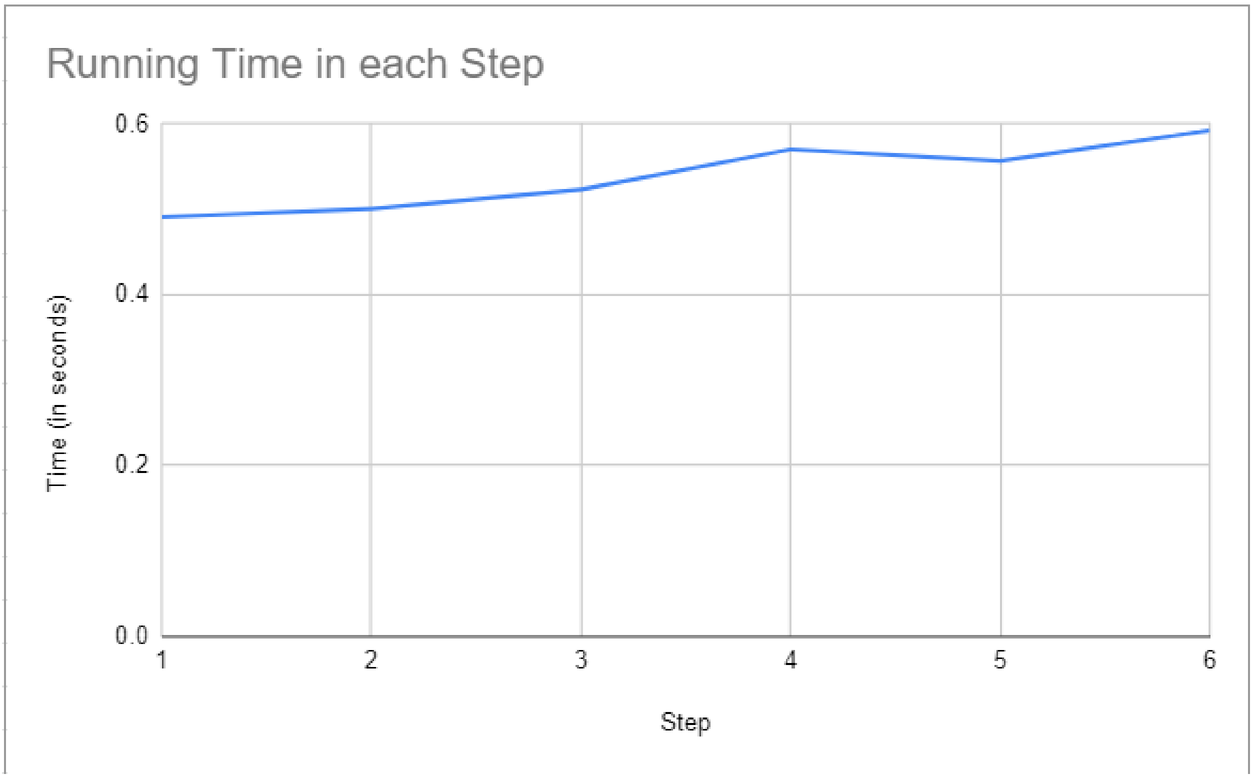

Fig. 2. The running time of the single for loop implementation

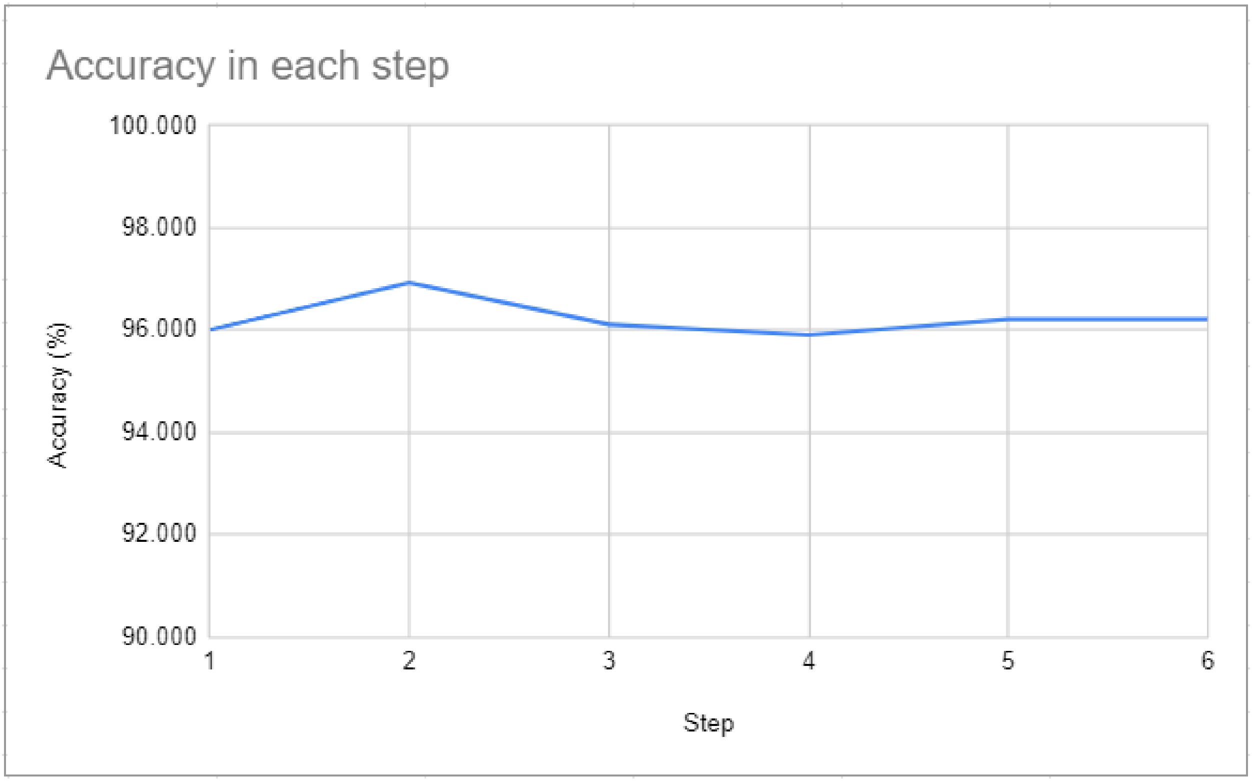

Fig. 3. The accuracy of the model

Question 3.3.b

As explained in the previous question, the time complexity of the model increases in each step linearly as the number of features increases. The reason is that the model checks more parameters as new features are added to the system. This is valid for both of my implementations (single and double for loop implementations).

Question 3.4

Best accuracy of the Bernoulli model (In step 2): 0.9693251533742331

The accuracy of the Multinomial model: 0.943762781186094

In this comparison, the Bernoulli model performs better than the multinomial model. However, this comparison is not ideal in order to compare the performance of two models.

The multinomial model does not use any feature selection method. Instead, uses all features to predict the result. Therefore, an experiment where both models use the same feature set would be more informative for a comparison the models' performances.