CS771 : Introduction to Machine Learning

Assignment - 1

Team: Profanities

Aryan Prajapati
210203
Material Science and Engineering
aryanp21@iitk.ac.in

Manish Kumar
210584
Material Science and Engineering
manishkv21@iitk.ac.in

Manasvi Jeevan Kweera
210582
Earth Sciences
manasvi21@iitk.ac.in

Kumar Gunjan
210542
Material Science and Engineering
kgunjan21@iitk.ac.in

Abstract

In this report, we present the solution for the first assignment of CS771 - Intro to Machine Learning. Our task involves demonstrating, through mathematical derivations, how a linear model can predict the time it takes for the upper signal in a simple arbiter PUF to reach the finish line. Using this derivation, we further show how the model can predict Response0 and Response1 for a COCO-PUF. Finally, we present the results of experiments using bluesklearn.svm.LinearSVC and bluesklearn.linear_model.LogisticRegression to learn the linear model for the COCO-PUF, examining how different hyperparameters affect training time and test accuracy.

1 Linear model to predict the time it takes for the upper signal to reach the finish line for Arbiter PUF

1.1 Arbiter PUFs – A Brief Introduction

Arbiter Physically Unclonable Functions (PUFs) are hardware systems that use unpredictable variations in data transmission speed to enhance security. They are made up of multiple multiplexers (muxes), each controlled by a selection bit. A specific set of selection bits forms a "challenge," and the signal that reaches the end of the PUF first is the "response." The response to a given challenge is unique to each PUF's hardware.

1.2 Using ML to find the time taken by upper signal to reach the finish line

Research shows that with data on the time taken by a few challenges, we can train a model to predict the time for any other challenge in a given arbiter PUF. Here's how it works:

t_i^u is the (unknown) time at which the upper signal leaves the i-th mux. t_i^l is the time at which the lower signal leaves the i-th mux. All muxes are different so that p_i, r_i, s_i, and q_i are distinct. Therefore, the answer is 0 if $t_{31}^u < t_{31}^l$ and 1 otherwise. (Here, we have assumed zero based indexing which implies $t_{-1}^u = t_{-1}^l = 0$)

Now, as given in the lecture slides -

$$t_n^u = (1 - c_n).(t_{n-1}^u + p_n) + c_n.(t_{n-1}^l + s_n) \tag{1}$$

$$t_n^l = (1 - c_n).(t_{n-1}^l + q_n) + c_n.(t_{n-1}^u + r_n) \tag{2}$$

Adding equations (1) and (2), we get -

$$t_n^u + t_n^l = t_{n-1}^u + t_{n-1}^l + (1 - c_n).(p_n + q_n) + c_n.(r_n + s_n) \tag{3}$$

For n = 0,

$$t_0^u + t_0^l = (1 - c_0)(p_0 + q_0) + c_0(r_0 + s_0) \text{ (Since, } t_{-1}^u = 0 \text{ \& } t_{-1}^l = 0) \tag{4}$$

For n = 1,

$$t_1^u + t_1^l = (1 - c_0).(p_0 + q_0) + c_0.(r_0 + s_0) + (1 - c_1).(p_1 + q_1) + c_1.(r_1 + s_1) \tag{5}$$

For n = 2,

$$t_2^u + t_2^l = (1-c_0).(p_0+q_0)+c_0.(r_0+s_0)+(1-c_1).(p_1+q_1)+c_1.(r_1+s_1)+(1-c_2).(p_2+q_2)+c_2.(r_2+s_2) \tag{6}$$

By induction-

$$t_{31}^u + t_{31}^l = \sum_{i=0}^{31} (1 - c_i).(p_i + q_i) + c_i.(r_i + s_i) \tag{7}$$

From the lecture slides we have expression for the difference of the equation (1) and (2),

$$\Delta_i = d_i \cdot \Delta_{i-1} + \alpha_i \cdot d_i + \beta_i \tag{8}$$

where,

$$\begin{aligned} \Delta_i &= t_i^u - t_i^l \\ d_i &= (1 - 2c_i) \\ \alpha_i &= (p_i - q_i + r_i - s_i)/2 \\ \beta_i &= (p_i - q_i - r_i + s_i)/2 \end{aligned}$$

