Assignment 1 : Introduction to Machine Learning [CS771]

Aditi Khandelia **Kushagra Srivastava** **Mahaarajan J** **Ruthvik Tunuguntala**
Department of Computer Science
Indian Institute of Technology, Kanpur

Abstract

This report contains the parts of solutions 1 and 3 of Assignment 1 using the format as given in the problem statement. Part 1 contains the mathematical derivation that a linear model can be developed for the given proble, whereas part 3 is our report of the experimentation that took place to find the optimal model and how different hyperparameters affect the accuracy and training time of the model.

1 Companion Arbiter PUF

1.1 Task 1

Let Δw and Δr be the time delays between the signals for the 1^{st} PUF and the 2^{nd} PUF respectively. Then we know that

$$\Delta w = w_w^T X + b_w$$
$$\Delta r = w_r^T X + b_r$$

(since the output of the arbiter PUF is a linear model).

Where X is a 32-dimensional vector with $x_i = d_i \cdot d_{i+1} \cdot ... \cdot d_{31}$ and $d_i = 1 - 2c_i$ (c_i being the $(i+1)^{th}$ bit of the 32-bit challenge).

Now from the problem statement we have:

$$[\Delta w - \Delta \tau] > \tau$$

for the response to be 1. (Where $\tau > 0$ is the secret threshold value of the companion Arbiter PUF).

Hence we have:

$$[(w_w - w_r)^T X - (b_w - b_r)]^2 - \tau^2 > 0$$

Which is simplified as:

$$[(W^T X + B)^2 - \tau^2] > 0$$

Where we have:

$$W = w_w - w_r$$
$$B = b_w - b_r$$

Now,

$$\begin{aligned}(W^T X + B)^2 &= (W^T X + B)(W^T X + B) \\ &= (W^T X)^2 + 2(W^T X)B + B^2\end{aligned}$$

Preprint. Under review.

Where

$$W = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{31} \end{bmatrix} \quad \text{and} \quad X = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{31} \end{bmatrix}$$

Both are 32-dimensional vectors as w_w and w_r were 32-dimensional, and the challenge has 32 bits. Hence the value of $W^T X = \sum_{i=0}^{31} \alpha_i x_i$.

So,

$$\left(\sum_{i=0}^{31} \alpha_i x_i \right)^2 = \sum_{i=0}^{31} \alpha_i^2 x_i^2 + 2 \sum_{i=0}^{31} \sum_{\substack{j=0 \\ j<i}}^{31} \alpha_i \alpha_j x_i x_j$$

Hence the equation becomes:

$$\sum_{i=0}^{31} \alpha_i^2 + 2 \sum_{i=0}^{31} \sum_{\substack{j=0 \\ j<i}}^{31} \alpha_i \alpha_j x_i x_j + 2B \sum_{i=0}^{31} \alpha_i x_i + (B^2 - \tau^2) > 0$$

Since $x_i \epsilon \{-1, 1\}$ the value of x_i^2 will be 1 for all values of i

Hence the condition for response can be written as:

$$[\mathbf{W^T X} + \mathbf{B}] > 0$$

Where:

$$\mathbf{B} = B^2 - \tau^2 + \sum_{i=0}^{31} \alpha_i^2 \quad , \quad \mathbf{X} = \begin{bmatrix} x_0 x_1 \\ x_0 x_2 \\ \vdots \\ x_{30} x_{31} \\ x_0 \\ x_1 \\ \vdots \\ x_{31} \end{bmatrix} \quad \text{and} \quad \mathbf{W} = \begin{bmatrix} \alpha_0 \alpha_1 \\ \alpha_0 \alpha_2 \\ \vdots \\ \alpha_{30} \alpha_{31} \\ 2B\alpha_0 \\ 2B\alpha_1 \\ \vdots \\ 2B\alpha_{31} \end{bmatrix}$$

