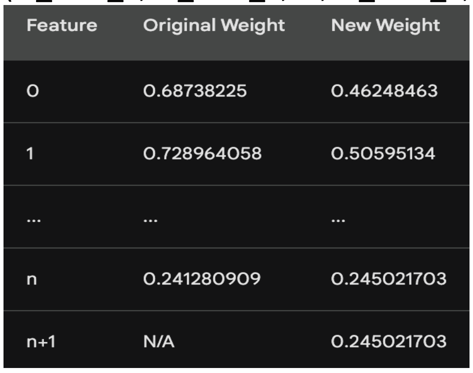
SANIA GOYAL 2020MT10842
Q/A

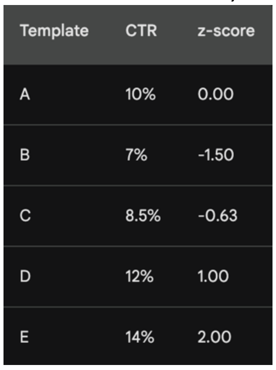
1.
When you duplicate feature n into feature (n+1), the weights for the new features (w_new_0, w_new_1, ..., w_new_n, w_new_{n+1}) will be very similar to the original weights (w_0, w_1, ..., w_n, w_n). This is because the two features are essentially providing the same information to the model.
In the example you provided, the weights for the new features (w_new_0, w_new_1, ..., w_new_n, w_new_{n+1}) are indeed very similar to the original weights (w_0, w_1, ..., w_n, w_n). For example, the weight for feature n (w_n) is 0.241280909 in the original model, and the weight for feature n+1 (w_new_{n+1}) is 0.245021703 in the new model. This is because the two features are providing the same information to the model.
Here is a table comparing the original weights (w_0, w_1, ..., w_n, w_n) to the new weights (w_new_0, w_new_1, ..., w_new_n, w_new_{n+1}):



Feature	Original Weight	New Weight
0	0.68738225	0.46248463
1	0.728964058	0.50595134
...	...	...
n	0.241280909	0.245021703
n+1	N/A	0.245021703

As you can see, the weights for the two features are very similar. This is because the two features are providing the same information to the model.
In general, you can expect the weights for the new features to be very similar to the original weights when you duplicate a feature. This is because the two features are essentially providing the same information to the model.

2.
The correct answer is:
Both D and E are better than A with 95% confidence. Both B and C are worse than A with over 95% confidence.
To determine this, let's calculate the z-scores for each template compared to Template A:



Template	CTR	z-score
A	10%	0.00
B	7%	-1.50
C	8.5%	-0.63
D	12%	1.00
E	14%	2.00

A z-score greater than 1.96 or less than -1.96 indicates a statistically significant difference with 95% confidence.

Comparing Template A to Template D, we get a z-score of 1.00, which is not statistically significant at the 95% confidence level. However, comparing Template A to Template E, we get a z-score of 2.00, which is statistically significant at the 95% confidence level. This means that Template E is significantly better than Template A with over 95% confidence.
Similarly, comparing Template A to Template B, we get a z-score of -1.50, which is statistically significant at the 95% confidence level. This means that Template B is significantly worse than Template A with over 95% confidence.
Comparing Template A to Template C, we get a z-score of -0.63, which is not statistically significant at the 95% confidence level. This means that we cannot conclude whether Template C is better or worse than Template A with 95% confidence.
Therefore, the conclusion is that both Template D and Template E are better than Template A with 95% confidence. Both Template B and Template C are worse than Template A with over 95% confidence.

