CS771: Assignment - 1

Group Name: Guesstimators

Group Members

Lakshika(210554)
Prajjwal Kumar(210736)
Siddharth Pathak (211034)
Soni Verma(211051)
Srishti Gupta(211059)
Vanshika Gupta(211146)

Abstract

This document contains the solutions of Assignment-1 of CS771 (Introduction of Machine Learning) by the group member mentioned above. This document submitted to the course instructor Prof. Purushottam Kar in the spring semester of 2023-24.

1 Question 1: [Companion Arbiter PUF]

Melbo is constantly innovating to come up with PUFs that cannot be broken by ML attacks. Recall that an arbiter PUF is a chain of k multiplexers, each of which either swaps the lines or keeps them intact, depending on what is the challenge bit fed into that multiplexer. The multiplexers each have delays which are hard to replicate but consistent. Let t^u, t^l respectively denote the time for the upper and lower signals to reach the finish line and let $\Delta := t^u - t^l$ denote the difference in the timings. Note that Δ can be negative or positive. Assume that $t^u = t^l$ never happens i.e., Δ is never exactly zero. If the upper signal reaches the finish line first i.e. $\Delta < 0$, the response is 0 else if the lower signal reaches first i.e. $\Delta > 0$, the response is 1.

Melbo came up with the idea of creating a PUF using multiple arbiter PUFs and decided to call it a Companion Arbiter PUF (CAR-PUF for short). A CAR-PUF uses 2 arbiter PUFs – a working PUF and a reference PUF, as well as a secret threshold value $\tau > 0$. Given a challenge, it is fed into both the working PUF and reference PUF and the timings for the upper and lower signals for both PUFs are measured. Let Δ_w, Δ_r be the difference in timings experienced for the two PUFs on the same challenge. The response to this challenge is 0 if $|\Delta_w - \Delta_r| \leq \tau$ and the response is 1 if $|\Delta_w, \Delta_r| > \tau$ where $\tau > 0$ is the secret threshold value. Melbo thinks that with multiple PUFs whose delays are being compared against a secret threshold value, it is very difficult for a linear machine learning model to can predict the responses if given a few thousand challenge-response pairs. Your job is to prove Melbo wrong! You will do this by showing that there does exist a linear model that can perfectly predict the responses of a CAR-PUF and this linear model can be estimated fairly accurately if given enough challenge-response pairs (CRPs).

Solution:

The fact that any arbiter PUFF can be broken by an ML attack was well-established in the lectures. We will make use of that fact and show that there exists a linear classifier that breaks the Companion Arbiter PUF. Let the linear classifier for working PUF and reference PUF for any challenge c (a boolean vector of size 32) are $U^t x + q$ and $V^t x + q$ respectively. Where x is a feature vector transformed from c. i^{th} coordinate of x is defined as $x_i = \Pi_{j=i}^{32}(1 - 2c_j)$. With the above pre-

Preprint. Under review.

information now, we will carry out our investigation of the existence of a linear classifier for the above problem.

$$
\begin{aligned}
|\Delta_W - \Delta_r| > \tau &\Rightarrow (\Delta_W - \Delta_r)^2 > \tau^2 \\
&\Rightarrow ((U^t x_i + p) - (V^t x_i + q))^2 > \tau^2 \\
&\Rightarrow ((U - V)^t x_i + (p - q))^2 > \tau^2 \\
&\Rightarrow ((U - V)^t x_i)^2 + (p - q)^2 + 2(U - V)^t (p - q) x_i > \tau^2 \\
&\Rightarrow ((U - V)^t x_i)^2 + (p - q)^2 + 2(U - V)^t (p - q) x_i > \tau^2 \\
&\Rightarrow [(U - V)^t x_i] * [(U - V)^t x_i] + 2(p - q)(U - V)^t x_i + (p - q)^2 - \tau^2 > 0 \\
&\Rightarrow (U - V)^t x_i x_i^t (U - V) + 2(p - q)(U - V)^t x_i + (p - q)^2 - \tau^2 > 0
\end{aligned}
$$

Let $U - V = Z_1,\ 2(p - q)(U - V) = Z_2$ and $(p - q)^2 - \tau^2 = a$. Now,

$$(U - V)^t x_i x_i^t (U - V) + 2(p - q)(U - V)^t x_i + (p - q)^2 - \tau^2 = Z_1^t (x_t x_i^t) Z_1 + Z_2^t x_i + a$$

