Narendra Gautam Sontu ML_HW2 002241534

[Link]

Q1) 1. **ML estimate of the mean ˆμ**

To estimate the mean (μ^) of a normally distributed random variable X∼N (μ, σ^2) from a dataset D={x_1} containing just a single sample, we turn to the concept of Maximum Likelihood Estimation (MLE). MLE aims to find the parameter values that maximize the likelihood function, which in turn reflects the probability of observing the given sample data under the assumed model.

For a normal distribution, the likelihood function L(μ, σ^2|x) for observing a single sample x_1 is given by the probability density function of the normal distribution:

$$L(\mu, \sigma^2|x_1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_1-\mu)^2}{2\sigma^2}\right)$$

To find the MLE of μ, we need to maximize this likelihood with respect to μ. This often involves taking the natural logarithm of the likelihood function to simplify the calculations, turning the product into a sum, which is easier to differentiate. However, since we only have a single observation, the log-likelihood simplifies to:

$$\log L(\mu, \sigma^2|x_1) = -\frac{1}{2}\log(2\pi\sigma^2) - \frac{(x_1-\mu)^2}{2\sigma^2}$$

Taking the derivative of this log-likelihood with respect to μ and setting it to zero for maximization yields:

$$\frac{d}{d\mu}\log L(\mu, \sigma^2|x_1) = \frac{x_1-\mu}{\sigma^2} = 0$$

Solving for μ, we find that μ^ =x1. Therefore, the MLE of the mean μ is simply the value of the single observed sample.

In the context of Maximum Likelihood Estimation, when we're dealing with a normal distribution and our dataset consists of just one sample, the process intuitively suggests that the best estimate for the mean (μ^) of the distribution is the observed value itself (x_1). This is because, with only one observation, our best guess at where the center of the distribution lies is at the point where we observed data. There's no other data point to suggest that the center (mean) of the distribution should be anywhere else. Hence, in this scenario, the observed sample directly serves as the ML estimate of the distribution's mean.

2. (Biased) ML estimate of the variance ˆσ2

When calculating the Maximum Likelihood Estimation (MLE) for the variance σ$^{\wedge 2}$ of a normally distributed random variable X∼N(μ,σ^2) from a dataset D={x_1 } that contains just a single sample, we follow a similar process to that used for estimating the mean. However, it's crucial to note that the resulting estimate of the variance will be biased in this case.

For a normal distribution, the likelihood function $L(\mu,\sigma^2|x)$ for observing a single sample x_1 is expressed by the probability density function:

$$L(\mu, \sigma^2|x_1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_1-\mu)^2}{2\sigma^2}\right)$$

Given that we have already determined the MLE of the mean to be $\mu\hat{} = x_1$, we can proceed to estimate the variance. To find the MLE of σ^2, we maximize the likelihood function with respect to σ^2. After taking the natural logarithm of the likelihood function, the log-likelihood becomes:

$$\log L(\mu, \sigma^2|x_1) = -\frac{1}{2}\log(2\pi\sigma^2) - \frac{(x_1-\mu)^2}{2\sigma^2}$$

Substituting $\mu\hat{}$ for μ, and differentiating with respect to σ^2, and setting it to zero, we get:

$$\frac{d}{d\sigma^2}\log L(\mu, \sigma^2|x_1) = -\frac{1}{2\sigma^2} + \frac{(x_1-\hat{\mu})^2}{2(\sigma^2)^2} = 0$$

Given that $\mu\hat{} = x_1$, the term $(x_1 - \mu\hat{})^2$ becomes $0$0, which complicates direct estimation from a single sample since variance measures the spread of data around the mean. In a typical scenario with multiple data points, the MLE of the variance $\sigma\hat{}^2$ for n samples would be:

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i-\hat{\mu})^2$$

However, for a single observation, this formula does not directly apply as it would lead to an estimate of 00, which does not provide a meaningful measure of spread.

3. Corresponding unbiased estimate of the variance:

When estimating the variance from a sample, the unbiased estimate is crucial because it corrects the bias introduced by the sample size in the Maximum Likelihood Estimation (MLE) of variance. However, with only a single sample point $D=\{x_1\}$), the situation is unique.

For a dataset $D=\{x_1, x_2, \ldots, x_n\}$ drawn from a normal distribution N(μ, σ2), the unbiased estimator of the population variance σ2 is given by:

$$\hat{\sigma}^2_{\text{unbiased}} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\hat{\mu})^2$$

This formula uses $n-1$ in the denominator instead of n, which corrects for the bias in the MLE of variance. The $n-1$ term is known as Bessel's correction, and it provides an unbiased estimate of the population variance by increasing the variance of the sample.

However, if the dataset D contains only a single sample, the formula for the unbiased estimate poses a problem:

- With only one sample, $n=1$, making the denominator $n-1=0$, which leads to division by zero. This mathematical issue reflects a deeper conceptual point: with only a single data point, it's impossible to estimate the spread or variance of the underlying population because there's no variation within the sample to measure.

4. Which Makes More Sense?

Given these considerations, neither estimate is truly meaningful nor practical when dealing with a single data point. However, if we must choose between them based on conceptual alignment with statistical principles:

The MLE approach technically produces a result (00), reflecting the absence of variability in a single-point dataset. This result, while mathematically defined, is misleading in a real-world context because it implies certainty (zero variance) that we cannot justifiably claim from a single observation.

The unbiased estimate fails to produce a result due to division by zero, which, paradoxically, may be seen as more sensible from a theoretical perspective. This failure underscores a fundamental truth in statistics: you cannot estimate variability from a single observation. The inability to compute an unbiased estimate reinforces the concept that variance estimation requires observations of spread or dispersion, which a single data point cannot provide.

Q3) 1. (a) MAP Estimation

For MAP estimation with a Dirichlet prior, the discriminant function for class c is:

$$g_c(x) = \log \pi_c + \sum_{j=1}^{d} \sum_{k=1}^{K_j} I_{j|k} \cdot \log \theta_{j|k|c}$$

Where:

- π_c *is* the class prior MAP estimate, given by:

$$\pi_c = \frac{N_c + \alpha_c - 1}{n + \sum_{c'} \alpha_{c'} - K}$$

- $\theta_{j|k|c}$ is the feature MAP estimate for feature j taking on value k under class c, given by:

$$\theta_{j|k|c} = \frac{N_{j|k|c} + \alpha_{j|k} - 1}{N_c + K_j \alpha_{j|k} - K_j}$$

- $I_{j|k}$ is an indicator function that is 1 if feature j takes on value k for instance x, and 0 otherwise.

where:

- $\boldsymbol{N_{kc}}$ is the number of times category *k* appears in feature *j* for class *c*,
- $\boldsymbol{N_c}$ is the total count of all categories for feature *j* in class *c*,
- $\boldsymbol{\alpha_k}$ is the prior (Dirichlet distribution parameter) for category *k*,
- The sum in the denominator iterates over all possible categories *k′* for feature *j*.

