EC60091: Machine Intelligence and Expert Systems

TERM PROJECT

USER AUTHENTICATION USING KEYSTROKE DYNAMICS

GROUP MEMBERS (Group No. 7):
Shubhankar Sanjay Banerjee (18EC10056)
Susmita Mazumdar (18EC10062)
Ankur Mishra (18EC10004)
Prerna Goel (18EC35022)
Bhargava Sai Macha (18EC10008)

COURSE INSTRUCTOR:
Prof. Sudipta Mukhopadhyay

Contents

Introduction

Increase in the number of software and devices for hacking and cracking causes gains in unauthorized access which results in manipulation of important data. Methods like user ID and password which are mostly used as security are now not reliable and secure due to the rapid increase in hackers and crackers. Also, this method no longer provides consistent security measures because passwords are prone to shoulder surfing and passwords can also be hacked. To gain secure and efficient access either the user must change his password frequently or the user should use the strong password (combination of alphabets, numeric and special symbols). Users do not respect these conditions as they feel them quite strict and difficult to be applied. The solution to the above said problems is keystroke dynamics. Keystroke Dynamics is a behavioral biometric approach to enhance computer access rights. It verifies the individual by its keystroke typing pattern. Keystroke biometric depends on the supposition that the composing example of every client is unique.

Approaches of User Authentication: *Object Based, Knowledge Based, and Biometric Based.*

Two categories are: *Physiological biometrics and Behavioural biometrics.*

Physiological Biometrics: It illustrates those features that describe who the user is depending on the physical attributes e.g. fingerprints, Iris and retina scanning. For this additional hardware required.

Behavioural Biometrics: It is based on typing pattern, Voice recognition and Signature style. Behavioral characteristics can be composed without the requirement of any extra hardware. This study will focus on Behavioral biometric

technique i.e. Keystroke Dynamics. Keystroke Dynamics is one of the recent Biometrics Technology used in upcoming research. This method analyzes the way a user types on a fatal, by monitoring the keyboard input. Since the input device is the remaining Keyboard, this approach is not exclusive.

Description of the Problem

In this paper we use keystroke dynamics to verify the identity of users when passwords are being pressed. These keystroke dynamics were used to extract the common features across all the 10 users. The alphabets and all pairs of alphabets which were typed in by every user. The corresponding hold time and latencies of the common features were extracted to create a feature vector for each of the 10 users. To solve the problem of novel impostors detection, a support vector machine (SVM) is employed because of its capability of presenting consistent algorithm status. The solution uses one-class SVM to capture the domain of the normal data patterns where probability lives.

The main objective of this Project Work is to utilize keystroke elements as biometric characteristics for individual verification and examinations. The execution of different classifiers in validation investigates biometric datasets. Recognizable proof of Genuine User/Imposter from Keystroke Dynamics Dataset by utilizing machine learning systems. To utilize keystroke elements as biometric quality for individual verification and investigations the execution of different classifiers in confirmation probes chose biometric datasets.

Focal points of Keystroke Dynamics compared to composed marks signature design can't be imitated. Most security frameworks permit a predetermined number of off base endeavours. After a couple of off base endeavours they obstruct the record. Contrasted with physiological biometric frameworks, for

example, finger print, Iris identification Keystroke flow does not require any additional equipment. Subsequently usage and arrangement cost is low.

Data Collection

A Python Script based GUI was used to collect the keystroke dynamics for 10 users.

The user just needs to press the ‘Start’ button given in the GUI and it starts recording the keys pressed by the user along with individual key press and key release time which later on is used to calculate the flight time(Inter key latency) and dwell time (hold time). It also records the mouse movements (mouse dynamics), however our problem statement required the use of just the keystroke dynamics.

This process was repeated 5-6 day a week for 4 weeks so as to have enough data to train the model. These feature vectors were then fed to a 1-class SVM with 5-fold cross-validation to train the SVM and then test it and report its accuracy of prediction.

Feature Extraction

Two basic main features are initially extracted from raw keystroke dynamics are *Key Hold time* and *Inter Key latency time*.

