

1. INTRODUCTION

1.1 Motivation

The geometric models for biometric hand features (skin fingerprint and palm model) are easy to hack in contrast with biometric bones-based system (phalangeal biometric models). This is because hand geometric model is based on external measurement of hand features (hand patterns; knuckle creases) which are highly exposable. Suspects leave traces everyday and everywhere they have utilized biometric resources. Even more aggravating, if suspects have committed crimes, third party presence or interfering on the same system might have defeated biometric cue traces when matching sample to suspect. However, in cyberspace context, utilization of external hand measurements and/or knuckle creases may allow criminals to easily utilize information of legitimate users for unauthorized access or activities. Another limitation of these features is that, it does not favor victims of skin diseases, in a situation where top layers of their skin (palm and fingerprints) are affected. Thus, victims may no longer get access to any external hand measurement-based biometric recognition system, even if they have fully recovered. However, these problems might have defeated crime detection, accident rescue, amnesia victims identification, missing persons, unknown deceased, and cyberspace authentication. In contrast, the phalangeal biometric model require intrinsic hand features such as length/width of Proximal Phalanx, Distal Phalanx etc. which are not as exposed as their counterparts. Therefore, hand biometric recognition based on intrinsic measurement of human bones stand as solution for human identification and crime detection. Moreover, an appropriate selection of intrinsic hand features yield an excellent result. Due to these reasons, evaluation of intrinsic hand features based on demographic cues by learning models aid in recognizing biometric hand information. It is well known that recognition of biometric hand features and demographic cues is a complex non-linear procedure that happens by a complex interaction of different redundant hand features. Pearson Correlation coefficient (PCC) features selection is generally appreciated as a stable feature representation of complex and non-linear dynamical behaviour of hand biometric information. This analysis minimize models training complexity, and reduce misclassification during recognition. Recognition from

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

selective intrinsic hand features by PCC is an effective and reliable method for critical infrastructure.

Studies of human demographic characteristics from biometric hand features have been extensively proved in the body of literature purposely on Artificial Intelligence (AI) learning or descriptive statistics analysis. It is being documented that external hand features (geometric features) are deployed in biometric recognition, such information involves measurement of lengths, circumference, thickness, width of fingers, palm, wrist, skin texture etc. to predict human sex, age, and human height. Karki and Singh demonstrate a strong relationship between the pattern of fingerprints and human gender. Meanwhile, in relationship of biometric information in cross-domain identity recognition is demonstrated. Among well-known biometric gesture recognition method is Discriminant analysis, which was followed by descriptive statistics approaches. Likewise, Thakar et al. propose method of gender determination using ridge characteristics and ridge density of fingerprints. These features were statistically analyzed and their results proved the usefulness of biometric information in determining demographic information. Another variation of the descriptive statistics model is a bootstrap estimate, where a large number of sample sizes similar to actual samples are drawn with replacement from samples, and desired statistic was determined within each sample. Bootstrap fails to approximate different ratios if they belong to similar finger measurements. Linear and curvilinear regression models were built to investigate two hundred and fifty (250) students using measurements of their height, hand length and breadth to predict gender. However, these models return low accuracy, features are not robust for biometric predictions. In Stevenage et al. proposed method to demonstrate restrictions while matching a suspect hand, analysis was carried out using Analysis of Variance (ANOVA).

As visited from the literature above, these models may not provide good prediction results, which lead to wrong conclusions. Misclassification from descriptive statistics, such as linear or curvilinear is correlated to cut-off point and there is no cut-off value that should be optimal. Nevertheless, AI algorithms are independent of any value of this cut-off.

In addition, misclassifications from classical and analytical statistics approaches happen because of nonlinearity from biometric features and model parameter settings. To handle nonlinearity of biometric features, AI algorithms remain as good choice. Multiple AI algorithms are combined using a fuzzy inference system, where hand geometry is particularly considered as intrinsic biometric cues from near-infrared human hand images to locate interphalangeal joints of human fingers. The paper employed pulse response, hand geometry and finger vein using Convolutional Neural Network fuzzy (CNN-fuzzy) inference for detection of a live human body to improve performance against counterfeit attempts. A part from the fuzzy recognition algorithms, Alias and Radzi, adopted an SVM for fingerprint classification. Gavrilova et al. in, used kinect sensor to extract human emotion cues in multimodal context, which are trained with SVM classifier. Recognition of human hand gestures is performed using ECOC-SVM. Their results indicate feasibility of using SVM in real-world applications. Henceforth, to further explore the capabilities of AI-based algorithms to learn intrinsic hand features in human demographic cues recognition, SVM demonstrates to be better as it strikes a better balance between nonlinear feature points to predict four demographic information (sex, height, foot size, and log-weight). The results obtained using the above-mentioned AI algorithms are quite promising and motivated us to explore other AI-based algorithms for recognition of demographic cues from biometric information. However, the major drawback of all these algorithms is that they do not prove to be robust in case of nonlinear features and have large parameter computation which lead to low accuracy. Therefore, to further explore the capabilities of AI-based algorithms and to overcome the drawbacks of different parameters estimation, the Error Correction Output Code (ECOC) scheme is used with Pearson correlation coefficient features (PCC) to recognize the demographic cues from biometric hand features.

Inspired by the above contributions, in this paper, we compare the performance of two improved AI algorithms for high-precision recognition of demographic cues from matching sample to suspect. However, in cyberspace context, utilization of external hand measurements and/or knuckle creases may allow criminals to easily utilize information of legitimate users for unauthorized access or activities. Another limitation of these features is that, it does not favor victims of skin diseases, in a situation where top layers of their

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

skin (palm and fingerprints) are affected. Thus, victims may no longer get access to any external hand measurement-based biometric recognition system, even if they have fully recovered. However, these problems might have defeated crime detection, accident rescue, amnesia victims identification, missing persons, unknown deceased, and cyberspace authentication. In contrast, the phalangeal biometric model require intrinsic hand features such as length/width of Proximal Phalanx, Distal Phalanx etc. which are not as exposed as their counterparts. Therefore, hand biometric recognition based on intrinsic measurement of human bones stand as solution for human identification and crime detection. Moreover, an appropriate selection of intrinsic hand features yield an excellent result. Due to these reasons, evaluation of intrinsic hand features based on demographic cues by learning models aid in recognizing biometric hand information. It is well known that recognition of biometric hand features and demographic cues is a complex non-linear procedure that happens by a complex interaction of different redundant hand features. Pearson Correlation coefficient (PCC) features selection is generally appreciated as a stable feature representation of complex and non-linear dynamical behavior of hand biometric information. This analysis minimize models training complexity, and reduce misclassification during recognition. Recognition from selective intrinsic hand features by PCC is an effective and reliable method for critical infrastructure.

Studies of human demographic characteristics from biometric hand features have been extensively proved in the body of literature purposely on Artificial Intelligence (AI) learning or descriptive statistics analysis. It is being documented that external hand features (geometric features) are deployed in biometric recognition, such information involves measurement of lengths, circumference, thickness, width of fingers, palm, wrist, skin texture etc. to predict human sex, age, and human height. Karki and Singh demonstrate a strong relationship between the pattern of fingerprints and human gender. Meanwhile, in relationship of biometric information in cross-domain identity recognition is demonstrated.

Among well-known biometric gesture recognition method is Discriminant analysis , which was followed by descriptive statistics approaches. Likewise, Thakar et al. propose method of gender determination using ridge characteristics and ridge density of fingerprints. These features were statistically analyzed and their results proved the usefulness of biometric

information in determining demographic information. Another variation of the descriptive statistics model is a bootstrap estimate, where a large number of sample sizes similar to actual samples are drawn with replacement from samples, and desired statistic was determined within each sample. Bootstrap fails to approximate different ratios if they belong to similar finger measurements. Linear and curvilinear regression models were built to investigate two hundred and fifty (250) students using measurements of their height, hand length and breadth to predict gender. However, these models return low accuracy, features are not robust for biometric predictions. In Stevenage et al. proposed method to demonstrate restrictions while matching a suspect hand, analysis was carried out using Analysis of Variance (ANOVA).

As visited from the literature above, these models may not provide good prediction results, which lead to wrong conclusions. Misclassification from descriptive statistics, such as linear or curvilinear is correlated to cut-off point and there is no cut-off value that should be optimal. . Their results indicate feasibility of using SVM in real-world applications. Henceforth, to further explore the capabilities of AI-based algorithms to learn intrinsic hand features in human demographic cues recognition, SVM demonstrates to be better as it strikes a better balance between nonlinear feature points to predict four demographic information (sex, height, foot size, and log-weight). The results obtained using the above-mentioned AI algorithms are quite promising and motivated us to explore other AI-based algorithms for recognition of demographic cues from biometric information. However, the major drawback of all these algorithms is that they do not prove to be robust in case of nonlinear features and have large parameter computation which lead to low accuracy. Therefore, to further explore the capabilities of AI-based algorithms and to overcome the drawbacks of different parameters estimation, the Error Correction Output Code (ECOC) scheme is used with Pearson correlation coefficient features (PCC) to recognize the demographic cues from biometric hand features. Inspired by the above contributions, in this paper, we compare the performance of two improved AI algorithms for high-precision recognition of demographic cues from and physiological hand data set.

1.2 Problem Definition

Addressing crime detection, cyber security and multi-modal gaze estimation in biometric information recognition is challenging. Thus, trained artificial intelligence (AI) algorithms such as Support vector machine (SVM) and adaptive neuro-fuzzy inference system (ANFIS) have been proposed to recognize distinct and discriminant features of biometric information (intrinsic hand features and demographic cues) with good classification accuracy

1.3 Objective

- ✓ We are using optimized AI algorithms ((ANFIS) with subtractive clustering (ANFIS-SC) and SVM with error correction output code (SVM-ECOC)) because it is shown to be effective for biometric information recognition.
- ✓ In this paper, we compare the performance of the ANFIS-SC and SVM-ECOC algorithms in their effectiveness at learning essential characteristics of intrinsic hand features and demographic cues based on Pearson correlation coefficient (PCC) feature selection.
- ✓ Furthermore, the accuracy of these algorithms are presented, and their recognition performances are evaluated by metrics.

1.4 Limitation of this Project

- ✓ In cyberspace context, utilization of external hand measurements and/or knuckle creases may allow criminals to easily utilize information of legitimate users for unauthorized access or activities.
- ✓ It does not favor victims of skin diseases, in a situation where top layers of their skin (palm and fingerprints) are affected.

1.5 Organization of Documentation

Chapter 2: Literature Survey

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

Literature survey is that the most vital step in software package development method. Before developing the tool it is necessary to see the time issue, economy n company. Once this stuff square measure, 10 next steps square measure to see that package and language might be used for developing the tool.

Once the programmers begin improving the tool the programmers want heap of external support. This support might be obtained from senior programmers, from different websites. Before building 5 the system on the top of thought square measure taken into consideration for developing the enforced system.

Chapter 3: System Analysis

Chapter three provided to hide the enforced system Analysis.

Chapter 4: System Design

Chapter 4 provided to cover the system modules and design.

Chapter 5: System Implementation and Results

Chapter 5 provided to cover software environment and Results.

Chapter 6: System Testing

This chapter is to cover testing of system.

Chapter 7: Conclusion and Future Scope

This chapter contains conclusion and further work with light scope

2 LITERATURE SURVEY

2.1 Introduction

Artificial intelligence (AI) is superior technique when there is a nonlinearity problem among biometric features recognition. AI can handle problems that have non-linear solutions irrespective of fitness of non-linear features. In addition, AI learning approach is suitable when needed to solve biometric features within certain time. Exploiting the benefits of AI algorithm, we proposed to build features from PCC into optimized AI algorithm. PCC-based feature selection minimized features dimensionality and AI learning complexity. This section describes architectures of the following adopted AI algorithms; (a) ANFIS-SC (b) SVM-ECOC and steps of the methods are described in flowchart.

2.1.1 Adaptive Neuro-Fuzzy Inference System

ANFIS is an AI approach, composing Artificial Neural Network (ANN) and fuzzy inference networks (FIS) as a single ANFIS model, leveraging individual limitations of ANN and FIS methods. One major advantage of ANFIS is its ability to make good representation of complex and non-linear connections among features. In addition, its soft computing and rapid convergence raised interest of using ANFIS in prediction of complicated relationships. ANFIS aim to enhance/optimize FIS parameters by referring learning protocol according to input-output relationship vector. ANFIS optimization is carried out such that calibration errors among trained samples and original samples are minimized. The following equations define and describe ANFIS:

However, ANFIS layers Equations suffer from thorny parameter estimation, which seriously lead to noise during recognition. Nature-inspired optimizers played significant role in estimating ANFIS parameters. In contrast, parameter estimation is achieved through clustering mechanisms. Clustering serves as robust technique to handle cluster information. Therefore, empowering ANFIS with clustering is a substitute to nonlinear features estimation. There exist other ANFIS parameter estimations according to clustering mechanism such as: Fuzzy C-mean (FCM); which optimize ANFIS parameters by minimizing FCM-based derived objective function. Its major limitation is inability to

estimate large number of parameters. Grid partition algorithm (GP); effectively handle ANFIS parameters for small sample size, its flexibility is due to inherent large amount of rules and parameters generation. However, its main challenge is inherent exponential growth of fuzzy rules due to increase in sample size. Subtractive clustering (SC); where generated clusters are utilized to derive iterative-based optimization clustering to estimate ANFIS parameters. This method has good representation, if high dimensional case is utilized for moderate biometric hand features. Optimization of ANFIS parameters using SC achieved best performance.

2.1.2 Subtractive Clustering

In SC, all biometric features are considered as competent cluster Centre. The competent cluster over entire input-output feature pose is computed as their Euclidean distance equation over the whole feature poses. The poses having high competent greater than the threshold value are considered cluster centers. The primary concept is to calculate the density index $\#i$, corresponding to the biometric hand features.

We have chosen feature with high competent as next cluster Centre. This loop in Eq. is iterated until the stopping criteria is achieved. The cluster centers are utilized to generate fuzzy rules and parameters which lead to design FIS model. The rules are fuzzified using Eq., which adaptively alter characteristics of FIS to make the pattern looks like corresponding antecedent function. The major contribution to the existing ANFIS-SC is made on lines in Algorithm, where we fine tune FIS and ANFIS-based model every iteration with presently labeled recognizer (calculated cluster).

Therefore, computed clusters are applied to generate iterative optimization-based clustering for ANFIS model parameters identification. In this paper, number of cluster centers are set to be 14 for 112 biometric hand pose $h_1; h_2; f_1 f_1 f_1 ; h_D$. Every biometric hand pose represents a particular member of cluster center. Thus, making same number of fuzzy rules set \bullet and cluster centers, and each correspond to features of cluster. Formulation Eq. is iterated until convergence, which lead to achieve sufficient cluster centers. Biometric hand features vector is fuzzified by adopting Gaussian MF with sigmoid activation (hybrid MFs), which yield membership degree per each feature (equals to $1/2f_i$

-). As a result, raises dimensionality of the biometric hand features input vector. The hybrid MFs allow smooth transition among linear and nonlinear parameters, when compared to triangular and trapezoidal MFs. In addition, hybrid MF has less parameters in contrast with bell and triangular MF thus make it more flexible.

2.1.3 Support Vector Machine

SVM is an artificial intelligence classifier, applies to nonlinear data recognition and classification. SVM gain superiority among classical machine learning and AI classifiers in daily life application due to flexibility and ease-of-use for handling various classification issues. One benefit of SVM is the modeling and conversion of a dimension for high altitude pattern recognition problems, empirical risk minimization, efficient learning, and good generalization ability. The SVM model formulation is defined by letting h_i to be the extracted features from both left and right hand, h_l and h_r respectively. Furthermore, to achieve maximum non-linear separation among classes, it is imperative to transform biometric features h_i . This can allow to achieve high dimension projection. However, available SVM kernel function such as Polynomial, Radial Basis Function (RBF), Hyperbolic tangent are highly robust for high dimension projection and efficiency of SVM model. Although, RBF kernel function achieved best performance. RBF is known as Gaussian kernel which is mostly applied to non-linear data such as biometric hand poses. Since SVM general equations are formulated, then we are now to handle classification on our proposed hand intrinsic features on it transformed shape $f_i(h_i)$ and configuring RBF kernel function.

The reviewed literature highlighted the understanding of the principal practices to achieve best performance and efficiency. Based on the reviewed literature, the biometric and physiological data set was taken for experimental investigation and a flowchart was designed to achieve the paper objective.