The dimension of $\mathbf{X}$ will be:

$$32 + \binom{32}{2} = 528$$

The map $\omega(X)$ will map X to $\mathbf{X}$ i.e it will be a map from R^{32} to R^{528}

From the definition of X we can represent X as $F(c)$ where F is a map from R^{32} to R^{32} (as $x_i = d_i \cdot d_{i+1} \cdot ... \cdot d_{31}$ where $d_i = 1 - 2c_i$)

Therefore we have a map $\boldsymbol{\phi}(c)$ which can be defined to be $\omega \circ F(c)$ only depending on the challenges and not any constants

Hence, the linear model to break this CAR-PUF can be given using the linear model $\mathbf{W}^\top \boldsymbol{X} + \mathbf{B}$, where given a challenge c, we can map it to $\boldsymbol{\phi}(c)$, and the response to this challenge can be given by:

$$\frac{1 + \text{sign}(\mathbf{W}^\top \boldsymbol{\phi}(c) + B)}{2}$$

Hence Proved

1.2 Task 2

```
import numpy as np
import sklearn
from scipy.linalg import khatri_rao
from sklearn.linear_model import LogisticRegression
import time as tm

# You are allowed to import any submodules of sklearn
# that learn linear models e.g. sklearn.svm etc
# You are not allowed to use other libraries such as keras, tensorflow etc
# You are not allowed to use any scipy routine other than khatri_rao

# SUBMIT YOUR CODE AS A SINGLE PYTHON (.PY) FILE INSIDE A ZIP ARCHIVE
# THE NAME OF THE PYTHON FILE MUST BE submit.py

# DO NOT CHANGE THE NAME OF THE METHODS my_fit, my_map etc BELOW
# THESE WILL BE INVOKED BY THE EVALUATION SCRIPT.
# CHANGING THESE NAMES WILL CAUSE EVALUATION FAILURE

# You may define any new functions, variables, classes here
# For example, functions to calculate next coordinate or step length

################################
# Non Editable Region Starting #
################################
def my_fit( X_train, y_train ):
# # ################################
# # #  Non Editable Region Ending  #
# # ################################

        X_train_final=my_map(X_train)

        #BEST RESULTS OBTAINED USING LOGISTIC REGRESSION
        C=125
        tol=0.001
        penalty='l2'
        solver='lbfgs'

        clf=LogisticRegression(C=C, tol=tol, penalty=penalty, solver=solver)
        clf.fit(X_train_final, y_train)
        w=clf.coef_
        b=clf.intercept_
        return w.T[:,0],b



################################
# Non Editable Region Starting #
################################
def my_map( X ):
################################
#  Non Editable Region Ending  #
################################
        no_challenges=X.shape[0]
        Xk=1-2*X
        Xkk=1-2*X
        Xk[:,0]=np.prod(Xk,axis=1)
        for i in range(1,32):
```

```
            Xk[:,i]=Xk[:,i-1]/Xkk[:,i-1]

    feat=[]
    Xk = Xk.reshape(-1,32,1)
    for i in range(no_challenges):
            result = np.triu(np.dot(Xk[i], Xk[i].T), k=1).reshape(-1)
            result = np.append(result[result != 0], Xk[i])
            feat.append(result)
    feat=np.array(feat)
    return feat
```

1.3 Task 3

General Remarks: We experimented with three models: LinearSVC, Logistic Regression, and Ridge Regression. The hyperparameter values which gave us the best model for LinearSVC is $\{C : 1, loss : squared\_hinge, penalty : l2, tol : 1\}$ with an accuracy of 0.992, for Logistic Regression is $\{C : 125, solver : liblinear, penalty : l2, tol : 0.0025\}$ with accuracy 0.9935 and for Ridge Regression is $\{alpha : 1.5, max\_iter : 1000, solver : saga, tol : 0.1\}$ with accuracy of 0.835625.
Clearly, the best accuracy was achieved with Logistic Regression, which we implemented to train our final model.