For i = 0,

$$\Delta_0 = \alpha_0.d_0 + \beta_0 \text{ (Since, } \Delta_{-1} = 0) \tag{9}$$

Let us take -

$$A_i = \alpha_i.d_i + \beta_i \tag{10}$$

Then,

$$\Delta_0 = A_0 \tag{11}$$

For i = 1,

$$\Delta_1 = d_1.\Delta_0 + \alpha_1.d_1 + \beta_1 \tag{12}$$

$$\Delta_1 = d_1.A_0 + A_1 \tag{13}$$

For i = 2,

$$\Delta_2 = d_2.d_1.A_0 + d_2.A_1 + A_2 \tag{14}$$

Similarly,

$$\Delta_{31} = \sum_{i=0}^{31} A_i. \prod_{j=i+1}^{31} d_j \tag{15}$$

So,

$$t_{31}^u - t_{31}^l = \sum_{i=0}^{31} [(p_i - q_i)(1 - c_i) + c_i.(s_i - r_i)] \prod_{j=i+1}^{31} (1 - 2c_j) \tag{16}$$

Adding equation (3) and (4), we get,

$$\begin{gathered} t_{31}^u = t_u = \frac{1}{2} \cdot \sum_{i=0}^{31} (1 - c_i)(p_i + q_i) + c_i.(r_i + s_i) + \\ \frac{1}{2} \cdot \sum_{i=0}^{31} [(1 - c_i).(p_i - q_i) + c_i.(s_i - r_i)] \prod_{j=i+1}^{31} (1 - 2c_j) \end{gathered} \tag{17}$$

If we take,

$$\begin{aligned} d_i &= (1 - 2.c_i) \\ \alpha_i &= \frac{(p_i + q_i + r_i + s_i)}{4} \\ \beta_i &= \frac{(p_i + q_i - r_i - s_i)}{4} \\ \gamma_i &= \frac{(p_i - q_i + s_i - r_i)}{4} \\ \delta_i &= \frac{(p_i - q_i + r_i - s_i)}{4} \end{aligned}$$

And simplifying equation (5), we get -

$$t_u = \sum_{i=0}^{31} \alpha_i + \gamma_{31} + \sum_{i=0}^{31} d_i.\beta_i + \sum_{i=0}^{31} (\delta_i + \gamma_{i-1})(d_i.d_{i+1}.d_{i+2}.....d_{31}) \text{ (Here, } \gamma_{-1} = 0) \tag{18}$$

Take $b = \sum_{i=0}^{31} \alpha_i + \gamma_{31}$

Now, converting this to matrix form, we get -

$$t_u(c) = W^T \phi(c) + b \tag{19}$$

where,

$$W = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{30} \\ \delta_0 \\ (\delta_1 + \gamma_0) \\ \vdots \\ (\delta_{31} + \gamma_{30} + \beta_{31}) \end{bmatrix} \tag{20}$$

and,

$$\phi(c) = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{30} \\ d_0 d_1 d_2 \cdots d_{31} \\ d_1 d_2 d_3 \cdots d_{31} \\ \vdots \\ d_{31} \end{bmatrix} \tag{21}$$

2 Dimensionality of the model W

The feature vector $\phi(c)$ and the weight vector W should match in dimensionality to ensure correct computation in the linear model of the PUF.

2.1 Feature Vector $\phi(c)$

The feature vector $\phi(c)$ is constructed as follows:

$$\phi(c) = [d_0, d_1, \dots, d_{30}, d_0 d_1 d_2 \dots d_{31}, d_1 d_2 d_3 \dots d_{31}, \dots, d_{31}]^T$$

Where:

- $d_i = 1 - 2c_i$
- Each d_i represents a transformed challenge bit.

The dimensionality of $\phi(c)$ is derived by considering all individual d_i terms and all unique product terms involving d_i's. Specifically:

- There are 32 individual d_i terms.
- There are $\binom{32}{2} = \frac{32 \times 31}{2} = 496$ unique product terms.

Therefore, the total number of features in $\phi(c)$ is:

$$32 + 496 = 528$$

2.2 Weight Vector W

To match the feature vector $\phi(c)$, the weight vector W must also have 528 elements.

Thus, the dimensionality of W is:

$$528 \times 1$$

This ensures that the linear model correctly computes the response based on the challenge vector.