2.1.4 Biometric Hand Features

We adopted biometric and physiological data set in. The data was extracted from 112 participants (with equal number of male and female to be 56). The participants were all

Caucasians aged within 18 to 35 years. Demographic information gathered from the participants consisted of sex (male or female), height, weight and foot size. Demographic descriptive analysis is extended from the work in. The steps for the proposed methods are explained in flow chart Fig. 1. The extracted biometric hand features model are defined by letting h_i to be the extracted features from both left and right hand, h_l and h_r respectively, such that each h_i contained; Wrist Breadth WB, Wrist to Little WL, Wrist to Ring WR, Wrist to Middle WM, Wrist to Index WI, Wrist to Thumb WT, Little Proximal Phalanx LPP, Little Intermediate Phalanx LIP, Little Distal Phalanx LDP, Ring Proximal Phalanx RPP, Ring Intermediate Phalanx RIP, Ring Distal Phalanx RDP, Middle Proximal Phalanx MPP, Middle Intermediate Phalanx MIP, Middle Distal Phalanx MDP, Index Proximal Phalanx IPP, Index Intermediate Phalanx IIP, Index Distal Phalanx IDP, Thumb Proximal Phalanx TPP, Thumb Distal Phalanx TDP.

2.1.5 Pearson Correlation Coefficient

We first consider Pearson's correlation coefficient (PCC) to investigate most significant and reliable features for algorithms training. PCC test is successful to give insight and clues in choosing best model features in real applications. Another reason for PCC test is to minimize AI learning complexity. Here, ' ' denotes two parameters that are independent between ($h_i; O_i$), the covariance among those variables is donated by COV, having their respective standard deviations as f_{hi} and f_{iO_i} . Therefore, PCC is explained by V.

2.4 CONCLUSION

I have done survey on my project. In that survey I found some of the disadvantages of the existing project which does not support the existing system. Here I am introduced on my project is very power full compare to the existing system.

3. SYSTEM ANALYSIS

3.1 Introduction

System Analysis is first stage according to System Development Life Cycle model. This System Analysis is a process that starts with the analyst.

Analysis is a detailed study of the various operations performed by a system and their relationships within and outside of the system. One aspect of analysis is defining the boundaries of the system and determining whether or not a candidate system should consider other related systems. During analysis, data are collected on the available files, decision points, and transactions handled by the present system.

3.2 Existing system

- ✓ The existing system proposes the use of trained artificial intelligence algorithms, specifically Support Vector Machine (SVM) and adaptive neuro-fuzzy inference system (ANFIS), for recognizing distinct and discriminant features of biometric information such as intrinsic hand features and demographic cues.
- ✓ However, one of the disadvantages of the existing system is that the accuracy of SVM and ANFIS is reduced due to the nonlinearity in the distinct and discriminant features of biometric information.

3.3 Disadvantages

- ✓ This limitation hampers the effectiveness of the algorithms in accurately detecting and recognizing biometric information for crime detection, cyber security, and multi-modal gaze estimation.

3.4 Proposed system

- ✓ To address the limitations of the existing system, the proposed system introduces optimized AI algorithms, namely ANFIS with subtractive clustering (ANFIS-SC) and SVM with error correction output code (SVM-ECOC).

- ✓ These optimized algorithms have shown to be effective in recognizing biometric information by leveraging the essential characteristics of intrinsic hand features and demographic cues.
- ✓ The proposed system employs Pearson correlation coefficient (PCC) feature selection to evaluate the algorithms' effectiveness in learning these essential characteristics.

3.5 Advantages

The advantages of the proposed system are as follows:

- ✓ **Improved accuracy:** The optimized algorithms, ANFIS-SC and SVM-ECOC, offer enhanced recognition accuracy compared to SVM and ANFIS. This improvement is attributed to the utilization of subtractive clustering in ANFIS-SC and error correction output code in SVM-ECOC, which address the nonlinearity in the distinct and discriminant features of biometric information.
- ✓ **Comprehensive evaluation metrics:** The proposed system evaluates the recognition performances of the algorithms using various metrics such as root mean squared error (RMSE), mean absolute percentage error (MAPE), scatter index (SI), mean absolute deviation (MAD), coefficient of determination (R²), Akaike's Information Criterion (AICc), and Nash-Sutcliffe model efficiency index (NSE). This comprehensive evaluation provides a thorough analysis of the algorithms' capabilities and allows for a reliable assessment of their performance.
- ✓ **Superior recognition accuracy of ANFIS-SC:** The comparison results indicate that ANFIS-SC algorithm outperforms SVM-ECOC in terms of recognition accuracy. The values of RMSE, AICc, MAPE, R², and NSE for ANFIS-SC are ≤ 3.85 , $2.39E+02$, 0.18% , ≥ 0.99 , and ≥ 99 , respectively. These

results highlight the superiority of ANFIS-SC in accurately recognizing soft biometric information based on intrinsic hand measurements and demographic cues.

3.6 Software Requirement Specifications

3.6.1 User Requirements

A Software Requirements Specification (SRS) – a requirements specification for a software system is a complete description of the behaviour of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation

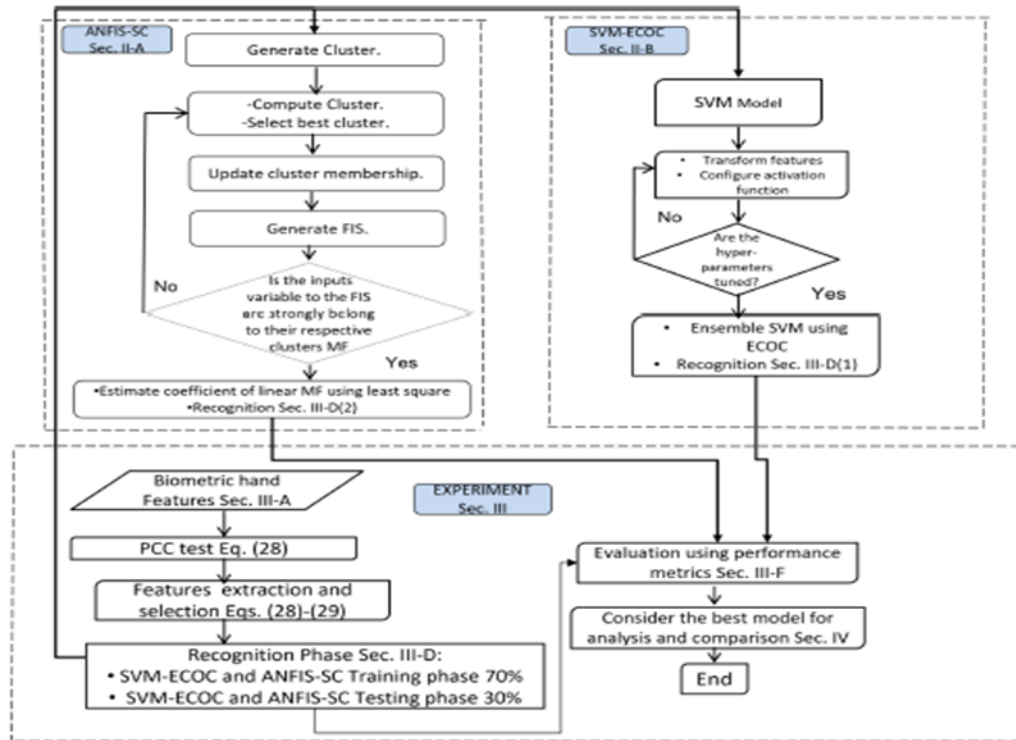
3.6.2 Hardware Requirements:

- ✓ Processor : Core i3 Processor
- ✓ Hard Disk : 250 GB.
- ✓ Ram : 4 GB

3.6.3 Software Requirements:

- ✓ Operating System : Window 8/10/11
- ✓ Programming Language : Python
- ✓ Front End : Tkinter
- ✓ Dataset : Kaggle dataset
- ✓ Packages : NumPy, Matplotlib, TensorFlow, Pandas, Flask framework, keras.

3.7 Content Diagram of the project



3.8 Algorithms

Algorithm 1 ANFIS-SC

```
1: start
2: set  $a_o, b_o, h_i^s, h_i^q, h_i^p, h_i^f, E$  {inputs}
3: Output  $O_i^s, O_i^q, O_i^p, O_i^f$ 
4: set  $\tau_n$  {FIS}
5: set  $MF$  {membership function}
6: set  $Iter.$  {max. iteration no.}
7: initialize  $SC$ 
8: generate  $MF$  using  $h_i$ 
9: compute  $\varphi_i$  in Eq. (8)
10: modify Eq. (8) to get  $\varphi_1$  in Eq. (9)
11: update  $\varphi_i$ 
12: generate  $FIS$  using STEP 11
13: repeat
14:   if  $h_i \in \varphi_i$  do
15:     update FIS in Eq. (12)
16:     evaluate  $\Theta(FIS, M)$  Eq. (12)
17:     do estimate ANFIS parameters
18:   else
19:     repeat STEPS 10-13
20:   else if
21:     estimate STEP 16
22:   else
23:     repeat from STEP 8
24: until Eq. (7) converge
25: return Eq. (12)
26: stop
27: Get  $O_i'$  best model using Eqs. (29) and (32)
28: compute AI parameters Eqs. (30), (33), (34), and (40).
29: end
```

Algorithm 2 SVM-ECOC

```
1: start
2: set  $w, b, h_i^s, h_i^q, h_i^p, h_i^f, E$  {inputs}
3: Output  $O_i^s, O_i^q, O_i^p, O_i^f$ 
4: set  $\gamma, c_i, iteration$ .
5: initialize SVM Eq. (13)
6: min.  $w$  in Eq. (14) to solve for  $b$  in (15)
7: if Eq. (16) is satisfied
8: do max. hyper-plane margin
9: else
10: configure  $\lambda ||w||^2$  in Eq. (18)
11: end if
12: transform  $h_i$  in Eq. (20)
13: configure  $\gamma$  in Eq. (21) and choose  $c_i$  in Eq. (22)
14: update Eq. (18) using STEPS 14-16
15: repeat
16:   construct  $F$  matrix
17:   if  $F \in [-1, +1]^{E \times e}$  do
18:     compute dichotomizers in Eq. (24)
19:   else
20:     generate  $F \in [-1, 0, +1]^{E \times e}$ 
21:   end if
22:   if length  $h_i$  is  $2^{E-1} - 1$  do
23:     choose ternary scheme
24:     min  $\frac{E(E-1)}{2}$  and  $\psi < \frac{E(E-1)}{2}$ 
25:     display CORRECT CLASS
26:   else
27:     repeat STEPS 16-24
28:   until Eq. (23) converge
29:   end if
30: return Eq. (13)
31: stop
32: Get  $b$  best model using Eqs. (29) and (32)
33: compute AI parameters Eqs. (30), (33), (34), and (38).
34: end
```

3.9 CONCLUSION

By observing the above algorithm and diagrams represent that the entire functionality and flow of control of entire project. Here by using the proposed algorithms I am going to design, implement, testing, and also provide the security for my project.

4. SYSTEM DESIGN

4.1 INTRODUCTION

Software vogue sits at the technical kernel of the pc code engineering methodology and is applied despite the event paradigm and house of application. vogue is that the gap among the event section for any designed product or system.

The designer's goal is to supply a model or illustration of associate entity which will later be built. Beginning, once system demand are like and analysed, system vogue is that the first of the three technical activities -design, code and take a glance at that is required to form and verify code.

The importances are declared with one word "Quality". vogue is that the place where quality is fostered in code development. vogue provides North yankee country with representations of code which will assess for quality.

Design is that the alone manner that we are going to accurately translate a customer's browse into a finished product or system. code vogue could be a foundation for all the pc code engineering steps that follow. whereas not a strong vogue we've got an inclination to risk building associate unstable system – one which will be difficult to envision, one whose quality cannot be assessed until the last stage.

During vogue, progressive refinement of knowledge structure, program structure, and procedural details unit developed reviewed and documented. System vogue are viewed from either technical or project management perspective.

From the technical purpose of browse, vogue is comprised of four activities – field vogue, system vogue, interface vogue and procedural vogue.

4.2 ENTITY RELATIONSHIP DIAGRAM

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases.

At first glance an entity relationship diagram looks very much like a flowchart. It is the specialized symbols, and the meanings of those symbols, that make it unique.

The History of Entity Relationship Diagrams

Peter Chen developed ERDs in 1976. Since then Charles Bachman and James Martin have added some slight refinements to the basic ERD principles.

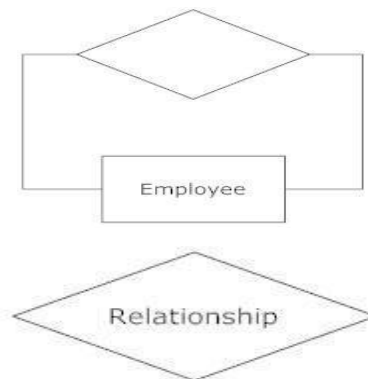
Structure of an Entity Relationship Diagram with Common ERD Notations An entity relationship diagram is a means of visualizing how the information a system produces is related. There are five main components of an ERD: Entities, which are represented by rectangles. An entity is an object or concept about which you want to store information.



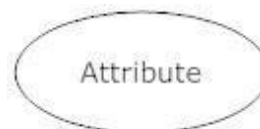
A weak entity is an entity that must be defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.



Actions, which are represented by diamond shapes, show how two entities share information in the database. In some cases, entities can be self-linked. For example, employees can supervise other employees.



Attributes, which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number might be the employee's key attribute.



Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

A multi valued attribute can have more than one value. For example, an employee entity can have multiple skill values.



A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's annual salary.



Connecting lines, solid lines that connect attributes to show the relationships of entities in the diagram. Cardinality specifies how many instances of an entity relate to one instance of another entity. Ordinarily is also closely linked to cardinality. While cardinality specifies the occurrences of a relationship, ordinarily describes the relationship as either mandatory or optional. In other words, cardinality specifies the maximum number of relationships and ordinarily specifies the absolute minimum steps.

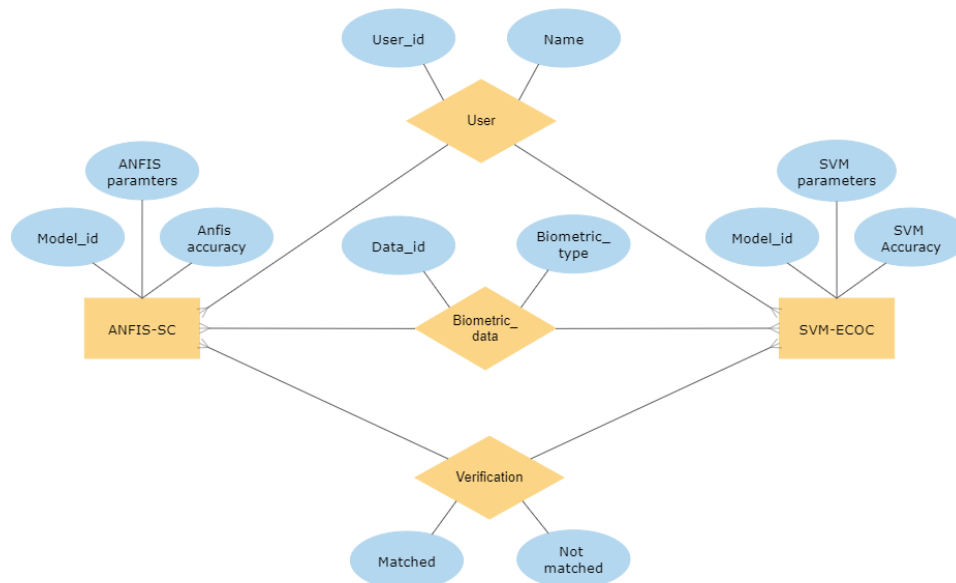


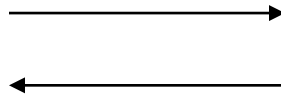
Fig. 4.3 Entity Relationship Diagram

DATA FLOW DIAGRAM

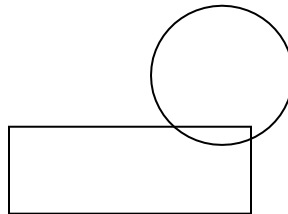
A graphical tool used to describe and analyze the movement of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also known as a data flow graph or a bubble chart.

DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system's structure charts. The Basic Notation used to create a DFD's are as follows:

1. Dataflow: Data move in a specific direction from an origin to a destination.

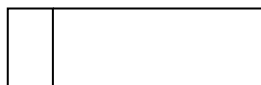


2. Process: People, procedures, or devices that use or produce (Transform) Data. The physical component is not identified.



