Assignment 1: Cross Connection PUF

NEURAL NEXUS

Group Members:

1.	Lekansh Bhatnagar	220586
2.	Kollamoram Karthik	220538
3.	Jiyanshu Dhaka	220481
4.	Harshit Agarwal	220438
5.	Rohit Karwa	220911
6.	Ayush	220259

Date of Submission: July 17th, 2024

University: Indian Institute of Technology Kanpur

Course: CS771 Introduction to Machine Learning

Instructor: Purushottam Kar **Total Marks:** 60

Contents

Abstract

This report explores the Cross-Connection PUF (COCO-PUF), a new variant of Physical Unclonable Functions (PUFs), which uses two arbiter PUFs and cross-connects their signals to generate two responses per challenge. The goal is to prove that despite the added complexity of COCO-PUFs, their responses can still be predicted using linear models. The mathematical derivation demonstrates how linear models can predict the arrival time of signals in a simple arbiter PUF and extends this to predict responses in a COCO-PUF. The dimensionality of the feature vector required for accurate predictions is determined. Practical experiments using Linear Support Vector Classifier (LinearSVC) and Logistic Regression on a provided dataset of challenge-response pairs validate the theoretical findings. This work concludes that COCO-PUF responses can indeed be effectively predicted using linear models, thereby challenging the presumed difficulty in modeling COCO-PUFs linearly

1 PART 1

1.1 Diagram and Notations

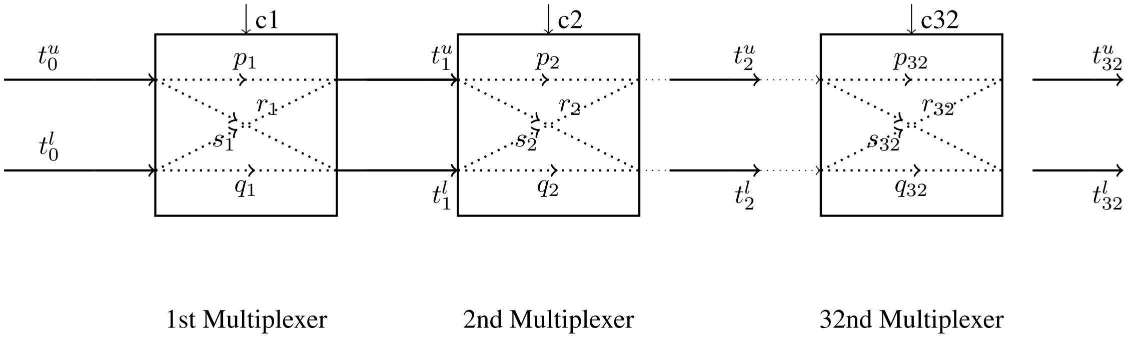

Figure 1: A simple arbiter PUF with 32 multiplexers

- c_i: challenge bit given to the i^{th} multiplexer. (Thus, $c_i \in \{0,1\}\ \forall i \in \{1,2,\ldots,32\}$)
- challenge vector **c** = $(c_1, c_2, \ldots, c_{32})$. (Thus, **c** $\in \{0,1\}^{32}$)
- p_i, q_i, r_i, s_i: delays associated with the i^{th} multiplexer.
- t_i^u: time taken by the, relatively, upper signal to reach $(i+1)^{th}$ multiplexer.
- t_i^l: time taken by the, relatively, upper signal to reach $(i+1)^{th}$ multiplexer.
- $t^u(\mathbf{c})\ or\ t_{32}^u$: time at which the upper signal emerges out of the final multiplexer.
- $t^l(\mathbf{c})\ or\ t_{32}^l$: time at which the lower signal emerges out of the final multiplexer.
- Time Lag Δ_i = t_i^u - t_i^l, $\forall i \in \{0,1,2,\ldots,32\}$.
- d_i = $(1-2c_i)$, $\forall i \in \{1,2,\ldots,32\}$. (please note that, $d_i \in \{-1,1\}$)
- α_i = $(p_i - q_i + r_i - s_i)/2$, $\forall i \in \{1,2,\ldots,32\}$.
- β_i = $(p_i - q_i - r_i + s_i)/2$, $\forall i \in \{1,2,\ldots,32\}$.

