**Q1.** When you duplicate feature (n) into feature (n+1) and retrain a new model, the relationship between the weights of the new features and the original weights can vary depending on several factors. However, some general patterns can be observed:

1. **Similarity in Weights for Duplicated Features:** In many cases, if the duplicated feature is providing similar information or has similar importance as the original feature, the weights assigned to the duplicated feature and the original feature in the new model may be similar.
2. **Equal Weights (In Special Cases):** If the duplicated feature is essentially a copy of the original feature, and the model has enough capacity to learn this, the weights for the duplicated features may end up being approximately equal.
3. **Different Weights (Complex Interactions):** If the duplicated feature introduces new information or interacts differently with other features, the weights for the duplicated feature and the original feature may differ.
4. **Redundancy Handling:** Some models, especially regularized ones, may adjust the weights to handle redundancy. In this case, the weights may be distributed differently to reduce overfitting or to better capture the underlying patterns.
5. **Regularization Effects:** The use of regularization techniques (e.g., L1 or L2 regularization) can also influence the relationship between the weights. Regularization may shrink certain weights towards zero or distribute the importance differently.

To summarize, the relationship can be influenced by the nature of the data, the model architecture, regularization, and the interactions among features. It's not possible to determine a specific relationship without more details about the data and the training process.

**Q2.** To determine which statement is true, we can analyze the confidence intervals for the click-through rates (CTR) of each template.

Let's calculate the 95% confidence intervals for each template:

Template A (control): 10% ± Margin of Error

Template B: 7% ± Margin of Error

Template C: 8.5% ± Margin of Error

Template D: 12% ± Margin of Error

Template E: 14% ± Margin of Error

Now, if the confidence interval of one template does not overlap with the confidence interval of another, we can have over 95% confidence that they are different.

Based on the information given, we can make the following observations:

Template E has a higher CTR than Template A (10% ± Margin of Error).

Template B has a lower CTR than Template A (7% ± Margin of Error).

Since the confidence intervals for C and D are not provided, we cannot make a definitive conclusion about their comparison with A.

Therefore, the correct statement is:

**b)** E is better than A with over 95% confidence, B is worse than A with over 95% confidence. You need to run the test for longer to tell where C and D compare to A with 95% confidence.

**Q3.** In logistic regression, the computational cost of each gradient descent iteration is dominated by the computation of the gradient and updating the weights. The specific computational cost depends on the implementation and the sparsity of the feature vectors.

Let's break down the key components:

1. **Gradient Computation:** In logistic regression, the gradient for the weights is computed as a sum over all training examples. For sparse data, where each training example has an average of (k) non-zero entries, the gradient computation can be done efficiently by considering only the non-zero entries. Therefore, the computational cost for gradient computation is roughly proportional to (k\*m).
2. **Weight Update:** The weight update involves updating each weight based on its corresponding gradient. This operation has a computational cost proportional to the number of features, which is (n).

Therefore, the overall computational cost per iteration of gradient descent for logistic regression with sparse data is approximately:

Cost ≈ (k\*m + n)

It's important to note that this is a rough approximation, and the actual performance may vary based on the specific implementation details and the optimizations used in the machine learning library being employed.Top of Form

**Q4.** Let's analyze each approach and its potential impact on the accuracy of the V2 classifier:

**Approach 1:**

1. **Description:** Run the 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 label them.
2. **Analysis:** These examples are likely to be close to the decision boundary of the V1 classifier, meaning they are challenging cases. Labelling these examples may help the V2 classifier improve its performance on ambiguous instances. However, if V1 had difficulty classifying these examples accurately, it may be challenging for the V2 classifier as well.

**Approach 2:**

1. **Description:** Get 10k random labelled stories from the 1000 news sources.
2. **Analysis:** This approach provides a diverse set of examples from the 1000 news sources, without specifically targeting challenging cases. It ensures that the training set is representative of the overall distribution of news stories.

**Approach 3:**

1. **Description:** Pick a random sample of 1 million stories from 1000 news sources and have them labelled. Pick the subset of 10k stories where the V1 classifier’s output is both wrong and farthest away from the decision boundary.
2. **Analysis:** By selecting cases where V1 was wrong and farthest from the decision boundary, this approach focuses on instances where the V1 classifier was most confident but incorrect. This might help correct cases where the V1 classifier was overly confident but made mistakes.

**Ranking:**

1. **Approach 3:** This approach may be the most beneficial for improving the accuracy of the V2 classifier by focusing on correcting instances where the V1 classifier was confidently wrong.
2. **Approach 2:** Randomly selecting labelled stories provides a diverse set of examples and is generally a good strategy for training a classifier on a representative dataset.
3. **Approach 1:** While focusing on challenging cases close to the decision boundary can be helpful, it may not be as effective as approaches 2 and 3 in improving overall accuracy.Bottom of Form

**Q5.** Let's derive the estimates for each method:

**1. Maximum Likelihood Estimate (MLE):** The MLE for the probability of heads, (p), is simply the ratio of the number of heads to the total number of coin tosses: MLE: (p)^=k/n

**2. Bayesian Estimate:** Here, we assume a continuous uniform prior on (p) from 0 to 1. The posterior distribution is a Beta distribution, and the Bayesian estimate is the expected value of this distribution.

For a binomial distribution with a Beta prior, the posterior distribution is also a Beta distribution: Beta(k+1,n−k+1)​

Therefore, the Bayesian estimate is Bayesian: (p)^=(k+1)/(n+2)​

**3. Maximum a Posteriori (MAP) Estimate:**

The MAP estimate is the mode of the posterior distribution. Therefore, the MAP estimate is MAP: (p)^​​ = n/k​

To summarize:

**MLE:** (p)^ = (n/k)​

**Bayesian Estimate:** (p)^ = (k+1/n+2)

**MAP Estimate:** (p)^ = (n/k)