3 Linear models to predict Response0 and Response1

3.1 Another type of PUF — COCO-PUF

In the COCO-PUF setup with two arbiter PUFs (PUF0 and PUF1), each PUF has its own multiplexers with different delays. When a challenge is given, it goes to both PUFs. Instead of comparing the lower and upper signals of PUF0 internally, the lower signals from PUF0 and PUF1 compete using Arbiter0 to produce Response0. Similarly, the upper signals compete using Arbiter1 to produce Response1.

3.2 Using ML to crack COCO-PUF

From the previous part, the expression for t^u_{31} is:

$$\begin{aligned} t^u_{31} = \frac{1}{2}\cdot\sum_{i=0}^{31}(1-c_i)(p_i+q_i)+c_i.(r_i+s_i)+ \\ \frac{1}{2}\cdot\sum_{i=0}^{31}[(1-c_i).(p_i-q_i)+c_i.(s_i-r_i)]\prod_{j=i+1}^{31}(1-2c_j) \end{aligned} \tag{22}$$

Similar expression for t^l_{31} can be derived by subtracting equation (4) from (3),

$$\begin{aligned} t^l_{31} = \frac{1}{2}\cdot\sum_{i=0}^{31}(1-c_i)(p_i+q_i)+c_i.(r_i+s_i)- \\ \frac{1}{2}\cdot\sum_{i=0}^{31}[(1-c_i).(p_i-q_i)+c_i.(s_i-r_i)]\prod_{j=i+1}^{31}(1-2c_j) \end{aligned} \tag{23}$$

$$2t^{l1}_{31} = \sum_{i=0}^{31}(1-c_i)(p^1_i+q^1_i)+c_i(s^1_i+r^1_i)-\sum_{i=0}^{31}\left[(1-c_i)(p^1_i-q^1_i)+c_i(s^1_i-r^1_i)\right]\left[\prod_{j=i+1}^{31}(1-2c_j)\right] \tag{24}$$

$$2t^{l0}_{31} = \sum_{i=0}^{31}(1-c_i)(p^0_i+q^0_i)+c_i(s^0_i+r^0_i)-\sum_{i=0}^{31}\left[(1-c_i)(p^0_i-q^0_i)+c_i(s^0_i-r^0_i)\right]\left[\prod_{j=i+1}^{31}(1-2c_j)\right] \tag{25}$$

Subtracting equation (8) from (7),

$$\begin{aligned} 2(t^{l1}_{31}-t^{l0}_{31}) = \sum_{i=0}^{31}(1-c_i)\left[p^1_i+q^1_i-p^0_i-q^0_i\right]+c_i\left[s^1_i+r^1_i-s^0_i-r^0_i\right]+ \\ \sum_{i=0}^{31}\left[(1-c_i)\left[p^0_i-q^0_i-p^1_i+q^1_i\right]+c_i\left[s^0_i-r^0_i-s^1_i+r^1_i\right]\right]\left[\prod_{j=i+1}^{31}(1-2c_j)\right] \end{aligned} \tag{26}$$

let, $(1-2c_i) = d_i$

$$2(t_{31}^{l1}-t_{31}^{l0})=\sum_{i=0}^{31}\left(\frac{1}{2}+\frac{d_i}{2}\right)\left[p_i^1+q_i^1-p_i^0-q_i^0\right]+\left(\frac{1-d_i}{2}\right)\left[s_i^1+r_i^1-s_i^0-r_i^0\right]+$$
$$\sum_{i=0}^{31}\left(\frac{1}{2}+\frac{d_i}{2}\right)(d_{i+1}...d_{31})\left[p_i^0-q_i^0-p_i^1+q_i^1\right]+\left(\frac{1-d_i}{2}\right)(d_{i+1}...d_{31})\left[s_i^0-r_i^0-s_i^1+r_i^1\right] \quad (27)$$

$$t_{31}^{l1}-t_{31}^{l0}=\sum_{i=0}^{31}\left(\frac{p_i^1+q_i^1+s_i^1+r_i^1-p_i^0-q_i^0-s_i^0-r_i^0}{4}\right)+d_i\left(\frac{p_i^1+q_i^1-s_i^1-r_i^1-p_i^0-q_i^0+s_i^0+r_i^0}{4}\right)+$$
$$(d_{i+1}\ldots d_{31})\left(\frac{p_i^0-q_i^0-p_i^1+q_i^1+s_i^0-r_i^0-s_i^1+r_i^1}{4}\right)+(d_i\ldots d_{31})\left(\frac{p_i^0-q_i^0-p_i^1+q_i^1-s_i^0+r_i^0+s_i^1-r_i^1}{4}\right) \quad (28)$$