1.2 Results obtained in class

$$t_i^u = (1-c_i)\cdot(t_{i-1}^u + p_i) + c_i \cdot (t_{i-1}^l + s_i) \tag{1}$$

$$t_i^l = (1-c_i)\cdot(t_{i-1}^l + q_i) + c_i \cdot (t_{i-1}^u + r_i) \tag{2}$$

$$\Delta_i \ = \ d_i \cdot \Delta_{i-1} \ + \ \alpha_i \cdot d_i \ + \ \beta_i$$

we can safely set $t_0^u = 0$ and $t_0^l = 0$. (absorb initial delays into p1, q1, r1 and s1)
with continued observation, we obtain the following recurring pattern:

$$\Rightarrow \Delta_i = \begin{cases} 0 & \text{if } i = 0 \\ \left(\sum_{j=1}^{i} w_j x_{ji}\right) + b_i & \text{if } i \neq 0 \end{cases} \tag{3}$$

where:

$$\Rightarrow x_{ji} = \prod_{r=j}^{i} d_r$$

$$\Rightarrow w_j = \begin{cases} \alpha_1 & \text{if } j = 1 \\ \alpha_j + \beta_{j-1} & \text{if } j \neq 1 \end{cases}$$

$$\Rightarrow b_i = \beta_i$$

1.3 Mathematical derivation

We can rewrite equation (1) as follows:

$$\Rightarrow t_i^u - t_{i-1}^u = p_i + c_i \cdot (s_i - p_i) - c_i \cdot \Delta_{i-1} \tag{4}$$

Summing the above equation over all values of i yields:

$$\Rightarrow \sum_{i=1}^{32}(t_i^u \ - \ t_{i-1}^u) \ = \ \sum_{i=1}^{32} c_i \cdot (s_i - p_i) \ - \ \sum_{i=2}^{32} c_i \cdot \Delta_{i-1} \ + \ \sum_{i=1}^{32} p_i \tag{5}$$

Expanding Δ_{i-1} using equation (3), we obtain:

$$\Rightarrow t_{32}^u \ - \ t_0^u \ = \ \sum_{i=1}^{32} c_i \cdot (s_i - p_i) \ + \ \sum_{i=1}^{32} p_i \ - \ \sum_{i=2}^{32} c_i \cdot b_{i-1} \ - \ \sum_{i=2}^{32}\sum_{j=1}^{i-1} c_i w_j x_{j(i-1)} \tag{6}$$

Assuming $t_0^u = 0$ and $b_0 = 0$, the above equation can be expressed as:

$$\Rightarrow t_{32}^u = \sum_{i=1}^{32}(s_i - p_i - b_{i-1}) \cdot c_i + \sum_{i=1}^{32} p_i - \sum_{j=1}^{31} w_j \left(\sum_{i=j+1}^{32} c_i x_{j(i-1)} \right) \tag{7}$$

$$\Rightarrow t_{32}^u = \sum_{i=1}^{32}(s_i - p_i - b_{i-1}) \cdot c_i + \sum_{j=1}^{31} w_j \left(- \sum_{i=j+1}^{32} c_i x_{j(i-1)} \right) + \sum_{i=1}^{32} p_i \tag{8}$$

The equation above, as evident, takes the form $W^T \phi(\mathbf{c}) + b = t^u(c)$,indicating that a linear model can predict the time for the upper signal to reach the finish line.

Please note that:

$$\Rightarrow t^u(\mathbf{c}) = t_{32}^u$$

$$\Rightarrow \phi(\mathbf{c}) = (\phi_i)_{63 \times 1} \in \mathbb{R}^{63}$$

$$\Rightarrow W = (w_i)_{63 \times 1}$$

$$\Rightarrow \phi_i = \begin{cases} -\sum_{j=i+1}^{31} c_j x_{i(j-1)} & \text{if } 1 \le i \le 31 \\ c_{i-31} & \text{if } 32 \le i \le 63 \end{cases}$$

$$\Rightarrow w_i = \begin{cases} w_i & \text{if } 1 \le i \le 31 \\ s_{i-31} - p_{i-31} - b_{i-32} & \text{if } 32 \le i \le 63 \end{cases}$$

$$\Rightarrow b = \sum_{i=1}^{32} p_i$$

1.4 Conclusion

The final equation:

$$\boxed{t_{32}^u = \sum_{i=1}^{32}(s_i - p_i - b_{i-1}) \cdot c_i + \sum_{j=1}^{31} w_j \left(- \sum_{i=j+1}^{32} c_i x_{j(i-1)} \right) + \sum_{i=1}^{32} p_i}$$

demonsrtates that the upper signal's time to reach the finish line, $t^u(\mathbf{c})$, can be predicted using a linear model with the aforementioned model parameters and feature vectors.

2 PART 2

2.1 Answer

The linear model we have derived in Part 1 predicts t^u**(c)** with the help of 63 weight parameters and one bias term. Therefore, the **dimensionality** of our model is **64**.