(b) Full Bayesian Estimation

For full Bayesian estimation, instead of point estimates, we integrate over all possible values of the parameters, weighted by their posterior probabilities. This often involves approximations or numerical methods since the exact integration can be complex. The predictive distribution can be expressed as an expectation:

$$P(x_j = k|C = c) = \int P(x_j = k|C = c, \theta)P(\theta|D)d\theta$$

where:

- $\boldsymbol{P(\theta|D)}$ is the posterior distribution of the parameters given the data *D*,
- $\boldsymbol{P(x_j = k|C = c, \theta)}$ is the likelihood of observing category *k* given parameters θ,
- The integration is over all possible parameter values θ.

The expected value of θ_{ik} under the posterior distribution, which can be used for prediction, is then:

$$E[\theta_{ik}] = \frac{\alpha'_k}{\sum_{k=1}^{K} \alpha'_k}$$

This is the full Bayesian estimate of the parameter θik , representing the probability of category k for feature i after observing the data and updating the beliefs according to the Dirichlet prior.

Difference with Uninformative Prior

When the Dirichlet prior is uninformative (i.e., each αj|k is the same and very small), the MAP estimates for π_c and $\theta_{j|k|c}$ converge to the frequencies in the data. Specifically:

- The class prior π_c simplifies to the relative frequency of class c in the data.
- The feature probabilities $\theta_{j|k|c}$ simplify to the relative frequencies of the feature values within each class.

In the case of an uninformative prior, the difference between MAP and full Bayesian estimation is minimal because the prior does not contribute significant information, and both estimates would rely heavily on the observed data. For full Bayesian estimation, the posterior distribution would closely resemble the likelihood, as the uninformative prior has little to no impact on the posterior.

2. Mushrooms Dataset:

In this data set descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family (pp. 500- 525). Each species is identified as edible, poisonous, or unknown edibility and not recommended. This latter class was combined with the poisonous one.

Creating a random split 80% of the dataset into training set and 20% into a test set. Training a categorical Na ̈ive Bayes classifier with different values of smoothing hyperparameter α, common amongst all features, ranging from 2−15 to 25.

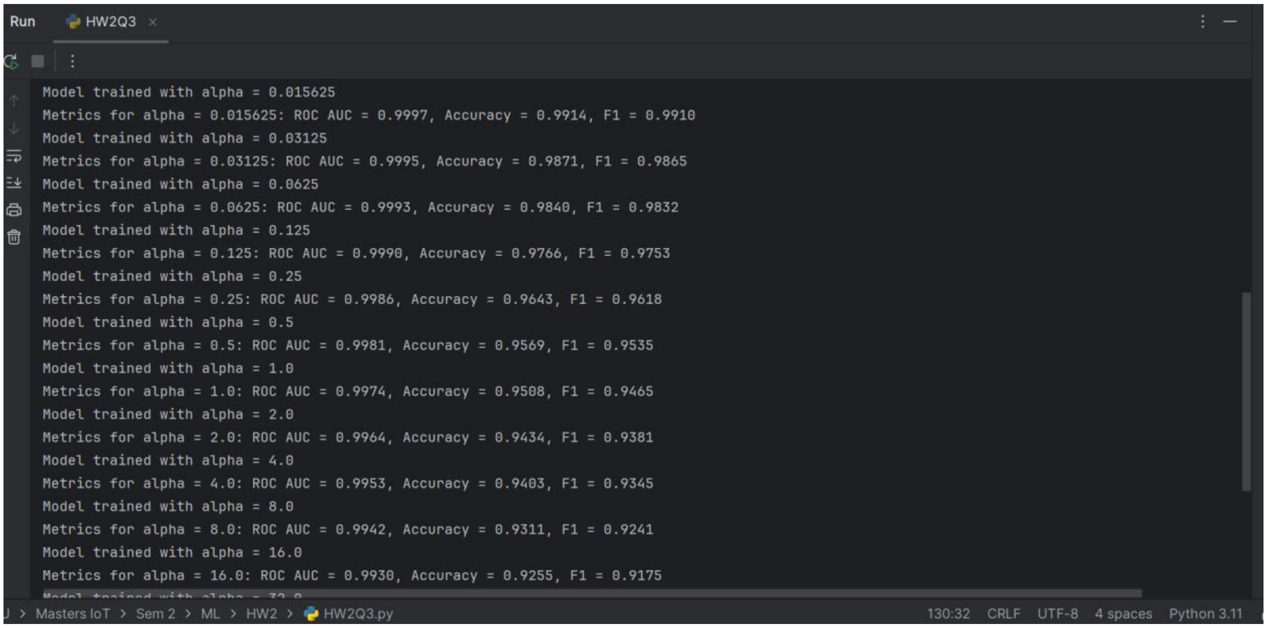
Output:

Training the categorical Na ̈ive Bayes classifier with different values of smoothing hyperparameter α, common amongst all features, ranging from 2−15 to 25.

Data split into training (80%) and test (20%) sets: 6499 training samples, 1625 test samples.

Training CategoricalNB models with varying alpha...

