

Software Design Document of Classification of Alzheimer's by DNA Analysis Project

Ahmed Samir, Fairuz Soufy, Omar Ehab, Sara Hassan
Lobna Shaheen, Nora El-Samanody, Omar El-Demrdash, Rawan El-Kady
Ashraf AbdelRaouf, Lamiaa Nabil, Mohamed Shahin

March 8, 2020

1 Introduction

1.1 Purpose

This software design document describes and presents a detailed description of the Classification of Alzheimer's (AD) by DNA Analysis project. The main purpose of this project is to be able to classify AD patients to healthy patients and people who carry AD. Early diagnosis of AD may help in slowing down the progression of the disease considerably. This document clarifies the purposes and features of the project.

1.2 Scope

The system is developed to reveal if the patient is healthy and if not, how much is the progression of the disease in his body. Either way, this classification helps the patient and the doctors to diagnose the disease early in which gives them a chance to slow down the progression of his disease depending on the stage the patient is in since early diagnosis is key in these type of situations.

1.3 Overview

This SDD document includes 8 main sections. The first section is an introduction to our system including our scope and purpose. The second section is the system overview illustrating our system work flow. The third section includes the architecture design of the system, activity diagram, sequence diagram and class diagram. The fourth section illustrates the database design in details. The fifth section illustrates our component design including the used algorithms and techniques. The sixth section illustrates the human interface design and describes how the user will interact with our system. The seventh section is the requirement matrix that shows which components satisfy each of the functional requirements.

1.4 Definitions and Acronyms

| Term | Definition |
|---------------|---|
| SDD | Software Design Document , used as the primary medium for communicating software design information. |
| Design Entity | An element of a design that is structurally and functionally distinct from other elements. |
| AD | Alzheimer's Disease. |
| SVR | Support Vector Regression. |
| DNA | Deoxyribonucleic acid, a self-replicating material which is present in nearly all living organisms as the main constituent of chromosomes. It is the carrier of genetic information.. |

2 System Overview

In the application, we are implementing a system that will be able to accurately and swiftly diagnose AD patients and categorize them into two classes scrupulously: healthy patients and patients with a high risk of developing the disease or disease carrier. Moreover, if the patient turns out to have AD we group them into three classes those being severe cases, moderate and mild. Our approach starts with the collection of the patient's sample. Then they are analyzed and filtered in order to get the desired chromosomes and the particular locations that is needed to be able to properly diagnose the patient concluding our preprocessing of the sample. The sample will then be compared with a model prebuilt already using a SVR that extract the needed features from the datasets. so it can be able to classify the sample correctly after this process if indeed the patients sample is positive for Alzheimer's it'll be compared with another model to be able to determine the severity of the disease in this specific sample.