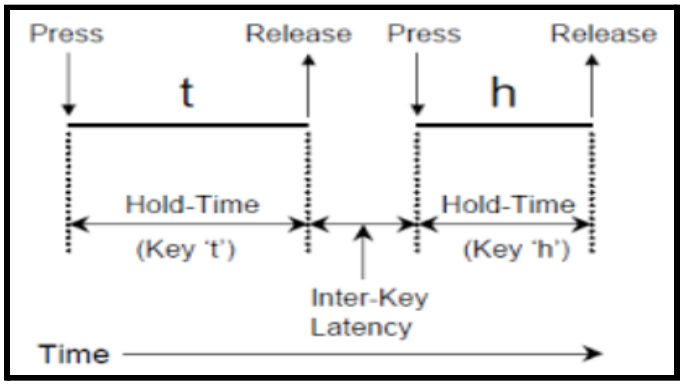


Figure 1: Defining the hold times and latencies

Hold TIme /Dwell Time (DT): Dwell time, also known as key hold time, is the duration for which a key is held down. It refers to how long a key was held pressing down or the amount of time between pressing and releasing a single key.

Latency /Flight Time (FT): Flight time is the duration between pressing a key and releasing the next key. It is also known as latency time, inter key time or interval time. It refers to the amount of time between pressing and releasing two successive keys. It involves key events (press or release) from two keys, which could be comparable or different characters When typing a text, flight time and dwell time are unique for each user, and is independent of overall typing speed. This is an important factor that is directly related to user acceptability to the technology. The technology should offer users as much comfort and transparency as possible by not overloading users with long inputs, memorization of complex strings, or providing huge amounts of repetitive input.

ppTime (PP): the latencies of when the two catches (keys) are press time ;
rpTime (RP): the latencies of when one catch (key) is discharged and the other is press

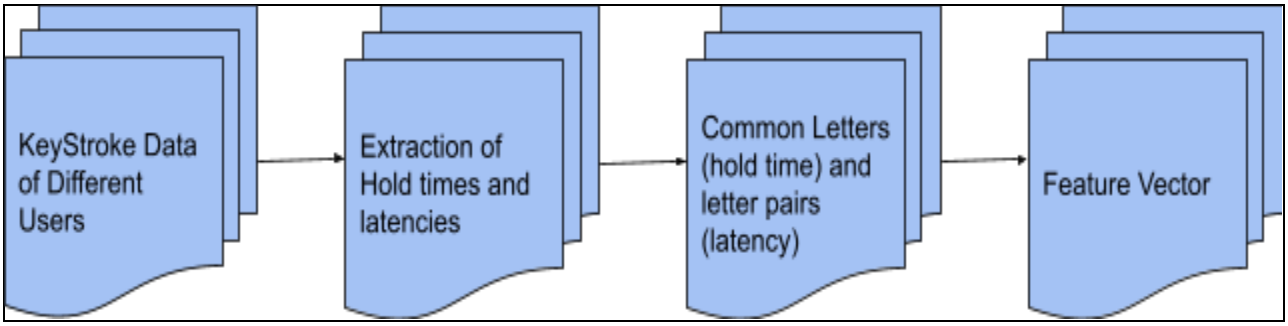


Figure 2: Flow of Feature Extraction

Thus the use of hold times and latencies as features is justified. The hold times and latencies were extracted from the data of each user. Though ideally all users should have written the same sentences, due to mismatch in the data between different groups, we had to extract common keypresses and latency pairs to make the feature vector. Thus finally the hold times and latencies of the same keys and key pairs respectively were taken as final features in the feature vector for each user.

To illustrate how exactly we made the feature vector, let us take a small example. Suppose there are only two users, and the keys pressed by User 1 were only A, B and C. The keys pressed by user 2 were A, B, K only. The successive key-pairs pressed by User 1 were AB and AC. For User 2, they were AB, AK only. Since the common values are A, B and AB, we take 2 (B and AB) out of these three only to form the feature vector. Now, in the figure we can also see the contents of the files for User 1&2. We form a feature vector and take some possible combinations of these for User 1 to obtain an 256X24 matrix which we take as the feature vector. The same is done for User 2. In the figure shown below a 2X4 matrix was formed for the sake of brevity.

The number of rows in each file (eg. a.txt), on an average, was found to be quite large. Thus keeping in mind the complexity and computation time we fixed the feature vector length to be 24 . To obtain these 24 common letters and letter pairs among all users, we will obtain 24^{24} feature vectors for a single user which is not feasible to train or even store efficiently. So, for each of the 24 common features obtained 256 randomly selected values were sampled to train the SVM. This gives a total of 256 feature vectors of length 24 per user.

Though there were many key presses like numericals , special functions /symbols etc but these presses weren't common for all users . Thus to maintain uniformity only the Alphabet key presses of all users are considered as the common features.