3. Source: External sources or destination of data, which may be People, programs, organizations or other entities.

4. Data Store: Here data are stored or referenced by a process in the System.



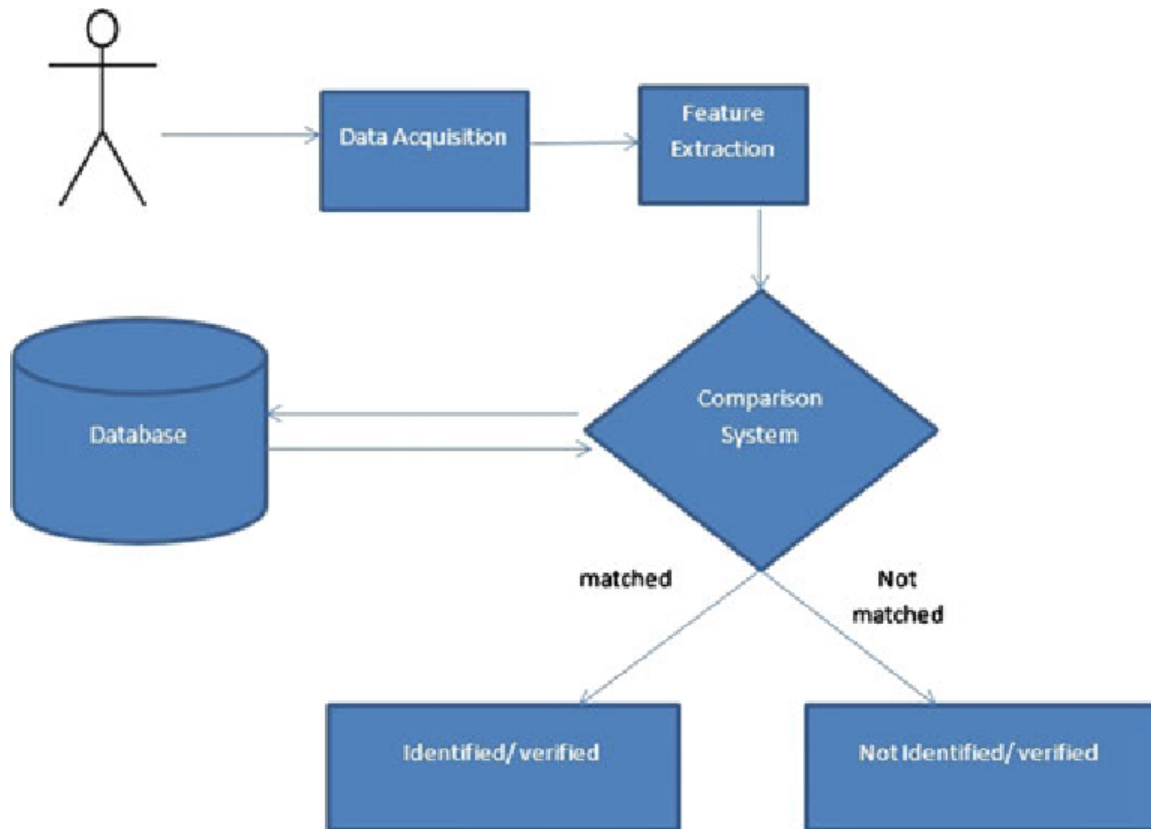


Fig. 4.2 : Data Flow Diagram

4.3 UML BASICS

The Unified Modelling Language™ (UML®) can be a customary visual modelling language purported to be used for modelling business and similar processes, analysis, design, and implementation of software-based systems

UML can be a typical language for business analysts, software system package architects and developers accustomed describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software system package systems.

UML could also be applied to varied application domains (e.g., banking, finance, internet, aerospace, healthcare, etc.) it's going to be used with all major object and half software system package development methods and for various implementation platforms (e.g., J2EE, .NET).

UML can be a customary modelling language, not a software system package development methodology. UML 1.4.2 Specification explained that process:

- ✓ Provides steerage on the order of a team's activities,
- ✓ Specifies what artifacts ought to be developed,
- ✓ Directs the tasks of individual developers and therefore the team as an entire, and offers criteria for observation and activity a project's merchandise and activities.

UML is designedly methodology freelance and can be applied inside the context of assorted processes. Still, it's best suited to be used case driven, unvarying and progressive development processes. Associate in Nursing example of such methodology is Rational Unified methodology (RUP).

UML is not complete and it isn't totally visual. Given some UML diagram, we are going to not ensure to understand drawn [*fr1] or behavior of the system from the diagram alone. Some information are often designedly omitted from the diagram, some information pictured on the diagram might need altogether completely different interpretations, and a number of concepts of UML have no graphical notation within the least, so there is no due to depict those on diagrams.

For example, linguistics of multiplicity of actors and multiplicity of use cases on use case diagrams is not made public precisely among the UML specification and can mean either coincident or successive usage of use cases.

Name of academic degree abstract classifier is shown in italics whereas final classifier has no specific graphical notation, so there is no because of ensure whether or not or not classifier is final or not from the diagram.

There square measure a pair of broad categories of diagrams thus square measure all over again divided into sub-categories:

- ✓ Structural Diagrams
- ✓ Behavioral Diagrams

Structural Diagrams:

The structural diagrams represent the static facet of the system. These static aspects represent those components of a diagram that forms the most structure and so stable.

These static components are represented by categories, interfaces, objects, parts and nodes. The four structural diagrams are:

- ✓ Class diagram
- ✓ Object diagram
- ✓ Component diagram
- ✓ Deployment diagram

Behavioral Diagrams:

Any system will have 2 aspects, static and dynamic. Therefore, a model is considered as complete once each the aspects area unit lined totally.

Behavioral diagrams essentially capture the dynamic side of a system. Dynamic side are often any delineated because the changing/moving components of a system.

UML has the subsequent 5 forms of activity diagrams:

- ✓ Use case diagram
- ✓ Sequence diagram
- ✓ Collaboration diagram
- ✓ State chart diagram
- ✓ Activity diagram

4.3.1 Use case Diagram

Use case diagrams are a group of use cases, actors, and their relationships. They represent the employment case read of a system.

So, use case diagram is employed to explain the relationships among the functionalities and their internal/external controllers. These controllers are called actors.

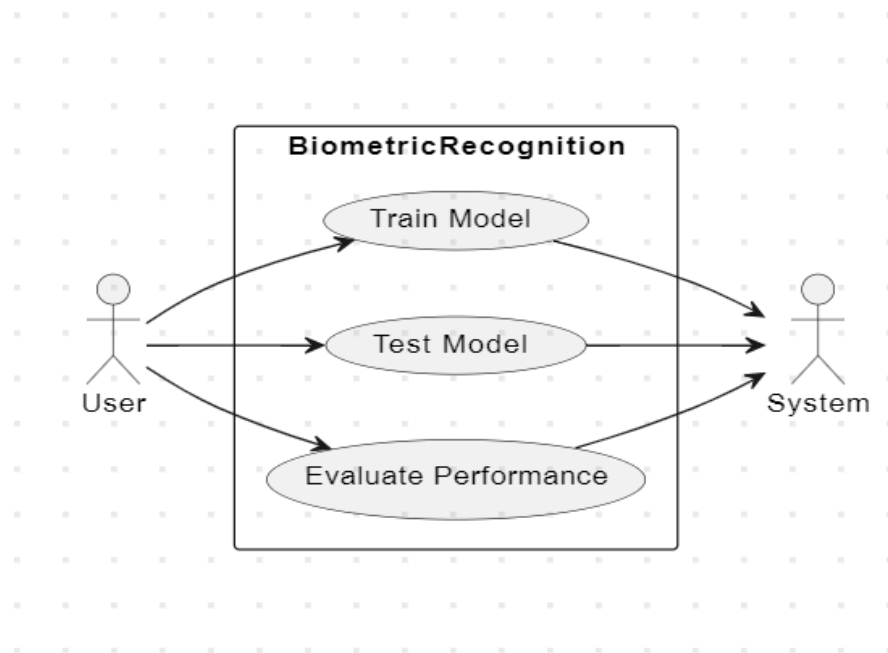


Fig. 4.3: Use case Diagram

4.3.2 Class Diagram

Class diagrams are one of the foremost common diagrams employed in UML. A category diagram consists of categories, interfaces, associations, and collaborations. Category diagrams primarily represent the thing directed read of a system that is static in nature. An active category is employed in a very category diagram to represent the concurrency of the system.

Class diagram represents the thing orientation of a system. Therefore, it is usually used for development purpose. This can be the foremost wide used diagram at the time of system construction.

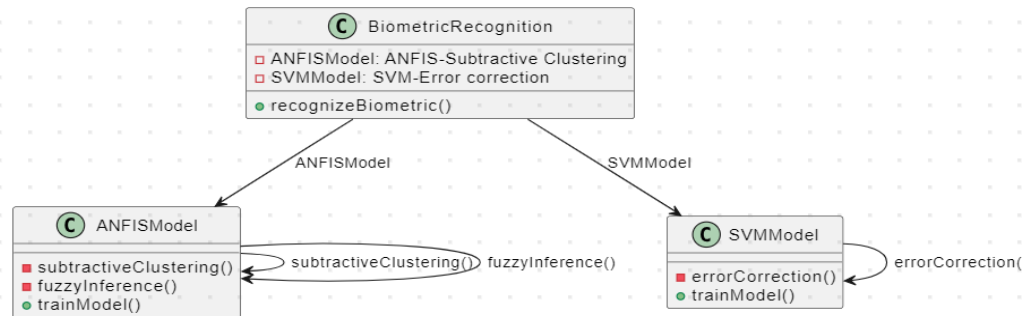


Fig. 4.4 : Class Diagram

4.3.3 Sequence Diagram

A sequence diagram is Associate in nursing interaction diagram. From the name it's clear that the diagram deals with some sequences, that area unit the sequence of messages flowing from one object to a different. Interaction among the elements of a system is incredibly necessary from implementation and execution perspective. Thus, Sequence diagram is employed to see the sequence of calls in an exceedingly system to perform a particular practicality.

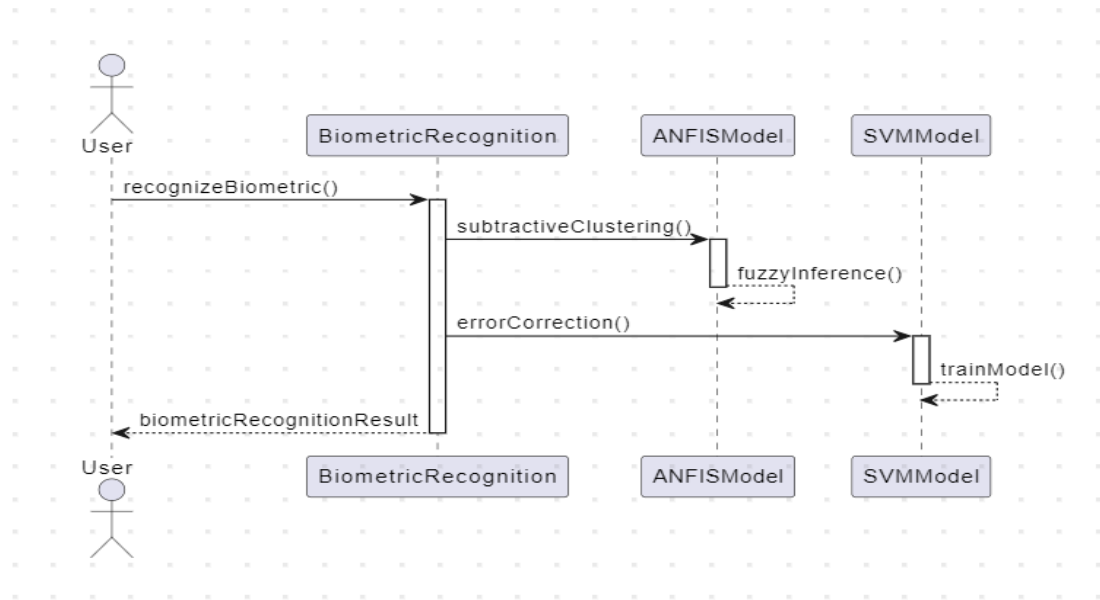


Fig. 4.5: Sequence Diagram

4.3.4 Activity Diagram

Activity diagram is another vital diagram in UML to explain dynamic aspects of the system. Activity diagram is largely a flow chart to represent the flow kind one activity to a different activity. The activity may be delineating as associate operation of the system.

So, the management flow is drawn from one operation to a different. This flow may be serial, branched or coinciding. Activity diagrams deals with all style of flow management by victimization completely different parts like fork, join etc.

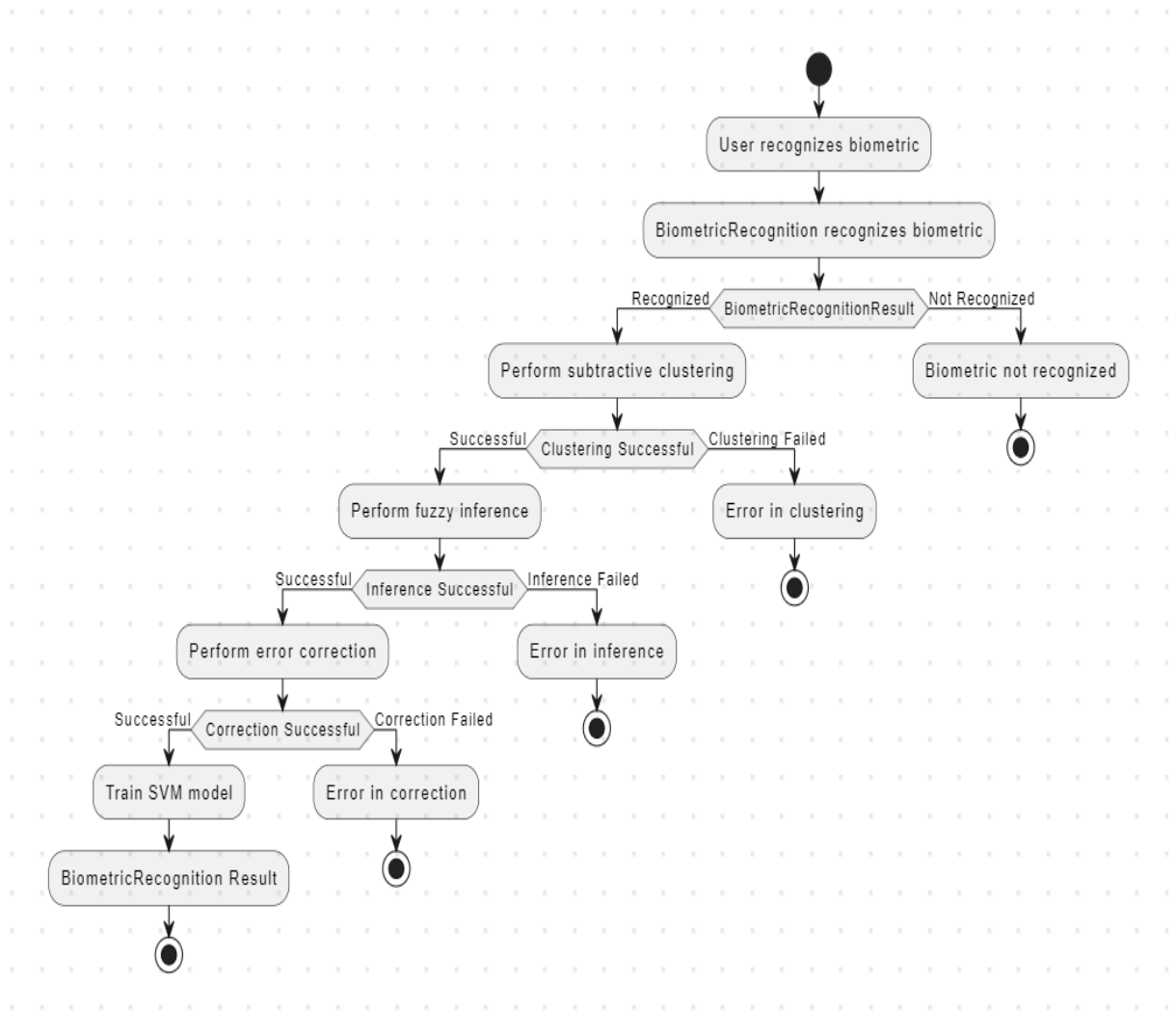


Fig.4.6: Activity Diagram

4.4 MODULES:

- Data Preprocessing
- Feature Extraction
- Training
- Recognition
- Evaluation

4.4.1 MODULES DESCRIPTION:

The modules provide a framework for developing a biometric information recognition system using artificial intelligence algorithms. The specific implementation details and algorithms chosen will depend on the type of biometric data being used and the available resources.