Let,

$$\alpha_i=\frac{p_i^1+q_i^1+s_i^1+r_i^1-p_i^0-q_i^0-s_i^0-r_i^0}{4}$$
$$\beta_i=\frac{p_i^1+q_i^1-s_i^1-r_i^1-p_i^0-q_i^0+s_i^0+r_i^0}{4}$$
$$\gamma_i=\frac{p_i^0-q_i^0-p_i^1+q_i^1+s_i^0-r_i^0-s_i^1+r_i^1}{4}$$
$$\delta_i=\frac{p_i^0-q_i^0-p_i^1+q_i^1-s_i^0+r_i^0+s_i^1-r_i^1}{4}$$

$$t_{31}^{l1}-t_{31}^{l0}=\sum_{i=0}^{31}\alpha_i+\sum_{i=0}^{31}d_i\beta_i+\sum_{i=0}^{31}(d_{i+1}\ldots d_{31})\gamma_i+\sum_{i=0}^{31}(d_i\ldots d_{31})\delta_i \quad (29)$$

$$t_{31}^{l1}-t_{31}^{l0}=\left(\sum_{i=0}^{31}\alpha_i+\gamma_{31}\right)+\left(\sum_{i=0}^{30}d_i\beta_i\right)+\delta_1(d_1....d_{31})+(\delta_2+\gamma_1)(d_2...d_{31})+.....+(\delta_{31}+\gamma_{30}+\beta_{31})d_{31} \quad (30)$$

Let, $\tilde{b}=\sum_{i=0}^{31}\alpha_i+\gamma_{31}$

$$t_{31}^{l1}-t_{31}^{l0}=\tilde{b}+\begin{bmatrix}\beta_0 & \beta_1 & \ldots & \beta_{30} & \delta_1 & (\delta_2+\gamma_1) & \ldots & (\delta_{31}+\gamma_{30}+\beta_{31})\end{bmatrix}\begin{bmatrix}d_0\\ d_1\\ \vdots\\ d_{30}\\ d_0d_1....d_{31}\\ d_1d_2....d_{31}\\ \vdots\\ d_{30}d_{31}\\ d_{31}\end{bmatrix} \quad (31)$$

$$t_{31}^{l1} - t_{31}^{l0} = \tilde{b} + \tilde{W}_0^T \tilde{\phi}_0(C), \text{where} \tag{32}$$

$$\tilde{W}_0 = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{30} \\ \delta_1 \\ \delta_2 + \gamma_1 \\ \vdots \\ \delta_{31} + \gamma_{30} + \beta_{31} \end{bmatrix} \tag{33}$$

and

$$\tilde{\phi}_0(C) = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{30} \\ d_0 d_1 \dots d_{31} \\ d_1 d_2 \dots d_{31} \\ \vdots \\ d_{31} \end{bmatrix} \tag{34}$$

Hence the Response0 can predicted using this linear model as -

$$\frac{1 + \text{sign}(\tilde{W}_0^\top \tilde{\phi}_0(c) + \tilde{b})}{2} = r_0(c) \tag{35}$$

Similar to the Response0, we can predict Response1 by using the expression of t_{31}^{u1} and t_{31}^{u0} in equation (7) and (8) respectively, we get-

$$t_{31}^{u1} - t_{31}^{u0} = \sum_{i=0}^{31} \alpha_i + \sum_{i=0}^{31} d_i \beta_i + \sum_{i=0}^{31} (d_{i+1} \dots d_{31}) \gamma_i + \sum_{i=0}^{31} (d_i \dots d_{31}) \delta_i \tag{36}$$

Here,

$$\begin{aligned} \alpha_i &= \frac{p_i^1 + q_i^1 + s_i^1 + r_i^1 - p_i^0 - q_i^0 - s_i^0 - r_i^0}{4} \\ \beta_i &= \frac{p_i^1 + q_i^1 - s_i^1 - r_i^1 - p_i^0 - q_i^0 + s_i^0 + r_i^0}{4} \\ \gamma_i &= \frac{p_i^1 - q_i^1 - p_i^0 + q_i^0 + s_i^1 - r_i^1 - s_i^0 + r_i^0}{4} \\ \delta_i &= \frac{p_i^1 - q_i^1 - p_i^0 + q_i^0 - s_i^1 + r_i^1 + s_i^0 - r_i^0}{4} \end{aligned}$$

Note: Here the value for the γ_i and δ_i in Response0 and Response1 are slighty different.

