CS771: Introduction to Machine Learning
Assignment 1

Team: Neurons

Rupesh Kumar Meena : 210881, EE
rupeshkm21@iitk.ac.in

Keerthi S : 210504, CHM
keerthis21@iitk.ac.in

Aditya Neelabha : 190064, Mech
adityan19@iitk.ac.in

Deepak Kumar : 220332, Math
deepakkr22@iitk.ac.in

Abstract

This document presents the submission from team **Neurons** for Assignment 1. We have addressed parts 1 and 3, providing detailed explanations, proofs, and tables as necessary.

1 Linear Model for Predicting the Time of Upper Signal in Arbiter PUF

1.1 Overview of Arbiter PUFs

A PUF consists of a series of k multiplexers, where each multiplexer either switches the connections or maintains the original paths based on the input challenge bit. Each multiplexer has unique delays that are challenging to reproduce but are consistent. Let t^u and t^l represent the time taken by the upper and lower signals to reach the output.

At the output, an arbiter (typically a flip-flop) determines which signal arrived first, thereby generating the final response.

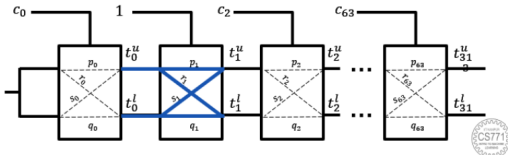

Figure 1: A simple arbiter PUF with 64 multiplexers

Define: $t_{u,i}$ = time when the upper signal exits the i-th PUF.
$t_{l,i}$ = time when the lower signal exits the i-th PUF.

Since each PUF has distinct characteristics, p_i, r_i, s_i, and q_i are different values. The decision rule states that if $t_{u,31} < t_{l,31}$, the output is 0; otherwise, it is 1. We use zero-based indexing, implying $t_{u,-1} = t_{l,-1} = 0$.

We know:

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

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

Adding equations (1) and (2) gives:

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

For $n = 0$:

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

For $n = 1$:

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

For $n = 2$:

$$t_{u,2}+t_{l,2} = (1-c_0)(p_0+q_0)+c_0(r_0+s_0)+(1-c_1)(p_1+q_1)+c_1(r_1+s_1)+(1-c_2)(p_2+q_2)+c_2(r_2+s_2) \tag{6}$$

By induction, we find:

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

From lecture notes, we have:

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

where,

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

For $i = 0$:

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

Let:

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

Then,

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

For $i = 1$:

$$\Delta_1 = d_1 \cdot \Delta_0 + \alpha_1 \cdot d_1 + \beta_1 = d_1 \cdot A_0 + A_1 \tag{12}$$

For $i = 2$:

$$\Delta_2 = d_2 \cdot d_1 \cdot A_0 + d_2 \cdot A_1 + A_2 \tag{13}$$

Thus, we have:

$$\Delta_{31} = \sum_{i=0}^{31} A_i \cdot \prod_{j=i+1}^{31} d_j \tag{14}$$

Therefore,

$$t^u_{31} - t^l_{31} = \sum_{i=0}^{31} [(p_i - q_i)(1 - c_i) + c_i(s_i - r_i)] \prod_{j=i+1}^{31} (1 - c_j) \tag{15}$$

Adding equations (3) and (4) yields:

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

Let:

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

After simplification, we obtain:

$$t^u = \sum_{i=0}^{31}\alpha_i + \gamma_{31} + \sum_{i=0}^{31} d_i \cdot \beta_i + \sum_{i=0}^{31}(\delta_i+\gamma_{i-1})\prod_{j=i}^{31} d_j \quad (\text{where } \gamma_{-1}=0) \tag{17}$$

Let $b = \sum_{i=0}^{31}\alpha_i + \gamma_{31}$.
In matrix form, this can be expressed as:

$$t^u(c) = W^T\phi(c) + b \tag{18}$$

where,

$$\phi(c) = \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_{30} \\ d_0d_1d_2\cdots d_{31} \\ d_1d_2d_3\cdots d_{31} \\ \vdots \\ d_{31} \end{pmatrix}$$

and,

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

2 2 Dimensionality of the Model W

From the previous part, we have:

$$W = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{30} \\ \delta_0 \\ \delta_1 + \gamma_0 \\ \vdots \\ \delta_{31} + \gamma_{30} + \beta_{31} \end{pmatrix}$$

Clearly, it is a 63×1 matrix. Hence, the dimension of the model W is 63×1.

