HW_3 Report - Feven, Mihret, Tagore , Group -1

PART1

Naive Bayes Theorem

Naive Bayes is a probabilistic classifier based on Bayes' theorem with the assumption of independence between features. The theorem can be expressed as follows:

$$P(C_k|x_1, x_2, \ldots, x_n) = \frac{P(C_k) \cdot P(x_1|C_k) \cdot P(x_2|C_k) \cdot \ldots \cdot P(x_n|C_k)}{P(x_1) \cdot P(x_2) \cdot \ldots \cdot P(x_n)}$$

Where:

- $P(C_k|x_1, x_2, \ldots, x_n)$ is the posterior probability of class C_k given the features $x_1, x_2, \ldots, x_n$.
- $P(C_k)$ is the prior probability of class C_k.
- $P(x_i|C_k)$ is the likelihood that feature x_i belongs to class C_k.
- $P(x_i)$ is the probability of feature x_i occurring in the dataset.

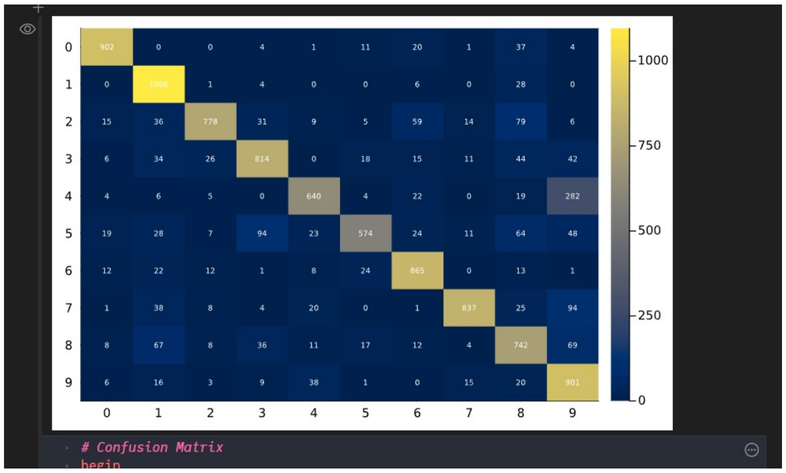
For the given problem, we are assuming that $P(c = 0) = P(c = 1) = \ldots = P(c = 9)$. Let $x = (x_1, x_2, \ldots, x_{784})$ be the vector of pixel values for a given image, and c is the class or digit, 0 to 9. Hence,

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \approx \prod_{i=1}^{784} P(x_i|c),$$

where $P(x_i|c) = N(\mu_i, \sigma_i|c)$ is modeled as a Gaussian distribution from the training data.
By using the **Training Gaussian Function** we are calculating the **means** and **variances** of features for each class in the training dataset. It iterates over unique labels and calculates means and variances for each feature.

By using the **Prediction Gaussian Function** we are calculating the **log-likelihoods** for each class given the features of the test dataset. It iterates over each sample in the test dataset and calculates log-likelihoods for each class.

Then the trained model is applied on the test dataset and computes the accuracy of the predictions. By using **Confusion matrix** we can evaluate the performance of the classifier for each digit class.



	0	1	2	3	4	5	6	7	8	9
0	902	0	0	4	1	11	20	1	37	4
1	0	1096	1	4	0	0	6	0	28	0
2	15	36	778	31	9	5	59	14	79	6
3	6	34	26	814	0	18	15	11	44	42
4	4	6	5	0	640	4	22	0	19	282
5	19	28	7	94	23	574	24	11	64	48
6	12	22	12	1	8	24	865	0	13	1
7	1	38	8	4	20	0	1	837	25	94
8	8	67	8	36	11	17	12	4	742	69
9	6	16	3	9	38	1	0	15	20	901

```
# Confusion Matrix
begin
```

Figure 1: Confusion matrix.

The Accuracy for each digit class is printed out in the programming file. This provides insights into how well the model performs on each digit individually. For example, digits 1 and 0 have higher accuracies 96.56 and 92.04 respectively, indicating that the model is better at distinguishing these digits. On the other hand, digits 4 and 5 have lower accuracies (65.17 and 64.35 respectively), meaning model struggles more with these digits.

Heatmaps of the mean pixel values for each digit class are plotted. This visualization allows us to see the average appearance of each digit according to the model. It can help us in understanding which features are important for distinguishing between different digits. The overall Accuracy of the model is 81.49. This indicates the percentage of correctly classified instances in the test dataset compared to the total number of instances.

PART 2A

We used ID3 method, a well-known decision tree approach which works by choosing the best characteristic at each node to partition data depending on the information gain. So we has to calculate the Information gain for each attribute and choose the attribute with high information gain as a splitting attribute. So let's calculate the information gain of each attribute. The first thing we should do is we need to compute the total Entropy of the whole dataset and entropy of each nodes.

Entropy Calculation

Let S be a dataset with n classes, and let p_i be the proportion of samples belonging to class i. The entropy of S, denoted by $H(S)$, can be calculated as follows:

$$H(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Where $p_i \log_2(p_i)$ is taken to be 0 whenever $p_i = 0$.

So let's calculate the information gain of each attribute. The first thing we should do is we need to compute the total Entropy of the whole dataset and entropy of each nodes.

Information Gain Calculation

Let $H(S)$ be the entropy of the original dataset S, $H(S|A)$ be the conditional entropy of S given attribute A, and $IG(S, A)$ be the information gain when splitting S on attribute A. The information gain can be calculated as follows:

$$IG(S, A) = H(S) - H(S|A)$$

Where:

$$H(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

$$H(S|A) = \sum_{j=1}^{m} \frac{|S_j|}{|S|} \times H(S_j)$$

Here, n is the number of classes in the dataset S, p_i is the proportion of samples belonging to class i, m is the number of distinct values of attribute A, S_j is the subset of S where attribute A has the j-th value, and $|S_j|$ represents the number of samples in subset S_j.

By applying those formula we were able to find the information gain of each attribute of the following (they are rounded to 2 decimal place) Information gains: $k_1 = 0.48, f = 0.02, c = 0.10, n_1 = 0.23, e = 0.007, x = 0.049, p_1 = 0.91, t = 0.19, w_1 = 0.24, s = 0.03, p_2 = 0.0, w_2 = 0.02, w = 0.25, s_3 = 0.20, k = 0.42, o = 0.03, s_1 = 0.28, s_2 = 0.272, p_3 = 0.318, e_1 = 0.13, u = 0.16, n = 0.04$ And comparing all the above information gains p_1 attribute got the highest information gain from all other attributes.
So the optimal attribute for the root of the decision tree will be p_1.

The number of entries that end up in each part of the split are:
Number of entries for p_1 = a:400
Number of entries for p_1 = l:400
Number of entries for p_1 = p:255
Number of entries for p_1 = n:3528
Number of entries for p_1 = f:2160
Number of entries for p_1 = c:192
Number of entries for p_1 = y:576
Number of entries for p_1 = s:576
Number of entries for p_1 = m:36

PART 2B

By using the Decision Tree package on the Mushrooms dataset and we built a decision tree. The maximum depth of the decision tree is set to 5, the complexity is limited to the number of nodes that can be accommodated within this depth. So, in this case, **the complexity would be a function of** 2^5, which is 32 nodes at maximum. we performed a test/train split and evaluate on the test data, we got

Accuracy on the test set: 0.978