Muhammed Şamil Arınç
21803683
07.11.2021

Answers:

1.1. Probability of getting heads from coin A is P1 and the probability to select A is P3. Hence, the probability of getting (A, heads) pair is P1*P3 and not getting it is 1-P1*P3. The probability of this event to happen 7 times in a row is $(1\text{-}P1*P3)^7$. After this event, (A, heads) should occur. Thus, the overall probability is $(1\text{-}P1*P3)^7$ *P1*P3.

1.2. The probability of heads from A is P1*P3 and from B is (1-P3)*P2. Therefore, the total probability of getting heads is P1*P3+(1-P3)*P2. Using the Bernoulli random variable and expected value sigma sum formula:
Let $P_x = P1 \cdot P3 + (1 - P3) \cdot P2$:

$$E[P_x] = \sum_{k=0}^{10} k \cdot C(10, k) \cdot P_x^k \cdot (1 - P_x)^{10-k}$$

1.3.b. Let probability of heads P(H), probability of tails P(T) and variable Y which is Oliver's prediction (either 'H' or 'T'). Using Bayes' Rule:

$$P(Y = H|H) = \frac{P(Y = H) \cdot P(H|Y = H)}{P(Y = H) \cdot P(H|Y = H) + P(Y = T) \cdot P(H|Y = T)}$$

And since all probabilities in the above formula is known, it can be calculated as:

$$P(Y = H|H) = \frac{0.99 \cdot 0.95}{0.99 \cdot 0.95 + 0.01 \cdot 0.01} = \frac{9405}{9406} = 0.9998937$$

1.3.c. This time the question asks the value which can be defined as P(Y=T|H) which is equal to:

$$P(Y = T|H) = P(Y = H|H)^c = 1 - P(Y = H|H) = \frac{1}{9406} = 1 - 0.9998937 = 0.0001063$$

2.1. For the distance metric, I used the Euclidean Distance method. Other distance metrics that I research about was Manhattan distance and Hamming distance. I eliminated Manhattan Distance since we don't have many features in our dataset so it is very likely to underfit. Furthermore, the Hamming distance is used to calculate the likelihood of a point with respect to a specified point which means that it does not calculate the distance between points A and B but calculates the distances from A to C, B to C and compares them. Thus, it may not be useful for kNN method. Overall, Euclidean distance seem more useful to me.

2.2. Since some of the data may not be useful at all or might be irrelevant with the problem we are dealing with, they may not change the result or decrease the score. Hence, we want to eliminate them by testing our data's performance with and without using them. If the elimination of some feature increases the model's score, it seems reasonable to delete this feature in our calculations.

2.3. Firstly, I wrote the kNN code and it eliminates 4 features using Backward Elimination since some of them do not affect score and some of them decreases it. However, after some time thinking about it, I figured out that since the data ranges change between features, it might be more logical to normalize all the data in 0-1 range. This way, all the features will have similar effects on Euclidean Distance calculation.

Deleted Feature	Score %	Elimination Time (s)	Test Time (s)
Nothing – Train	74.27	None	3.25
Nothing – Test	72.08		0.96
Pregnancies – Train	74.76	32.44	2.98
Pregnancies – Test	71.43		0.89
Insulin – Train	74.76	25.93	2.69
Insulin – Test	76.62		0.81
Skin Thickness – Train	75.41	19.96	2.44
Skin Thickness – Test	75.97		0.76
Diabetes Pedigree Function – Train	75.41	11.69	2.31
Diabetes Pedigree Function – Test	75.97		0.70

Table 1. First Trial Before Normalization

Deleted Feature	Score %	Elimination Time	Test Time
Nothing – Train	73.45	None	3.30
Nothing – Test	74.03		1.08
Skin Thickness – Train	77.36	32.41	2.88
Skin Thickness – Test	77.92		0.91

Table 2. After Normalization

2.4. It can be seen on the table that the elimination time and test time is decreasing with increasing number of eliminated features. It may seem logical since the number of calculations done during the prediction process is also decreasing with decreasing number of available features. Besides, the normalization process seems very reasonable since the overall performance of the model is increased overall. This is because, when normalization is not done, some of the data has very significant effects on the prediction while others have less. Normalization makes the data ranges equal for all the features and it leads to better results in the model.

Apart from that, after the Skin Thickness data is deleted, the accuracy of the model increases significantly for the after-normalization model. Therefore, it can be concluded that the Skin Thickness may not be relevant with diabetes. Hence, the elimination of irrelevant features increases the performance as expected.

3.1. While trying to analyze the data given, I figured out that there are some words that are not occurring in both the spam labeled messages and ham labeled messages. Therefore, they will not have an effect on the prediction of our model. This is because, these words cannot give any information about the category of the message either spam or ham. Hence, I deleted these words from data in the first step.

After preprocessing the data, I trained my model with the given data and equations. The confusion matrix of the model is:

Training Set		Actual	
		1	0
Prediction	1	581	50
	0	16	3264
Score:		98.31	
Time:		0.7412	

Test Set		Actual	
		1	0
Prediction	1	108	18
	0	32	820
Score:		94.89	
Time:		0.1902	

Table 3. Confusion Matrixes

3.2. Since we are trying to find Spam or Ham which needs binary outputs, the parameters we need during the prediction process are $P(X_j = x_j \,|Y = y_k)$ where y_k is either '0' or '1'. Since j goes from 1 to number of words, and k goes 0 to 1, the number of parameters is (# of words) * 2 from this term. Also, there is another term in the formula which is $Y = y_k$where k is either 0 or 1 and the number of parameters coming from here is 2. Hence, the number of parameters we need to estimate is:

Let N is the number of words in vocabulary list:

$$2 \cdot N + 2$$

Nonetheless, since $Y = 0$ is complement of $Y = 1$, we can use them as one parameter if needed.

3.3.a.

# of features and set	Model Score %	Time
600 – Train	97.09	15.10
600 – Test	94.68	3.88
500 – Train	96.86	12.78
500 – Test	94.58	3.27
400 – Train	96.60	10.15
400 – Test	94.27	2.60
300 – Train	96.01	7.82
300 – Test	94.48	2.00
200 – Train	95.30	5.16
200 – Test	94.58	1.32
100 – Train	93.15	2.79
100 – Test	91.92	0.69

Table 5. Score and Time for different number of features

Train Set		Actual	
		1	0
Predicted	1	570	87
	0	27	3227

Test Set		Actual	
		1	0
Predicted	1	126	38
	0	14	800

Table 6. Confusion Matrixes when # of features is 600

3.3.b. The time elapsed for algorithm to finish increases with increasing number of features. When the number of features used to predict new data changes, obviously the time it takes computer to calculate the prediction parameters change accordingly. This is because, the loop inside prediction function iterates number of features times and it is the main loop that creates complexity.

3.4. For Multinomial Naïve Bayes Classifier, the time it takes to predict a data is extremely lower than Bayesian Naïve Bayes Classifier. Furthermore, even if their scores are very similar to each other, Multinomial's score is still a bit higher than Bayesian. Even if I expect Bayesian to take less time to predict as it uses 600 parameters while Multinomial uses all 3458 features, however, it turns out that it is slower. Hence, it may seem plausible to use Multinomial Model over Bernoulli Model.