So finally the Response1 can similarly be predicted as -

$$\frac{1+\text{sign}(\tilde{W}_1^\top \tilde{\phi}_1(c)+\tilde{b})}{2} = r_1(c) \tag{37}$$

Here,

$$\tilde{W}_1 = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{30} \\ \delta_1 \\ \delta_2+\gamma_1 \\ \vdots \\ \delta_{31}+\gamma_{30}+\beta_{31} \end{bmatrix} \tag{38}$$

and

$$\tilde{\phi}_1(C) = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{30} \\ d_0 d_1 \dots d_{31} \\ d_1 d_2 \dots d_{31} \\ \vdots \\ d_{31} \end{bmatrix} \tag{39}$$

4 Dimensionality of the Linear Models $\tilde{W}_0$ and $\tilde{W}_1$

From the previous part, the weight vectors $\tilde{W}_0$ and $\tilde{W}_1$ are constructed as follows:

$$\tilde{W}_0 = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{30} \\ \delta_1 \\ \delta_2+\gamma_1 \\ \vdots \\ \delta_{31}+\gamma_{30}+\beta_{31} \end{bmatrix}$$

and

$$\tilde{W}_1 = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{30} \\ \delta_1 \\ \delta_2 + \gamma_1 \\ \vdots \\ \delta_{31} + \gamma_{30} + \beta_{31} \end{bmatrix}$$

Note: Here the γ_i and δ_i for these two models are slightly different.
Clearly, both $\tilde{W}_0$ and $\tilde{W}_1$ should match the dimensionality derived from the feature vector $\phi(c)$.
Hence, the dimensionality for both $\tilde{W}_0$ and $\tilde{W}_1$ is 528×1.

5 Code for the two linear models W_0, b_0, W_1, b_1

Code zip file

6 Outcomes of the Experiments

6.1 Training Set

The model undergoes training using the public dataset trn.txt and is then tested using tst.txt. Throughout this process, we vary the hyperparameters, recording the resulting accuracy and training time for each configuration.

6.1.1 Effect of changing the loss hyperparameter in LinearSVC

The loss function in sklearn.svm.LinearSVC determines how effectively the model handles the dataset. This model offers two types of loss functions: hinge and squared hinge.

In our observations, we used specific hyperparameters like C = 1, tolerance = 1e-3, penalty = L2, and dual = True for LinearSVC to analyze their impact on performance.

Loss	Training Time (in seconds)	Accuracy y0	Accuracy y1
Hinge	1.78	0.9245	0.9981
Squared Hinge	5.47	0.9214	0.9948

Table 1: Loss Hyperparameters

6.1.2 Effect of changing the C hyperparameter in LinearSVC and Logistic Regression model

Regularization is a method used to prevent models from becoming too complex and fitting too closely to training data, which can lead to overfitting. The parameter C plays a crucial role in this process: it represents the inverse of regularization strength. Smaller values of C result in stronger regularization, penalizing large coefficients in the model.

In both Logistic Regression and Linear SVC, adjusting C influences how closely the model adapts to the training data. Increasing C tightens the fit to the training data, potentially causing overfitting, while decreasing C promotes simpler decision boundaries, aiding generalization but risking underfitting.

Our observations were based on experiments using specific settings: for LinearSVC, we used hyperparameters like loss = Squared Hinge, tolerance = 1e-3, penalty = L2, and dual = False. Similarly, for Logistic Regression, we used tolerance = 1e-3, penalty = L2, and dual = False.