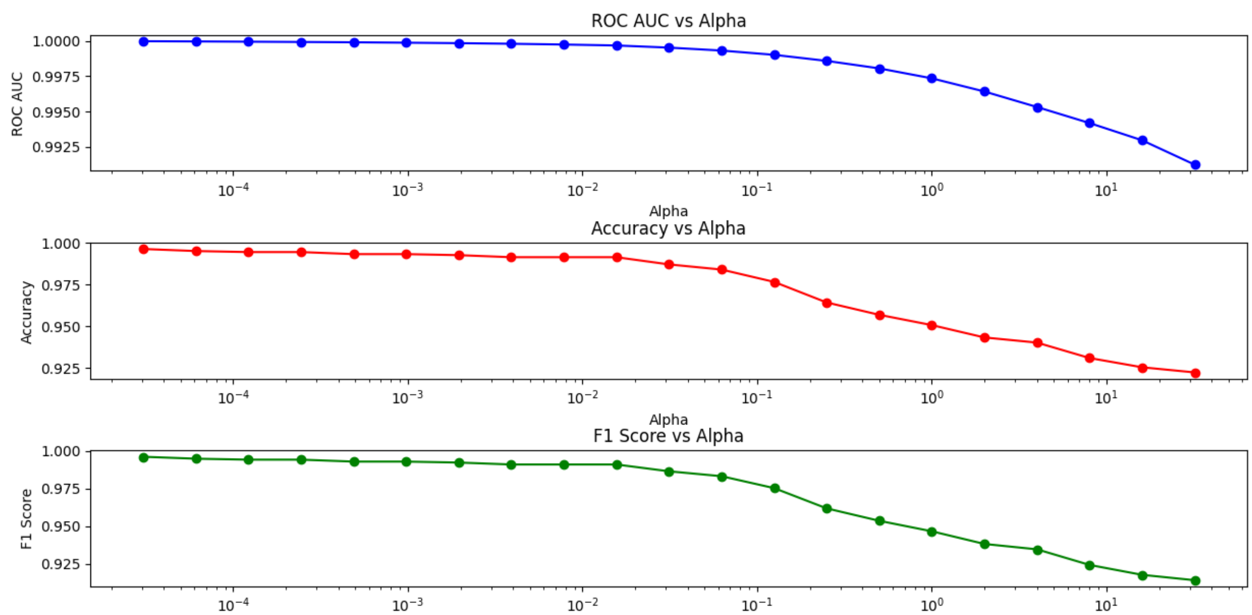
```
Run   HW2Q3 ×
C:\Users\sontu\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:\Users\sontu\Desktop\NEU\Masters IoT\Sem 2\ML\HW2\HW2Q3.py"
Dataset loaded successfully.
Categorical features encoded.
Data split into features and target.
Data split into training (80%) and test (20%) sets: 6499 training samples, 1625 test samples.
Training CategoricalNB models with varying alpha...
Model trained with alpha = 3.0517578125e-05
Metrics for alpha = 3.0517578125e-05: ROC AUC = 1.0000, Accuracy = 0.9963, F1 = 0.9962
Model trained with alpha = 6.103515625e-05
Metrics for alpha = 6.103515625e-05: ROC AUC = 1.0000, Accuracy = 0.9951, F1 = 0.9949
Model trained with alpha = 0.0001220703125
Metrics for alpha = 0.0001220703125: ROC AUC = 1.0000, Accuracy = 0.9945, F1 = 0.9943
Model trained with alpha = 0.000244140625
Metrics for alpha = 0.000244140625: ROC AUC = 0.9999, Accuracy = 0.9945, F1 = 0.9943
Model trained with alpha = 0.00048828125
Metrics for alpha = 0.00048828125: ROC AUC = 0.9999, Accuracy = 0.9932, F1 = 0.9930
Model trained with alpha = 0.0009765625
Metrics for alpha = 0.0009765625: ROC AUC = 0.9999, Accuracy = 0.9932, F1 = 0.9930
Model trained with alpha = 0.001953125
Metrics for alpha = 0.001953125: ROC AUC = 0.9998, Accuracy = 0.9926, F1 = 0.9923
Model trained with alpha = 0.00390625
Metrics for alpha = 0.00390625: ROC AUC = 0.9998, Accuracy = 0.9914, F1 = 0.9910
Model trained with alpha = 0.0078125
U > Masters IoT > Sem 2 > ML > HW2 > HW2Q3.py        130:32  CRLF  UTF-8  4 spaces  Python 3.11
```



Model trained with alpha = 3.0517578125e-05
Metrics for alpha = 3.0517578125e-05: ROC AUC = 1.0000, Accuracy = 0.9963, F1 = 0.9962
Model trained with alpha = 6.103515625e-05
Metrics for alpha = 6.103515625e-05: ROC AUC = 1.0000, Accuracy = 0.9951, F1 = 0.9949
Model trained with alpha = 0.0001220703125
Metrics for alpha = 0.0001220703125: ROC AUC = 1.0000, Accuracy = 0.9945, F1 = 0.9943
Model trained with alpha = 0.000244140625
Metrics for alpha = 0.000244140625: ROC AUC = 0.9999, Accuracy = 0.9945, F1 = 0.9943
Model trained with alpha = 0.00048828125
Metrics for alpha = 0.00048828125: ROC AUC = 0.9999, Accuracy = 0.9932, F1 = 0.9930
Model trained with alpha = 0.0009765625
Metrics for alpha = 0.0009765625: ROC AUC = 0.9999, Accuracy = 0.9932, F1 = 0.9930
Model trained with alpha = 0.001953125
Metrics for alpha = 0.001953125: ROC AUC = 0.9998, Accuracy = 0.9926, F1 = 0.9923
Model trained with alpha = 0.00390625
Metrics for alpha = 0.00390625: ROC AUC = 0.9998, Accuracy = 0.9914, F1 = 0.9910
Model trained with alpha = 0.0078125
Metrics for alpha = 0.0078125: ROC AUC = 0.9997, Accuracy = 0.9914, F1 = 0.9910
Model trained with alpha = 0.015625
Metrics for alpha = 0.015625: ROC AUC = 0.9997, Accuracy = 0.9914, F1 = 0.9910
Model trained with alpha = 0.03125
Metrics for alpha = 0.03125: ROC AUC = 0.9995, Accuracy = 0.9871, F1 = 0.9865
Model trained with alpha = 0.0625
Metrics for alpha = 0.0625: ROC AUC = 0.9993, Accuracy = 0.9840, F1 = 0.9832
Model trained with alpha = 0.125
Metrics for alpha = 0.125: ROC AUC = 0.9990, Accuracy = 0.9766, F1 = 0.9753
Model trained with alpha = 0.25
Metrics for alpha = 0.25: ROC AUC = 0.9986, Accuracy = 0.9643, F1 = 0.9618

Model trained with alpha = 0.5
Metrics for alpha = 0.5: ROC AUC = 0.9981, Accuracy = 0.9569, F1 = 0.9535
Model trained with alpha = 1.0
Metrics for alpha = 1.0: ROC AUC = 0.9974, Accuracy = 0.9508, F1 = 0.9465
Model trained with alpha = 2.0
Metrics for alpha = 2.0: ROC AUC = 0.9964, Accuracy = 0.9434, F1 = 0.9381
Model trained with alpha = 4.0
Metrics for alpha = 4.0: ROC AUC = 0.9953, Accuracy = 0.9403, F1 = 0.9345
Model trained with alpha = 8.0
Metrics for alpha = 8.0: ROC AUC = 0.9942, Accuracy = 0.9311, F1 = 0.9241
Model trained with alpha = 16.0
Metrics for alpha = 16.0: ROC AUC = 0.9930, Accuracy = 0.9255, F1 = 0.9175
Model trained with alpha = 32.0
Metrics for alpha = 32.0: ROC AUC = 0.9912, Accuracy = 0.9225, F1 = 0.9139

Plotting the predictive performance of the trained classifier on your test set, as measured by ROC AUC, accuracy, and F1 scores.