⇒ Hence, (Dimensionality) **D = 64**

3 PART 3

3.1 Notations

In the context of this study, we use the following notations:

- $t_1^l(c)$: Arrival time of the lower signal from PUF1 for challenge c.
- $t_0^l(c)$: Arrival time of the lower signal from PUF0 for challenge c.
- $t_1^u(c)$: Arrival time of the upper signal from PUF1 for challenge c.
- $t_0^u(c)$: Arrival time of the upper signal from PUF0 for challenge c.
- b_b, b_a, b_c, b_d: Bias terms associated with PUF0 and PUF1.
- W_a, W_b, W_c, W_d: Weight vectors associated with PUF0 and PUF1.
- $\phi(c)$: Feature map representing the challenge vector c.
- $\Delta_0(c)$: Time difference between $t_0^l(c)$ and $t_1^l(c)$.
- $\Delta_1(c)$: Time difference between $t_0^u(c)$ and $t_1^u(c)$.
- $\mathbf{r_0(c)}$: Response0 for challenge vector c, indicating the decision based on $\Delta_0(c)$.
- $\mathbf{r_1(c)}$: Response1 for challenge vector c, indicating the decision based on $\Delta_1(c)$.

These notations are integral to the formulation and understanding of the linear models used to predict responses from the COCO-PUF based on signal arrival times.

3.2 Inferences from Part 1

Both the lower signal and upper signal undergo identical mathematical treatment, except for the changes in delays. Therefore, the time taken by the lower signal to reach the finish line can also be predicted by a similar linear model of the same dimensionality. Therefore, we can write the following four equations:

$$\Rightarrow t_1^l(c) \;=\; W_a^T\phi(c) \;+\; b_a \tag{9}$$
$$\Rightarrow t_0^l(c) \;=\; W_b^T\phi(c) \;+\; b_b \tag{10}$$
$$\Rightarrow t_1^u(c) \;=\; W_c^T\phi(c) \;+\; b_c \tag{11}$$
$$\Rightarrow t_0^u(c) \;=\; W_d^T\phi(c) \;+\; b_d \tag{12}$$

It is important to note that since $\phi(c)$ depends only on c and not on the delays, it remains consistent for both the signals and for both the arbiter PUFs.

3.3 Predicting Response 0

Let's start by calculating $\Delta_0(c)$,

$$\Rightarrow \Delta_0(c) \;=\; t_0^l(c) \;-\; t_1^l(c) \tag{13}$$
$$\Rightarrow \Delta_0(c) \;=\; W_b^T\phi(c) \;-\; W_a^T\phi(c) \;+\; b_b \;-\; b_a \tag{14}$$
$$\Rightarrow \Delta_0(c) \;=\; (W_b^T \;-\; W_a^T)\,\phi(c) \;+\; (b_b \;-\; b_a) \tag{15}$$
$$\Rightarrow \Delta_0(c) \;=\; (W_b \;-\; W_a)^T\,\phi(c) \;+\; (b_b \;-\; b_a) \tag{16}$$

Given vectors W_a, W_b and scalars b_a, b_b, we define:

$$W_0 = W_b - W_a$$

$$b_0 = b_b - b_a$$

Then, equation (16) can be written as:

$$\Rightarrow \Delta_0(c) = W_0^T\phi(c) + b_0 \tag{17}$$

As we know, Response0, denoted as $\mathbf{r_0(c)}$, is determined by the arrival times of the lower signals from PUF0 and PUF1. Specifically:

$$\mathbf{r_0(c)} = \begin{cases} 1, & \text{if } \Delta_0(c) > 0 \\ 0, & \text{if } \Delta_0(c) < 0 \end{cases}$$

This can also be expressed using the sign function:

$$\mathbf{r_0(c)} = \frac{1 + \text{sign}(\Delta_0(c))}{2}$$

where $\Delta_0(c) = W_0^T \phi(c) + b_0$ represents the difference in signals from PUF0 and PUF1. Equation (17) can be utilized to express this relationship formally:

$$\Rightarrow \boxed{\mathbf{r_0(c)} = \frac{1 + \text{sign}(W_0^T \phi(c) + b_0)}{2}} \tag{18}$$

In this expression, $\text{sign}(x)$ denotes the sign function, which returns $+1$ if $x > 0$and -1 if $x < 0$.