3 Linear Models to Predict Response_0 and Response_1

3.1 Another Type of PUF — COCO-PUF

A COCO-PUF uses 2 arbiter PUFs, say PUF0 and PUF1 – each PUF has its own set of multiplexers with possibly different delays. Given a challenge, it is fed into both the PUFs. However, the way responses are generated is different. Instead of the lower and upper signals of PUF0 competing with each other, Melbo makes the lower signal from PUF0 compete with the lower signal from PUF1 using an arbiter called Arbiter0 to generate a response called Response0. If the signal from PUF0 reaches first, Response0 is 0 else if the signal from PUF1 reaches first, Response0 is 1. Melbo also makes the upper signal from PUF0 compete with the upper signal from PUF1 using a second arbiter called Arbiter1 to generate a response called Response1. If the signal from PUF0 reaches first, Response1 is 0 else if the signal from PUF1 reaches first, Response1 is 1. Thus, on each challenge, the COCO-PUF generates two responses instead of one response

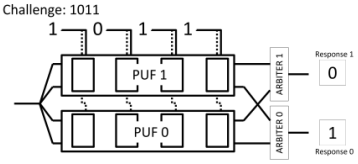

Figure 2: A COCO-PUF with 4-bit challenges and 2-bit responses

Figure 2: A COCO-PUF with 4-bit challenges and 2-bit responses

3.2 Calculations for COCO-PUF

From the part 1, the expression for t_{31}^{u} is:

$$t_u^{31} = \frac{1}{2}\sum_{i=0}^{31}(1-c_i)(p_i+q_i)+c_i(r_i+s_i)+\frac{1}{2}\sum_{i=0}^{31}\left[(1-c_i)(p_i-q_i)+c_i(s_i-r_i)\right]\prod_{j=i+1}^{31}(1-2c_j)$$

$$t_l^{31} = \frac{1}{2}\sum_{i=0}^{31}(1-c_i)(p_i+q_i)+c_i(r_i+s_i)-\frac{1}{2}\sum_{i=0}^{31}\left[(1-c_i)(p_i-q_i)+c_i(s_i-r_i)\right]\prod_{j=i+1}^{31}(1-2c_j)$$

$$2t_{l1}^{31} = \sum_{i=0}^{31}(1-c_i)(p_{1i}+q_{1i})+c_i(s_{1i}+r_{1i})-\sum_{i=0}^{31}(1-c_i)(p_{1i}-q_{1i})+c_i(s_{1i}-r_{1i})\prod_{j=i+1}^{31}(1-2c_j) \tag{7}$$

$$2t_{l0}^{31} = \sum_{i=0}^{31}(1-c_i)(p_{0i}+q_{0i})+c_i(s_{0i}+r_{0i})-\sum_{i=0}^{31}(1-c_i)(p_{0i}-q_{0i})+c_i(s_{0i}-r_{0i})\prod_{j=i+1}^{31}(1-2c_j) \tag{8}$$

Subtracting equation (8) from (7):

$$2(t_{l0}^{31} - t_{l1}^{31}) = \sum_{i=0}^{31}(1-c_i)\left(p_{0i}+q_{0i}-p_{1i}-q_{1i}\right)+\sum_{i=0}^{31}c_i\left(s_{0i}+r_{0i}-s_{1i}-r_{1i}\right)$$

$$+\sum_{i=0}^{31}(1-c_i)\left(p_{1i}-q_{1i}-p_{0i}+q_{0i}\right)+\sum_{i=0}^{31}c_i\left(s_{1i}-r_{1i}-s_{0i}+r_{0i}\right)\prod_{j=i+1}^{31}(1-2c_j)$$

Letting $(1-2c_i) = d_i$:

$$2(t_{l0}^{31} - t_{l1}^{31}) = \sum_{i=0}^{31}\left(\frac{1}{2}+\frac{d_i}{2}\right)\left(p_{0i}+q_{0i}-p_{1i}-q_{1i}\right)$$

$$+\sum_{i=0}^{31}\left(1-\frac{d_i}{2}\right)\left(s_{0i}+r_{0i}-s_{1i}-r_{1i}\right)$$

$$+\sum_{i=0}^{31}\left(\frac{1}{2}+\frac{d_i}{2}\right)\left(d_{i+1}\dots d_{31}\right)\left(p_{1i}-q_{1i}-p_{0i}+q_{0i}\right)$$

$$+\sum_{i=0}^{31}\left(1-\frac{d_i}{2}\right)\left(d_{i+1}\dots d_{31}\right)\left(s_{1i}-r_{1i}-s_{0i}+r_{0i}\right)$$