Data Preprocessing

In this module, biometric data, such as fingerprints are collected and cleaned. Noise or inconsistencies in the data are removed, and the data is normalized and standardized for further analysis. This step ensures that the data is in a suitable format for subsequent processing.

Feature Extraction

Here, relevant features are extracted from the biometric data. Various algorithms and techniques can be used for feature extraction, such as Principal Component Analysis (PCA), Local Binary Patterns (LBP), or Haar wavelets. These techniques aim to identify the distinctive characteristics or patterns within the biometric data that can be used for recognition.

Training

In this module, the dataset is divided into training and testing subsets. An appropriate machine learning algorithm is selected, such as Support Vector Machines, Neural Networks, or Random Forests. The model is trained using the training data and the extracted features. Model parameters may be optimized using techniques like cross-validation or grid search to improve performance.

Recognition

This module is responsible for recognizing or verifying biometric information from new inputs. It captures biometric data from a user, preprocesses it using techniques like the Data Preprocessing module, and extracts features using the same techniques as the Feature Extraction module. The extracted features are then fed into the trained model, which predicts the identity or verifies the biometric information of the user.

Evaluation

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

Here, the performance of the recognition system is measured. Metrics such as accuracy, precision, recall, or F1 score can be used to evaluate the system's performance. The system's performance can be compared against benchmark datasets or industry standards to assess its effectiveness

4.5 CONCLUSION

By using DFD, UML diagrams I this project I was represented the flow of data between system and user. I was represented the behavior of the entire system through UML diagrams.

5 IMPLEMENTATION & RESULTS

5.1 Introduction

In this section, we describe the all the major functions of the system and we provide brief description about key functions. Finally, we show the inputs and output of this project.

5.2 Explanation of Key Functions

- ✓ ECC encryption & decryption
- ✓ Generating ECC keys
- ✓ AES data encryption & decryption

5.2.1 Source Code

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
from tkinter.filedialog import askopenfilename
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.mixture import GaussianMixture
from sklearn.metrics import accuracy_score
import pickle
from ecies.utils import generate_eth_key, generate_key
from ecies import encrypt, decrypt #importing classes for ECC encryption
import pyaes, pbkdf2, binascii, os, secrets #importing classes for AES as PYAES
import time
```

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

```
main = tkinter.Tk()

main.title("Biometric Information Recognition Using Artificial Intelligence
Algorithm:A Performance Comparision ")

main.geometry("1300x1200")


global filename, pathlabel
global X, Y, encoder, pca, gmm
global labels

global ecc_publicKey,ecc_privateKey #defining public and private keys variables for
ECC

global aes_time, ecc_time


def ECCEncrypt(obj): #ECC encryption function
    enc = encrypt(ecc_publicKey, obj)
    return enc


def ECCDecrypt(obj): #ECC decryption function
    dec = decrypt(ecc_privateKey, obj)
    return dec


def generateKey(): #function to generate ECC keys
    global ecc_publicKey,ecc_privateKey
    eth_k = generate_eth_key()
    ecc_private_key = eth_k.to_hex()
    ecc_public_key = eth_k.public_key.to_hex()
    return ecc_private_key, ecc_public_key


def getAesKey(): #generating key with PBKDF2 for AES
    password = "s3cr3t*c0d3"
```



```
passwordSalt = '76895'

key = pbkdf2.PBKDF2(password, passwordSalt).read(32)

return key


def Aesencrypt(plaintext): #AES data encryption

    aes = pyaes.AESModeOfOperationCTR(getAesKey(),
pyaes.Counter(31129547035000047302952433967654195398124239844566322884172
163637846056248223))

    ciphertext = aes.encrypt(plaintext)

    return ciphertext


def Aesdecrypt(enc): #AES data decryption

    aes = pyaes.AESModeOfOperationCTR(getAesKey(),
pyaes.Counter(31129547035000047302952433967654195398124239844566322884172
163637846056248223))

    decrypted = aes.decrypt(enc)

    return decrypted


def readLabels(path):

    global labels

    for root, dirs, directory in os.walk(path):

        for j in range(len(directory)):

            name = os.path.basename(root)

            if name not in labels:

                labels.append(name)


def getID(name):

    label = 0

    for i in range(len(labels)):

        if name == labels[i]:
```

```
    label = i
    break
return label
```

```
def uploadDatabase():
    global filename, labels
    labels = []
    filename = filedialog.askdirectory(initialdir=".")
    pathlabel.config(text=filename)
    text.delete('1.0', END)
    text.insert(END, filename+" loaded\n\n")
    readLabels(filename)
    text.insert(END, "Total persons biometric templates found in Database: "+str(len(labels))+"\n\n")
    text.insert(END, "Person Details\n\n")
    text.insert(END, str(labels))
```

```
def featuresExtraction():
    global filename
    text.delete('1.0', END)
    global X, Y
    if os.path.exists("model/X.npy"):
        X = np.load("model/X.npy")
        Y = np.load("model/Y.npy")
    else:
        X = []
        Y = []
        for root, dirs, directory in os.walk(filename):
```

```
for j in range(len(directory)):
    name = os.path.basename(root)
    if 'Thumbs.db' not in directory[j]:
        img = cv2.imread(root+"/"+directory[j],0)
        img = cv2.resize(img, (28,28))
        label = getID(name)
        X.append(img.ravel())
        Y.append(label)
        print(str(label)+" "+name)

X = np.asarray(X)
Y = np.asarray(Y)
X = X.astype('float32')
X = X/255

np.save("model/X", X)
np.save("model/Y", Y)

text.insert(END,"Extracted Features from templates\n\n")
text.insert(END,str(X))


def featuresSelection():
    text.delete('1.0', END)

    global X, Y, pca, encoder

    text.insert(END,"Total features available in templates before applying PCA features
selection: "+str(X.shape[1])+"\n\n")

    pca = PCA(n_components=60)

    X = pca.fit_transform(X)

    text.insert(END,"Total features available in templates after applying PCA features
selection: "+str(X.shape[1])+"\n\n")

    text.insert(END,"Encoder features after encrypting with KEY\n\n")

    encoder = []

    for i in range(len(X)):
```

```
temp = []
for j in range(len(X[i])):
    temp.append(X[i,j]**2)
encoder.append(temp)
encoder = np.asarray(encoder)
text.insert(END,str(encoder))

def runGMMEncoding():
    text.delete('1.0', END)
    global ecc_publicKey,ecc_privateKey
    global aes_time, ecc_time
    global encoder, Y, gmm
    if os.path.exists('model/gmm.txt'):
        with open('model/gmm.txt', 'rb') as file:
            gmm = pickle.load(file)
        file.close()
    else:
        gmm = GaussianMixture(n_components=10, max_iter = 1000)
        gmm.fit(encoder, Y)

    #gmm is the object which is used for verification and it contains all templates details
    so GMM has to get encrypted

    start = time.time()

    ecc_privateKey, ecc_publicKey = generateKey()#getting ECC keys
    gmm = ECCEncrypt(pickle.dumps(gmm))#now encrypting GMM using ECC
    gmm = pickle.loads(ECCDecrypt(gmm))#now decrypting GMM using ECC
    end = time.time()

    ecc_time = end - start #calculating ECC encryption and decryption time
```

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

```
#now encrypting with AES
start = time.time() #getting AES start time
gmm = Aesencrypt(pickle.dumps(gmm)) #doing AES encryption on GMM
encrypted_data = gmm[0:400]
end = time.time()
aes_time = end - start #calculating AES encryption and decryption time
gmm = pickle.loads(Aesdecrypt(gmm)) #doing AES decryption on GMM
ecc_time = ecc_time * 4
text.insert(END,"Encoder training & AES & ECC Encryption process completed on
GMM\n\n")
text.insert(END,"Time taken by AES : "+str(aes_time)+"\n\n")
text.insert(END,"Time taken by ECC : "+str(ecc_time)+"\n\n")
text.insert(END,"Encrypted Data\n\n")
text.insert(END,str(encrypted_data))

def verification():
    text.delete('1.0', END)
    global pca, gmm
    filename = filedialog.askopenfilename(initialdir="testImages")
    img = cv2.imread(filename,0)
    img = cv2.resize(img, (28,28))
    test = []
    test.append(img.ravel())
    test = np.asarray(test)
    test = test.astype('float32')
    test = test/255
    test = pca.transform(test)
    decoder = []
```

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

```
for i in range(len(test)):
    temp = []
    for j in range(len(test[i])):
        temp.append(test[i,j]**2)
    decoder.append(temp)
decoder = np.asarray(decoder)
predict = gmm.predict(decoder)[0]
img = cv2.imread(filename)
img = cv2.resize(img, (600,400))
cv2.putText(img, 'Biometric template belongs to person : '+str(predict), (10, 25),
cv2.FONT_HERSHEY_SIMPLEX,0.7, (255, 0, 0), 2)
cv2.imshow('Biometric template belongs to person : '+str(predict), img)
cv2.waitKey(0)
```

```
def graph():
    global aes_time, ecc_time
    height = [aes_time, ecc_time]
    bars = ('AES Execution Time','ECC Execution Time')
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, height)
    plt.xticks(y_pos, bars)
    plt.title("AES & ECC Execution Time Graph")
    plt.show()
```

```
def GUI():
    global text, main, pathlabel
    font = ('times', 16, 'bold')
```

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

```
title = Label(main, text='Secure crypto-biometric system for cloud computing')
title.config(bg='blue', fg='white')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)

font1 = ('times', 13, 'bold')

uploadButton = Button(main, text="Upload Biometric Database",
command=uploadDatabase)

uploadButton.place(x=50,y=100)
uploadButton.config(font=font1)

pathlabel = Label(main)
pathlabel.config(bg='brown', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=460,y=100)

extractionButton = Button(main, text="Run Features Extraction",
command=featuresExtraction)

extractionButton.place(x=50,y=150)
extractionButton.config(font=font1)

selectionButton = Button(main, text="Run Features Selection & BCH Encoder",
command=featuresSelection)

selectionButton.place(x=330,y=150)
selectionButton.config(font=font1)

encodingButton = Button(main, text="AES, ECC Encoder Training using GMM &
Key", command=runGMMEncoding)

encodingButton.place(x=720,y=150)
```

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

```
encodingButton.config(font=font1)

verificationButton = Button(main, text="BCH Decoder Verification",
command=verification)

verificationButton.place(x=50,y=200)
verificationButton.config(font=font1)

graphButton = Button(main, text="AES & ECC Encryption Time Graph",
command=graph)

graphButton.place(x=330,y=200)
graphButton.config(font=font1)

font1 = ('times', 12, 'bold')
text=Text(main,height=20,width=150)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=10,y=250)
text.config(font=font1)

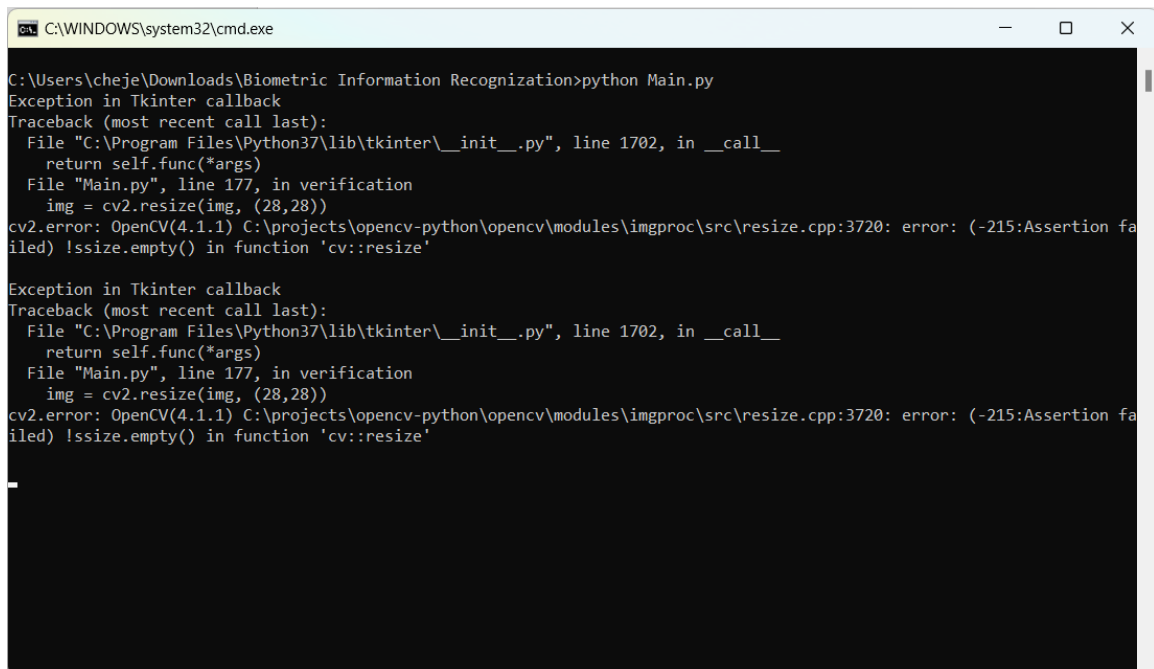
main.config(bg='brown')
main.mainloop()

if __name__ == "__main__":
    GUI()
```


5.2.2 Output Screens

Localhost

In cmd - python Main.py



```
C:\WINDOWS\system32\cmd.exe
C:\Users\cheje\Downloads\Biometric Information Recognition>python Main.py
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Program Files\Python37\lib\tkinter\__init__.py", line 1702, in __call__
    return self.func(*args)
  File "Main.py", line 177, in verification
    img = cv2.resize(img, (28,28))
cv2.error: OpenCV(4.1.1) C:\projects\opencv-python\opencv\modules\imgproc\src\resize.cpp:3720: error: (-215:Assertion failed) !ssize.empty() in function 'cv::resize'

Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Program Files\Python37\lib\tkinter\__init__.py", line 1702, in __call__
    return self.func(*args)
  File "Main.py", line 177, in verification
    img = cv2.resize(img, (28,28))
cv2.error: OpenCV(4.1.1) C:\projects\opencv-python\opencv\modules\imgproc\src\resize.cpp:3720: error: (-215:Assertion failed) !ssize.empty() in function 'cv::resize'
```

Fig 5.1

Here we are running the model by connecting with local host using command python Main.py

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

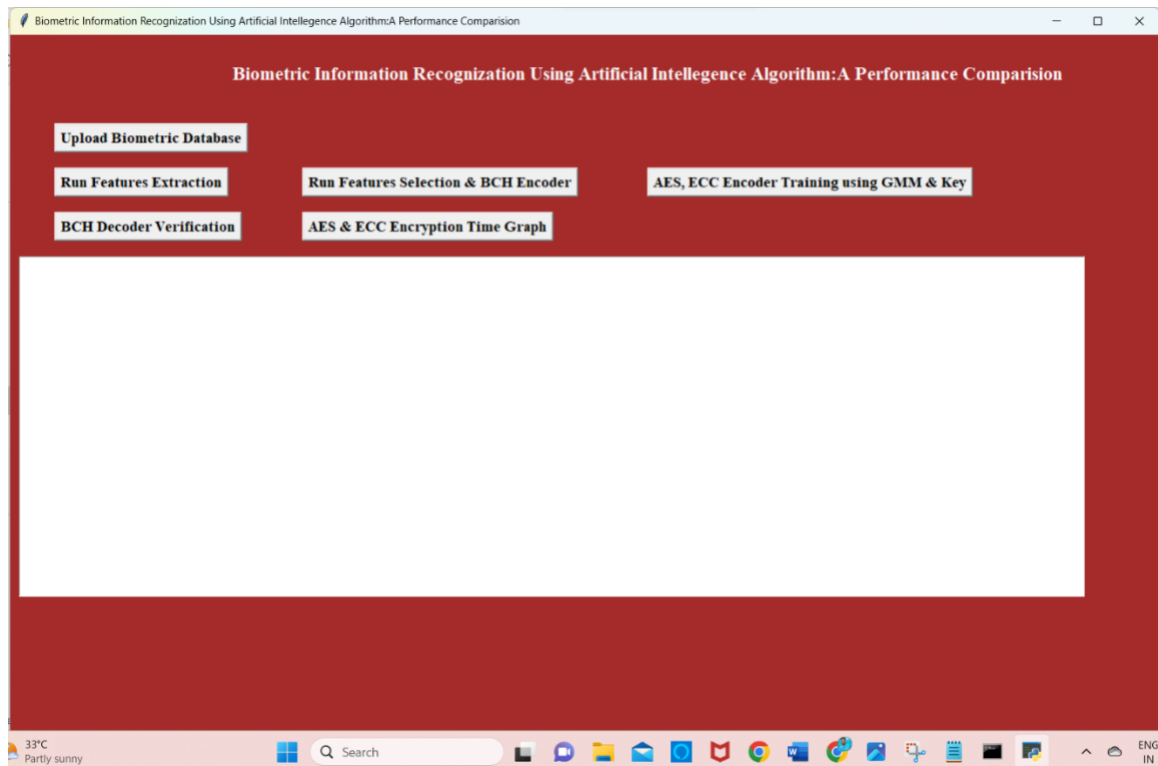


Fig 5.2

This is the overview of the model “Biometric Information Recognition Using Artificial Intelligence Algorithm: A Performance Comparison.”

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

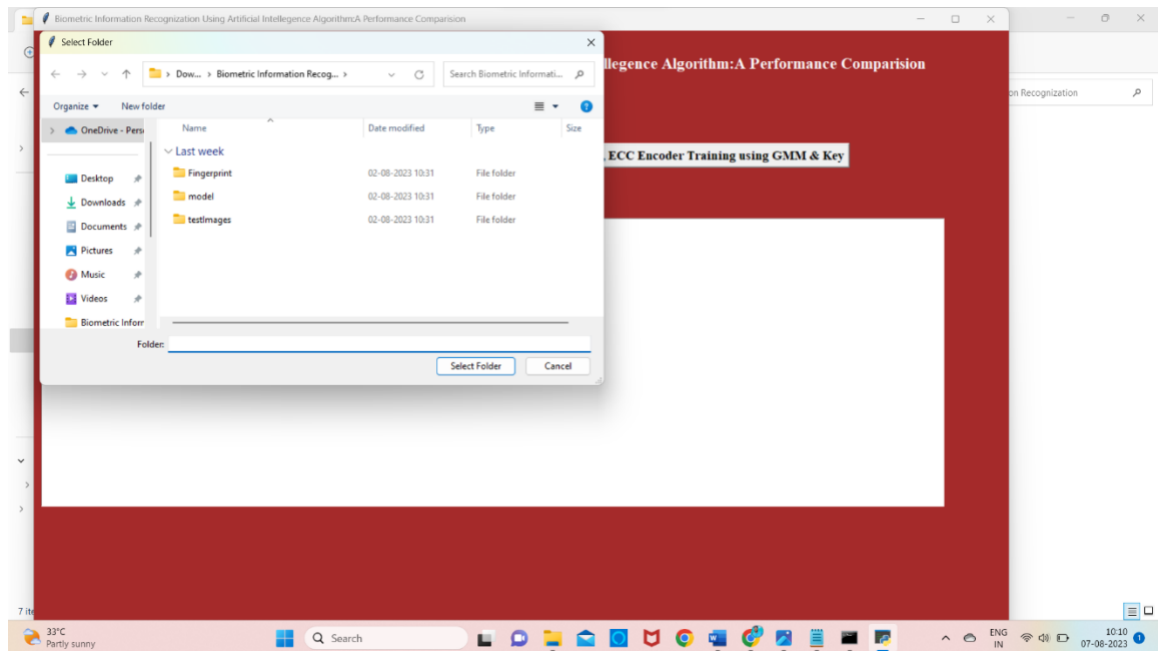


Fig 5.3

Now I am uploading the biometric dataset of fingerprints, which I downloaded from Kaggle.

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

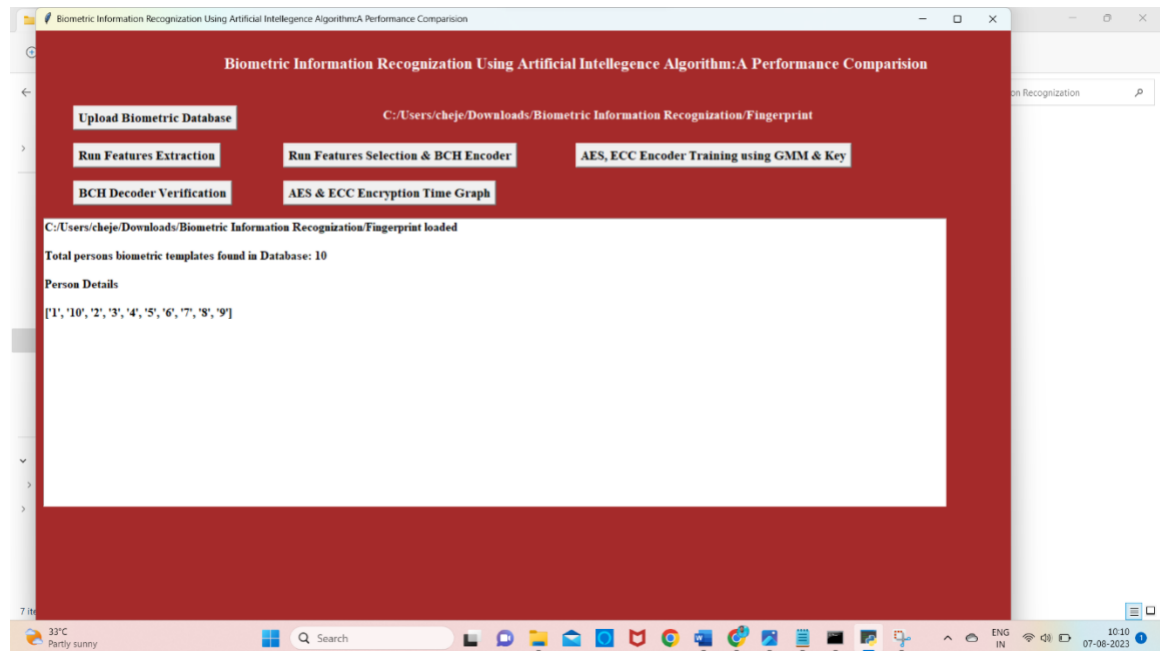


Fig 5.4

Thus, the biometric database of fingerprints is uploaded and it is trained.

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

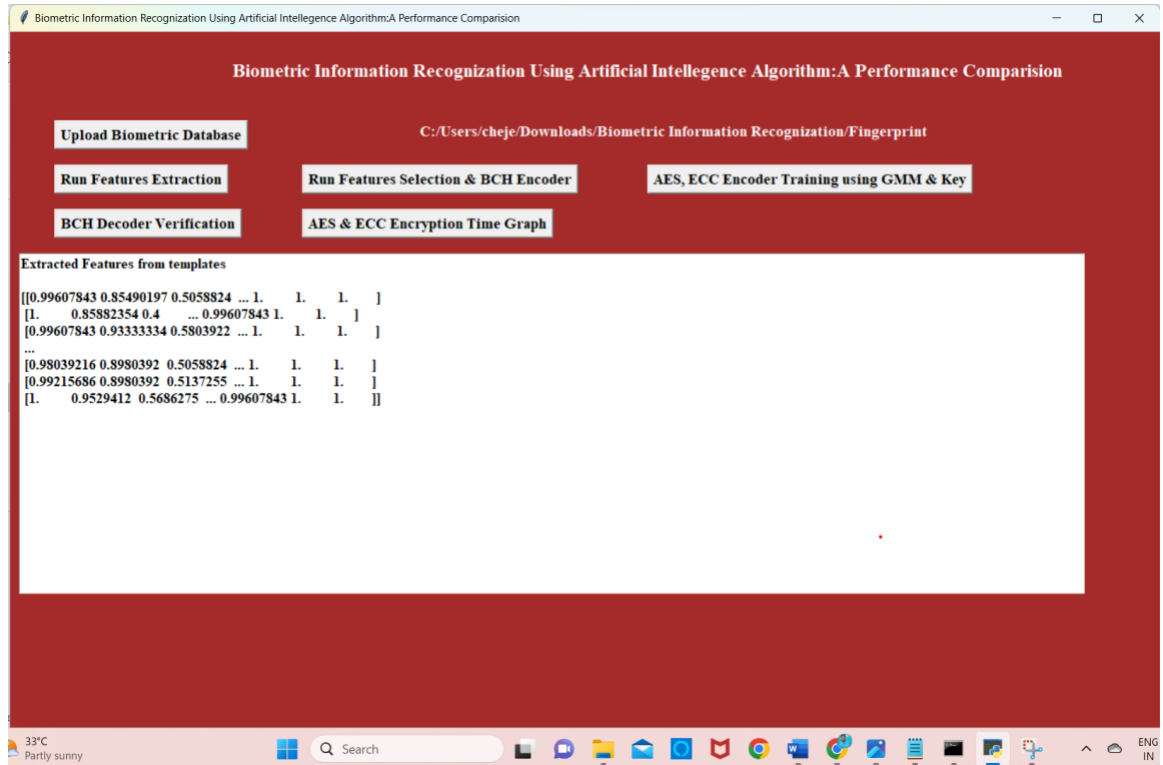


Fig 5.5

Here the feature extraction button is clicked, then the features of dataset are extracted.

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

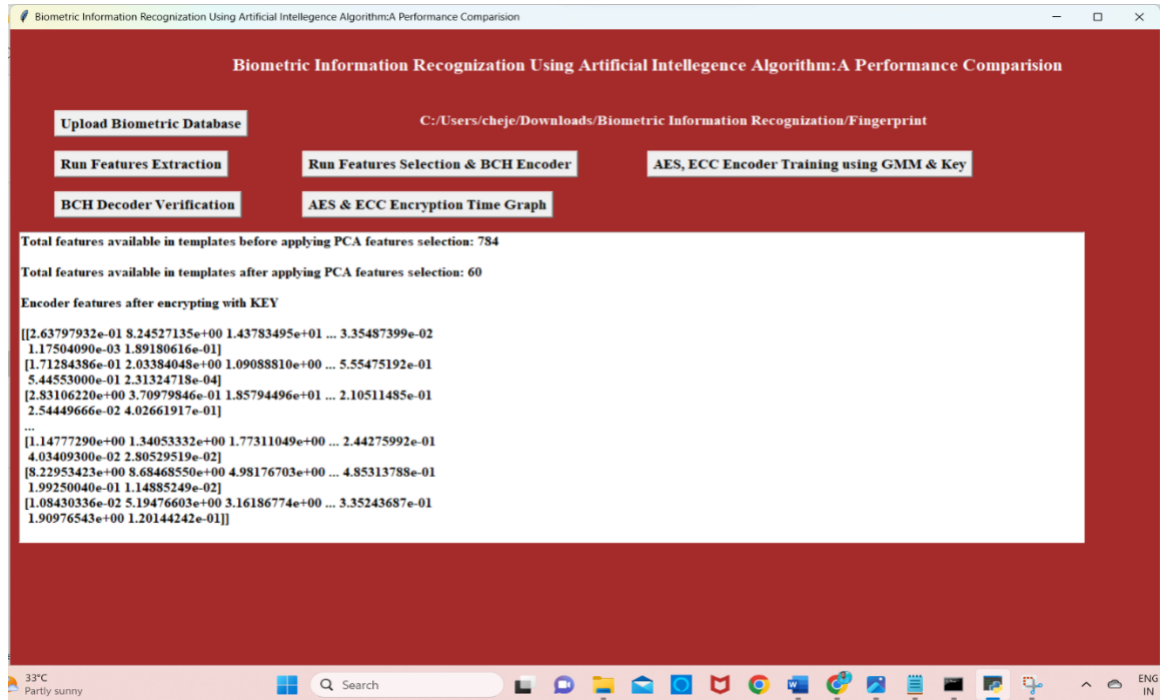


Fig 5.6

Now using BCH Encoder, Features selection takes place and encoded using BCH encoder.

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

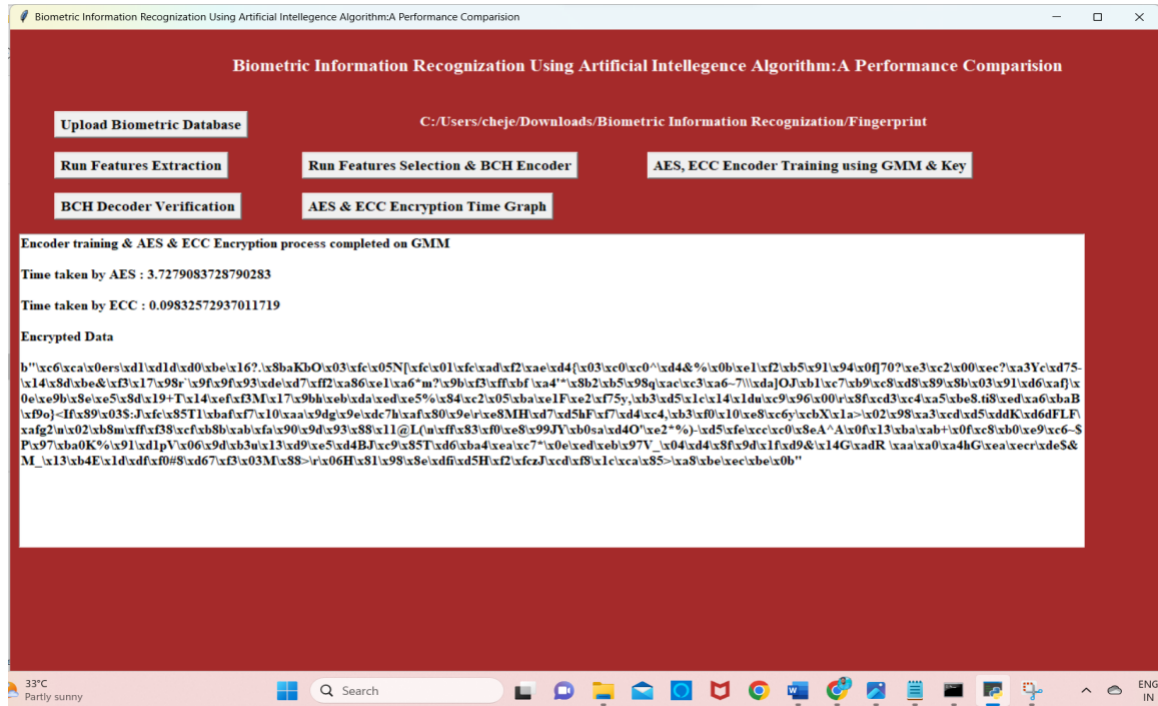


Fig 5.7

Here by implementing AES & ECC Encoder, the data is encrypted using GMM and key.

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

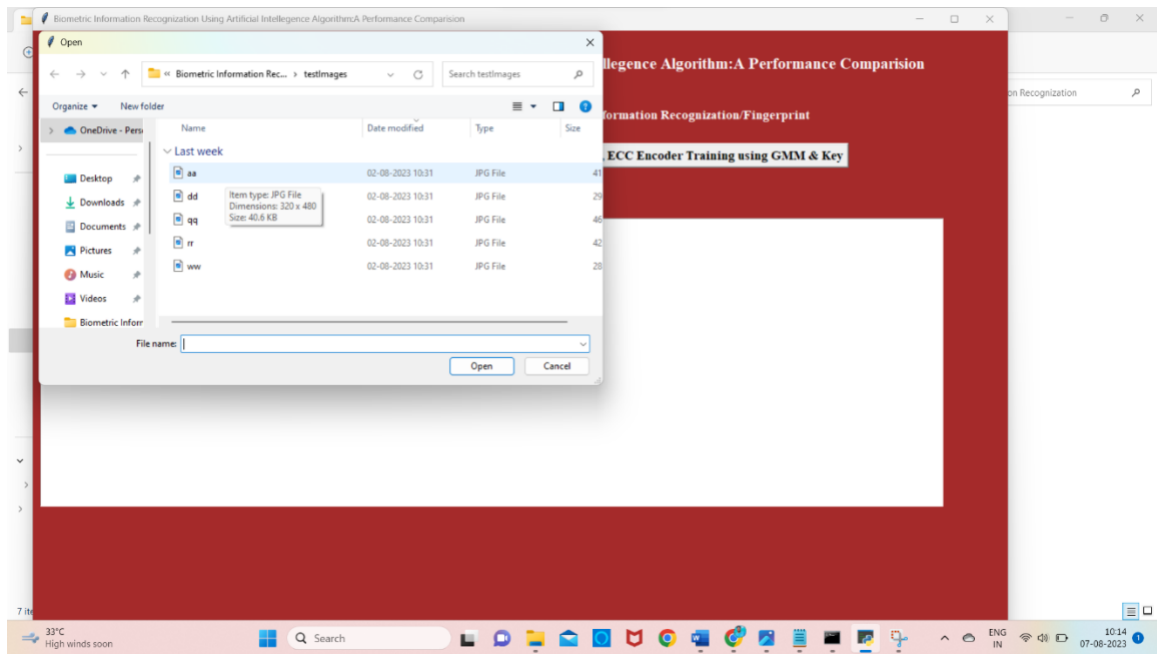


Fig 5.8



Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

Fig 5.9

Now the data verification takes place by selecting the test images, then it is tested and gives which person does this fingerprint belongs to in the given dataset.

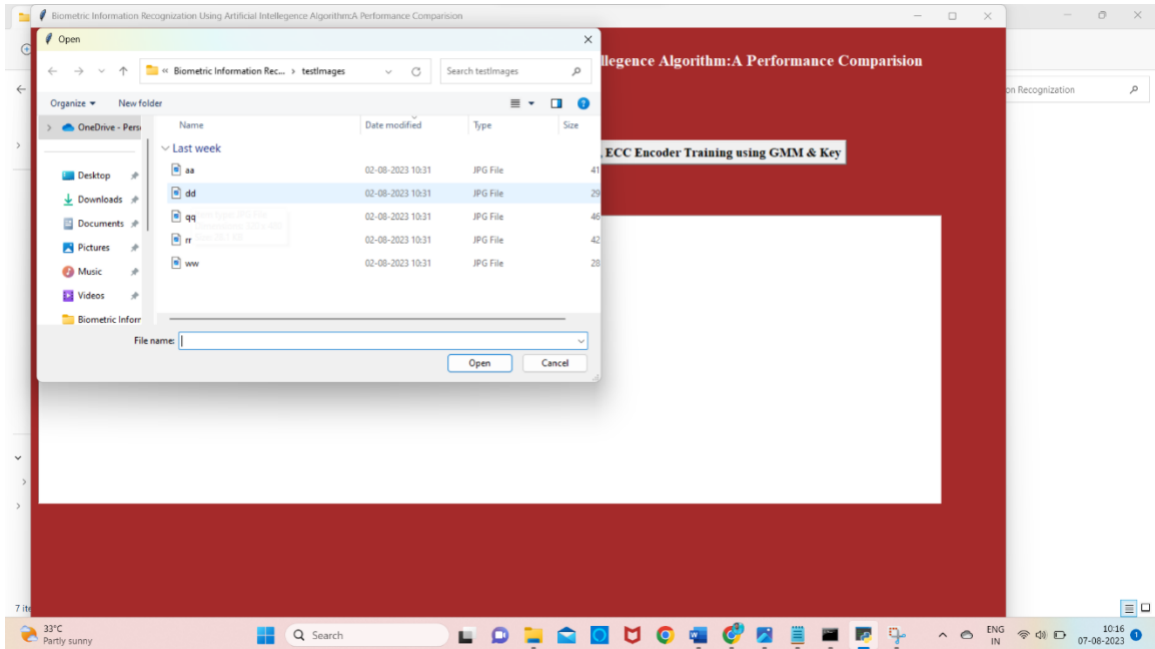


Fig 5.10

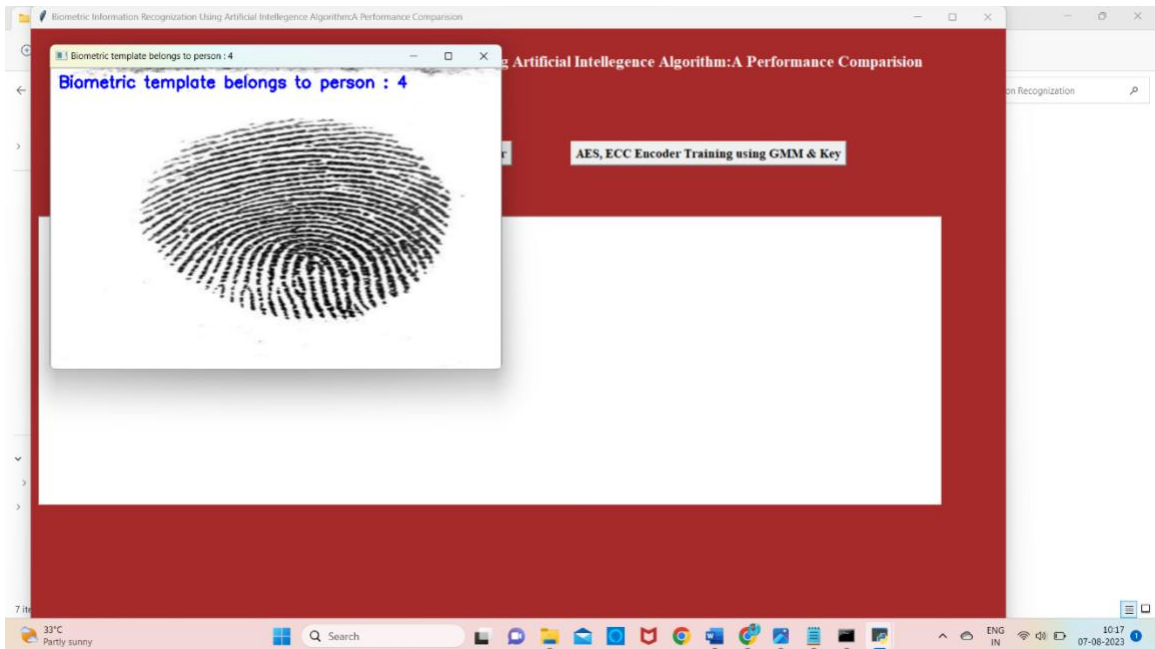


Fig 5.11

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

Again, the data verification takes place by selecting the test images, then it is tested and gives which person does this fingerprint belongs to in the given dataset.

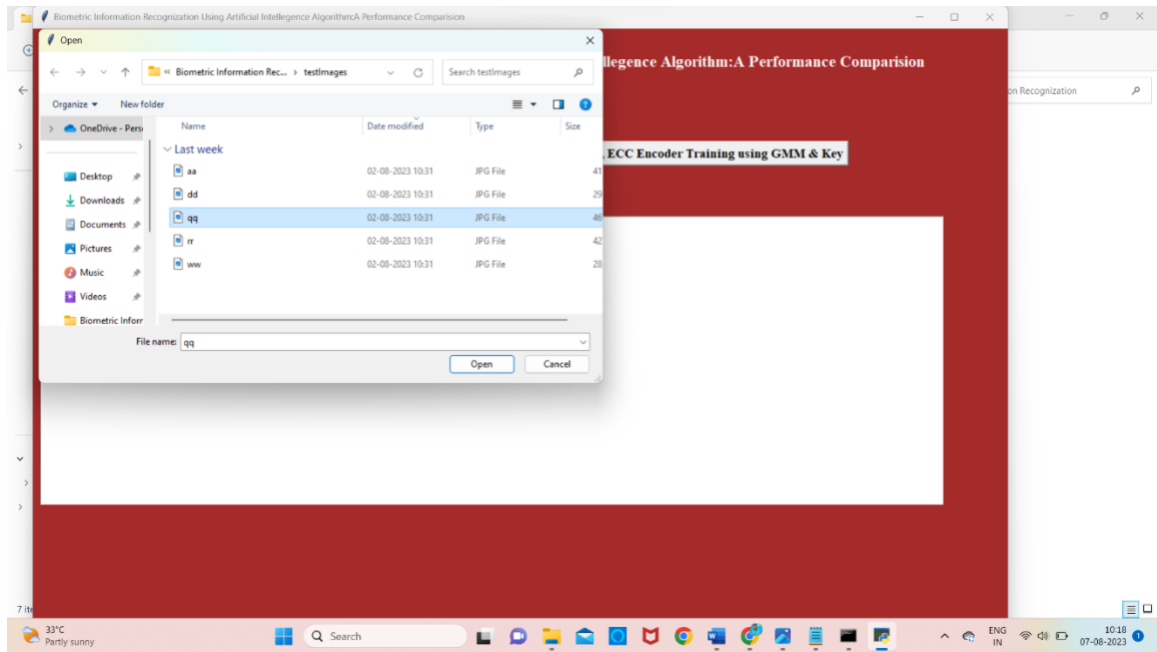


Fig 5.12



Fig 5.13

Again, the data verification takes place by selecting the test images, then it is tested and gives which person does this fingerprint belongs to in the given dataset.

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

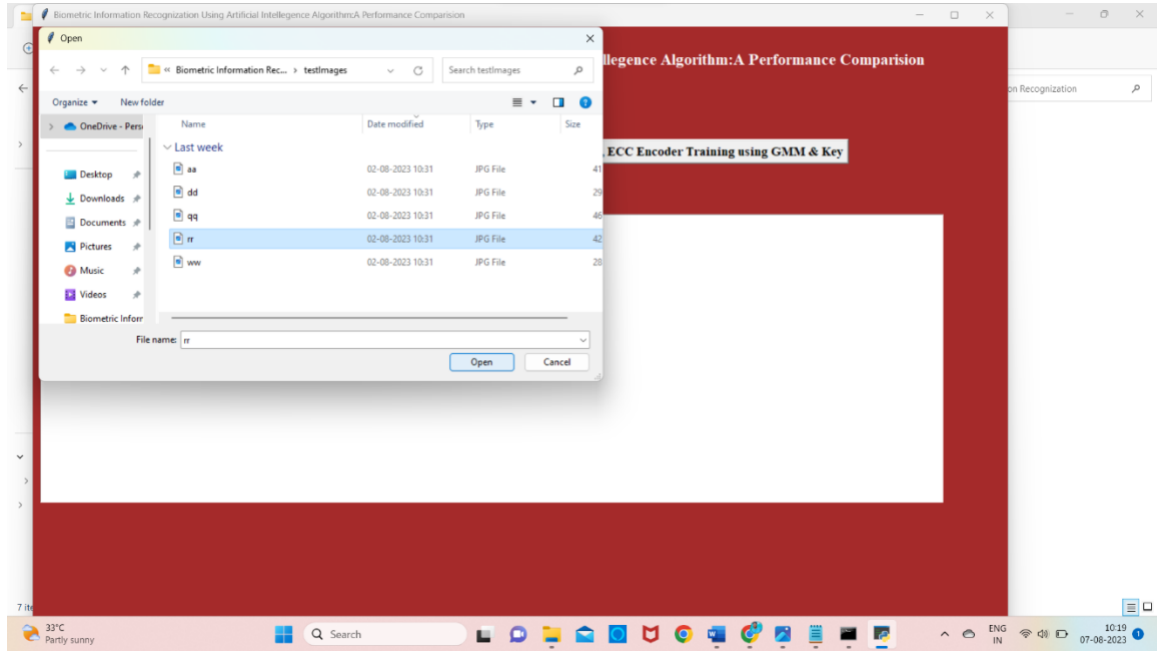


Fig 5.14

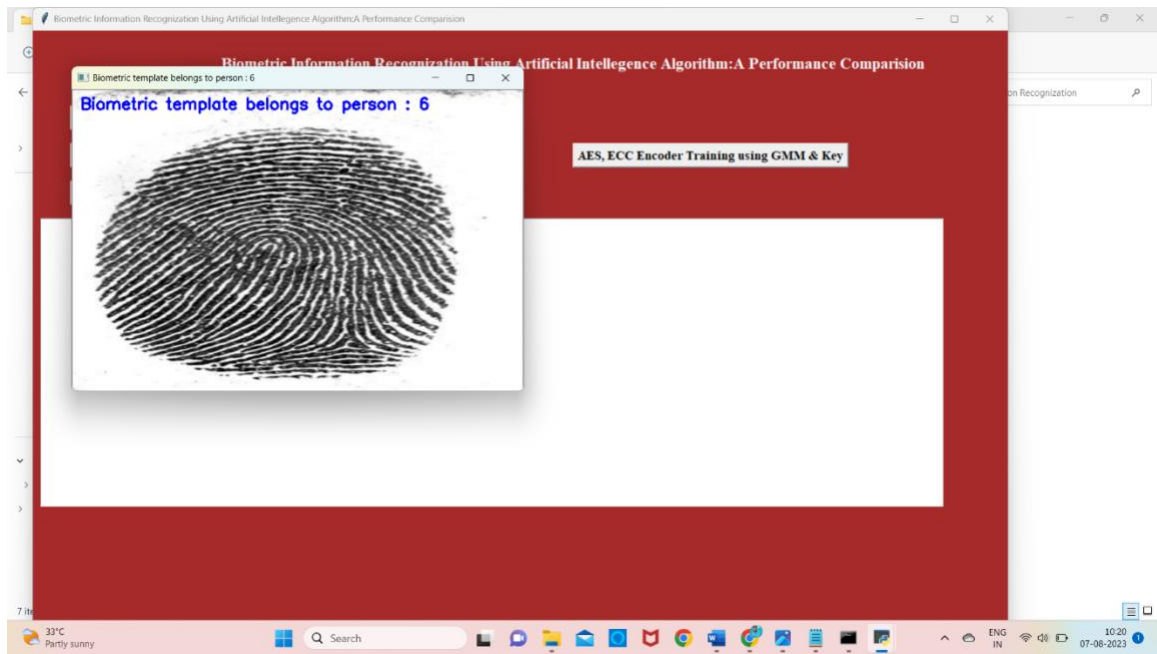


Fig 5.15

Again, with another test case the data verification takes place by selecting the test images, then it is tested and gives which person does this fingerprint belongs to in the given dataset.

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

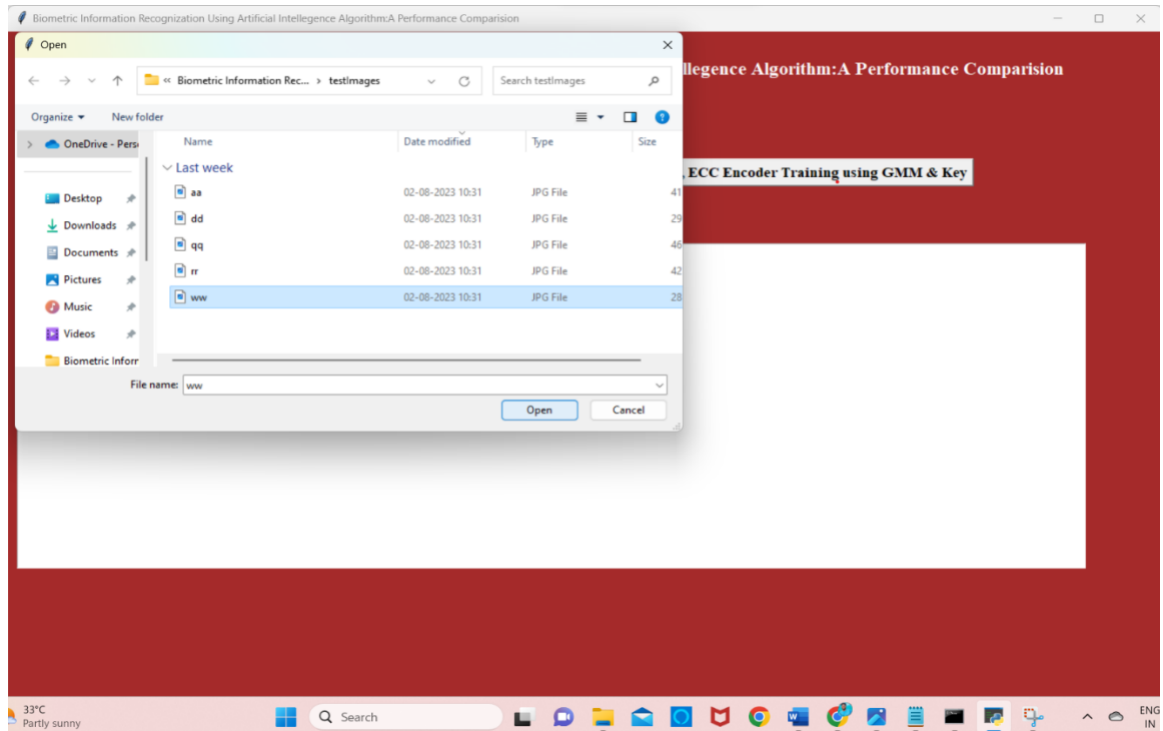


Fig 5.17

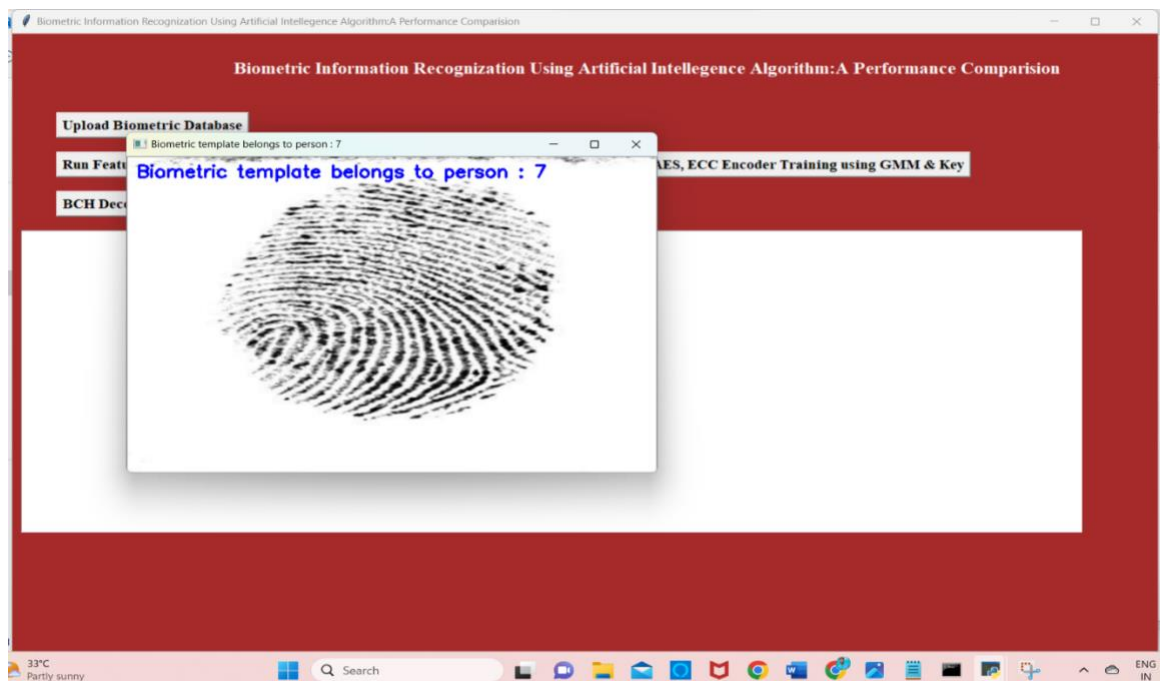


Fig 5.18

Finally, every test image is verified and gives results.

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

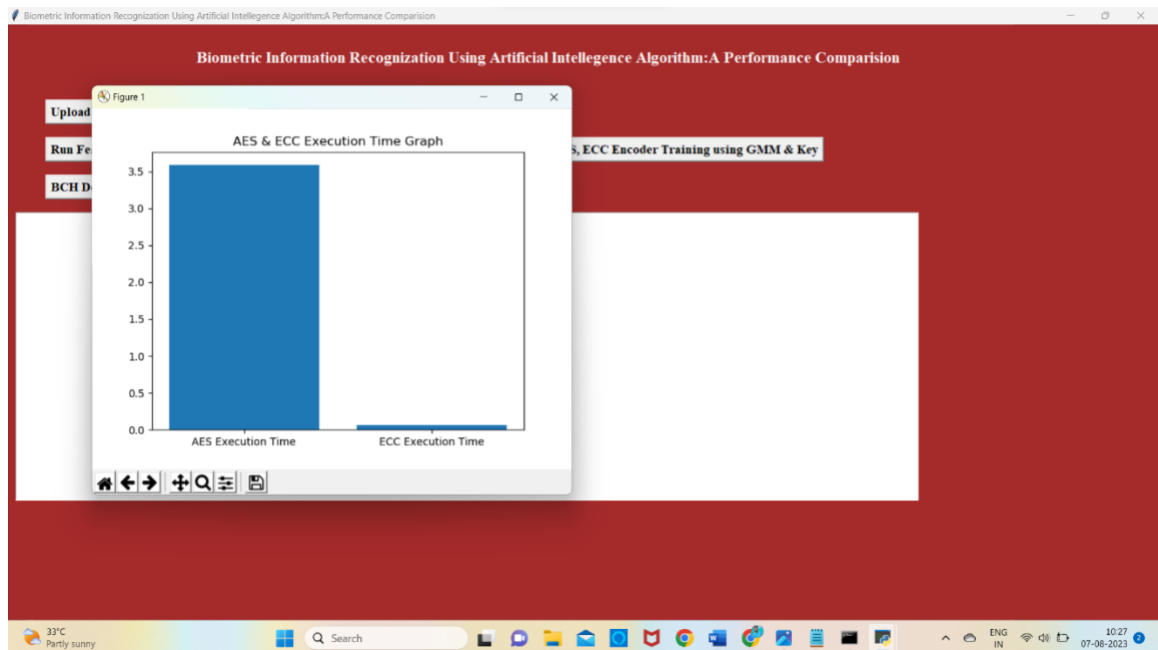


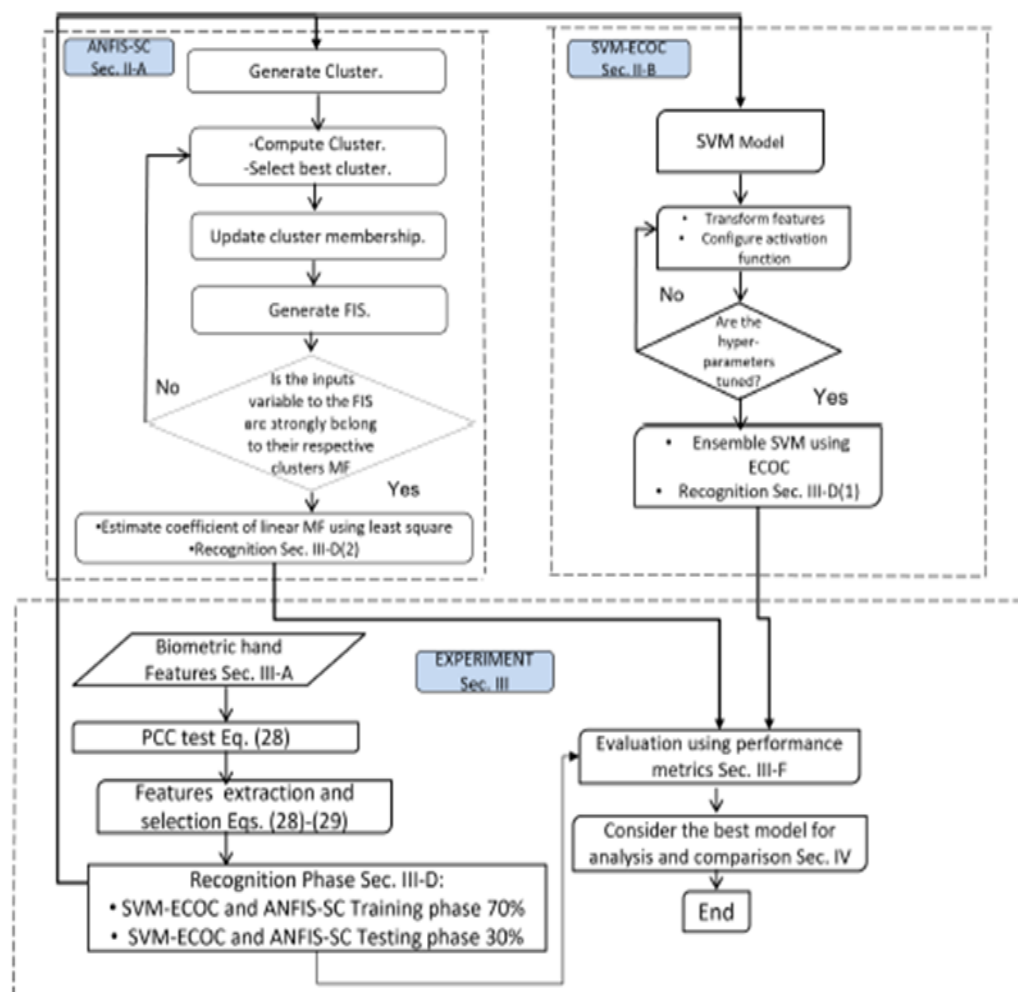
Fig 5.19

Finally, the Execution time graph between AES & ECC is displayed.

5.2.3 Result Analysis

The result analysis describes that the entire project was executed successfully and having quality and performance by analyzing the flow of data and output screens. In my project the modules like Data Preprocessing, Feature Extraction, Training, Recognition and Evaluation modules provide a framework for developing a biometric information recognition system using artificial intelligence algorithms. The specific implementation details and algorithms chosen will depend on the type of biometric data being used and the available resources.

5.3 Method of Implementation



SOFTWARE AND TECHNOLOGY DESCRIPTION

Python

Below are some facts about Python. Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. Programmers must type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech- giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc. The biggest strength of Python is huge collection of standard libraries which can be used for the following –

Machine Learning

GUI Applications (like Kivy, Tkinter, PyQt etc.)

Web frameworks like Django (used by YouTube, Instagram, Dropbox)

Image processing (like Opencv, Pillow)

Web scraping (like Scrapy, BeautifulSoup, Selenium)

Test frameworks

Multimedia

Advantages of Python

Let us see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it *contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading,*

databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be **extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities** to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print '**Hello World**'. But in Python, just a print statement will do. It is also quite **easy to learn, understand, and code**. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need

curly braces to define blocks, and **indentation is mandatory**. These further aids the readability of the code.

8. Object-Oriented

This language supports both the **procedural and object-oriented** programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

9. Free and Open-Source

Like we said earlier, Python is **freely available**. But not only can you **download Python** for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it is not the same with Python. Here, you need to **code only once**, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages.

Advantages of Python over other languages

1. Less Coding

Almost all the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have

to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 GitHub annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac, or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and **machine learning**, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in **slow execution**. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the **client-side**. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbonnelle**.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is **dynamically-typed**. This means that you don't need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can **raise run-time errors**.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like **JDBC (Java Data Base Connectivity)** and **ODBC (Open Data Base Connectivity)**, Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

History of Python

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van

Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners¹, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

What is Machine Learning : -

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of *building models of data*.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models *tunable parameters* that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these

tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Categories Of Machine Learning :-

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into *classification* tasks and *regression* tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as *clustering* and *dimensionality reduction*. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