Now, We have $x_i = (x_{i1}, x_{i2}, ..., x_{i32})$

$$x_i x_i^t = \begin{bmatrix} (x_{i1})^2 & x_{i1}x_{i2} & x_{i1}x_{i3} & \dots & x_{i1}x_{i32} \\ x_{i2}x_{i1} & (x_{i2})^2 & x_{i2}x_{i3} & \dots & x_{i2}x_{i32} \\ x_{i3}x_{i1} & x_{i3}x_{i2} & (x_{i3})^2 & \dots & x_{i3}x_{i32} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{i32}x_{i1} & x_{i32}x_{i2} & x_{i32}x_{i3} & \dots & (x_{i32})^2 \end{bmatrix}$$

$$x_i x_i^t = \begin{bmatrix} 1 & x_{i1}x_{i2} & x_{i1}x_{i3} & \dots & x_{i1}x_{i32} \\ x_{i2}x_{i1} & 1 & x_{i2}x_{i3} & \dots & x_{i2}x_{i32} \\ x_{i3}x_{i1} & x_{i3}x_{i2} & 1 & \dots & x_{i3}x_{i32} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{i32}x_{i1} & x_{i32}x_{i2} & x_{i32}x_{i3} & \dots & 1 \end{bmatrix}$$

Since this is a symmetric matrix with diagonal elements 1, the number of unique unknowns from this matrix will be $32(32 - 1)/2 = 496$. Now, let us simplify our linear classifier. Consider $Z_1 = (Z_1^{(1)}, Z_2^{(1)}, Z_3^{(1)}, ..., Z_{32}^{(1)})$ and $Z_2 = (Z_1^{(2)}, Z_2^{(2)}, Z_3^{(2)}, ..., Z_{32}^{(2)})$.

$$
\begin{aligned}
(\Delta_W - \Delta_r)^2 - \tau^2 &= Z_1^t (x_t x_i^t) Z_1 + Z_2^t x_i + a \\
&= \sum_{1 \le j < k \le 32} (2 Z_j^{(1)} Z_k^{(1)}) x_{ij} x_{ik} + Z_2^t x_i + a + \sum_{j=1}^{32} \left(Z_j^{(1)} \right)^2 \\
&= \sum_{1 \le j < k \le 32} (2 Z_j^{(1)} Z_k^{(1)}) \Pi_{n=j}^{32} (1 - 2c_{in}) \Pi_{n=k}^{32} (1 - 2c_{ik}) + Z_2^t x_i + a + \sum_{j=1}^{32} \left(Z_j^{(1)} \right)^2 \\
&= \sum_{1 \le j < k \le 32} (2 Z_j^{(1)} Z_k^{(1)}) \Pi_{n=j}^{k-1} (1 - 2c_{in}) + Z_2^t x_i + a + \sum_{j=1}^{32} \left(Z_j^{(1)} \right)^2 \\
&= \underbrace{\sum_{1 \le j < k \le 32} (2 Z_j^{(1)} Z_k^{(1)}) \Pi_{n=j}^{k-1} (1 - 2c_{in}) + \sum_{j=1}^{32} Z_j^{(2)} \Pi_{k=j}^{32} (1 - 2c_{jk})}_{\mathbf{W^t \phi(c_i)}} + \underbrace{a + \sum_{j=1}^{32} \left(Z_j^{(1)} \right)^2}_{b} \\
&= \sum_{j=1}^{31} \sum_{k=j+1}^{32} (2 Z_j^{(1)} Z_k^{(1)}) \Pi_{n=j}^{k-1} (1 - 2c_{in}) + \sum_{j=1}^{32} Z_j^{(2)} \Pi_{k=j}^{32} (1 - 2c_{jk}) + a + \sum_{j=1}^{32} \left(Z_j^{(1)} \right)^2 \\
&= W^t \phi(c_i) + b
\end{aligned}
$$

Where, $W^t = (W_1^t, W_2^t)$ (say)

$$W_1 = \Big[2Z_1^{(1)}Z_2^{(1)}, 2Z_1^{(1)}Z_3^{(1)}, 2Z_1^{(1)}Z_4^{(1)}, \ldots, 2Z_1^{(1)}Z_{32}^{(1)},$$
$$2Z_2^{(1)}Z_3^{(1)}, 2Z_2^{(1)}Z_4^{(1)}, \ldots, 2Z_2^{(1)}Z_{32}^{(1)},$$
$$\vdots$$
$$2Z_{30}^{(1)}Z_{31}^{(1)}, 2Z_{30}^{(1)}Z_{32}^{(1)},$$
$$2Z_{31}^{(1)}Z_{32}^{(1)}\Big]^t$$