Finally, we can express:

$$t_{l0}^{31} - t_{l1}^{31} = \sum_{i=0}^{31}\frac{1}{4}\left(p_{0i}+q_{0i}+s_{0i}+r_{0i}-p_{1i}-q_{1i}-s_{1i}-r_{1i}\right)$$

$$+d_i\frac{1}{4}\left(p_{0i}+q_{0i}-s_{0i}-r_{0i}-p_{1i}-q_{1i}+s_{1i}+r_{1i}\right)$$

$$+(d_{i+1}\dots d_{31})\frac{1}{4}\left(p_{1i}-q_{1i}-p_{0i}+q_{0i}+s_{1i}-r_{1i}-s_{0i}+r_{0i}\right)$$

$$+(d_{i}\dots d_{31})\frac{1}{4}\left(p_{1i}-q_{1i}-p_{0i}+q_{0i}-s_{1i}+r_{1i}+s_{0i}-r_{0i}\right)$$

Let,

$$\alpha_i = \frac{p_{0i}+q_{0i}+s_{0i}+r_{0i}-p_{1i}-q_{1i}-s_{1i}-r_{1i}}{4}$$
$$\beta_i = \frac{p_{0i}+q_{0i}-s_{0i}-r_{0i}-p_{1i}-q_{1i}+s_{1i}+r_{1i}}{4}$$
$$\gamma_i = \frac{p_{1i}-q_{1i}-p_{0i}+q_{0i}+s_{1i}-r_{1i}-s_{0i}+r_{0i}}{4}$$
$$\delta_i = \frac{p_{1i}-q_{1i}-p_{0i}+q_{0i}-s_{1i}+r_{1i}+s_{0i}-r_{0i}}{4}$$

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

Let,

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

Then,

$$t_{l0}^{31}-t_{l1}^{31}=e_b+[\beta_0,\beta_1,\dots,\beta_{30},\delta_1,(\delta_2+\gamma_1),\dots,(\delta_{31}+\gamma_{30}+\beta_{31})]$$

$$\begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_{30} \\ d_0d_1\cdots d_{31} \\ d_1d_2\cdots d_{31} \\ \vdots \\ d_{30}d_{31} \\ d_{31} \end{pmatrix}$$

Thus,

$$t_{l0}^{31}-t_{l1}^{31}=e_b+W_{f0}^{T}\phi_{g0}(C)$$

where

$$W_{f0} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{30} \\ \delta_1 \\ \delta_2 + \gamma_1 \\ \vdots \\ \delta_{31} + \gamma_{30} + \beta_{31} \end{pmatrix}$$

and

$$\phi_{g0}(C) = \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_{30} \\ d_0 d_1 \cdots d_{31} \\ d_1 d_2 \cdots d_{31} \\ \vdots \\ d_{31} \end{pmatrix}$$

Hence, the Response $r_0(c)$ can be predicted using this linear model as:

$$r_0(c) = \frac{1 + \text{sign}(W_{g0}^{\top} \phi_{f0}(c) + b_e)}{2}$$

Similarly, for Response $r_1(c)$, we use the expressions of t_{u0}^{31} and t_{u1}^{31} from equations (7) and (8):

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

Here,

$$\alpha_i = \frac{p_{0i} + q_{0i} + s_{0i} + r_{0i} - p_{1i} - q_{1i} - s_{1i} - r_{1i}}{4}$$

$$\beta_i = \frac{p_{0i} + q_{0i} - s_{0i} - r_{0i} - p_{1i} - q_{1i} + s_{1i} + r_{1i}}{4}$$

$$\gamma_i = \frac{p_{0i} - q_{0i} - p_{1i} + q_{1i} + s_{0i} - r_{0i} - s_{1i} + r_{1i}}{4}$$

$$\delta_i = \frac{p_{0i} - q_{0i} - p_{1i} + q_{1i} - s_{0i} + r_{0i} + s_{1i} - r_{1i}}{4}$$

Note: Here the values for γ_i and δ_i in Response 0 and Response 1 are slightly different.