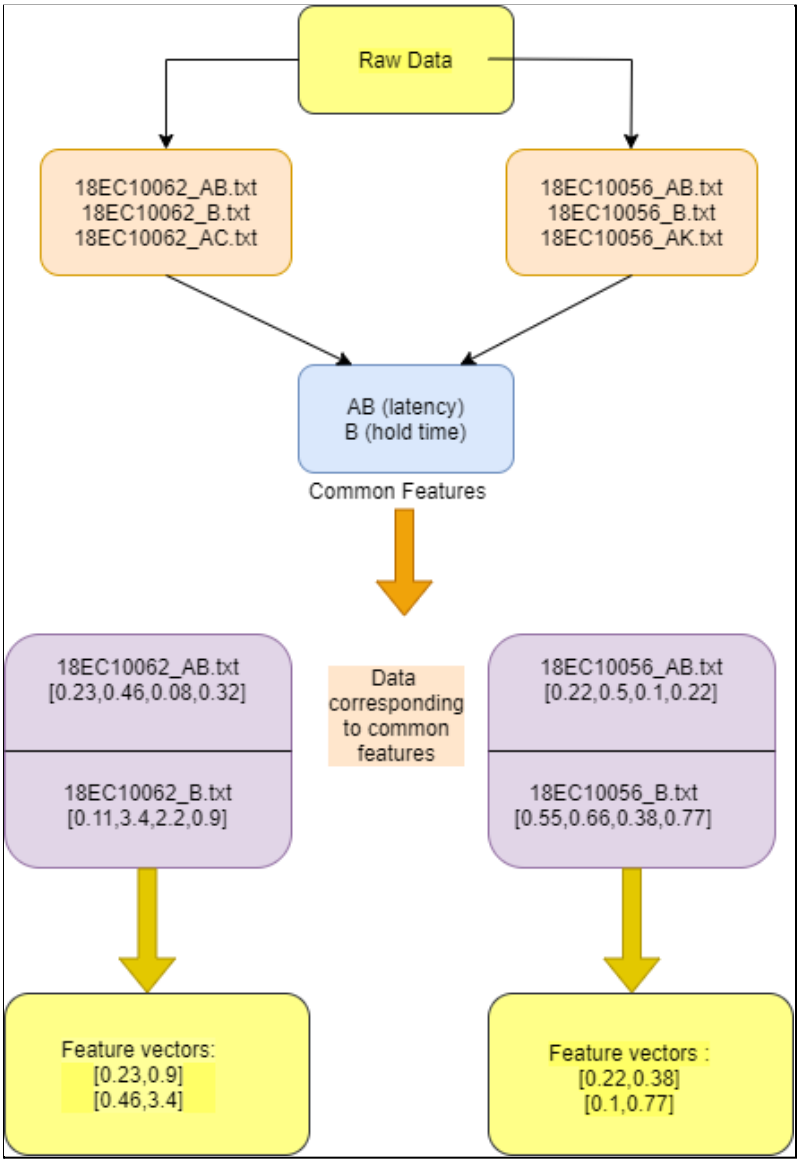


Figure 3: An example of Feature Extraction

Data Standardization and Pre-Processing

The training data was standardized to have mean 0 and variance 1. The mean of the training data was subtracted from the testing data and the testing data was then divided by the variance of the training data. The results with and without normalization are presented.

Principal component analysis was performed and 8 principal components were selected from the 24. The results with and without PCA are also presented.

Brief Theory of One Class SVMs

Support vector machines (SVMs) are supervised learning models that analyze data and recognize patterns, and that can be used for both classification and regression tasks.

Typically, the SVM algorithm is given a set of training examples labeled as belonging to one of two classes. An SVM model is based on dividing the training sample points into separate categories by as wide a gap as possible, while penalizing training samples that fall on the wrong side of the gap. The SVM model then makes predictions by assigning points to one side of the gap or the other.

Sometimes oversampling is used to replicate the existing samples so that you can create a two-class model, but it is impossible to predict all the new patterns of fraud or system faults from limited examples. Moreover, collections of even limited examples can be expensive.

Therefore, in one-class SVM, the support vector model is trained on data that has only one class, which is the “normal” class. It infers the properties of normal cases and from these properties can predict which examples are unlike the normal examples. This is useful for anomaly detection because the scarcity of training examples is what defines anomalies: that is, typically there are very few examples of the network intrusion, fraud, or other anomalous behavior.