and

$$W_2 = Z_2$$

$\phi(c) = \begin{bmatrix}\phi_1(c)\\ \phi_2(c)\end{bmatrix}$ is the feature vector for this problem for any boolean vector c of size 32. Where

$$\phi_1^t(c) = \Big[(1-2c_1), \Pi_{k=1}^{2}(1-2c_k), \Pi_{k=1}^{3}(1-2c_k), \ldots, \Pi_{k=1}^{31}(1-2c_k),$$
$$(1-2c_k), \Pi_{k=2}^{3}(1-2c_k), \ldots, \Pi_{k=2}^{31}(1-2c_k),$$
$$\vdots$$
$$(1-c_{30}), \Pi_{k=30}^{31}(1-2c_k),$$
$$(1-c_{31})\Big]$$

and

$$\phi_2^t(c) = \Big[\Pi_{k=1}^{32}(1-2c_k), \Pi_{k=2}^{32}(1-2c_k), \Pi_{k=3}^{32}(1-2c_k), \ldots, (1-2c_{32})\Big]$$

Also, we have an intercept term which is equal to $b = a + \sum_{j=1}^{32}\left(Z_j^{(1)}\right)^2$.

Note that there is no redundancy in the notation of feature vector $\phi(c)$. The size of the feature vector for any challenge will be $\binom{32}{2} + 32 = 528$. Hence, we get a linear classifier to solve for and predict the output of the Companion Arbiter PUF. Let c be a challenge i.e. boolean vector of size 32. if $W^t\phi(c) + b > 0$ then the response will be 1. if $W^t\phi(c) + b \leq 0$ then the response will be 0. In other words, the response of Companion Arbiter PUF will be $\frac{1+sign(W^t+b)}{2}$. Here we need to take care that we are assuming that sign of 0 is taken as -1.

2 Question 2:

Done in *.py* file, submitted via the code submission page. We are mentioning the code of submit.py here also.

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

# 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  #
################################
        # Use this method to train your model using training CRPs
        # X_train has 32 columns containing the challeenge bits
        # y_train contains the responses

        # THE RETURNED MODEL SHOULD BE A SINGLE VECTOR AND A BIAS TERM
        # If you do not wish to use a bias term, set it to 0
        feature = my_map(X_train)
        model = linear_model.LogisticRegression(penalty='l2',tol=1e-4,
        max_iter = 100000,C= 70.7,random_state = 42)
        model.fit(feature,y_train)
        w = model.coef_[0,:]
        b = model.intercept_[0]
        return w, b


################################
# Non Editable Region Starting #
################################
def my_map( X ):
################################
#  Non Editable Region Ending  #
################################
        # Use this method to create features.
        # It is likely that my_fit will internally call my_map to create features
        # for train points
        X = 1-2*X
        n_rows, n_col = X.shape
        for i in range(1,n_col):
                X[:,n_col-i-1] = X[:,n_col-i-1] * X[:,n_col-i]

        out_prod = np.einsum('ij,ik->ijk', X, X)
        unique_variables = np.triu_indices(n_col,k=1)
        feat = out_prod[:, unique_variables[0], unique_variables[1]]
        feat = np.concatenate((X,feat),axis=1)