3.4 Predicting Response 1

Similarly to Response 0, Response 1, denoted as $\mathbf{r_1(c)}$, is determined by the arrival times of the upper signals from PUF0 and PUF1. The difference in arrival times, $\Delta_1(c)$, can be calculated as follows:

$$\begin{aligned} &\Rightarrow \Delta_1(c) = t_0^u(c) - t_1^u(c) && (19) \\ &\Rightarrow \Delta_1(c) = (W_d^T \phi(c) + b_d) - (W_c^T \phi(c) + b_c) && (20) \\ &\Rightarrow \Delta_1(c) = (W_d^T - W_c^T)\phi(c) + (b_d - b_c) && (21) \\ &\Rightarrow \Delta_1(c) = (W_d - W_c)^T \phi(c) + (b_d - b_c) && (22) \end{aligned}$$

Given vectors W_c, W_d and scalars b_c, b_d, define:

$$W_1 = W_d - W_c$$

$$b_1 = b_d - b_c$$

Then, equation (22) can be written as:

$$\Rightarrow \Delta_1(c) = W_1^T \phi(c) + b_1 \tag{23}$$

As we know, $\mathbf{r_1(c)}$, Response 1, is determined similarly:

$$\mathbf{r_1(c)} = \begin{cases} 1, & \text{if } \Delta_1(c) > 0 \\ Contents0, & \text{if } \Delta_1(c) < 0 \end{cases}$$

This can also be expressed using the sign function:

$$\mathbf{r_1(c)} = \frac{1 + \text{sign}(\Delta_1(c))}{2}$$

where $\Delta_1(c) = W_1^T \phi(c) + b_1$ represents the difference in signals from PUF0 and PUF1 for Response 1. Equation (23) formalizes this relationship:

$$\Rightarrow \boxed{\mathbf{r_1(c)} = \frac{1 + \text{sign}(W_1^T \phi(c) + b_1)}{2}} \tag{24}$$

In this expression, $\text{sign}(x)$ denotes the sign function, returning $+1$ if $x > 0$ and -1 if $x < 0$.

3.5 Conclusion

We have shown that linear models effectively predict Response0 and Response1 for a COCO-PUF based on signal arrival times. The models are formulated as:

$$\mathbf{r_0}(\mathbf{c}) = \frac{1 + \text{sign}(W_0^T \phi(c) + b_0)}{2}$$

$$\mathbf{r_1}(\mathbf{c}) = \frac{1 + \text{sign}(W_1^T \phi(c) + b_1)}{2}$$

These equations categorize responses using feature vector $\phi(c)$ that abstract away PUF-specific delays.

4 PART 4

4.1 Answer

The linear models derived in Part 3 predict Response0 and Response1 of a COCO-PUF using weight vectors of dimensionality 63 and a bias term each. Thus, the total **dimensionality** required to predict both Response0 and Response1 is **64**. This dimensionality corresponds to the requirement for predicting the arrival times of the upper signal in a simple arbiter PUF.

$\Rightarrow$ Hence, (Dimensionality) **D = 64**

5 PART 6

5.1 Linear SVC

If we choose to use **LinearSVC** for both the Responses, we get the following results by altering the parameters:

5.1.1 Loss Hyperparameter

1. **Hinge**: Training time for $Hinge\ Loss$ is found to be **3.75 seconds** and it gives test accuracy for **Response 0 as 0.98392** and for **Response 1 as 0.99444**.

2. **Squared Hinge**: Training time for $Squared\ Hinge\ Loss$ is found to be **6.05 seconds** and it gives test accuracy for **Response 0 as 0.9801** and for **Response 1 as 0.99660**.

   We can thus summarize for this specific case that, Hinge Loss is giving almost the same accuracy as Squared Hinge Loss (even better for Response 0) and takes a lot less time to train the model.

5.1.2 C Hyperparameter

1. **High**: Training time for $High\ Value$ of **C(=100)** is found to be **9.076 seconds** and it gives test accuracy for **Response 0 as 0.95656** and for **Response 1 as 0.99564**.

2. **Medium**: Training time for $Medium\ Value$ of **C(=1)** is found to be **6.27 seconds** and it gives test accuracy for **Response 0 as 0.9801** and for **Response 1 as 0.99660**.

3. **Low**: Training time for $Low\ Value$ of **C(=0.01)** is found to be **3.66 seconds** and it gives test accuracy for **Response 0 as 0.9778** and for **Response 1 as 0.983**.

We can thus summarize for this specific case that, for **Medium value of C**, the model has the most accurate predictions. But, we can also see that **Low value of C** gives about 1 percent less accuracy (on average for both responses) than for Medium value, but it takes a lot less time to train the model. Thus, we can select any one value in between these two to optimize the model for time and accuracy.

5.1.3 Tolerance

1. **High**: Training time for $High\ Value$ of **tol(=0.1)** is found to be **4.86 seconds** and it gives test accuracy for **Response 0 as 0.97978** and for **Response 1 as 0.99674**.

2. **Medium**: Training time for $Medium\ Value$ of **tol(=0.001)** is found to be **5.74 seconds** and it gives test accuracy for **Response 0 as 0.9801** and for **Response 1 as 0.99660**.

3. **Low**: Training time for $Low\ Value$ of **tol(=0.00001)** is found to be **6.32 seconds** and it gives test accuracy for **Response 0 as 0.9801** and for **Response 1 as 0.99660**.