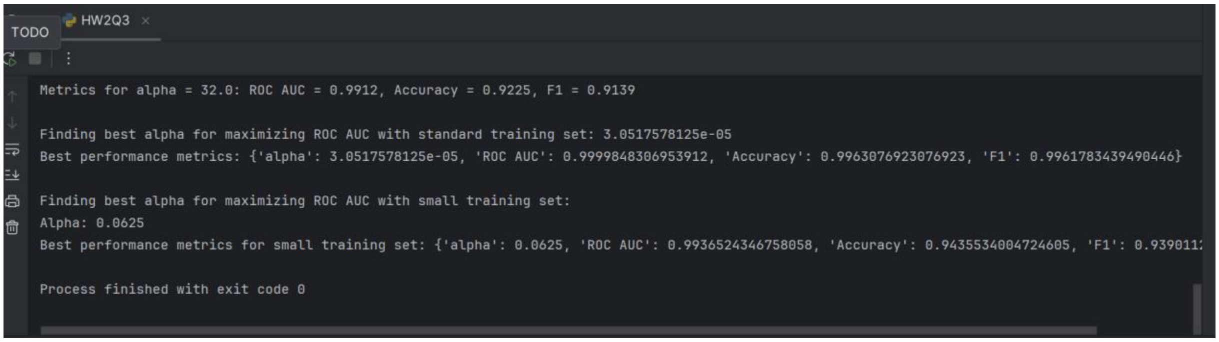
The **parameters of the model for the α value that maximizes the ROC AUC** with standard training set: 3.0517578125e-05

Best performance metrics: {'alpha': 3.0517578125e-05, 'ROC AUC': 0.9999848306953912, 'Accuracy': 0.9963076923076923, 'F1': 0.9961783439490446}

Repeating the same experiment with split in which you use only 1% of the dataset for training, and 99% of the dataset for testing.

Finding best alpha for maximizing ROC AUC with small training set:
Alpha: 0.0625

Best performance metrics for small training set: {'alpha': 0.0625, 'ROC AUC': 0.9936524346758058, 'Accuracy': 0.9435534004724605, 'F1': 0.9390112842557765}



```
Metrics for alpha = 32.0: ROC AUC = 0.9912, Accuracy = 0.9225, F1 = 0.9139

Finding best alpha for maximizing ROC AUC with standard training set: 3.0517578125e-05
Best performance metrics: {'alpha': 3.0517578125e-05, 'ROC AUC': 0.9999848306953912, 'Accuracy': 0.9963076923076923, 'F1': 0.9961783439490446}

Finding best alpha for maximizing ROC AUC with small training set:
Alpha: 0.0625
Best performance metrics for small training set: {'alpha': 0.0625, 'ROC AUC': 0.9936524346758058, 'Accuracy': 0.9435534004724605, 'F1': 0.939011

Process finished with exit code 0
```

Based on the provided output for training a categorical Naïve Bayes classifier on the Mushroom dataset with standard and small training sets,

Metric	Standard Training Set (80%)	Small Training Set (1%)
Best Alpha	$3.0517578125 \times 10{-}5$	0.0625
ROC AUC	0.9999848306953912	0.9936524346758058
Accuracy	0.9963076923076923	0.9435534004724605
F1 Score	0.9961783439490446	0.9390112842557765

So, from the table:

- The **Best Alpha** value required for optimal performance increases when the training set size is reduced, indicating a need for greater smoothing to counter the lack of data.
- Both **ROC AUC** and **Accuracy** metrics show a decline when the model is trained on a much smaller dataset, though the ROC AUC remains relatively high, suggesting the model's robustness in distinguishing between classes under varied conditions.
- The **F1 Score**, which balances precision and recall, also decreases with the smaller training set, reflecting challenges in maintaining performance with limited training data.

Overall, **ROC AUC may be more informative than accuracy**, especially when classes are imbalanced or the cost of false negatives significantly outweighs false positives (e.g., classifying poisonous mushrooms as edible). ROC AUC provides a more comprehensive view of the model's performance across various threshold settings, making it a crucial metric for evaluating classifiers in sensitive applications.

4Q) Sentence Classification dataset:

The dataset is randomly splited into a training set (80%) and a test set (20%). The division ensured that the model was trained and evaluated on distinct subsets of the data, providing a fair assessment of its predictive capabilities.

I have trained a Multinomial Naïve Bayes classifier, utilizing the “bag of words” representation of the text data.

The smoothing hyperparameter α was varied across a predefined range, specifically from 2^{-15} to 2^5. Each value of α introduces a different degree of smoothing, affecting how the model handles words do not present in the training data

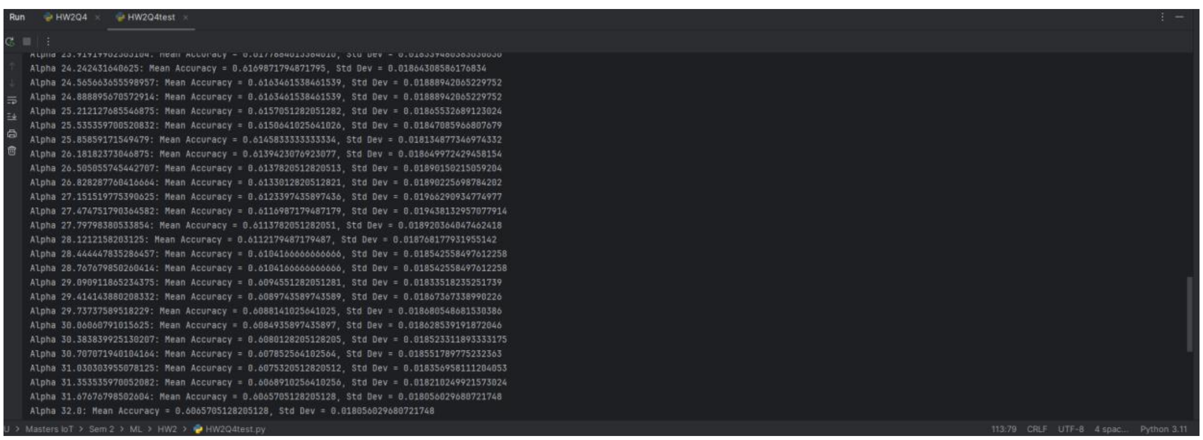
Output:

C:\Users\sontu\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:\Users\sontu\Desktop\NEU\Masters IoT\Sem 2\ML\HW2\HW2Q4test.py"

```
Alpha 3.0517578125e-05: Mean Accuracy = 0.8395833333333333, Std Dev = 0.016815472294326513
Alpha 0.3232625325520833: Mean Accuracy = 0.8099358974358974, Std Dev = 0.01520701140910948
Alpha 0.6464945475260416: Mean Accuracy = 0.8179487179487179, Std Dev = 0.01429788975845861
Alpha 0.9697265625: Mean Accuracy = 0.8182692307692307, Std Dev = 0.017623348204258905
Alpha 1.2929585774739583: Mean Accuracy = 0.822275641025641, Std Dev = 0.01648989229674391
Alpha 1.6161905924479165: Mean Accuracy = 0.8192307692307692, Std Dev = 0.01420980334886387
Alpha 1.939422607421875: Mean Accuracy = 0.8158653846153846, Std Dev = 0.014540319615257518
Alpha 2.262654622395833: Mean Accuracy = 0.8081730769230769, Std Dev = 0.013682932183420876
Alpha 2.5858866373697915: Mean Accuracy = 0.8017628205128204, Std Dev = 0.01362022363989948
Alpha 2.90911865234375: Mean Accuracy = 0.796474358974359, Std Dev = 0.017027112660800467
Alpha 3.232350667317708: Mean Accuracy = 0.7884615384615384, Std Dev = 0.016132124861689637
Alpha 3.5555826822916665: Mean Accuracy = 0.7815705128205128, Std Dev = 0.01722787467420315
Alpha 3.878814697265625: Mean Accuracy = 0.7770833333333333, Std Dev = 0.01667916061399215
Alpha 4.202046712239583: Mean Accuracy = 0.771474358974359, Std Dev = 0.01634650298493122
Alpha 4.525278727213541: Mean Accuracy = 0.7668269230769231, Std Dev = 0.01634737579925146
Alpha 4.8485107421875: Mean Accuracy = 0.7629807692307692, Std Dev = 0.01751371132496809
Alpha 5.171742757161458: Mean Accuracy = 0.7572115384615384, Std Dev = 0.018662208941841247
Alpha 5.494974772135416: Mean Accuracy = 0.7503205128205128, Std Dev = 0.018237652019974907
Alpha 5.818206787109375: Mean Accuracy = 0.7455128205128205, Std Dev = 0.019003884781503555
Alpha 6.141438802083333: Mean Accuracy = 0.7419871794871795, Std Dev = 0.018795525732648696
Alpha 6.464670817057291: Mean Accuracy = 0.7379807692307692, Std Dev = 0.01923447851216756
Alpha 6.78790283203125: Mean Accuracy = 0.7363782051282051, Std Dev = 0.019308514499922883
Alpha 7.111134847005208: Mean Accuracy = 0.7315705128205128, Std Dev = 0.01849324719349759
Alpha 7.434366861979166: Mean Accuracy = 0.7254807692307692, Std Dev = 0.019859297834955783
Alpha 7.757598876953125: Mean Accuracy = 0.7221153846153846, Std Dev = 0.019688289743229826
Alpha 8.080830891927082: Mean Accuracy = 0.716025641025641, Std Dev = 0.020183543976208702
Alpha 8.404062906901041: Mean Accuracy = 0.7120192307692308, Std Dev = 0.0197006055660314
Alpha 8.727294921875: Mean Accuracy = 0.7080128205128206, Std Dev = 0.017207157628451476
Alpha 9.050526936848957: Mean Accuracy = 0.7040064102564103, Std Dev = 0.018068668429092837
Alpha 9.373758951822916: Mean Accuracy = 0.7011217948717948, Std Dev = 0.017607959106676816
Alpha 9.696990966796875: Mean Accuracy = 0.6975961538461538, Std Dev = 0.017277494231666468
Alpha 10.020222981770832: Mean Accuracy = 0.6940705128205128, Std Dev = 0.01807498448921027
Alpha 10.343454996744791: Mean Accuracy = 0.6907051282051282, Std Dev = 0.01867367338990229
Alpha 10.66668701171875: Mean Accuracy = 0.6873397435897436, Std Dev = 0.01934100055218342
```