Thus, the Response $r_1(c)$ can similarly be predicted as:

$$r_1(c) = \frac{1 + \text{sign}(W_{g1}^{\top}\phi_{f1}(c) + b_e)}{2}$$

Here,

$$W_{f1} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{30} \\ \delta_1 \\ \delta_2 + \gamma_1 \\ \vdots \\ \delta_{31} + \gamma_{30} + \beta_{31} \end{pmatrix}$$

and

$$\phi_{g1}(C) = \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_{30} \\ d_0 d_1 \cdots d_{31} \\ d_1 d_2 \cdots d_{31} \\ \vdots \\ d_{31} \end{pmatrix}$$

4 Dimensionality of the Linear Models W_0 and W_1

From the previous part, we have:

$$W_{f0} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{30} \\ \delta_1 \\ \delta_2 + \gamma_1 \\ \vdots \\ \delta_{31} + \gamma_{30} + \beta_{31} \end{pmatrix}$$

and

$$W_{f1} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{30} \\ \delta_1 \\ \delta_2 + \gamma_1 \\ \vdots \\ \delta_{31} + \gamma_{30} + \beta_{31} \end{pmatrix}$$

Note: Here the values for γ_i and δ_i for these two models are slightly different.

Clearly, both W_{f0} and W_{f1} are 63×1 matrices. Hence, the dimensionality for both W_{f0} and W_{f1} is 63×1.

6 Outcomes of the Experiments

6.1 Training Set

The model is trained on the public `trn.txt` and tested on the public `tst.txt` with different sets of hyperparameters. Their accuracy and training time are recorded.

a) Effect of changing the hyperparameters in LogisticRegression

Regularization is a technique used to prevent overfitting by penalizing large coefficients in the model. The C parameter represents the inverse of regularization strength, where smaller values of C correspond to stronger regularization.

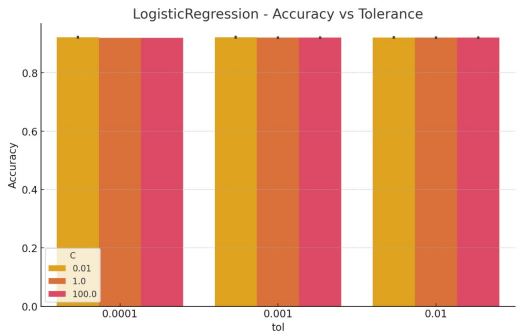

Figure 3: LogisticRegression - Accuracy vs Tolerance

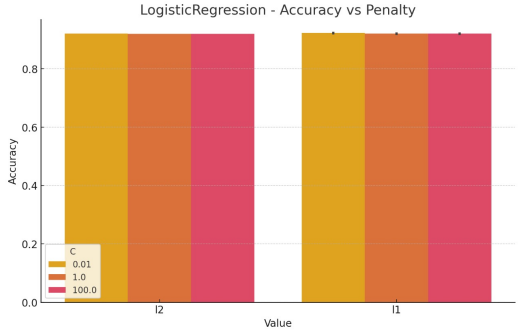

Figure 4: LogisticRegression - Accuracy vs Penalty

Figure 5: LogisticRegression - Training Time vs Tolerance

Figure 6: LogisticRegression - Training Time vs Penalty

b) Effect of changing the hyperparameters in LinearSVC

The loss function characterizes how well the model performs on a given dataset. `sklearn.svm.LinearSVC` provides two loss functions, namely: hinge and squared hinge.