The following optimization problem is to be solved:

$$\min_{w,\,\xi_i,\,\rho} \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho$$

$$\text{subject to:}$$

$$(w \cdot \phi(x_i)) \geq \rho - \xi_i \qquad \text{for all } i = 1, \ldots, n$$

$$\xi_i \geq 0 \qquad \text{for all } i = 1, \ldots, n$$

The solution for w and gives the plane and the decision is taken as:

$$f(x) = \operatorname{sgn}((w \cdot \phi(x_i)) - \rho) = \operatorname{sgn}(\sum_{i}^{n} \alpha_i K(x, x_i)$$

The parameter v, also known as the margin of the One-Class SVM, corresponds to the probability of finding a new, but regular, observation outside the frontier. For the kernel K(x,x),

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

where $\sigma \in R$ is a kernel parameter and $\|x - x'\|$ is the dissimilarity measure.

The dissimilarity is the L2 norm. We define $\gamma = 1/2\sigma^2$. Intuitively, the gamma parameter defines how far the influence of a single training example reaches, with low values meaning far and high values meaning close.

Hyperparameters of the One Class SVM :

- **Kernels :**

  The main hyperparameter of the SVM is the kernel. It maps the observations into some feature space. Ideally the observations are more easily (linearly) separable after this transformation. There are multiple standard kernels for these transformations, e.g. the linear kernel, the polynomial kernel and the radial kernel. The choice of the kernel and their hyperparameters greatly affect the separability of the classes (in classification) and the performance of the algorithm.

- **Nu (*v*) :**

  The parameter nu is an upper bound on the fraction of margin errors and a lower bound of the fraction of support vectors relative to the total number of training examples. For example, if we set it to 0.05 we are guaranteed to find at most 5% of your training examples being misclassified (at the cost of a small margin, though) and at least 5% of your training examples being support vectors.[Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier].

- **Gamma Parameter :**

  The parameter gamma is usually used for polynomial / radial kernel functions.Gamma parameter of RBF (radial basis function) controls the distance of influence of a single training point. Low values of gamma indicates a large similarity radius which results in more points being grouped together. For high values of gamma, the points need to be very close to each other in order to be considered in the same group (or class). Therefore, models with very large gamma values tend to overfit.

Cross Validation

For user i among n users, data corresponding to user i and the n-1 remaining users is shuffled randomly. The data corresponding to user i is divided into 80:20 ratio, out of which 80% data of user i will go for training. Now a portion of data (totally 5% of size of user is data) from n-1 users is added to 20% data of user i for testing so that the resulting ratio for testing becomes 80:20. (80% from user i and 20% from other users). The above operation is repeated 5 times.

Results

- For different combinations of the hyperparameters (kernel type , gamma , nu) as listed above, the classification accuracy for the test classes for each of these Roll Number / Users on applying one-class SVM are plotted below.

KERNEL TYPE :

- **LINEAR KERNEL :**

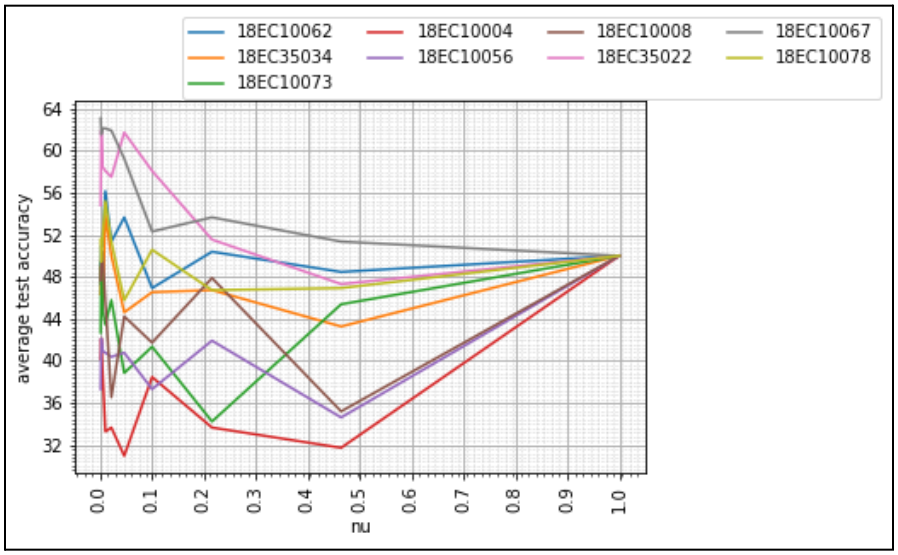
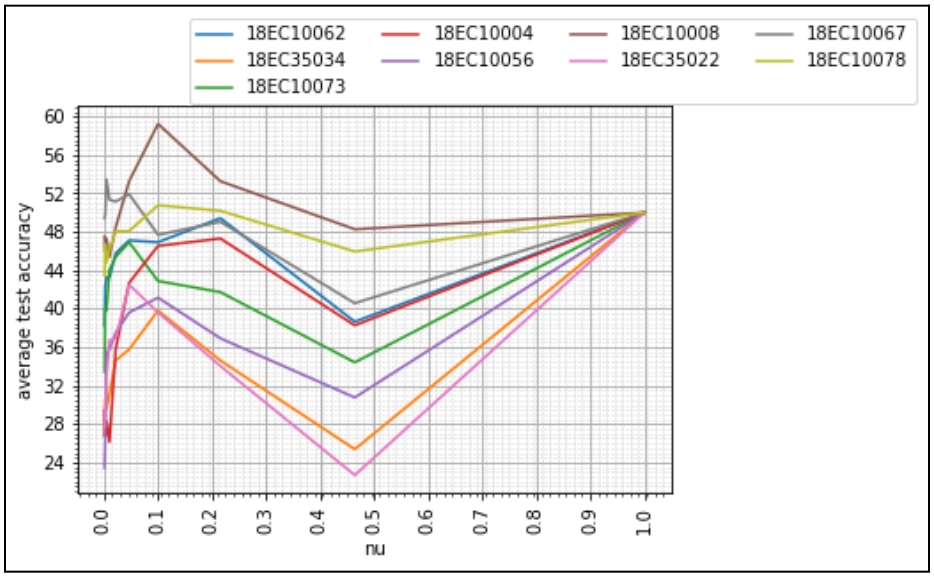