be used, instead of using programming logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

Challenges in Machines Learning :-

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are –

Quality of data – Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task – Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons – As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems – Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting – If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality – Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment – Complexity of the ML model makes it quite difficult to be deployed in real life.

Applications of Machines Learning: -

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML –

Emotion analysis

Sentiment analysis

Error detection and prevention

Weather forecasting and prediction

Stock market analysis and forecasting

Speech synthesis

Speech recognition

Customer segmentation

Object recognition

Fraud detection

Fraud prevention

Recommendation of products to customer in online shopping

How to Start Learning Machine Learning?

Arthur Samuel coined the term “**Machine Learning**” in 1959 and defined it as a “**Field of study that gives computers the capability to learn without being explicitly programmed**”.

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine

Learning Engineer Is the Best Job of 2019 with a 344% growth and an average base salary of **\$146,085** per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So, this article deals with the Basics of Machine Learning and the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let us get started!!!

How to start learning ML?

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don't know these, never fear! You don't need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on math as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the

collection, analysis, and presentation of data. So, it is no surprise that you need to learn it!!!

Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.

So if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as **Fork Python** available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to learning ML (Which is the fun part!!!) It is best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

(a) Terminologies of Machine Learning

Model – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.

Feature – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

Target (Label) – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.

Training – The idea is to give a set of inputs(features) and it is expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.

Prediction – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Types of Machine Learning

Supervised Learning – This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.

Unsupervised Learning – This involves using unlabeled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.

Semi-supervised Learning – This involves using unlabeled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.

Reinforcement Learning – This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

Advantages of Machine learning: -

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you do not need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and improve the algorithms on their own. A common example of this is anti-virus software; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As **ML algorithms** gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data, you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

Disadvantages of Machine Learning: -

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

Python Development Steps: -

Guido Van Rossum published the first version of Python code (version 0.9.0) at outsources in February 1991. This release included already exception handling, functions, and the core data types of lists, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting Unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it." Some changes in Python 7.3:

Print is now a function

Views and iterators instead of lists

The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.

There is only one integer type left, i.e., int. long is int as well.

The division of two integers returns a float instead of an integer. "/" can be used to have the "old" behavior.

Text Vs. Data Instead of Unicode Vs. 8-bit

Purpose: -

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional, and procedural, and has a large and comprehensive standard library.

Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

Python is Interactive – you can sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you must scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Modules Used in Project: -

TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

A powerful N-dimensional array object

Sophisticated (broadcasting) functions

Tools for integrating C/C++ and Fortran code

Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac :

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows and Mac :

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your **System Requirements**. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a **Windows 64-bit operating system**. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. [Download the Python Cheatsheet here.](#) The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: <https://www.python.org>

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison



Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.



Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Looking for a specific release?
Python releases by version number:

Release version	Release date	Click for more	
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Files					
Version	Operating System	Description	MD5 Sum	File Size	GP6
Gzipped source tarball	Source release		68111671e5b2db4ae77b9ab01b709be	23017663	SG
XZ compressed source tarball	Source release		d33e4aaf66097051c2eca45ee3604803	17131432	SG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583da71a402cha0ce08e6	34896416	SG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a45773b9e4a936b241f	26882845	SG
Windows help file	Windows		d63090573a2c96b2ac56cade6b47cd2	8131761	SG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/v64	9b00c3cfd9ec0b0abe83184a072ba2	7504391	SG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/v64	a702b4b0ad76d9bdc3543a583e543e00	26882848	SG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/v64	28cb1c608b6d73a8e53a3bdf351b4bd2	1362904	SG
Windows x86 embeddable zip file	Windows		9fab1b8128b41879fda94133574139d8	6741626	SG
Windows x86 executable installer	Windows		33cc602942a5446a3d9e451a76394789	25663848	SG
Windows x86 web-based installer	Windows		1b670cfa5d317df82c30983ea371687c	1324608	SG

- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.

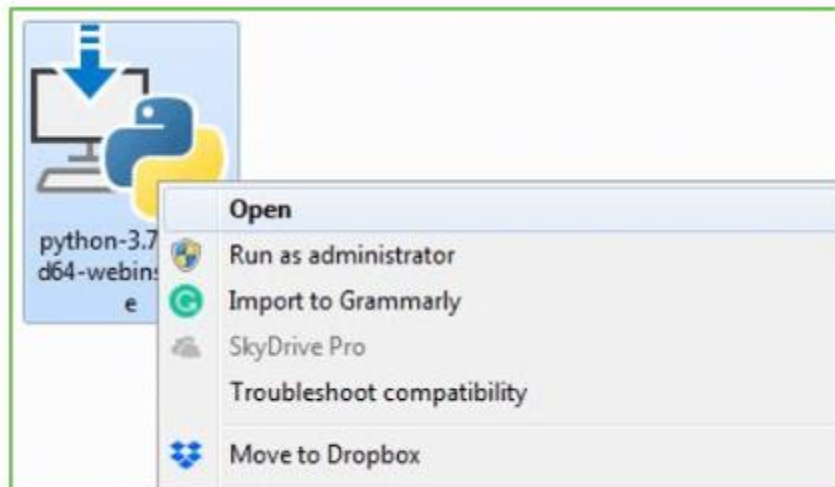
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.



Step 3: Click on Install NOW After the installation is successful. Click on Close.



With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

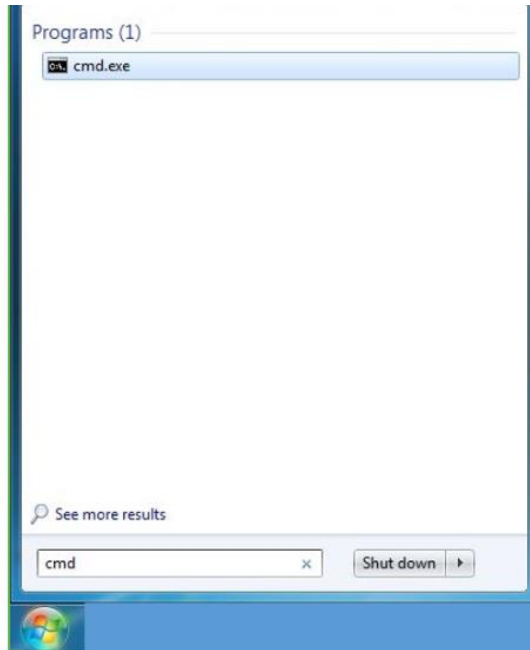
Note: The installation process might take a couple of minutes.

Verify the Python Installation

Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type **python -V** and press Enter.