1.3.1 Changing the loss hyperparameter in `LinearSVC` (`hinge` vs `squared hinge`)

The best results for LinearSVC were obtained when we used hyperparameter values as follows: $\{C : 1, loss : squared\_hinge, penalty : l2, tol : 1\}$. Finding accuracy values for loss being calculated using squared hinge loss and hinge loss while varying C and tol near the values of the best model using LinearSVC gives us the observations below in Table 1 and 2.

Table 1: Accuracy and Training Time Comparison

C	Tolerance	Penalty	Squared Hinge Accuracy	Squared Hinge Training Time
0.5	0.5	l2	0.993	3.542
0.5	0.75	l2	0.99225	3.122
0.5	1	l2	0.9915	3.050
1	0.5	l2	0.992375	3.371
1	0.75	l2	0.9925	3.514
1	1	l2	0.992875	3.555
2	0.5	l2	0.990125	3.394
2	0.75	l2	0.991875	3.367
2	1	l2	0.991875	3.427
3	0.5	l2	0.991625	3.591
3	0.75	l2	0.992	3.344
3	1	l2	0.991375	3.198

Table 2: Accuracy and Training Time Comparison

C	Tolerance	Penalty	Hinge Accuracy	Hinge Training Time
0.5	0.5	l2	0.987875	2.959971
0.5	0.75	l2	0.988125	3.054396
0.5	1	l2	0.988	2.797376
1	0.5	l2	0.990125	3.241517
1	0.75	l2	0.990875	3.241567
1	1	l2	0.990625	3.299687
2	0.5	l2	0.991375	3.360085
2	0.75	l2	0.990875	3.294506
2	1	l2	0.99025	3.521247
3	0.5	l2	0.99025	3.633900
3	0.75	l2	0.988875	3.226441
3	1	l2	0.9915	3.205128

The best accuracy was found using Squared Hingle loss in the case of LinearSVC.Changing the loss parameter from hinged to square hinged increased the training time when the model used lower values of C such as 0.5 and 1. When we use relatively higher values of C to train the model like 2 and 3 it falls in some cases and increases in the other by small values.

1.3.2 Setting C in `LinearSVC` and `LogisticRegression` to high/low/medium values

Varying the values of C to high, low, and medium for two different values of tolerance generated the outcomes as given below in Table 3 and Table 4.
From the outcomes, it can be inferred that for logistic regression, accuracy increases with increasing the value of C, slightly falling at the end. The time taken to train the model increased with increasing C for lower tolerance values and is inconsistent with relatively higher values of C. For LinearSVC, accuracy first increases with increasing the value of C reached maximum, then decreases. The training time increases with the increasing values of C slightly falling at the end.

Table 3: Accuracy and Training Time Comparison

Tolerance	C	LinearSVC Accuracy	LinearSVC Training Time
0.0025	0.001	0.956	0.6047
0.0025	0.01	0.9836	1.1979
0.0025	0.1	0.9901	3.3570
0.0025	1	0.9929	3.3827
0.0025	10	0.9905	3.2677
0.0025	100	0.9898	3.2314
0.0025	1000	0.9901	3.1917
0.75	0.001	0.957	0.3422
0.75	0.01	0.9824	0.5699
0.75	0.1	0.9891	1.5754
0.75	1	0.9923	3.2283
0.75	10	0.9920	3.2381
0.75	100	0.9900	3.2246
0.75	1000	0.9895	3.1936

Table 4: Accuracy and Training Time Comparison

Tolerance	C	LogisticRegression Accuracy	LogisticRegression Training Time
0.0025	0.001	0.886	0.6118
0.0025	0.01	0.9549	0.7860
0.0025	0.1	0.984	1.1980
0.0025	1	0.9913	1.5097
0.0025	10	0.993	2.3460
0.0025	100	0.993375	2.4140
0.0025	1000	0.993375	3.1397
0.75	0.001	0.8335	0.2924
0.75	0.01	0.833125	0.2859
0.75	0.1	0.88475	0.3627
0.75	1	0.88475	0.3551
0.75	10	0.884625	0.3614
0.75	100	0.884625	0.3648
0.75	1000	0.884625	0.3664