Fig : Linear Kernel

- **POLYNOMIAL KERNEL (default degree =3)**

gamma param set to default auto = 1/n_features



- **RBF KERNEL :**

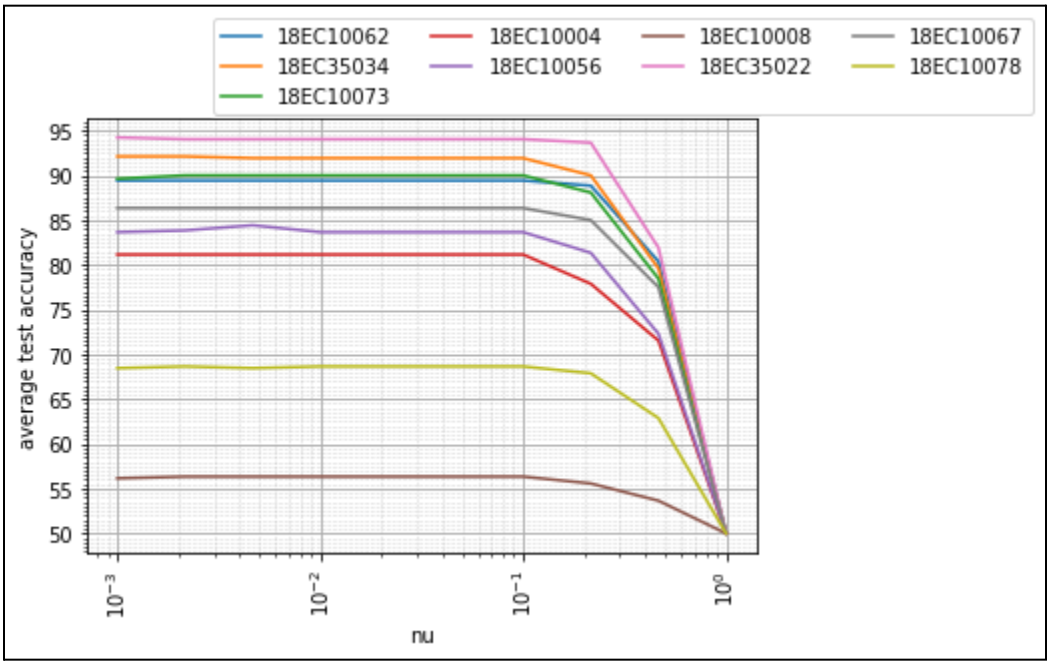


Fig: RBF Kernel

- As can be seen the best testing accuracy results are obtained when RBF kernel is used. Thus for all the further results shown below the RBF kernel is used .