A screenshot of a Windows Command Prompt window. The title bar shows 'C:\Windows\system32\cmd.exe'. The window displays the following text: 'Microsoft Windows [Version 6.1.7601] Copyright (c) 2009 Microsoft Corporation. All rights reserved. C:\Users\DELL>python -U Python 3.7.4 C:\Users\DELL>'. The command and its output are highlighted with an orange box.

Step 5: You will get the answer as 3.7.4

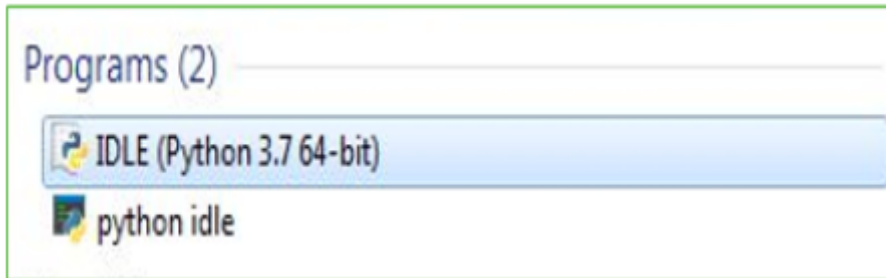
Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

Check how the Python IDLE works

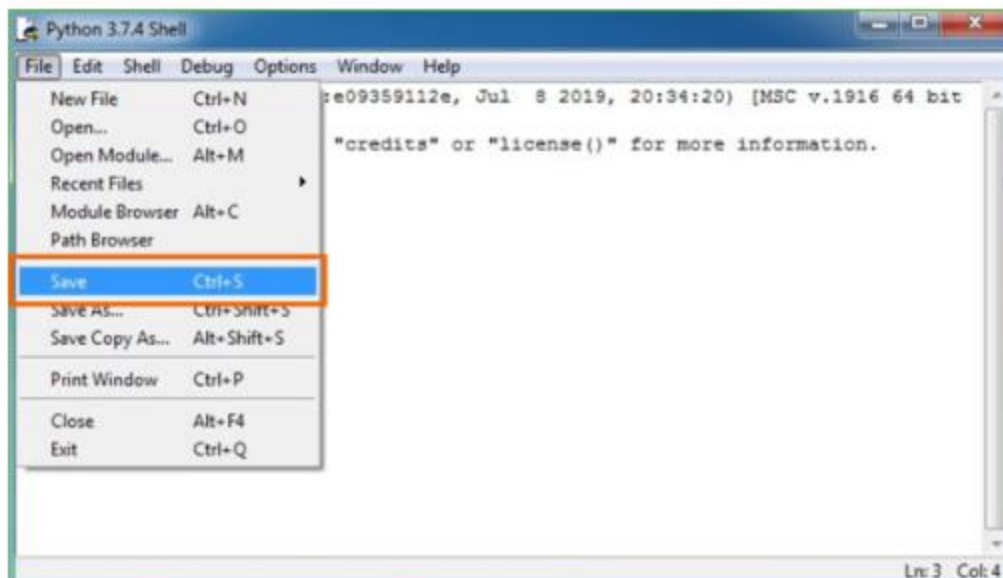
Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.



Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. **Click on File > Click on Save**



Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. **enter print**

5.4 CONCLUSION

By observing the above methods of implementation, output screens, result analysis the implementation phase was completed successfully with quality, good performance and also satisfying all user requirements. It satisfies all java application development standards.

6. SYSTEM TESTING

6.1 INTRODUCTION

Testing is that the debugging program is one amongst the leading crucial aspects of the pc programming triggers, while not programming that works, the system would ne'er turn out relate in Nursing output of that it had been designed. Testing is best performed once user development is asked to help in characteristic all errors and bugs. The sample knowledge are used for testing. It is not amount however quality of the information used the matters of testing .Testing is aimed toward guaranteeing that the system was accurately relate in Nursing with efficiency before live operation commands.

TESTING OBJECTIVES

The most objective of testing is to uncover a bunch of errors, consistently and with minimum effort and time. Stating formally, Testing may be a method of corporal punishment a program with intent of finding miscalculation.

- ✓ A productive check is one that uncovers Associate in nursing hitherto undiscovered error.
- ✓ A decent legal action is one that has likelihood of finding miscalculation, if it exists.
- ✓ The check is insufficient to find probably gift errors.
- ✓ The code additional or less confirms to the standard and reliable standards.

6.1.1 TYPES OF TESTING

6.1.1.1 Unit Testing

Unit testing they have a tendency to test every module separately and integrate with the general system. Unit testing focuses verification efforts on the littlest unit of code style within the module. This is often conjointly called module testing. The module of the system is tested individually. As an example the validation check is completed for variable the user input given by the user that validity of the information entered. It's terribly straightforward to search out error rectify the system.

Every Module will be tested victimization the subsequent 2 Strategies: recording machine Testing and White Box Testing.

6.1.1.2 Black Box Testing

Recording machine checking may be a code testing techniques during which practicality of the code below test (SUT) is tested while not staring at the interior code structure, implementation details and data of internal ways of the code. This type of testing is predicated entirely on the code needs and specifications .In recording machine Testing they have a tendency to simply concentrate on inputs and output of the package while not bothering concerning internal data of the code program.

The on top of recording machine will be any package you wish to check. For example, Associate in Nursing software like Windows, a web site like Google ,a information like Oracle or maybe the own custom application. Under recording machine testing, you can check these applications by simply that specialize in the inputs and outputs while not knowing their internal code implementation.

Types of Black Box Testing

There are many varieties of recording machine testing however following the outstanding ones.

- ✓ **Functional testing:** This recording machine testing kind is said to purposeful needs of a system; it's done by code testers.
- ✓ **Non-Functional testing:** This sort of recording machine testing isn't associated with testing of a selected practicality, however non-functional needs like performance, measurability, usability.
- ✓ **Regression testing:** Regression testing is completed once code fixes, upgrades or the other system maintenance to visualize the new code has not affected the prevailing code.

6.1.1.3 White Box Testing

White Box Testing is that the testing of a code solution's internal committal to writing and infrastructure. It focuses totally on Traffic Redundancy Elimination security, the flow of inputs and outputs through the applying, and rising style and value. White box testing is additionally called clear, open, structural, and glass box testing. It is one amongst 2 elements of the "box testing" approach of code testing.

6.1.1.4. System Testing

Once the individual module testing is completed, modules are assembled and integrated to perform as a system. The top down testing that began from higher level to lower level module was allotted to visualize whether or not the whole system is playacting satisfactorily. There are 3 main types of System testing: Alpha Testing, Beta Testing, and Acceptance Testing.

- ✓ **Alpha Testing:** This refers to the system checking that allotted by the test team with the Organization.
- ✓ **Beta Testing:** This refers to the system testing that performed by a particular cluster of friendly customers.
- ✓ **Acceptance Testing:** This refers to the system testing that performed by the client to see whether or not or to not settle for the delivery of the system.

6.2 TEST STRATEGY AND APPROACH

Field Testing will be performed manually and functional tests will be written in detail.

➤ **Test objectives**

- ✓ All field entries must work properly.
- ✓ Pages must be activated from the identified link.
- ✓ The entry screen, messages and responses must not be delayed.

➤ **Features to be tested**

- ✓ Verify that the entries are of the correct format.
- ✓ No duplicate entries should be allowed.
- ✓ All links should take the user to the correct page.

6.2.1 TEST CASES

TC No	Test Case	Input	Expected Output	Observed Output	Result
1	Give Fingerprint	Select Fingerprint	Verified	--do--	Pass
2	Fingerprint in JPG format	Select Fingerprint in JPG format	Fingerprint must be in JPG format	Fingerprint given in PDF format	Fail
3	Fingerprint in JPG format	Select Fingerprint image	Fingerprint must be in JPG format	Fingerprint given in JPG format	Pass

Table 6.1: Test Cases Table

6.3 VALIDATION

Testing process starts with a test plan. This plan identifies all the testing related activities that must be performed and specifies the schedules, allocates the resources, and specified guidelines for testing. During the testing of the unit the specified test cases are executed and the actual result compared with expected output. The final output of the testing phase is the test report and the error report.

Test Data:

Here all test cases that are used for the system testing are specified. The goal is to test the different functional requirements specified in Software Requirements Specifications (SRS) document.

Unit Testing:

Each individual module has been tested against the requirement with some test data.