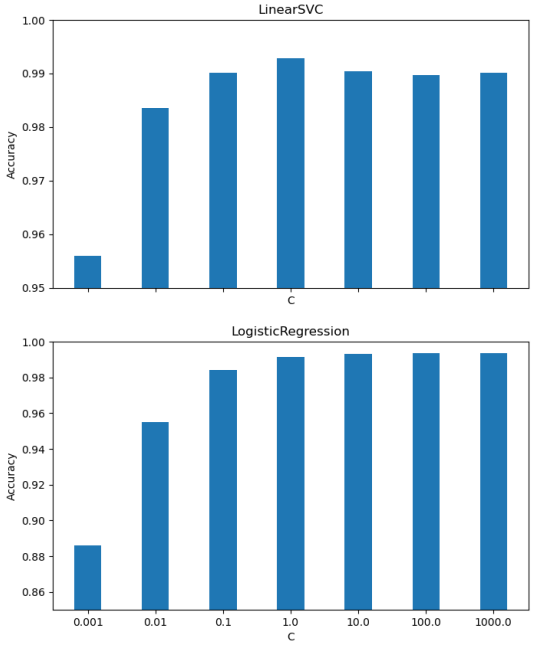

Figure 1: Accuracy Plot by Changing Value of C at tol = 0.0025

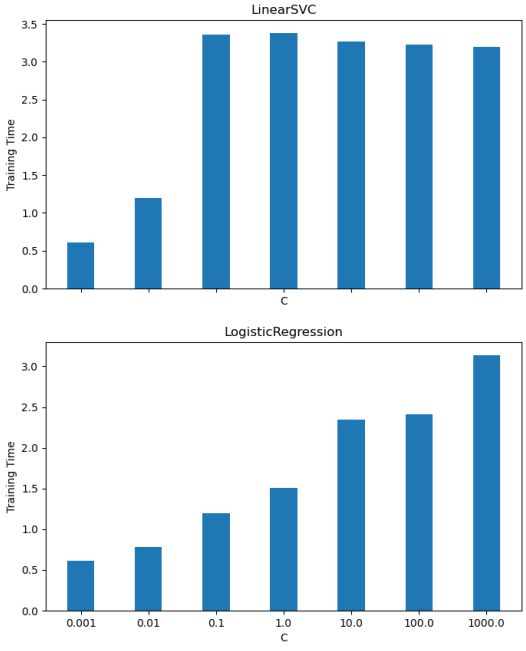

Figure 2: Training time Plot by Changing Value of C at tol = 0.0025

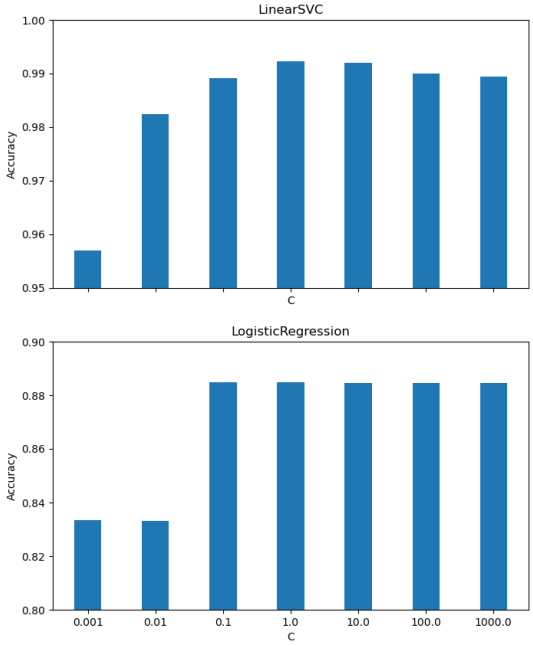

Figure 3: Accuracy Plot by Changing Value of C at tol = 0.75

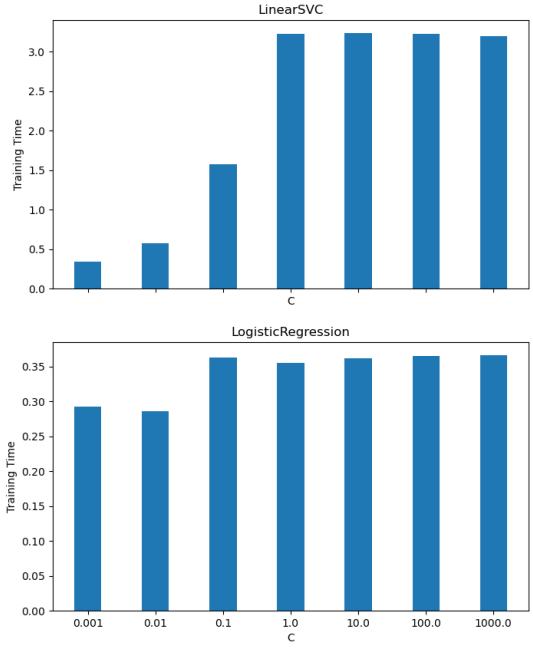

Figure 4: Training time Plot by Changing Value of C at tol = 0.75

1.3.3 Changing tol in `LinearSVC` and `LogisticRegression` to high/low/medium values

Varying the values of C to high, low, and medium values generated outcomes as given below in Table 5 and Table 6.
From the outcomes of the experimentation, it can be inferred that for LinearSVC, there is very little variation, and no regular trend can be found for both the accuracy and the time taken to train the model. For Logistic Regression, variation is high, and both the accuracy and the time taken to train the model decrease with the increasing tolerance value.

Table 5: Comparison of LinearSVC and Logistic Regression (Excluding LinearSVC)

C	Tolerance	Logistic Regression Accuracy	Logistic Regression Training Time