   We can thus summarize for this specific case that, for **Medium value of tol**, the model has the most accurate predictions and it takes the least amount of time to train the model among the three cases.

5.1.4 Penalty

1. **l2**: Training time for **penalty as l2** is found to be **4.819 seconds** and it gives test accuracy for **Response 0 as 0.9801** and for **Response 1 as 0.99660**.

2. **l1**: Training time for **penalty as l1** is found to be **21.43 seconds** and it gives test accuracy for **Response 0 as 0.9800** and for **Response 1 as 0.99648**.

   We can thus summarize for this specific case that, for **l2 penalty**, the model has the most accurate predictions and it takes the least amount of time to train the model among the two cases.

5.2 Logistic Regression

If we choose to use **LogisticRegression** for both the Responses, we get the following results by altering the parameters:

5.2.1 C Hyperparameter

1. **High**: Training time for $High\ Value$ of **C(=100)** is found to be **4.229 seconds** and it gives test accuracy for **Response 0 as 0.9808** and for **Response 1 as 0.9984**.

2. **Medium**: Training time for $Medium\ Value$ of **C(=1)** is found to be **3.043 seconds** and it gives test accuracy for **Response 0 as 0.9803** and for **Response 1 as 0.9953**.

3. **Low**: Training time for $Low\ Value$ of **C(=0.01)** is found to be **3.747 seconds** and it gives test accuracy for **Response 0 as 0.9647** and for **Response 1 as 0.9672**.

   We can thus summarize for this specific case that, for **Medium value of C**, the model has the most accurate predictions and takes the least amount of time to train the model.

5.2.2 Tolerance

1. **High**: Training time for $High\ Value$ of **tol(=0.1)** is found to be **4.92 seconds** and it gives test accuracy for **Response 0 as 0.9803** and for **Response 1 as 0.9953**.

2. **Medium**: Training time for $Medium\ Value$ of **tol(=0.001)** is found to be **3.023 seconds** and it gives test accuracy for **Response 0 as 0.9803** and for **Response 1 as 0.9953**.

3. **Low**: Training time for $Low\ Value$ of **tol(=0.00001)** is found to be **3.736 seconds** and it gives test accuracy for **Response 0 as 0.9803** and for **Response 1 as 0.9953**.

   We can thus summarize for this specific case that, for **all values of tol**, the model has the most same accuracies and it takes the least amount of time to train the model for **medium value of tol** among the three cases.

5.2.3 Penalty

1. **l2**: Training time for **penalty as l2** is found to be **3.051 seconds** and it gives test accuracy for **Response 0 as 0.9803** and for **Response 1 as 0.9953**.

2. **l1**: Training time for **penalty as l1** is found to be **35.13 seconds** and it gives test accuracy for **Response 0 as 0.9810** and for **Response 1 as 0.9978**.

   We can thus summarize for this specific case that, for **l2 penalty**, the model has slightly less accurate results but it takes a lot less amount of time to train the model among the two cases.

6 PLOTS

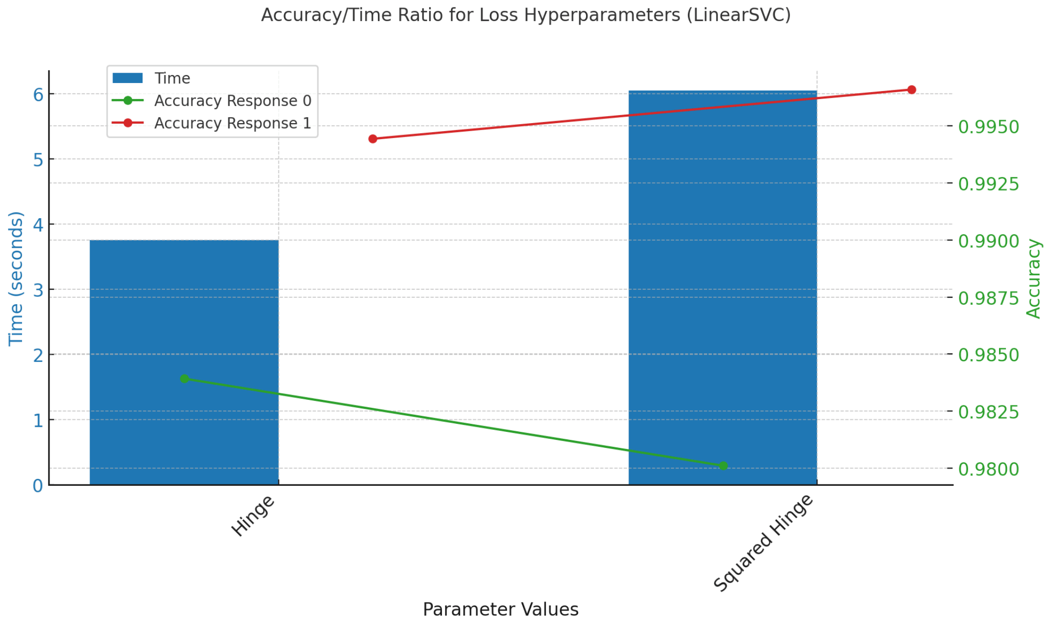

Figure 2: Accuracy/Time Ratio for Loss Hyperparameters (LinearSVC)