Test Report:

The module is working properly provided the user must enter information. All data entry forms have tested with specified test cases and all data entry forms are working properly.

Error Report:

If the user does not enter data in specified order, then the user will be prompted with error messages. Error handling was done to handle the expected and unexpected errors.

6.4 CONCLUSION

By observing all sample test cases which were taken in my project was successfully done. The testing strategy like system testing, integration testing, unit tastings are used in my project to test the established system. Finally, I conclude that my project has been working very accurately with quality and good performance.

7. CONCLUSION

In this work, we initialized AI algorithms using ECOC and subtractive clustering optimization schemes as potential enabler in conventional AIs to handle complex parameters estimation. The work aimed to investigate correlation among intrinsic hand measurements and demographic features. It particularly used AI algorithms and LOG for comparison to analyze and recognize sex, height, weight, and foot-size from 21 hand features extracted through measurement of hand bones. The compared models are realized according to proper parameters chosen. The evaluation metrics show that AI recognition performed better than LOG while predicting demographic characteristics. Our results have shown to agree with and performed better than previous method. Specifically, ANFIS-SC based learning demonstrates high performance in terms of accuracy and speed, when compared to SVM-ECOC learning. ANFIS-SC learning results qualified it worthy in many applications such as crime detection and soft-biometric features recognition. This work has direct application in both biometric and forensic industries. The major limitations of both methods are related to accuracy and high number of computational burden for a limited number of the hand corpus. This means that, on restricted hand corpus, the quality of the biometric recognition of AI algorithms would be already rather good. Moreover, adding more demographic and psychological cues such as race/ethnicity, life expectancy, and facial marks can further improve the recognition capabilities by reducing the number of classification errors. In addition, AI algorithms yield premature and slow convergence.

8. BIBLIOGRAPHY

References

1. M. Drahansky, M. Dolezel, J. Urbanek, E. Brezinova, and T.-H. Kim, "Influence of skin diseases on fingerprint recognition," J. Biomed. Biotechnol., vol. 2012, pp. 1fi14, May 2012.
2. S. A. Haider, Y. Rehman, and S. M. U. Ali, "Enhanced multimodal biometric recognition based upon intrinsic hand biometrics," Electronics, vol. 9, no. 11, p. 1916, Nov. 2020.
3. J. L. Blanch and S. S. Christensen, "Biometric basics: Options to gather data from digital devices locked by a biometric key," Dept. Justice J. Federal Law Pract., United State Attorney's Bull., vol. 66, no. 2018, p. 3, Jan. 2018.
4. T. Kanchan and P. Rastogi, "Sex determination from hand dimensions of north and south Indians," J. Forensic Sci., vol. 54, no. 3, pp. 546fi550, May 2009.
5. R. Karki and P. Singh, "Gender determination from fingerprints," J. Universal College Med. Sci., vol. 2, no. 1, pp. 12fi15, May 2014.
6. R. Guest, O. Miguel-Hurtado, S. V. Stevenage, G. J. Neil, and S. Black, "Biometrics within the SuperIdentity project: A new approach to spanning multiple identity domains," in Proc. Int. Carnahan Conf. Secur. Technol. (ICCST), Oct. 2014, pp. 1fi6.
7. S.-C. Jee, S. Bahn, and M. H. Yun, "Determination of sex from various hand dimensions of Koreans," Forensic Sci. Int., vol. 257, p. 521.e1, Dec. 2015.
8. S.-C. Jee and M. H. Yun, "Estimation of stature from diversified hand anthropometric dimensions from Korean population," J. Forensic Legal Med., vol. 35, pp. 9fi14, Oct. 2015.

Biometric Information Recognition Using Artificial Algorithms: A Performance Comparison

9. M. K. Thakar, P. Kaur, and T. Sharma, "Validation studies on gender determination from fingerprints with special emphasis on ridge characteristics," *Egyptian J. Forensic Sci.*, vol. 8, no. 1, pp. 1fi7, Dec. 2018.
10. J. C. Loehlin, S. E. Medland, and N. G. Martin, "Relative finger lengths, sex differences, and psychological traits," *Arch. Sexual Behav.*, vol. 38, no. 2, pp. 298fi305, Apr. 2009.
11. A. K. Agnihotri, S. Agnihotri, N. Jeebun, and K. Googoolye, "Prediction of stature using hand dimensions," *J. Forensic Legal Med.*, vol. 15, no. 8, pp. 479fi482, Nov. 2008.
12. S. V. Stevenage, C. Walpole, G. J. Neil, and S. M. Black, "Testing the reliability of hands and ears as biometrics: The importance of viewpoint," *Psychol. Res.*, vol. 79, no. 6, pp. 989fi999, Nov. 2015.
13. O. Al-Jarrah and A. Halawani, "Recognition of gestures in Arabic sign language using neuro-fuzzy systems," *Artif. Intell.*, vol. 133, nos. 1fi2, pp. 117fi138, 2001.
14. F. A. Mufarroha and F. Utaminingrum, "Hand gesture recognition using adaptive network based fuzzy inference system and K-nearest neighbor," *Int. J. Technol.*, vol. 8, no. 3, pp. 559fi567, 2017.
15. N. A. Alias and N. H. M. Radzi, "Fingerprint classification using support vector machine," in *Proc. 5th ICT Int. Student Project Conf. (ICT-ISPC)*, May 2016, pp. 105fi108.
16. M. L. Gavrilova, F. Ahmed, A. H. Bari, R. Liu, T. Liu, Y. Maret, B. K. Sieu, and T. Sudhakar, "Multi-modal motion-capture-based biometric systems for emergency response and patient rehabilitation," in *Research Anthology on Rehabilitation Practices and Therapy*. Hershey, PA, USA: IGI Global, 2021, pp. 653fi678.
17. W. K. Wong, F. H. Juwono, and B. T. T. Khoo, "Multi-features capacitive hand gesture recognition sensor: A machine learning approach," *IEEE Sensors J.*, vol. 21, no. 6, pp. 8441fi8450, Mar. 2021.

18. O. Miguel-Hurtado, R. Guest, S. V. Stevenage, G. J. Neil, and S. Black, "Comparing machine learning classifiers and linear/logistic regression to explore the relationship between hand dimensions and demographic characteristics," *PLoS ONE*, vol. 11, no. 11, Nov. 2016, Art. no. e0165521.
19. S. Escalera, O. Pujol, and P. Radeva, "Separability of ternary codes for sparse designs of error-correcting output codes," *Pattern Recognit. Lett.*, vol. 30, no. 3, pp. 285-297, 2009.
20. S. Escalera, O. Pujol, and P. Radeva, "On the decoding process in ternary error-correcting output codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 120-134, Jan. 2010.
21. T. Windeatt and R. Ghaderi, "Coding and decoding strategies for multiclass learning problems," *Inf. Fusion*, vol. 4, no. 1, pp. 11-21, Mar. 2003.
22. J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise Reduction in Speech Processing*. Berlin, Germany: Springer, 2009, pp. 1-4.
23. B. Kaur and G. Joshi, "Lower order Krawtchouk moment-based featureset for hand gesture recognition," *Adv. Hum.-Comput. Interact.*, vol. 2016, pp. 1-10, Mar. 2016.
24. S. Abdullahi and M. Gaya, "Forecasting of HOSR for different mobile carriers in Kano using conventional and intelligent techniques," *Int. J. New Comput. Archit. Appl.*, vol. 9, no. 1, pp. 1-11, 2019.
25. S. B. Abdullahi, K. Muangchoo, A. B. Abubakar, A. H. Ibrahim, and K. O. Aremu, "Data-driven AI-based parameters tuning using grid partition algorithm for predicting climatic effect on epidemic diseases," *IEEE Access*, vol. 9, pp. 55388-55412, 2021.