        return feat
```

3 Question 3:

Report outcomes of experiments with both the sklearn.svm.LinearSVC and sklearn.linear model.LogisticRegression methods when used to learn the linear model. In particular, report how various hyperparameters affected training time and test accuracy using tables and/or charts. Report these experiments with both LinearSVC and LogisticRegression methods even if your own submission uses just one of these methods or some totally different linear model learning method (e.g. RidgeClassifier) In particular, you must report how at least 2 of the following affect training time and test accuracy:
(a) changing the loss hyperparameter in LinearSVC (hinge vs squared hinge)
(b) setting C in LinearSVC and LogisticRegression to high/low/medium values

(c) changing tol in LinearSVC and LogisticRegression to high/low/medium values
(d) changing the penalty (regularization) hyperparameter in LinearSVC and LogisticRegression (l_2 vs l_1)

Solution:
(b) Linear SVC:

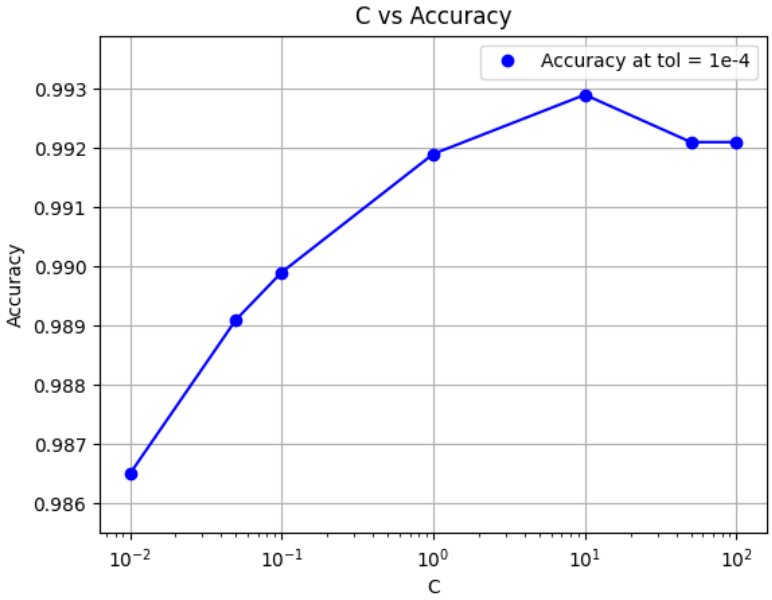