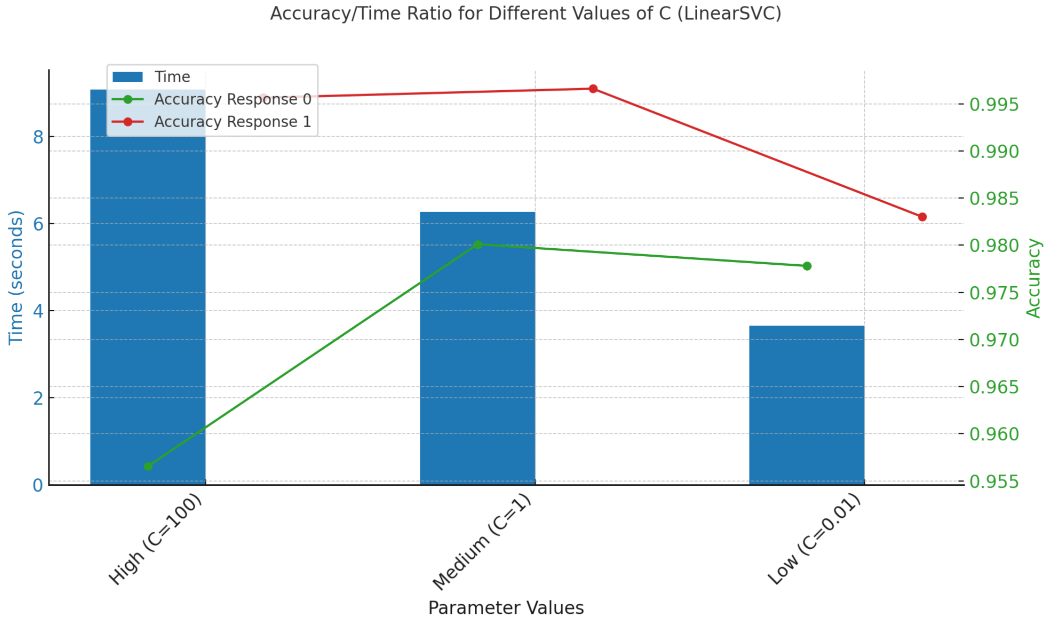

Figure 3: Accuracy/Time Ratio for Different Values of C (LinearSVC)

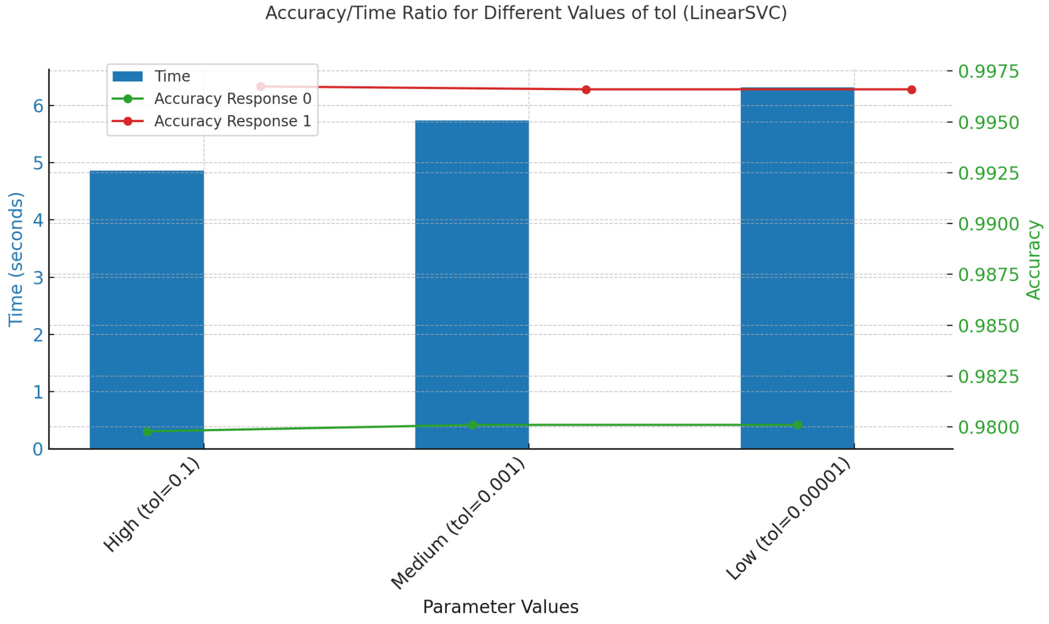

Figure 4: Accuracy/Time Ratio for Different Values of tol (LinearSVC)

