smartphone Side-Channel Attacks and Defenses

**Module 4 lAB Manual**

**Lab Manual Development Institution:** Colorado School of Mines

**Lab Manual Contributors**:

Weiping Pei\*, Ryan Jackson+, Chuan Yue@

(\*: PhD student and research/teaching assistant; +: Master Student; @: Faculty.)

# Model training and Evaluation

**Lab Description:** In this lab, you will build machine learning models to perform input inference as well as user fingerprinting attacks, and you will also evaluate the accuracy of your attacks. In more details, you will select, apply, and compare some appropriate machine learning algorithms such as Random Forest and SVM (Support Vector Machine). You will use some popular machine learning software packages such as [Weka](https://www.cs.waikato.ac.nz/ml/weka/) and [scikit-learn](http://scikit-learn.org/stable) to create programs for training the machine learning classifiers. You will evaluate the accuracy of your trained classifiers using techniques such as cross-validation. We suggest you to use Python and [scikit-learn](http://scikit-learn.org/stable) to write your programs so that this lab manual will be more straightforward to you. This lab consists of eight STEPs.

The high-level **learning outcomes** and the corresponding **assessment** of this lab are summarized as follows. In other words, upon completion of this lab, students should be able to:

* **Create** data structures to contain the feature vectors and label lists.
  + Assessed by the tasks and outputs specified in STEP 1.
* **Construct** the training and testing datasets by splitting the entire dataset.
  + Assessed by the tasks and outputs specified in STEP 2.
* **Create** machine learning models with training and hyper-parameters tuning for performing input inference attacks.
  + Assessed by the tasks and outputs specified in STEP 3.
* **Evaluate** the accuracy of machine learning models based on the test data.
  + Assessed by the tasks and outputs specified in STEP 4.
* **Compare** different machine learning models and their accuracy.
  + Assessed by the tasks and outputs specified in STEP 5.
* **Analyze** the impact of different features and datasets on model accuracy.
  + Assessed by the tasks and outputs specified in STEPs 6 and 7.
* **Create** machine learning models with training, hyper-parameters tuning, and evaluation for performing user fingerprinting attacks.
  + Assessed by the tasks and outputs specified in STEP 8.

**Lab Environment:** Linux, Mac, or Windows.

**Lab Files that are Needed:** TheLab Manual file and the feature\_data.csv file.

**Learning Setting:** This lab module is for students to complete outside the classroom, so it can be used in either face to face or online courses.

**Prerequisites:** Java or Python Programming, Basic Cybersecurity and Machine Learning knowledge and skills, Linux or Windows Systems, Computer Networks.

**Length of Completion:** 600 minutes.

**Level of Instruction:** Senior undergraduate students or graduate students in CS or related STEM programs. The lab exercise should be further simplified if it will be used for freshmen, sophomores, or none-CS major students.

**Interconnection with Other Labs:** This lab module is standalone by itself; however, if needed, an instructor can use the details in the course project manual and the other four lab manuals to provide additional hints to students.

**Assessment Guideline:** Students should follow the steps to answer all the questions. Based on the points assigned to each individual question, the instructor will grade each answer (together with the additional materials if specified for the question) in terms of its correctness (60%), clarity (20%), and concision (20%).

### **Lab Exercise/step 1 (download the feature data file and read the content into data structures)**

Download the file feature\_data.csv which contains the feature vectors and the labels (i.e., the key pressed and the user ID) produced in Lab 3. Each row, except for the first row for column headers, corresponds to a single keypress. Each column, except for ‘key\_pressed’ and ‘user\_id’, contains some feature extracted from the motion data of a user’s smartphone while the user was typing. You will first build a machine learning model to predict what key was pressed based on those features.

Some of those features such as the frequency related ones are on the complex plane, but most machine learning packages only accept real value inputs. For the sake of simplicity, you’ll discard the complex portion of the complex numbers and only use the real portion (scikit-learn’s machine learning algorithms do this automatically when passed with complex numbers).