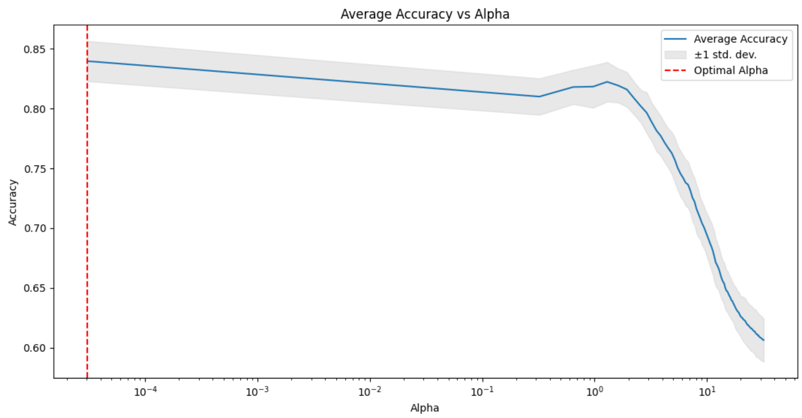
Alpha 10.989919026692707: Mean Accuracy = 0.6842948717948718, Std Dev = 0.0200160192256359
Alpha 11.313151041666666: Mean Accuracy = 0.6807692307692308, Std Dev = 0.01982694138403771
Alpha 11.636383056640625: Mean Accuracy = 0.6762820512820513, Std Dev = 0.02075786242278357
Alpha 11.959615071614582: Mean Accuracy = 0.6714743589743589, Std Dev = 0.01967088957213885
Alpha 12.282847086588541: Mean Accuracy = 0.6693910256410256, Std Dev = 0.01930555851177024
Alpha 12.6060791015625: Mean Accuracy = 0.6671474358974359, Std Dev = 0.018272822908710868
Alpha 12.929311116536457: Mean Accuracy = 0.6642628205128205, Std Dev = 0.01840043231800196
Alpha 13.252543131510416: Mean Accuracy = 0.6605769230769231, Std Dev = 0.01774598056287042
Alpha 13.575775146484375: Mean Accuracy = 0.6575320512820513, Std Dev = 0.01735987832572468
Alpha 13.899007161458332: Mean Accuracy = 0.6552884615384615, Std Dev = 0.01674745513904442
Alpha 14.222239176432291: Mean Accuracy = 0.6538461538461539, Std Dev = 0.01771701276899254
Alpha 14.54547119140625: Mean Accuracy = 0.6503205128205128, Std Dev = 0.01795380473299148
Alpha 14.868703206380207: Mean Accuracy = 0.6479166666666667, Std Dev = 0.0169944012494027
Alpha 15.191935221354166: Mean Accuracy = 0.6471153846153846, Std Dev = 0.01725683675490947
Alpha 15.515167236328125: Mean Accuracy = 0.6453525641025641, Std Dev = 0.01714485629077576
Alpha 15.838399251302082: Mean Accuracy = 0.6434294871794872, Std Dev = 0.01679509607079324
Alpha 16.16163126627604: Mean Accuracy = 0.642147435897436, Std Dev = 0.016842602246091454
Alpha 16.48486328125: Mean Accuracy = 0.6399038461538461, Std Dev = 0.016534823966669155
Alpha 16.808095296223957: Mean Accuracy = 0.6392628205128205, Std Dev = 0.01606920716390419
Alpha 17.131327311197914: Mean Accuracy = 0.6375, Std Dev = 0.01630455295088431
Alpha 17.454559326171875: Mean Accuracy = 0.6358974358974359, Std Dev = 0.01603987971439000
Alpha 17.777791341145832: Mean Accuracy = 0.6344551282051282, Std Dev = 0.01721793360316524
Alpha 18.10102335611979: Mean Accuracy = 0.6326923076923077, Std Dev = 0.01758444451839129
Alpha 18.42425537109375: Mean Accuracy = 0.6317307692307692, Std Dev = 0.01724029295208916
Alpha 18.747487386067707: Mean Accuracy = 0.6304487179487179, Std Dev = 0.01706394256112631
Alpha 19.070719401041664: Mean Accuracy = 0.6298076923076923, Std Dev = 0.01707731555891643
Alpha 19.393951416015625: Mean Accuracy = 0.6286858974358974, Std Dev = 0.01753976129578445
Alpha 19.717183430989582: Mean Accuracy = 0.6270833333333333, Std Dev = 0.01668942260877803
Alpha 20.04041544596354: Mean Accuracy = 0.6261217948717949, Std Dev = 0.017178111789566165
Alpha 20.3636474609375: Mean Accuracy = 0.6253205128205128, Std Dev = 0.01768154323838328
Alpha 20.686879475911457: Mean Accuracy = 0.625, Std Dev = 0.018036263906126213
Alpha 21.010111490885414: Mean Accuracy = 0.6238782051282051, Std Dev = 0.01862853919187206
Alpha 21.333343505859375: Mean Accuracy = 0.6233974358974359, Std Dev = 0.01859711024095001
Alpha 21.656575520833332: Mean Accuracy = 0.6229166666666667, Std Dev = 0.01875068484394712
Alpha 21.97980753580729: Mean Accuracy = 0.622275641025641, Std Dev = 0.018947492192098674
Alpha 22.30303955078125: Mean Accuracy = 0.6211538461538462, Std Dev = 0.01866144439488416
Alpha 22.626271565755207: Mean Accuracy = 0.6201923076923077, Std Dev = 0.01873469862900661
Alpha 22.949503580729164: Mean Accuracy = 0.6192307692307693, Std Dev = 0.01836861307721151
Alpha 23.272735595703125: Mean Accuracy = 0.6190705128205128, Std Dev = 0.01841283466657872
Alpha 23.595967610677082: Mean Accuracy = 0.6182692307692308, Std Dev = 0.01870115939063340
Alpha 23.91919962565104: Mean Accuracy = 0.6177884615384616, Std Dev = 0.018539480383630656
Alpha 24.242431640625: Mean Accuracy = 0.6169871794871795, Std Dev = 0.01864308586176834
Alpha 24.565663655598957: Mean Accuracy = 0.6163461538461539, Std Dev = 0.01888942065229752
Alpha 24.888895670572914: Mean Accuracy = 0.6163461538461539, Std Dev = 0.01888942065229752
Alpha 25.212127685546875: Mean Accuracy = 0.6157051282051282, Std Dev = 0.01865532689123024

Alpha 25.535359700520832: Mean Accuracy = 0.6150641025641026, Std Dev = 0.01847085966807679
Alpha 25.85859171549479: Mean Accuracy = 0.6145833333333334, Std Dev = 0.018134877346974332
Alpha 26.18182373046875: Mean Accuracy = 0.6139423076923077, Std Dev = 0.018649972429458154
Alpha 26.505055745442707: Mean Accuracy = 0.6137820512820513, Std Dev = 0.01890150215059204
Alpha 26.828287760416664: Mean Accuracy = 0.6133012820512821, Std Dev = 0.01890225698784202
Alpha 27.151519775390625: Mean Accuracy = 0.6123397435897436, Std Dev = 0.01966290934774977
Alpha 27.474751790364582: Mean Accuracy = 0.6116987179487179, Std Dev = 0.01943813295707791
Alpha 27.79798380533854: Mean Accuracy = 0.6113782051282051, Std Dev = 0.018920364047462418
Alpha 28.1212158203125: Mean Accuracy = 0.6112179487179487, Std Dev = 0.018768177931955142
Alpha 28.444447835286457: Mean Accuracy = 0.6104166666666666, Std Dev = 0.01854255849761225
Alpha 28.767679850260414: Mean Accuracy = 0.6104166666666666, Std Dev = 0.01854255849761225
Alpha 29.090911865234375: Mean Accuracy = 0.6094551282051281, Std Dev = 0.01833518235251739
Alpha 29.414143880208332: Mean Accuracy = 0.6089743589743589, Std Dev = 0.01867367338990226
Alpha 29.73737589518229: Mean Accuracy = 0.6088141025641025, Std Dev = 0.018680548681530386
Alpha 30.06060791015625: Mean Accuracy = 0.6084935897435897, Std Dev = 0.018628539191872046
Alpha 30.383839925130207: Mean Accuracy = 0.6080128205128205, Std Dev = 0.01852331189333317
Alpha 30.707071940104164: Mean Accuracy = 0.607852564102564, Std Dev = 0.018551789775232363
Alpha 31.030303955078125: Mean Accuracy = 0.6075320512820512, Std Dev = 0.01835695811120405
Alpha 31.353535970052082: Mean Accuracy = 0.6068910256410256, Std Dev = 0.01821024992157302
Alpha 31.67676798502604: Mean Accuracy = 0.6065705128205128, Std Dev = 0.018056029680721748
Alpha 32.0: Mean Accuracy = 0.6065705128205128, Std Dev = 0.018056029680721748