Figure 1: The graph shows that the accuracy generally increases as the control parameter (C) increases, reaching a peak accuracy around C=10 and then slightly decreasing.

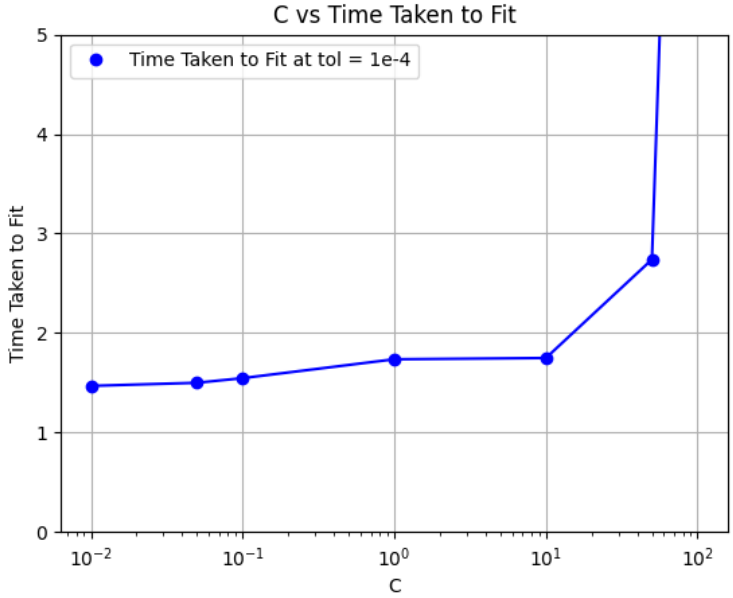

Figure 2: The graph suggests a roughly linear relationship between the control parameter (C) and the fitting time. As C of the model increases, the time required to fit the model also generally increases.

Logistic Regression:

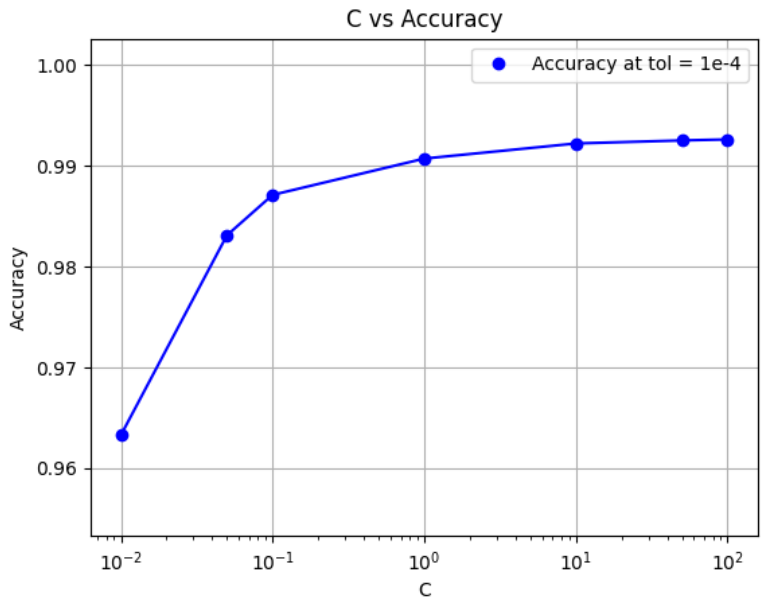

Figure 3: The graph suggests that accuracy generally increases as the control parameter (C) is increased. This means the controller performs better with a higher control parameter setting. The accuracy seems to approach a maximum value, indicating a possible saturation point where further increases in C might not significantly improve accuracy.

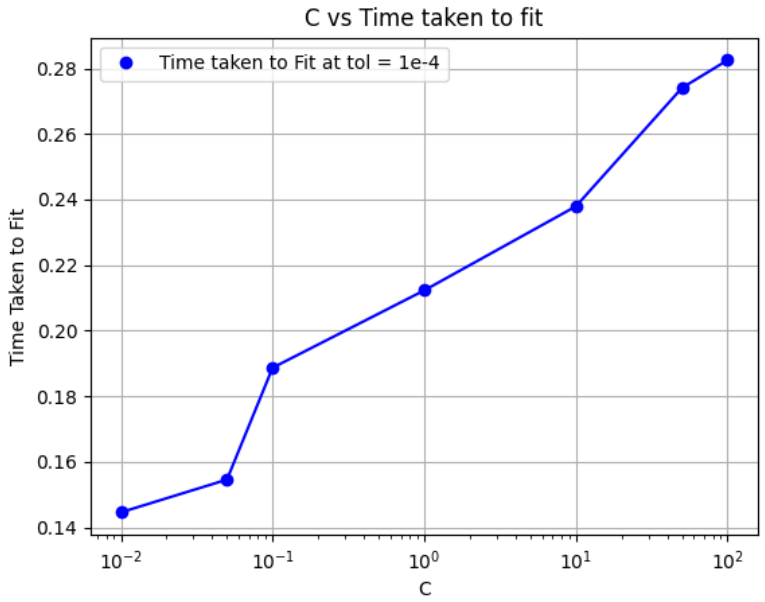

Figure 4: The graph suggests a positive correlation between C and fitting time.This means that the time required to fit the model increases proportionally with the C.

(c) Linear SVC:

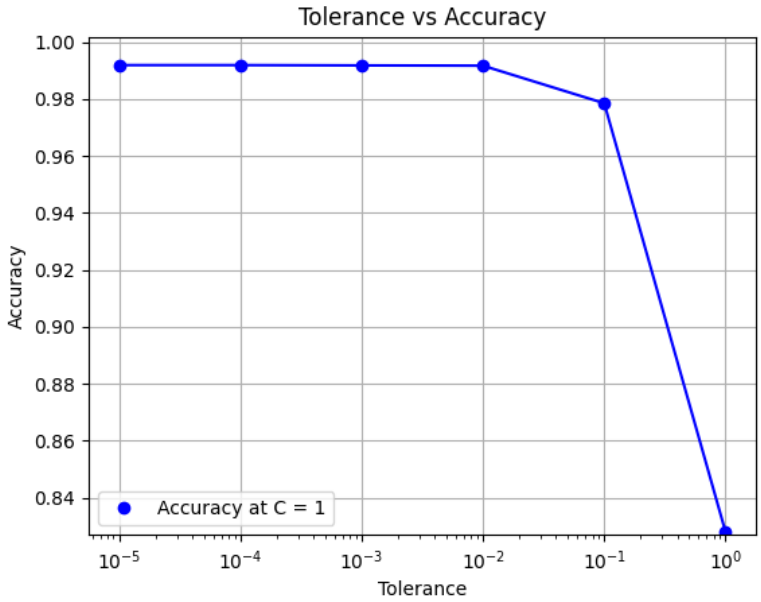