C value	Training Time (in seconds)	Accuracy y0	Accuracy y1
0.00001	1.2023	0.957	0.9637
0.0001	1.3886	0.9667	0.9742
0.001	1.8473	0.9721	0.9812
0.01	2.1969	0.9739	0.9870
0.1	2.9476	0.9740	0.9913

Table 2: C Hyperparameters in LinearSVC

C value	Training Time (in seconds)	Accuracy y0	Accuracy y1
0.01	2.2964	0.9739	0.9870
0.1	2.4157	0.9740	0.9913
1	2.4452	0.9739	0.9922
10	2.3271	0.9738	0.9921
100	2.3628	0.9738	0.9921

Table 3: C Hyperparameters in Logistic Regression

6.1.3 Effect of changing the tol hyperparameter in LinearSVC and Logistic Regression model

The tol hyperparameter, short for tolerance, determines how precisely optimization algorithms in LinearSVC and LogisticRegression converge during training. It sets a threshold for when the iterative process stops, based on changes in the objective function or coefficients between iterations. A smaller tol means the model requires tighter convergence criteria, which can extend training time but potentially enhance accuracy. On the other hand, a larger tol speeds up training but may sacrifice precision. Choosing the right tol value strikes a balance between computational efficiency and model performance.

In our observations, we used specific configurations:

For LinearSVC: loss = Squared Hinge, C = 10, penalty = L2, dual = False.

For Logistic Regression: C = 100, penalty = L2, solver = 'lbfgs', dual = False.

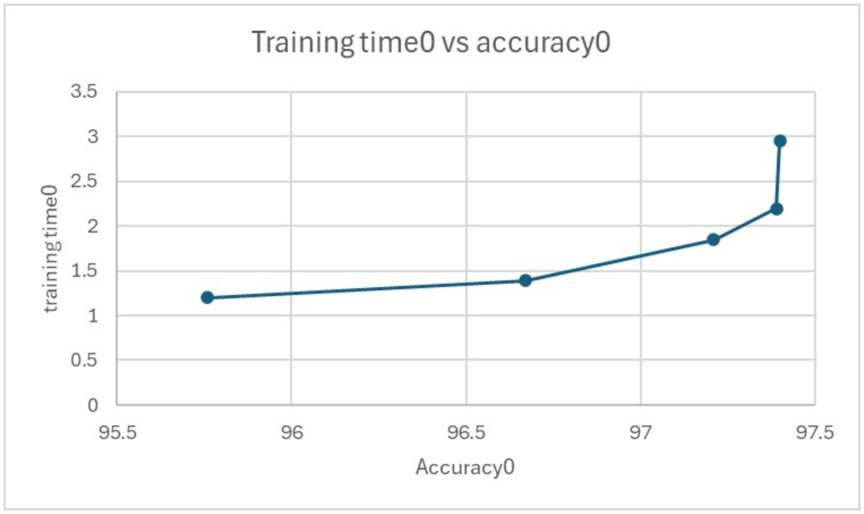

Figure 1: Training Time vs C for LinearSVC

tol value	Training Time (in seconds)	Accuracy y0	Accuracy y1
10^{-1}	3.2062	0.9738	0.9925
10^{-2}	2.4108	0.9738	0.9925
10^{-3}	2.5727	0.9738	0.9924
10^{-4}	1.5324	0.9751	0.9923
10^{-5}	2.6791	0.9694	0.9793

Table 4: tol Hyperparameters in LinearSVC

6.1.4 Effect of changing the penalty hyperparameter in LinearSVC and Logistic Regression model

In Logistic Regression, the penalty hyperparameter determines the type of regularization applied: '$l1$' penalizes absolute coefficient values, promoting sparsity, while '$l2$' penalizes squared coefficients, leading to smoother decision boundaries. Adjusting this hyperparameter helps control overfitting, encouraging simpler models that generalize better to new data.

Similarly, in LinearSVC, the penalty hyperparameter also controls regularization: 'l1' induces sparsity by penalizing absolute coefficients, resulting in sparse solutions, whereas 'l2' penalizes squared coefficients, promoting smaller values and smoother boundaries. By adjusting this hyperparameter, we manage model complexity, reducing overfitting and improving generalization to unseen data.

In our observations, for LinearSVC with hyperparameters loss = Squared Hinge, C = 1, tolerance = 1e-3, and dual = False, we used the liblinear solver with the L1 penalty. And for Logistic Regression with C = 100, tolerance = 1e-3, and dual = False, we utilized the lbfgs solver, which is the default solver for 'l2' penalty because the 'l1' penalty cannot be used with lbfgs solver due to compatibility constraints.