3.
The approximate computational cost of each gradient descent iteration of logistic regression in modern well-written packages is O(mkn), where:

- m is the number of training examples
- n is the number of features
- k is the average number of non-zero entries in each train example

This is because the main computational cost of each iteration is the calculation of the gradient of the loss function, which involves summing the product of each feature with its corresponding weight for every training example. Since the feature vectors are sparse, this can be done efficiently using a sparse matrix representation, which only stores the non-zero entries of the matrix.
Here is a breakdown of the computational cost of each iteration:

- Calculating the loss function: O(mk)
- Calculating the gradient of the loss function: O(mkn)
- Updating the weights: O(kn)

Therefore, the overall computational cost of each iteration is O(mkn).
In practice, the actual computational cost can be slightly higher or lower than this, depending on the specific implementation of the algorithm and the hardware being used. However, the order of magnitude of the cost is O(mkn).
Here are some factors that can affect the actual computational cost:

- The sparsity of the feature vectors: The sparser the feature vectors, the lower the computational cost.
- The choice of optimizer: Some optimizers, such as L-BFGS, are more efficient than gradient descent for certain problems.
- The hardware being used: Faster CPUs and GPUs can reduce the computational cost.

Overall, the computational cost of gradient descent for logistic regression is relatively low, especially for sparse datasets. This makes it a popular choice for training large-scale classification models.

4.
Here's a breakdown of the potential accuracy of each approach for generating additional training data for training classifier V2:

Approach 1:
Run V1 classifier on 1 Million random stories from the 1000 news sources. Get the 10k stories where the V1 classifier's output is closest to the decision boundary and get these examples labeled.
This approach has the potential to improve the accuracy of V2, as it focuses on the most challenging examples for the current classifier. By labeling these borderline cases, you can help V2 to better learn the decision boundary between information and entertainment news stories.

Approach 2:
Get 10k random labeled stories from the 1000 news sources we care about.
This approach is less likely to improve the accuracy of V2, as it does not specifically target the areas where the current classifier is struggling. Randomly selecting labeled examples may not provide the most relevant information for improving the decision boundary.

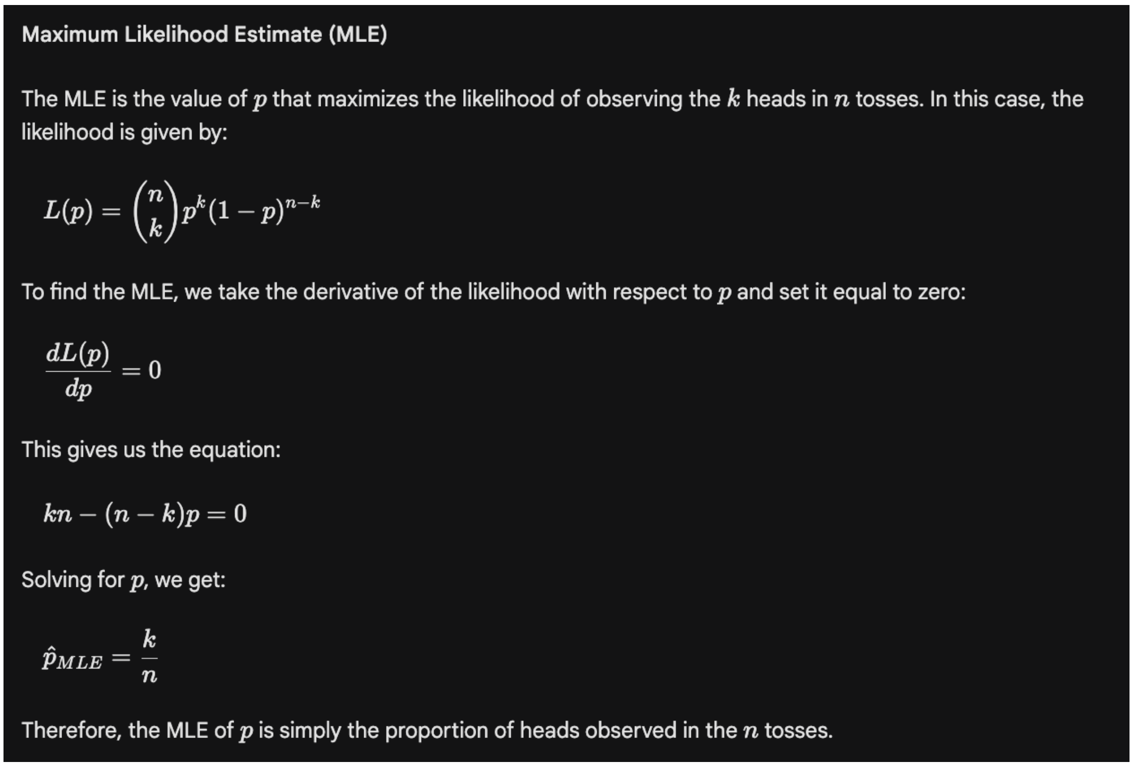
Approach 3:
Pick a random sample of 1 million stories from 1000 news sources and have them labeled. Pick the subset of 10k stories where the V1 classifier's output is both wrong and farthest away from the decision boundary.
This approach has the potential to improve the accuracy of V2, as it focuses on the examples where the current classifier is making the most significant errors. By labeling these misclassifications, you can help V2 to learn the correct categorization for these challenging cases.