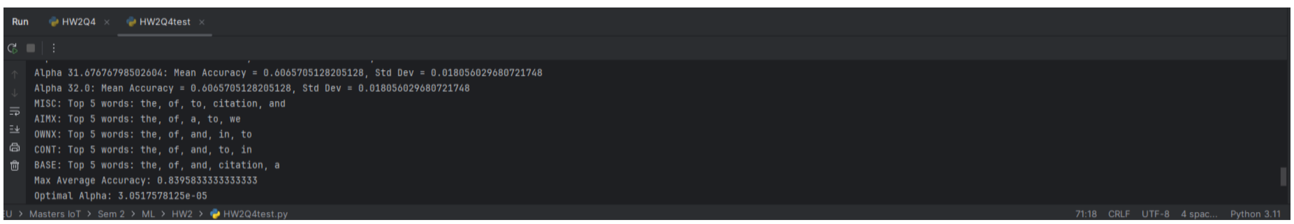


```
Run   HW2Q4   HW2Q4test
Alpha 24.242431640625: Mean Accuracy = 0.6169871794871795, Std Dev = 0.01864308586176834
Alpha 24.565663655598957: Mean Accuracy = 0.6163461538461539, Std Dev = 0.01888942065229752
Alpha 24.888895670572914: Mean Accuracy = 0.6163461538461539, Std Dev = 0.01888942065229752
Alpha 25.212127685546875: Mean Accuracy = 0.6157051282051282, Std Dev = 0.01865532689123024
Alpha 25.535359700520832: Mean Accuracy = 0.6150641025641026, Std Dev = 0.01847085966807679
Alpha 25.85859171549479: Mean Accuracy = 0.6145833333333334, Std Dev = 0.018134877346974332
Alpha 26.18182373046875: Mean Accuracy = 0.6139423076923077, Std Dev = 0.018649972429458154
Alpha 26.505055745442707: Mean Accuracy = 0.6137820512820513, Std Dev = 0.01890150215059204
Alpha 26.828287760416664: Mean Accuracy = 0.6133012820512821, Std Dev = 0.01890225698784202
Alpha 27.151519775390625: Mean Accuracy = 0.6123397435897436, Std Dev = 0.01966290934774977
Alpha 27.474751790364582: Mean Accuracy = 0.6116987179487179, Std Dev = 0.019438132957077914
Alpha 27.79798380533854: Mean Accuracy = 0.6113782051282051, Std Dev = 0.018920364047462418
Alpha 28.1212158203125: Mean Accuracy = 0.6112179487179487, Std Dev = 0.018768177931955142
Alpha 28.444447835286457: Mean Accuracy = 0.6104166666666666, Std Dev = 0.018542558497612258
Alpha 28.767679850260414: Mean Accuracy = 0.6104166666666666, Std Dev = 0.018542558497612258
Alpha 29.090911865234375: Mean Accuracy = 0.6094551282051281, Std Dev = 0.01833518235251739
Alpha 29.414143880208332: Mean Accuracy = 0.6089743589743589, Std Dev = 0.01867367338990226
Alpha 29.73737589518229: Mean Accuracy = 0.6088141025641025, Std Dev = 0.018680548681530386
Alpha 30.06060791015625: Mean Accuracy = 0.6084935897435897, Std Dev = 0.018628539191872046
Alpha 30.383839925130207: Mean Accuracy = 0.6080128205128205, Std Dev = 0.018523311893333175
Alpha 30.707071940104164: Mean Accuracy = 0.607852564102564, Std Dev = 0.018551789775232363
Alpha 31.030303955078125: Mean Accuracy = 0.6075320512820512, Std Dev = 0.018356958111204053
Alpha 31.353535970052082: Mean Accuracy = 0.6068910256410256, Std Dev = 0.018210249921573024
Alpha 31.67676798502604: Mean Accuracy = 0.6065705128205128, Std Dev = 0.018056029680721748
Alpha 32.0: Mean Accuracy = 0.6065705128205128, Std Dev = 0.018056029680721748
U > Masters IoT > Sem 2 > ML > HW2 > HW2Q4test.py
```

Plot of Avg Accuracy vs Alpha



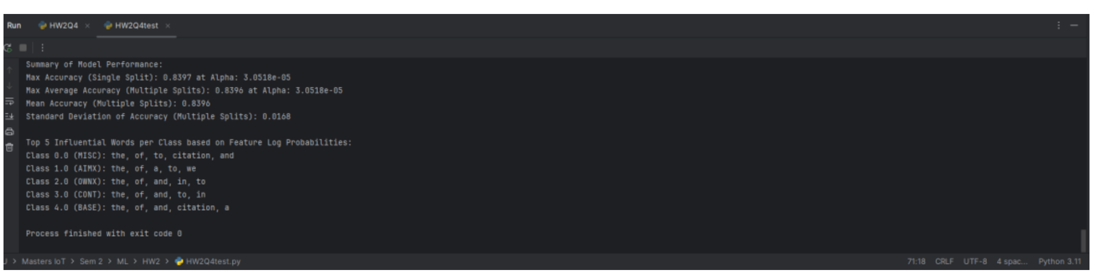
Now, Extracting the top 5 words for each class,



Model Summary and Top 5 Influential Words,

Summary of Model Performance:
Max Accuracy (Single Split): 0.8397 at Alpha: 3.0518e-05
Max Average Accuracy (Multiple Splits): 0.8396 at Alpha: 3.0518e-05
Mean Accuracy (Multiple Splits): 0.8396
Standard Deviation of Accuracy (Multiple Splits): 0.0168