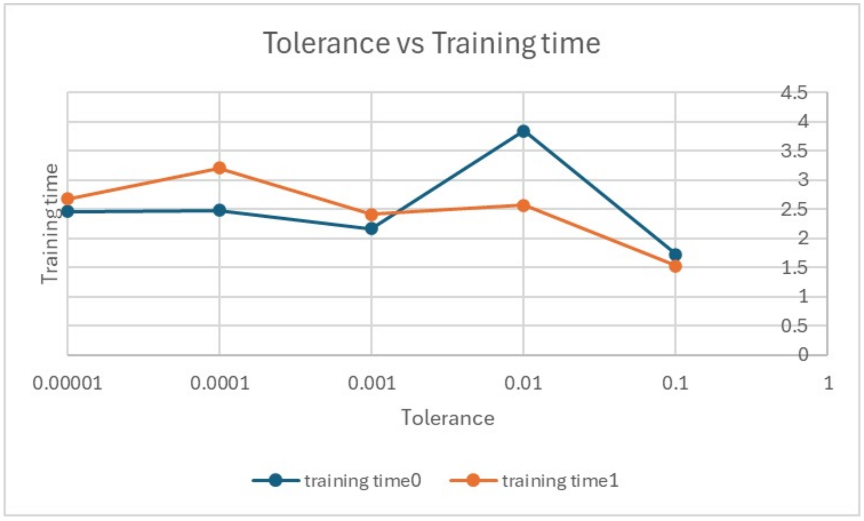

Figure 2: Accuracy vs C for LinearSVC

tol value	Training Time (in seconds)	Accuracy y0	Accuracy y1
10^{-1}	1.39	0.9808	0.9990
10^{-2}	1.20	0.9808	0.9990
10^{-3}	1.18	0.9808	0.9990
10^{-4}	0.99	0.9808	0.9990
10^{-5}	1.01	0.9808	0.9990