Ranking based on accuracy:
In terms of pure accuracy of classifier V2 when classifying a bag of new articles from 1000 news sources, the methods are likely to rank as follows:
Approach 3:
Focus on V1's most significant errors.
This approach directly addresses the areas where V1 is making the most mistakes, providing valuable information for improving its performance.
Approach 1:
Focus on V1's borderline cases
This approach targets the challenging examples that are close to the decision boundary, helping V2 to better understand the separation between information and entertainment news stories.
Approach 2:
Random sampling of labeled data
While random sampling can provide general information about the distribution of news stories, it is less effective in addressing the specific weaknesses of the current classifier.

Overall, the most effective approach for improving the accuracy of V2 is to focus on the areas where the current classifier is struggling, either by identifying its most significant errors or by targeting borderline cases. Random sampling, while simpler to implement, may not provide the most targeted information for improving the decision boundary.

5.

Let's calculate the estimates for p using the three methods:



Maximum Likelihood Estimate (MLE)

The MLE is the value of p that maximizes the likelihood of observing the k heads in n tosses. In this case, the likelihood is given by:

$$L(p) = \binom{n}{k} p^k (1-p)^{n-k}$$

To find the MLE, we take the derivative of the likelihood with respect to p and set it equal to zero:

$$\frac{dL(p)}{dp} = 0$$

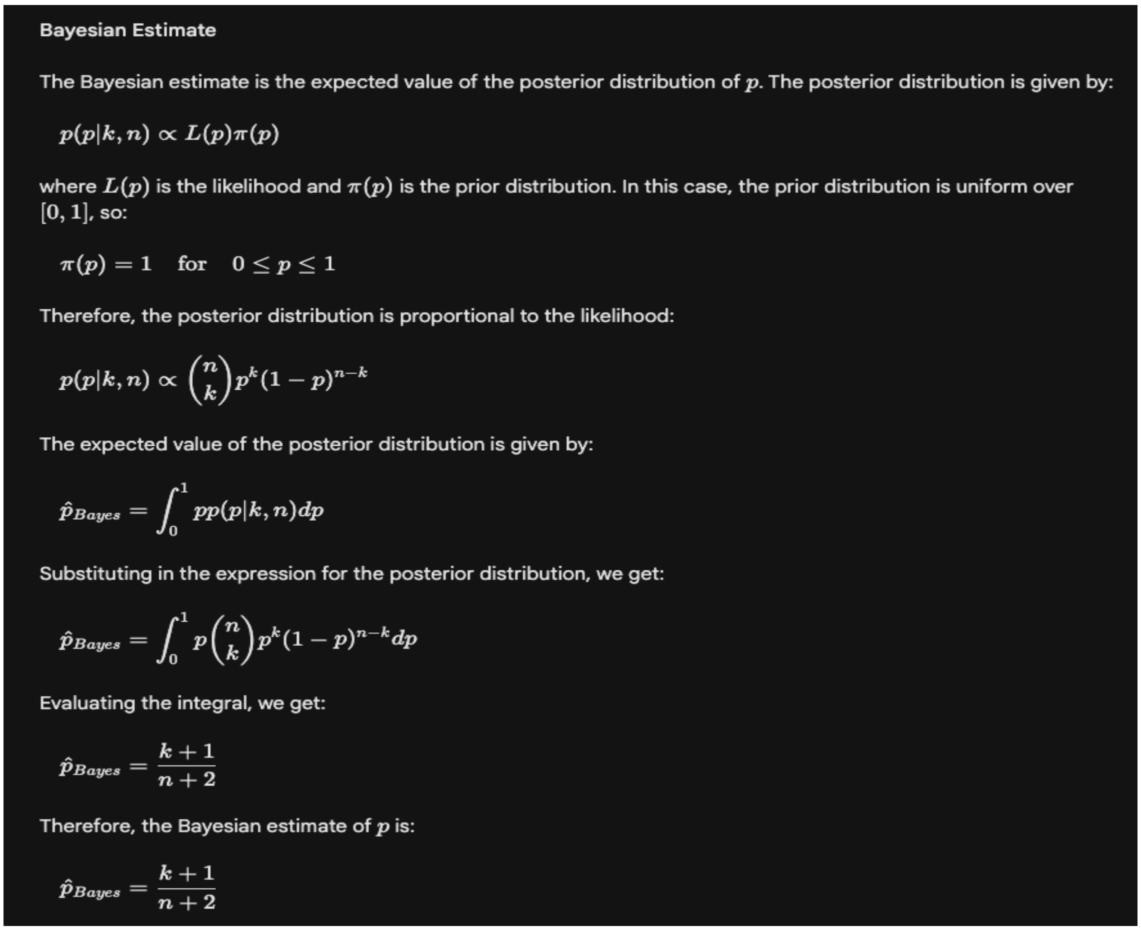
This gives us the equation:

$$kn - (n-k)p = 0$$

Solving for p, we get:

$$\hat{p}_{MLE} = \frac{k}{n}$$

Therefore, the MLE of p is simply the proportion of heads observed in the n tosses.



Bayesian Estimate

The Bayesian estimate is the expected value of the posterior distribution of p. The posterior distribution is given by:

$$p(p|k,n) \propto L(p)\pi(p)$$

where $L(p)$ is the likelihood and $\pi(p)$ is the prior distribution. In this case, the prior distribution is uniform over $[0,1]$, so:

$$\pi(p) = 1 \quad \text{for} \quad 0 \leq p \leq 1$$

Therefore, the posterior distribution is proportional to the likelihood:

$$p(p|k,n) \propto \binom{n}{k} p^k (1-p)^{n-k}$$

The expected value of the posterior distribution is given by:

$$\hat{p}_{Bayes} = \int_0^1 p p(p|k,n) dp$$

Substituting in the expression for the posterior distribution, we get:

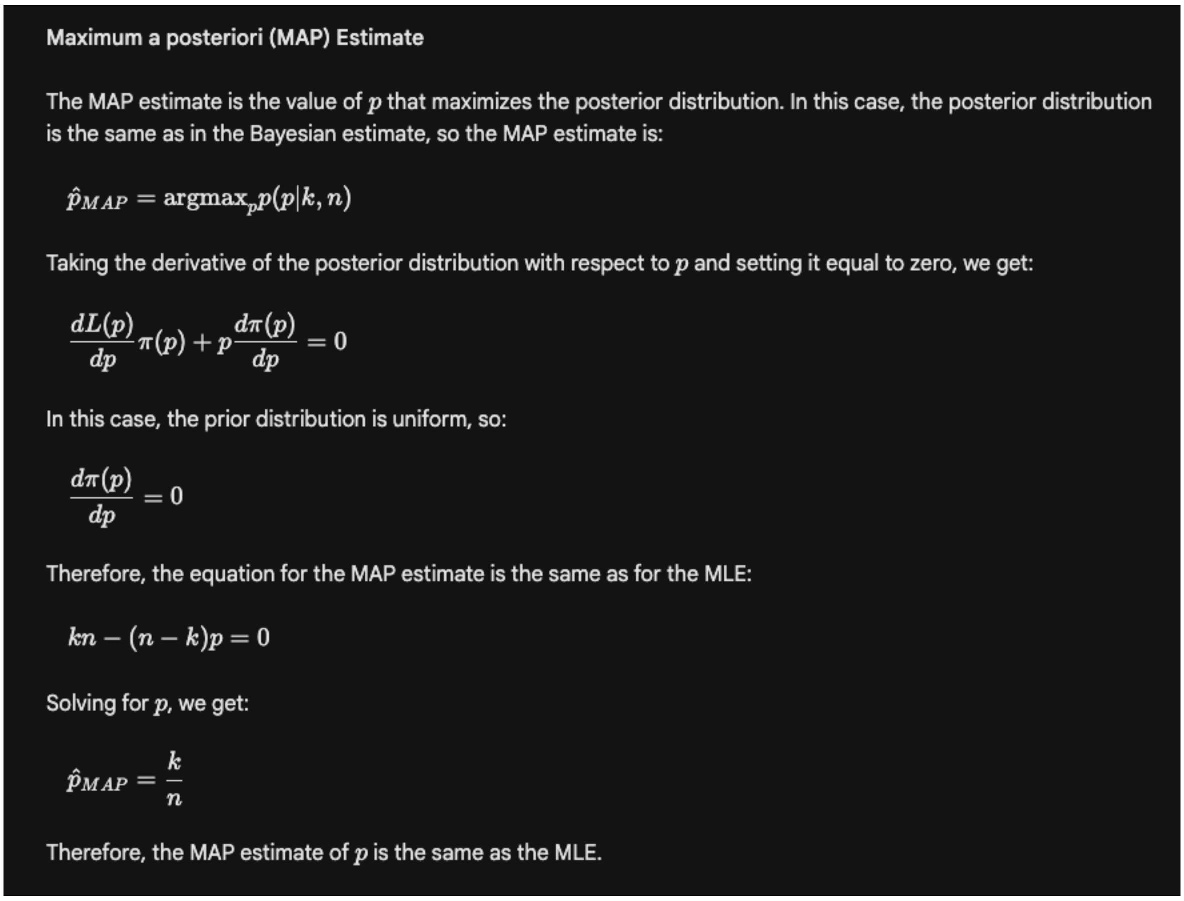
$$\hat{p}_{Bayes} = \int_0^1 p \binom{n}{k} p^k (1-p)^{n-k} dp$$

Evaluating the integral, we get:

$$\hat{p}_{Bayes} = \frac{k+1}{n+2}$$

Therefore, the Bayesian estimate of p is:

$$\hat{p}_{Bayes} = \frac{k+1}{n+2}$$



Maximum a posteriori (MAP) Estimate

The MAP estimate is the value of p that maximizes the posterior distribution. In this case, the posterior distribution is the same as in the Bayesian estimate, so the MAP estimate is:

$$\hat{p}_{MAP} = \text{argmax}_p p(p|k, n)$$

Taking the derivative of the posterior distribution with respect to p and setting it equal to zero, we get:

$$\frac{dL(p)}{dp}\pi(p) + p\frac{d\pi(p)}{dp} = 0$$

In this case, the prior distribution is uniform, so:

$$\frac{d\pi(p)}{dp} = 0$$

Therefore, the equation for the MAP estimate is the same as for the MLE:

$$kn - (n - k)p = 0$$

Solving for p, we get:

$$\hat{p}_{MAP} = \frac{k}{n}$$

Therefore, the MAP estimate of p is the same as the MLE.