GENERIC CLASSIFICATION RESULTS :

A snippet of the Classification results for the USERS using the one class SVM is attached below. This also contains the metadata for a particular user in a particular fold (1-5) for each parameter variation like 'nu' . The metadata has the complete information like

- Training Dataset Size : The number of feature vectors (out of the 256 created vectors for each user) used for training . This is usually 80% of 256 = 204
- Test Dataset Size : The number of feature vectors used for testing ! . This also has a ratio of 80% of test data from the same user and the rest 20% of it from other users.
- Classification Metrics : The classification metrics like testing accuracy , precision , recall , specificity are also listed for each run

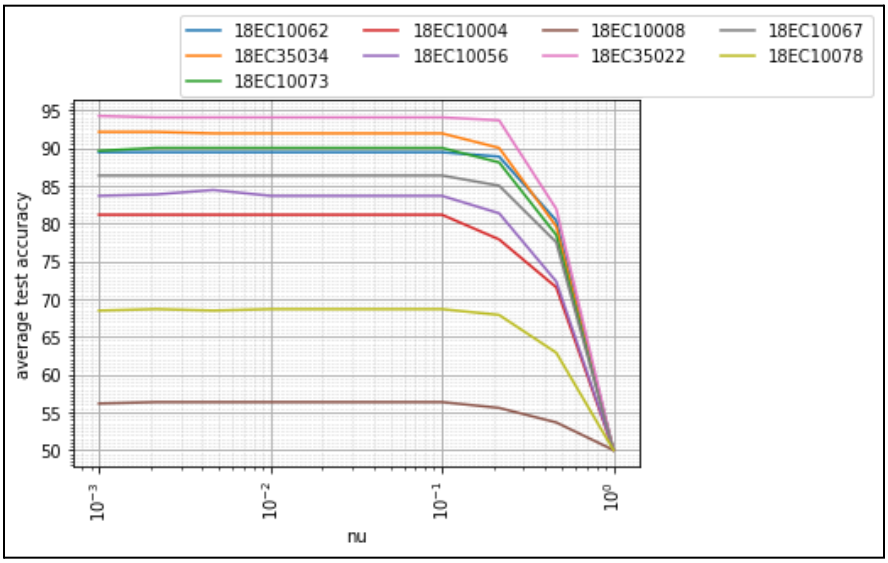
The complete results for all users/ roll numbers for all the folds / parameters variation are listed and printed by the code.

```
For User 18EC10062
Validation Fold 1:
Training Dataset Size : 204
No of Training Misclassifications : 26
Training Error : 12.75 percent
Testing Dataset Size : 65
Correct Decissions: 51
Incorrect Decissions: 14
Testing Accuracy: 80.77 percent
Testing Error: 19.23 percent
Precision: 95.24 percent
Recall: 76.92 percent
Specificity: 84.62 percent

Training Dataset Size : 204
No of Training Misclassifications : 25
Training Error : 12.25 percent
Testing Dataset Size : 65
Correct Decissions: 51
Incorrect Decissions: 14
Testing Accuracy: 80.77 percent
Testing Error: 19.23 percent
Precision: 95.24 percent
Recall: 76.92 percent
Specificity: 84.62 percent

Training Dataset Size : 204
No of Training Misclassifications : 26
Training Error : 12.75 percent
```

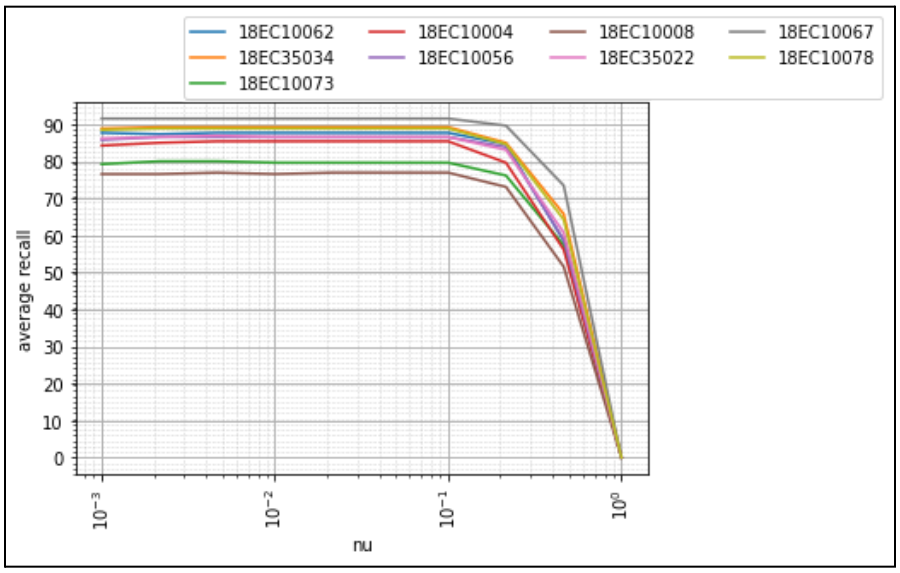
NU VARIATION

(kernel type fixed as RBF)