Top 5 Influential Words per Class based on Feature Log Probabilities:
Class 0.0 (MISC): the, of, to, citation, and
Class 1.0 (AIMX): the, of, a, to, we
Class 2.0 (OWNX): the, of, and, in, to
Class 3.0 (CONT): the, of, and, to, in
Class 4.0 (BASE): the, of, and, citation, a



Q2) ① Given, $X \in \{1, \dots, K\}$

Denoted by

$$\theta_k = P(X=k)$$

$D = \{x_1, x_2, \dots x_N\}$

each sample x_i can fall into ~~no~~ one of K categories, with θ_k being the prob of observing k^{th} category.

prob of observing each individual sample:

$$p(D|\theta) = \prod_{i=1}^{N} \theta_{x_i}$$

Since, $N_k = \sum_{i=1}^{N} 1_{x_i = k}$

the no of times the outcome k occurs in D,

$$\Rightarrow p(D|\theta) = \prod_{k=1}^{K} \theta_k^{N_k}$$

↳ the formula captures the likelihood of the Dataset D as a function of θ, with each θ_k raised to ~~to~~ the power of the number of times the k^{th} category occurs in dataset.

② Dirichlet prior distribution.

$$P(\theta) \propto \prod_{k=1}^{K} \theta_k^{\alpha_k - 1} \quad \text{—— ①}$$

where, $\alpha_k \geq 1$ for $k = 1, \ldots, K$

we need to show posterior distribution $p(\theta|D)$ is also a Dirichlet distribution,

Posterior distribution is proportional to the product of the likelihood and the prior:

$$p(\theta|D) \propto p(D|\theta) \cdot p(\theta) \quad \text{—— ②}$$

Substituting, $P(D|\theta)$ & ① in ②,

$$p(\theta|D) \propto \left(\prod_{k=1}^{K} \theta_k^{N_k}\right) \cdot \left(\prod_{k=1}^{K} \theta_k^{\alpha_k - 1}\right)$$

$$\Rightarrow \quad p(\theta|D) \propto \prod_{k=1}^{K} \theta_k^{N_k + \alpha_k - 1}$$

This shows that the posterior distribution is also a Dirichlet distribution with updated hyperparameters

$\alpha_k' = N_k + \alpha_k$, where N_k is the count of category k in the dataset, & α_k are the original hyperparameters of the prior.

Hence, the posterior distribution is $Dir(\theta|\alpha_1', \alpha_2', \ldots \alpha_K')$ with the updated hyperparameters reflecting the info. gained the data D..

③ To show that if $\alpha_1 = \alpha_2 = \dots = \alpha_K = 1$,

Here, the Dirichlet distribution, when all its parameters $(\alpha_1, \alpha_2, \dots \alpha_K)$ are set to 1, does indeed become a uniform distribution over the simplex of valid θ values.

Dirichlet distribution, defined for a vector $\theta = (\theta_1, \theta_2, \dots \theta_K)$ where each θ_K represents the probability of the kth category and $\sum_{K=1}^{K} \theta_K = 1$,

$$p(\theta|\alpha) \propto \prod_{K=1}^{K} \theta_K^{\alpha_K - 1},$$

where $\alpha = (\alpha_1, \alpha_2 \dots \alpha_K)$, when all $\alpha_K = 1$

$$\Rightarrow p(\theta|\alpha) \propto \prod_{K=1}^{K} \theta_K^{1-1}$$

$$= \prod_{K=1}^{K} \theta_K^{0}$$

$$= \prod_{K=1}^{K} 1 = 1$$

This means the prob density function of θ doesn't depend on θ at all and is constant across all possible values of θ that satisfy $\sum_{K=1}^{K} \theta_K = 1$,

When all $\alpha_K = 1$, the Dirichlet distribution simplifies to a uniform distribution over the simplex, meaning all configurations of probability vector θ are equally likely. This "uninformative" prior indicates no preference for any outcomes, embodying a state of complete ignorance about the prob of categories, making it ideal for situations where we lack prior knowledge about the distribution of outcomes.

(4) One collected sample $D = \{x_1\}$,

Considering an uninformative prior for a categorical random variable X with K possible categories, the situation simplifies the estimation of θ_K, the prob of observing outcome k.

Let's consider the three estimators given under this specific scenario; assuming $N_k = 1$ for the observed category

$N_k = 0$ for all others.

For, uninformative proior, $\alpha_k = 1$

$$\alpha_0 = \sum_{k=1}^{K} \alpha_k = K$$

(i) Max likelihood Estimator ($\hat{\theta}_{ML}$)

$$\hat{\theta}_{ML,k} = \frac{N_k}{N}$$

we have one sample, $N = 1$

$\therefore$ observed category k, $N_k = 1$

$\hat{\theta}_{ML,k} = 1$ for that category

$\hat{\theta}_{ML,k} = 0$ for all other category's.

(ii) Maximum A posterior (MAP) Estimator ($\hat{\theta}_{MAP}$):

$$\hat{\theta}_{MAP,k} = \frac{N_k + \alpha_k - 1}{N + \alpha_0 - K}$$

Under the uninformative prior ($\alpha_k = 1$ for all k), for the observed category k, $N_k = 1$

for all others, $N_k = 0$

→ Observed category, $\hat{\theta}_{MAP,k} = \frac{1}{N+K-K} = \frac{1}{1+K-K} = \frac{1}{1} = 1$

→ all other categories, $\hat{\theta}_{MAP,k} = \frac{0}{1+K-K} = \frac{0}{1} = 0$

(iii) Full Bayesian Estimation:

$$E[\theta_k | D] = \frac{N_k + \alpha_k}{N + \alpha_0}$$

Same, $N_k = 1$, observed

$N_k = 0$, for all others.

$$E[\theta_k | D] = \frac{1+1}{1+K} = \frac{2}{1+K}$$, for observed category.

$$E[\theta_{k'} | D] = \frac{0+1}{1+K} = \frac{1}{1+K}$$, for any other k'

Which Makes More Sense:

→ ML estimator puts all the prob mass on the observed category, which might not be reasonable with 1 sample, as it ignores the possibility of other outcomes entirely.

→ MAP estimator, with an uninformative prior, suggests similar behavior to the ML estimator but adjusted by the prior, however, with 1 observation & $\alpha_k = 1$, still may not provide meaningfull estimate.

→ Full Bayesian Mean estimator distributes the prob across all categories, given non-zero prob even to the unobserved categories.

∴ Given the context of having only one sample & using a uninformative prior, the Full Bayesian Mean estimator makes most sense.