Table 5: tol Hyperparameters in Logistic Regression

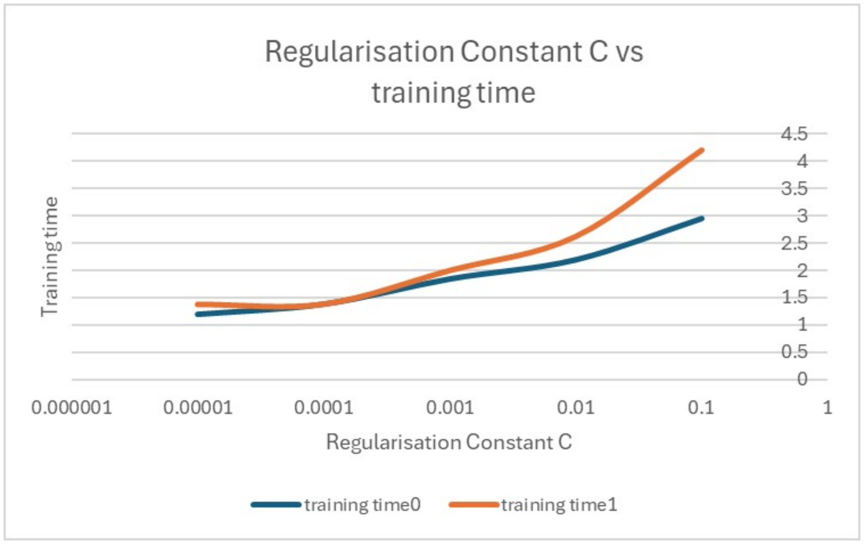

Figure 3: Training Time vs C for Logistic Regression

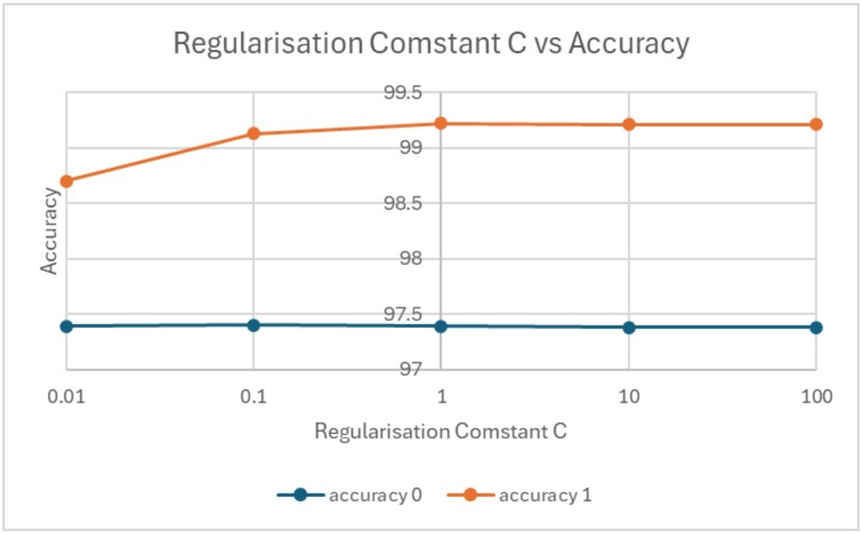

Figure 4: Enter Caption

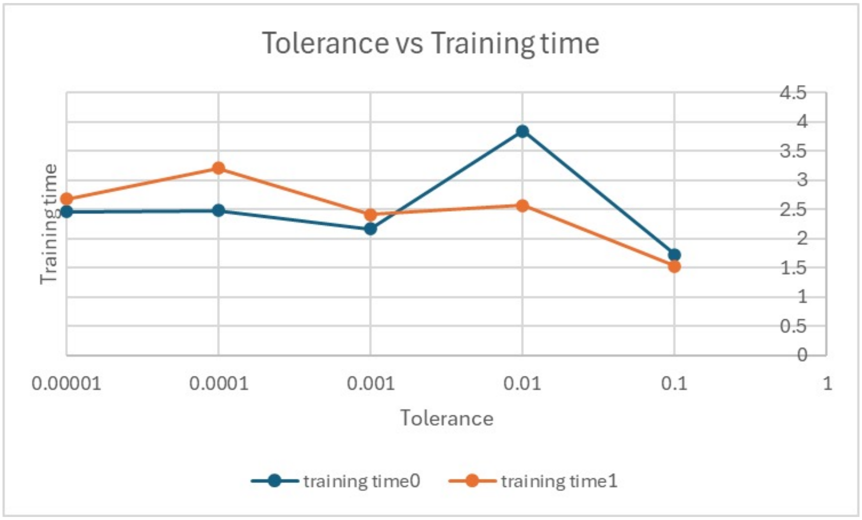

Figure 5: Training Time vs tol for LinearSVC

Penalty	Training Time (in seconds)	Accuracy y0	Accuracy y1
L1	17.68	0.9241	0.9876
L2	1.53	0.942	0.9809

Table 6: Penalty Hyperparameters in LinearSVC

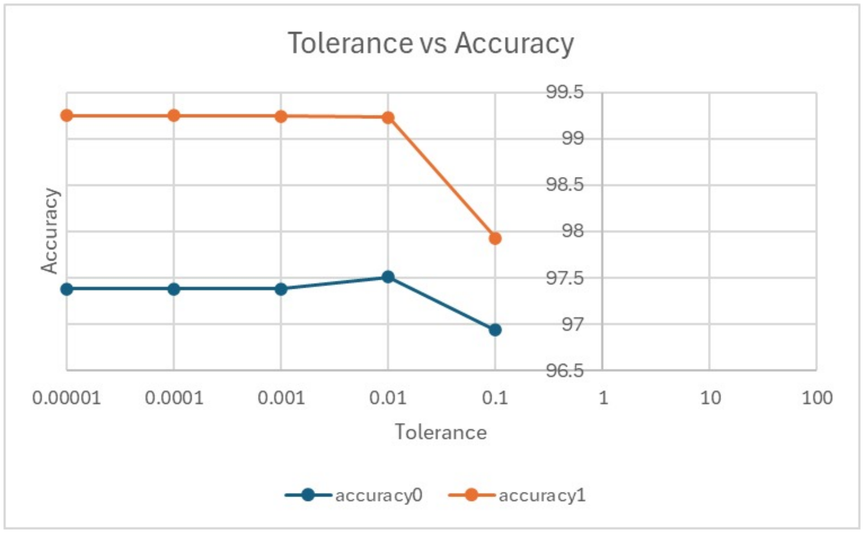

Figure 6: Accuracy vs tol for LinearSVC

Solver	Penalty	Training Time (in seconds)	Accuracy y0	Accuracy y1
liblinear	L1	18.12	0.9808	0.9982
lbfgs	L2	1.25	0.9808	0.9990

Table 7: Penalty Hyperparameters in Logistic Regression