Write Python code to read the content in feature\_data.csv into two data structures: one is a 2-dimensional matrix (denoted as “X”) that contains all of the feature vectors where each row is one feature vector, and the other is a list (denoted as “y”) that contains the `key\_pressed’ column. Keep both data structures ordered such that the feature vector at index i corresponds to the keypress at index i for all possible values of i. Note that you discard the user IDs for now because you will first build the models to perform the input inference attacks; you will use the ‘user\_id’ column later in STEP 8 for user fingerprinting attacks.

**Question 1**: Please describe and explain what might be a better way to deal with these complex numbers that wouldn’t involve discarding potentially useful information.  
(Total score: 5 points. Grading rubric:  
100% points for a clear description and a clear explanation;  
60% points for only a clear description or a clear explanation;  
30% points for a vague description and/or a vague explanation.)

**Question 2**: What is your code for constructing the two data structures?  
(Total score: 5 points. Grading rubric:  
100% points for correct and complete code;  
60% points for partially correct and partially complete code;  
30% points for partially correct or partially complete code.)

### **LAB EXERCISE/STEP 2 (spilt the data into training and testing data)**

You’ll use 80% of the data for training your machine learning models and 20% for testing the models. You should stratify the data such that the classes remain balanced despite this split (e.g. the proportion of ‘6’ pressed in the training data roughly equals the proportion of ‘6’ pressed in the test data). You’ll then have 4 sets of data: labels for training, labels for testing, feature vectors for training, and feature vectors for testing. Here, the “label” refers to what key was pressed.

Some machine learning models (like support vector machines) assume input data standardized to zero mean and unit variance. You can perform a transformation to accomplish this on the training data, and then apply the same transformation to the test data.

Write code to accomplishes all of these in a function which takes “X” (the feature vectors) and “y” (the list of labels) as the input and returns the transformed four sets of data as the output.

**Question 3**: Please explain why it is important to stratify the classes as described above.  
(Total score: 5 points. Grading rubric:  
100% points for a clear explanation;  
50% points for a vague explanation.)

**Question 4**: What is your code for accomplishing all of these in STEP 2?  
(Total score: 10 points. Grading rubric:  
100% points for correct and complete code;  
60% points for partially correct and partially complete code;  
30% points for partially correct or partially complete code.)

### **LAB EXERCISE/STEP 3 (Train and tune a machine learning model)**

Choose a machine learning algorithm or model available in scikit-learn and read about it to get a basic understanding of how it works. We recommend you to try the Random Forest algorithm. Train your chosen model on the training data obtained from the previous step. You should read about the k-fold cross-validation method and use it to tune the hyper-parameters (e.g. number of trees in the random forest) of your model. The more folds you use, the longer this process will take. For the purposes of this lab, choosing 3 folds should be alright. This [scikit-learn document on tuning the hyper-parameters](http://scikit-learn.org/stable/modules/grid_search.html) should be helpful to you.

**Question 5**: What is the algorithm or model you used? What hyper-parameters are available for you to tune? How did you tune your model?(Hint: You can use GridSearchCV method in scikit-learn to tune your model).  
(Total score: 10 points. Grading rubric:  
100% points for answering all the three questions;  
60% points for answering two of the three questions;  
30% points for answering one of the three questions.)

**Question 6**: What hyper-parameters worked best for your model? What values did you experiment with?  
(Total score: 5 points. Grading rubric:  
100% points for answering both questions;  
50% points for answering one of the two questions.)

### **LAB EXERCISE/STEP 4 (test your machine learning model)**

Give your trained model the feature vectors that you held out for testing. Compare your model’s classification results to the actual correct labels in the testing set of labels. You’ll use accuracy as the performance metric since it is the most straightforward one in this lab.

**Question 7**: How is accuracy calculated for a classification problem? What accuracy did your model achieve? (Hint: Your model should be much better than random guessing which would give a 10% accuracy since you have 10 balanced classes).  
(Total score: 10 points. Grading rubric:  
100% points for answering both questions, and *accuracy >= 25%*;  
60% points for answering both questions, and *25% > accuracy >=15%*;  
30% points for answering one of the two questions.)

**Question 8**: What is your code for testing your trained machine learning model?  
(Total score: 5 points. Grading rubric:  
100% points for correct and complete code;  
60% points for partially correct and partially complete code;  
30% points for partially correct or partially complete code.)