1	0.001	0.99125	1.760859013
1	0.0025	0.99125	1.444149971
1	0.005	0.99125	1.471183062
1	0.01	0.990625	1.215860844
1	0.05	0.984125	0.77090621
1	0.1	0.972875	0.584295988
1	0.5	0.88475	0.366055965
1	0.75	0.88475	0.375716209
1	1	0.833125	0.281864882
125	0.001	0.993125	3.853279829
125	0.0025	0.9935	3.063081026
125	0.005	0.992875	1.912871838
125	0.01	0.992125	1.566137791
125	0.05	0.9845	0.74539423
125	0.1	0.972625	0.565456152
125	0.5	0.884625	0.365886927
125	0.75	0.884625	0.378480196
125	1	0.833125	0.270027161

Table 6: Comparison of LinearSVC and Logistic Regression (Excluding Logistic Regression)

C	Tolerance	LinearSVC Accuracy	LinearSVC Training Time
1	0.001	0.99225	3.215604067
1	0.0025	0.992625	3.148298025
1	0.005	0.992875	3.224939108
1	0.01	0.99225	3.198827982
1	0.05	0.992	3.235346317
1	0.1	0.992375	3.221129179
1	0.5	0.99175	3.218056202
1	0.75	0.992375	3.240312099
1	1	0.992375	3.237003803
125	0.001	0.991125	3.185766935
125	0.0025	0.991	3.173813105
125	0.005	0.990875	3.194851875
125	0.01	0.991625	3.211688995
125	0.05	0.990375	3.210982084
125	0.1	0.9905	3.191169977
125	0.5	0.99075	3.213047981
125	0.75	0.990875	3.163609982
125	1	0.9905	3.192769051