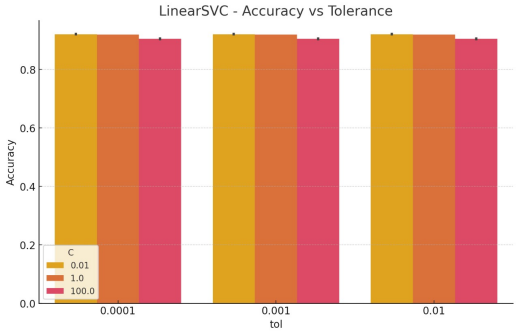

Figure 7: LinearSVC - Accuracy vs Tolerance

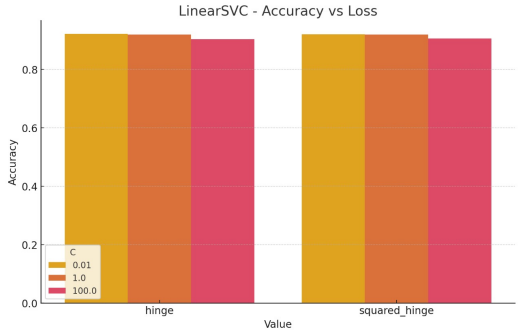

Figure 8: LinearSVC - Accuracy vs Loss

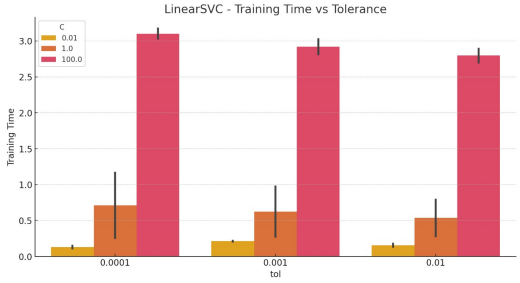

Figure 9: LinearSVC - Training Time vs Tolerance

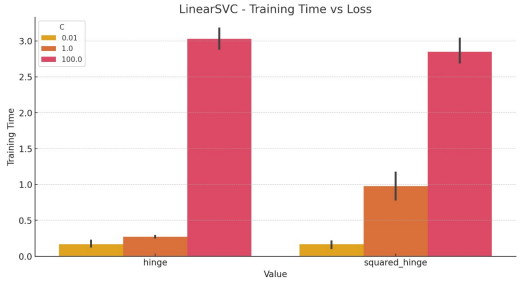

Figure 10: LinearSVC - Training Time vs Loss

Model	Hyperparameter	Value	C	tol	Training Time	Accuracy
LinearSVC	loss	hinge	0.01	0.0001	0.146911	0.921625
LinearSVC	loss	hinge	0.01	0.001	0.216954	0.921625
LinearSVC	loss	hinge	0.01	0.01	0.135156	0.921625
LinearSVC	loss	hinge	1	0.0001	0.255343	0.920875
LinearSVC	loss	hinge	1	0.001	0.274222	0.92
LinearSVC	loss	hinge	1	0.01	0.283396	0.92
LinearSVC	loss	hinge	100	0.0001	3.171903	0.904375
LinearSVC	loss	hinge	100	0.001	3.024502	0.904375
LinearSVC	loss	hinge	100	0.01	2.892286	0.905875
LinearSVC	loss	squared_h	0.01	0.0001	0.113703	0.920875
LinearSVC	loss	squared_h	0.01	0.001	0.207061	0.920875
LinearSVC	loss	squared_h	0.01	0.01	0.176339	0.920875
LinearSVC	loss	squared_h	1	0.0001	1.165411	0.9205
LinearSVC	loss	squared_h	1	0.001	0.975823	0.9205
LinearSVC	loss	squared_h	1	0.01	0.791162	0.920125
LinearSVC	loss	squared_h	100	0.0001	3.031998	0.905875
LinearSVC	loss	squared_h	100	0.001	2.815472	0.905875
LinearSVC	loss	squared_h	100	0.01	2.700102	0.905875
LogisticRegression	penalty	l2	0.01	0.0001	0.130856	0.920875
LogisticRegression	penalty	l1	0.01	0.0001	0.477687	0.923
LogisticRegression	penalty	l2	0.01	0.001	0.1222	0.920875
LogisticRegression	penalty	l1	0.01	0.001	0.381838	0.923
LogisticRegression	penalty	l2	0.01	0.01	0.244288	0.920875
LogisticRegression	penalty	l1	0.01	0.01	0.251836	0.922
LogisticRegression	penalty	l2	1	0.0001	0.251123	0.9205
LogisticRegression	penalty	l1	1	0.0001	1.156214	0.9205
LogisticRegression	penalty	l2	1	0.001	0.145757	0.9205
LogisticRegression	penalty	l1	1	0.001	0.760736	0.920625
LogisticRegression	penalty	l2	1	0.01	0.225008	0.9205
LogisticRegression	penalty	l1	1	0.01	0.448613	0.92075
LogisticRegression	penalty	l2	100	0.0001	0.375652	0.9205
LogisticRegression	penalty	l1	100	0.0001	0.993722	0.9205
LogisticRegression	penalty	l2	100	0.001	0.391621	0.9205
LogisticRegression	penalty	l1	100	0.001	0.886039	0.920625
LogisticRegression	penalty	l2	100	0.01	0.49778	0.9205
LogisticRegression	penalty	l1	100	0.01	0.442883	0.92075

Table 1: Effect of Changing the Hyperparameters in LinearSVC and LogisticRegression