### **LAB EXERCISE/STEP 5 (repeat steps 3 and 4 for some other machine learning algorithms)**

Try other machine learning algorithms (e.g. Support Vector Machine or Artificial Neural Network). Note: if you use a neural network, make your life easier by using a single hidden layer unless you have previous experience. Here are a few algorithms that you can import and use in scikit-learn:

from sklearn.ensemble import RandomForestClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.neural\_network import MLPClassifier

The scikit-learn online documentation (please Google search the latest one) for each of them is quite good.

**Question 9**: What other machine learning models did you choose? What hyper-parameters did you try and which worked best? What accuracy did these new models achieve? You should try at least one other model.  
(Total score: 10 points. Grading rubric:  
100% points for answering all the three questions;  
60% points for answering two of the three questions;  
30% points for answering one of the three questions.)

### **LAB EXERCISE/STEP 6 (Try using only time domain features)**

Use two of your tested algorithms that performed well, and train them without using the frequency domain features (i.e., removing all the columns that contain the string “freq” from the data read in from the CSV file). This will result in much shorter feature vectors in “X” with less information for your models to learn from.

**Question 10**: What is the accuracy now without using the frequency domain features? Compared with previous results, which is better? Please explain your comparison results.  
(Total score: 10 points. Grading rubric:  
100% points for answering both questions with a clear explanation;  
60% points for answering one of the two questions with a clear explanation;  
30% points for answering one of the two questions.)

### **LAB EXERCISE/STEP 7 (Try using only data from one user)**

As shown in the “user\_id” column, the dataset contains keystrokes from 30 different users. Let’s see how the results change if you use data from only one user. Repeat steps 1-4 with only the data from User 1. Ignore the data from all other users. Use the full feature vectors (e.g. both time domain and frequency domain data). Use whichever machine learning model that performed well in the previous steps.

**Question 11**: Which machine learning model did you use? What are hyper-parameters that worked best? What is the accuracy for discerning which key was pressed using only data from User 1? Compared with experiments using all data, which is better? What is an explanation of your comparison result? (Hint: the training dataset’s size may affect accuracy.)  
(Total score: 10 points. Grading rubric:  
100% points for answering all the five questions;  
80% points for answering four of the five questions;  
60% points for answering three of the five questions;  
40% points for answering two of the five questions;  
20% points for answering one of the five questions.)

### **LAB EXERCISE/STEP 8 (Infer which user is typing)**

Repeat steps 1 through 4, but this time use the “user\_id” column as the label (in the list “y”) instead of the “key\_pressed” column in order to perform the user fingerprinting attacks. In other words, you are now discarding the “key\_pressed” data and training a model to infer who is typing given the feature vectors. Use both time domain and frequency domain data. Use whichever machine learning algorithm that gave you the best performance before.

**Question 12**: What is your accuracy in inferring which user was typing some given keystroke? Which machine learning model did you use and what are the hyper-parameters that worked best?  
(Total score: 15 points. Grading rubric:  
100% points for answering both questions, and *accuracy >= 85%*;  
60% points for answering both questions, and *85% > accuracy >=50%*;  
30% points for answering one of the two questions.)

### **Puzzler**

This is an advanced activity for students who complete the regular activities early.

**Question 13**: Propose two concrete ideas on how you could improve this machine learning approach for performing input inference or user fingerprinting attacks. Explain why you think your ideas would be good. Examples would include ways to achieve higher accuracy or ways to evaluate your models more thoroughly than just reporting accuracy alone. Changing model hyper-parameters doesn’t count since you should’ve already tuned these. Choosing different models doesn’t count either, not because it wouldn’t help, but because you should’ve already tried that. Your answer to Question 1 can be counted as one idea here, but only if you try it and discuss the results.  
(Total score: 15 bonus points. Grading rubric:  
100% points for proposing two concrete ideas with the clear explanations;  
30% points for proposing one concrete idea with the clear explanation.)

## What to submit

Please answer all the 12 questions (Question 13 is optional) in this lab exercise. Please feel free to directly reuse this Word document to provide and submit your answers. Please submit your complete code for performing all the model training, tuning, and evaluation steps in this lab.