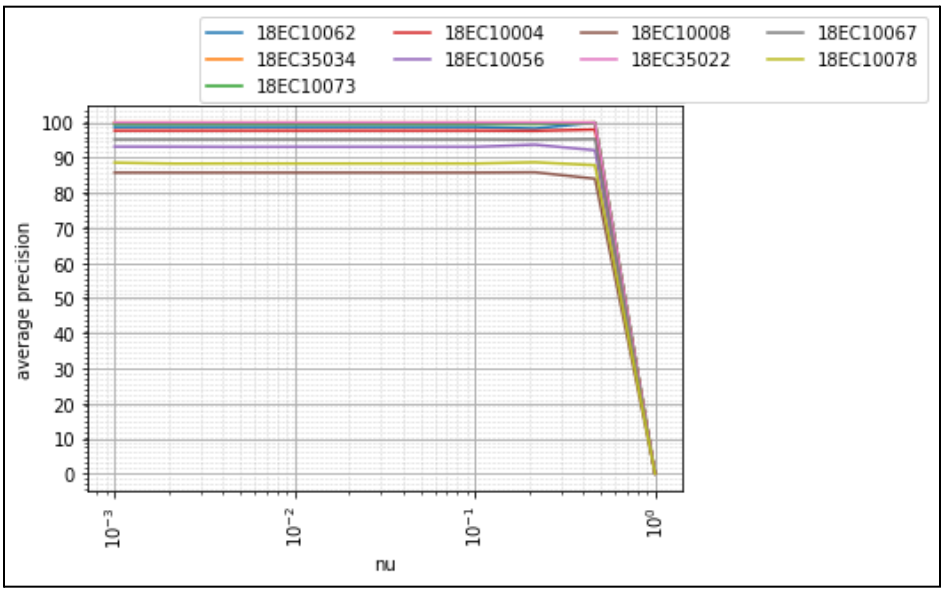
1. **Average Testing accuracy vs Nu**



2. Average recall vs Nu :



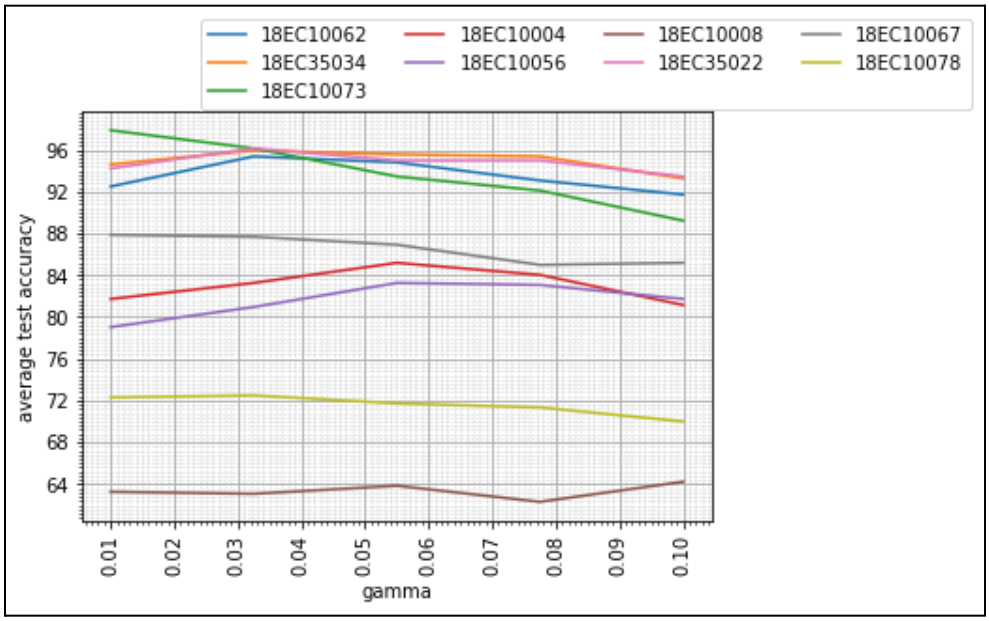
3. Average Precision vs Nu



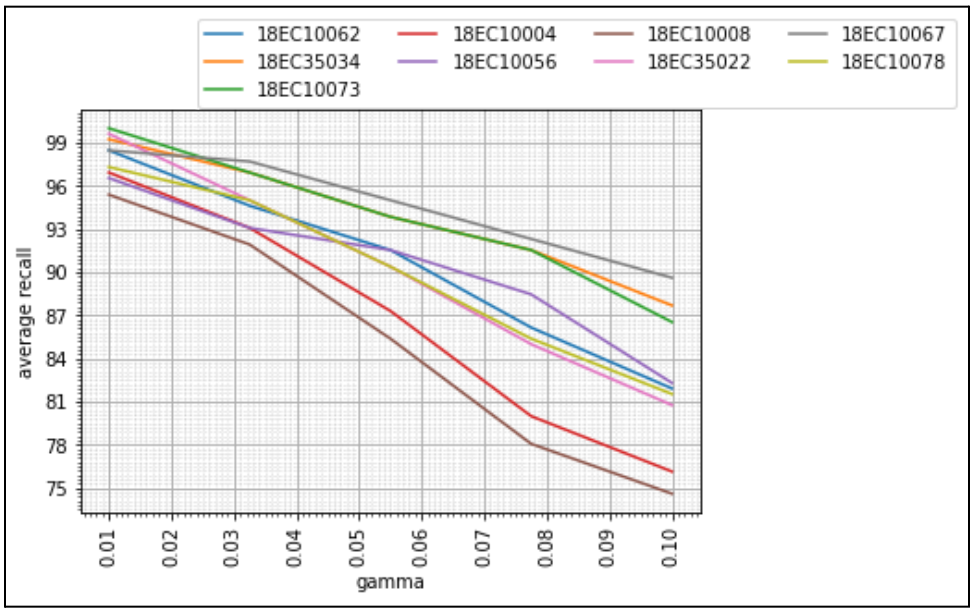
GAMMA VARIATION

(kernel type fixed as RBF)

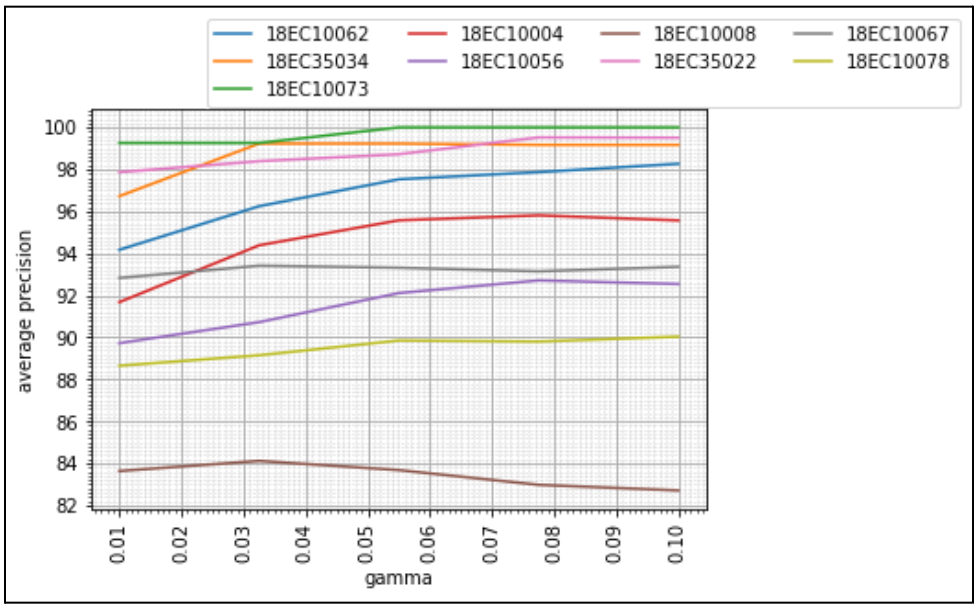
1. **Average Testing accuracy vs Gamma**



2. **Average recall vs Gamma :**



3. Average Precision vs Gamma:



Conclusion:

The main task of this term project to use the user's keystroke dynamics to authenticate / classify the users has been successfully completed. The results plots generated validates this task completion .So it can be summarised that given a set of proper hyperparameters the performance metrics like accuracy /precision/recall obtained by the above implementation is quite good (mostly above 80% for almost all users) ensuring a proper user classification ! . The high recall values specially suggest a lower chance of misclassification of the actual user thus proving it's convenience in being used as an user authentication system. The high accuracy implies a low chance of security breach.