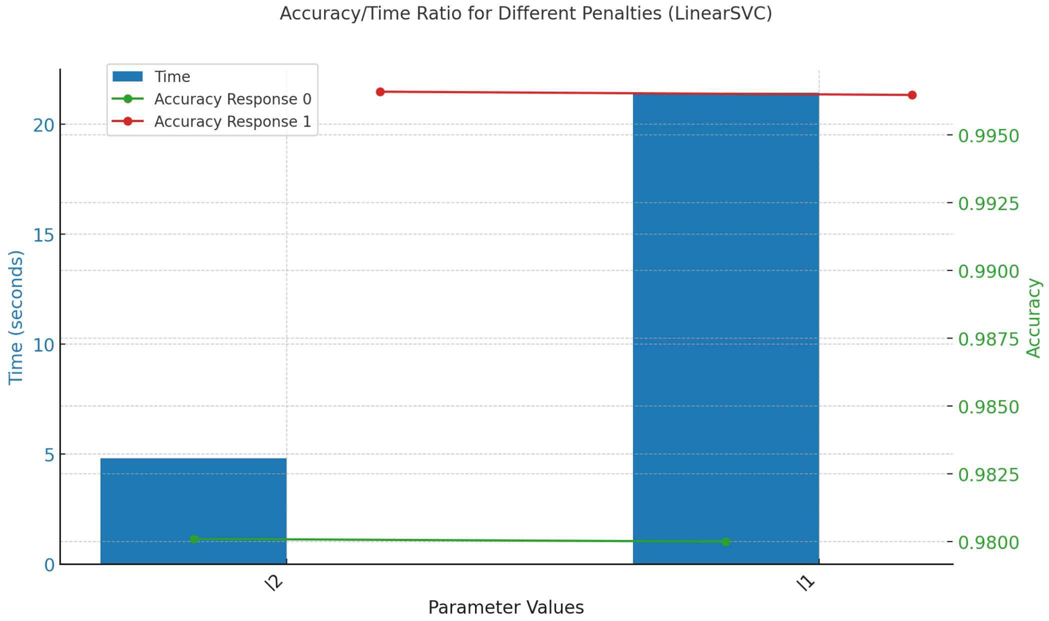

Figure 5: Accuracy/Time Ratio for Different Penalties (LinearSVC)

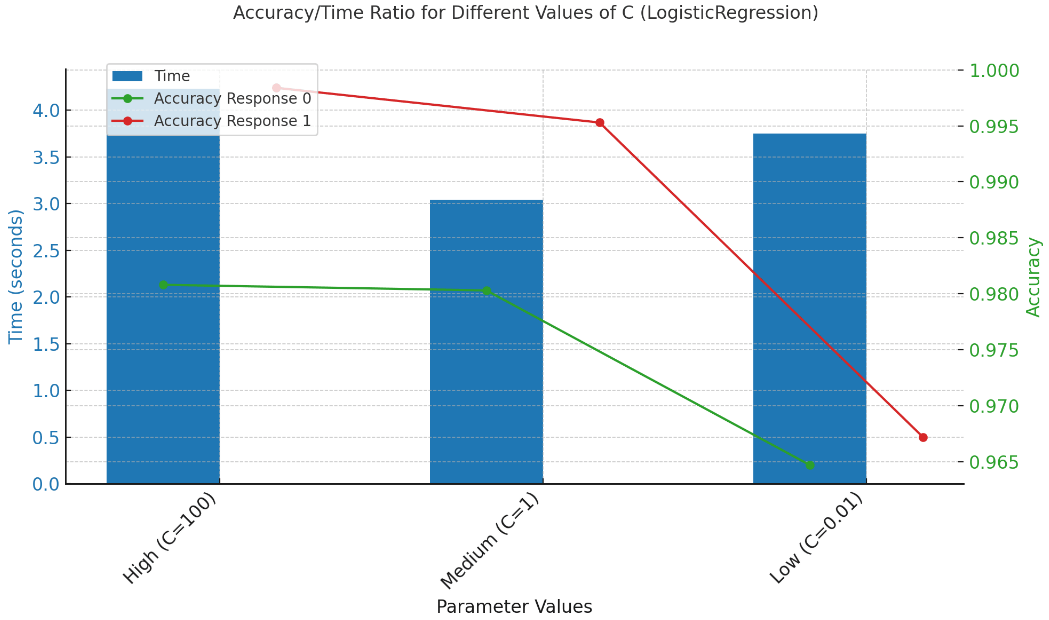

Figure 6: Accuracy/Time Ratio for Different Values of C (LogisticRegression)