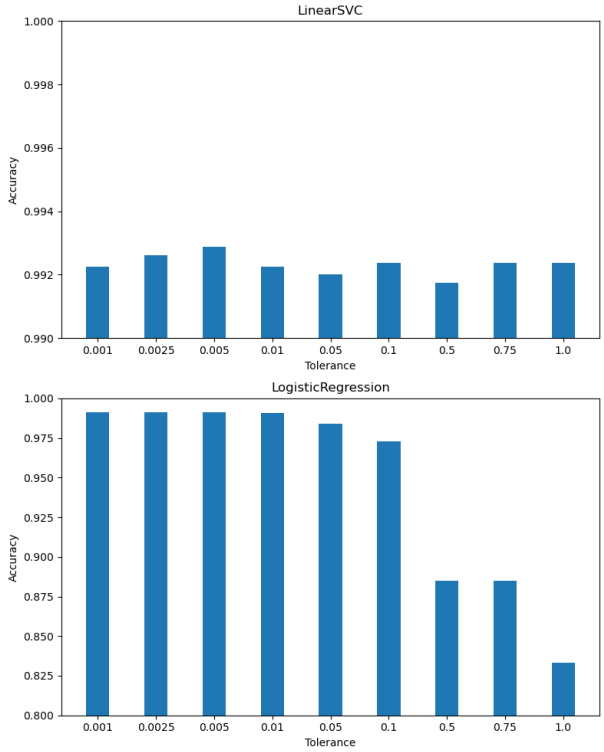

Figure 5: Accuracy Plot by Changing Value of tol at C = 1

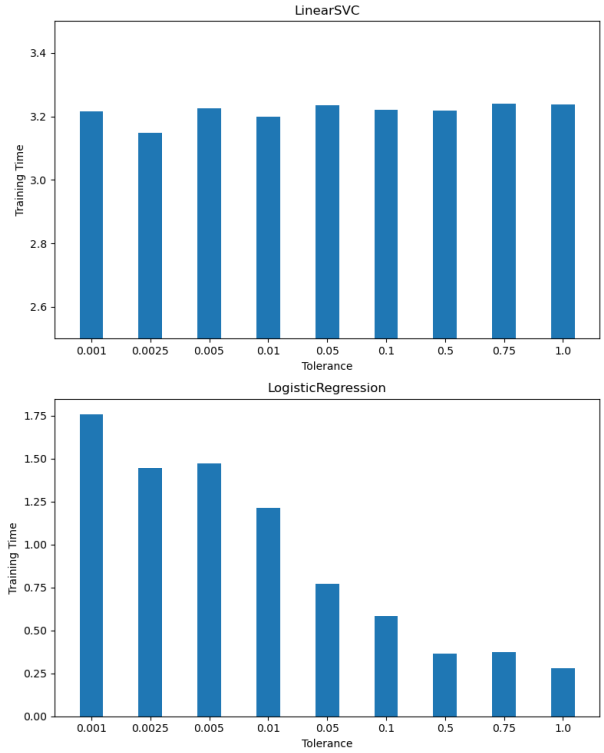

Figure 6: Training time Plot by Changing Value of tol at C = 1

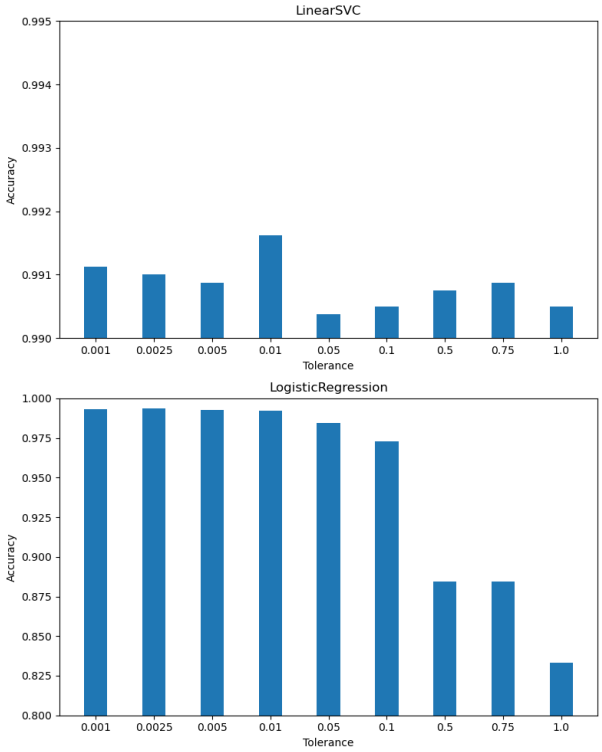

Figure 7: Accuracy Plot by Changing Value of tol at C = 125

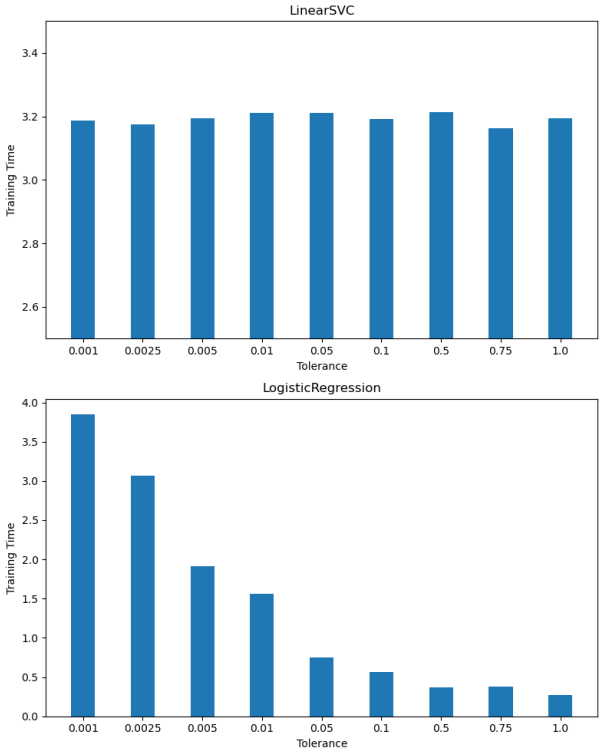

Figure 8: Training time Plot by Changing Value of tol at C = 125

1.3.4 Changing the penalty (regularization) hyperparameter in `LinearSVC` and `Logistic regression` (l2 vs l1)

The outcomes of changing the penalty parameter are given below in Table 7 and Table 8. In the case of Logistic Regression, changing the penalty from l2 to l1 increases the accuracy in most cases whereas the training time always increases. For LinearSVC on changing penalty from l2 to l1, Accuracy decreases while there is no regular trend for the training time.

Table 7: Comparison of LinearSVC and Logistic Regression (Excluding LinearSVC)

C	Tolerance	Penalty	Logistic Regression Accuracy	Logistic Regression Training Time
1	0.0025	l2	0.99125	1.447992086
1	0.75	l2	0.88475	0.362996817
125	0.0025	l2	0.9935	3.033905983
125	0.75	l2	0.884625	0.370075941
1	0.0025	l1	0.992375	15.21927691
1	0.75	l1	0.88625	0.391086102
125	0.0025	l1	0.993125	27.12482405
125	0.75	l1	0.885875	0.374642849

Table 8: Comparison of LinearSVC and Logistic Regression (Excluding Logistic Regression)

C	Tolerance	Penalty	LinearSVC Accuracy	LinearSVC Training Time
1	0.0025	l2	0.99225	3.203393936
1	0.75	l2	0.992875	3.164654016
125	0.0025	l2	0.989875	3.141861916
125	0.75	l2	0.990375	3.195630789
1	0.0025	l1	0.99225	30.35474205
1	0.75	l1	0.878	0.381443024
125	0.0025	l1	0.993125	34.10513306
125	0.75	l1	0.888	0.397687912