Figure 5: The graph suggests a general trade-off between tolerance and accuracy. As tolerance (acceptable error) increases (moving right on the x-axis), accuracy (closeness to the actual value) generally decreases (moving down on the y-axis). This means that allowing for a larger margin of error typically results in less precise measurements or outputs.

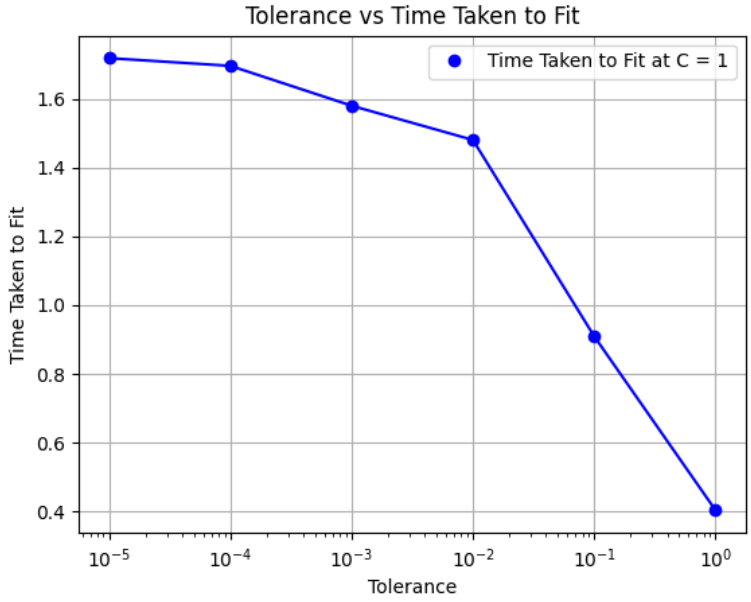

Figure 6: The graph suggests that achieving a smaller acceptable error generally requires more time to fit the model or process the data.

Logistic Regression:

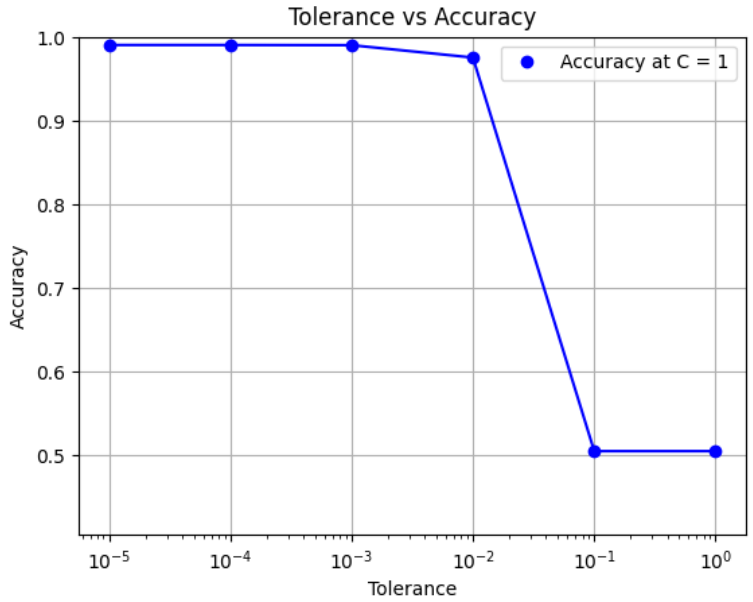

Figure 7: The graph suggests a general trade-off between tolerance and accuracy. As tolerance increases (moving right on the x-axis), accuracy seems to generally decrease (moving down on the y-axis). This means that allowing for a larger margin of error typically results in less precise measurements or outputs.

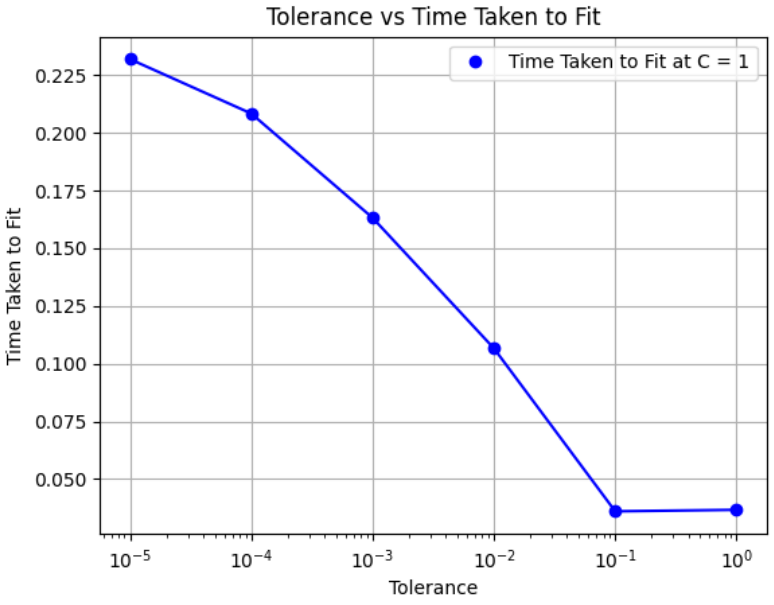

Figure 8: The graph suggests that achieving a tighter tolerance (smaller allowable difference in fit) generally requires more time to fit the model.