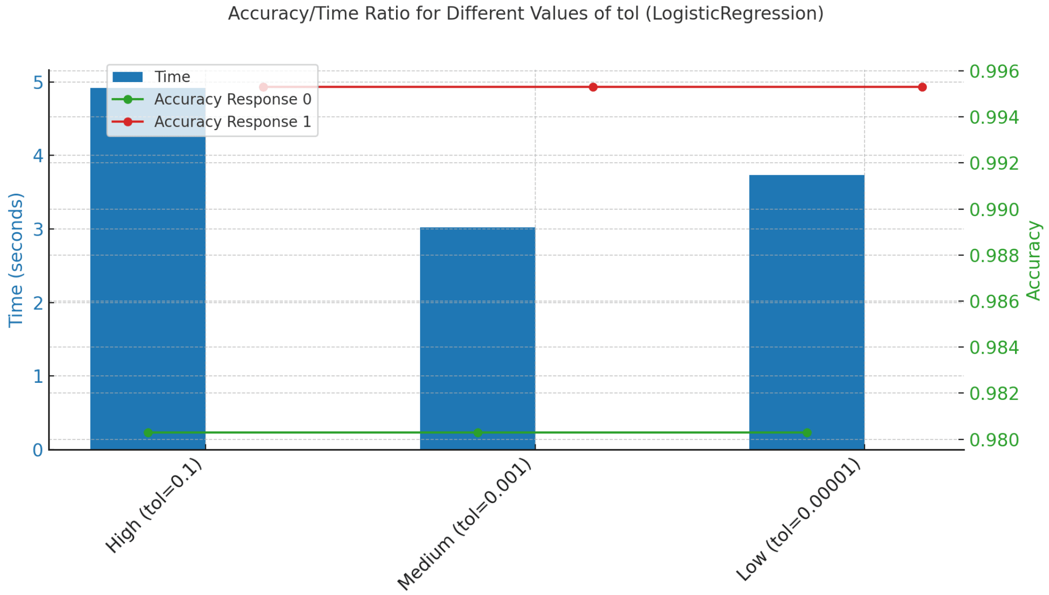

Figure 7: Accuracy/Time Ratio for Different Values of tol (LogisticRegression)

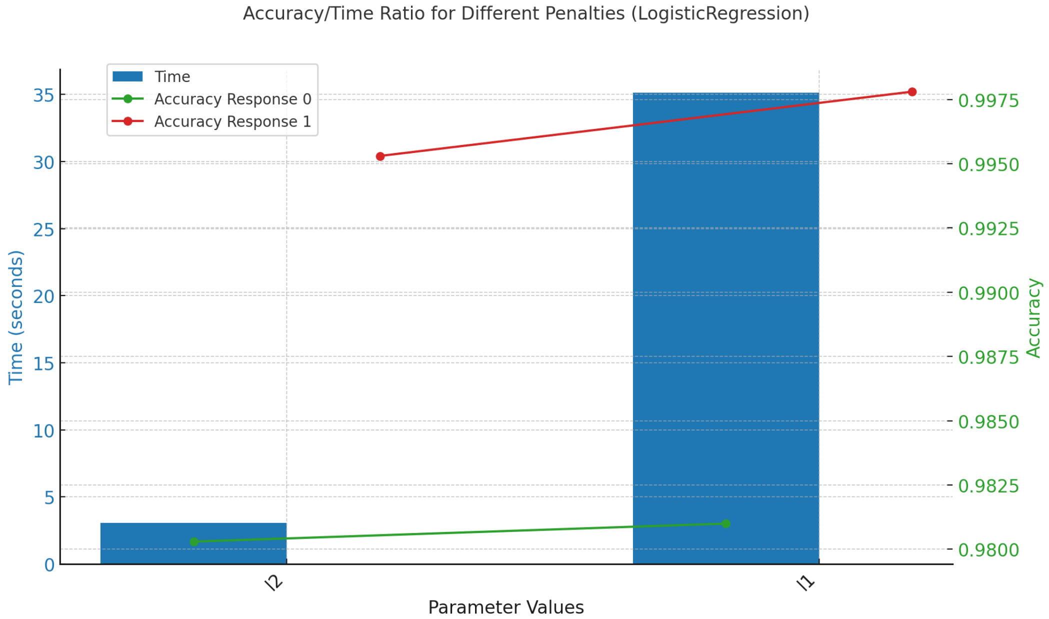

Figure 8: Accuracy/Time Ratio for Different Penalties (LogisticRegression)