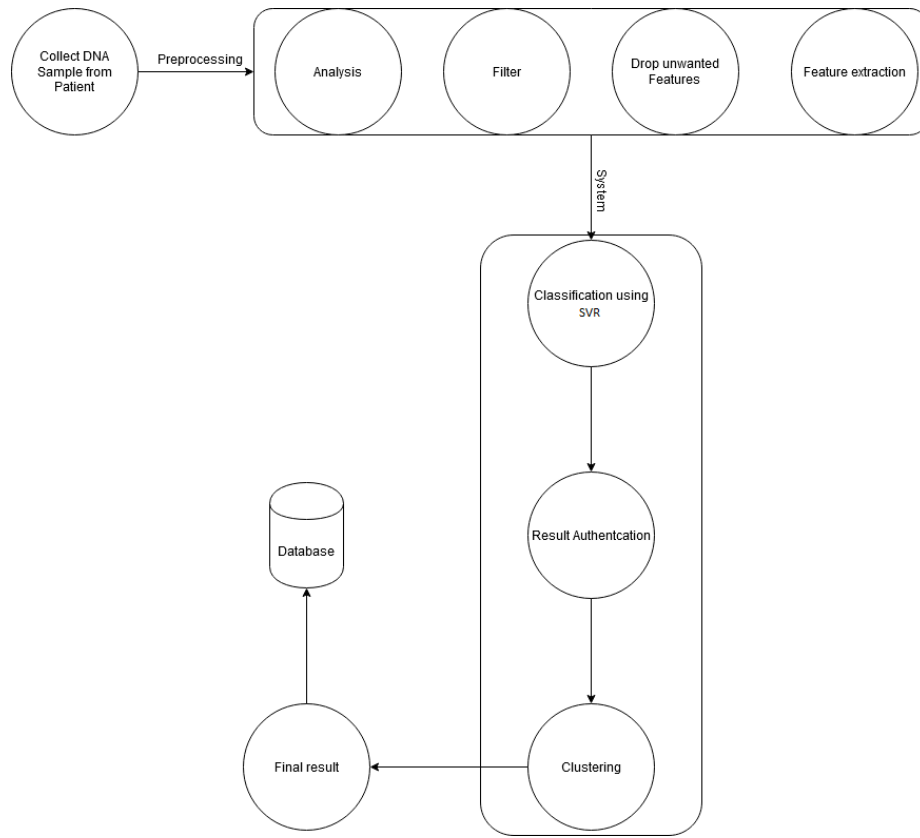


Figure 1: System Overview

3 System Architecture

3.1 Architectural Design

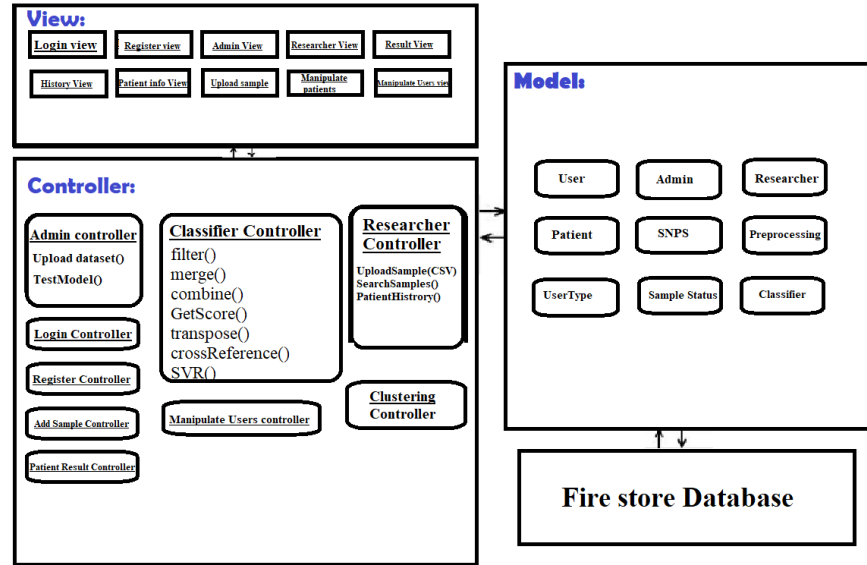


Figure 2: Architecture Diagram

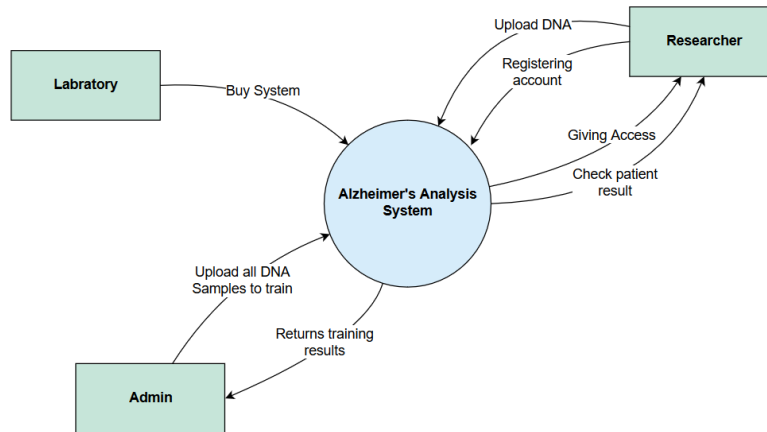


Figure 3: Context Diagram

3.2 Decomposition Description

3.2.1 Class Diagram

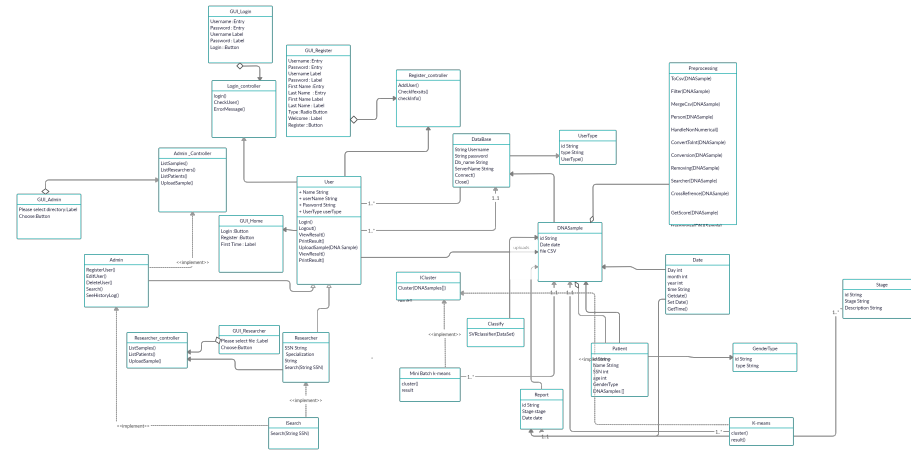


Figure 4: Class Diagram

3.2.2 User

1. Class Name: User
2. Super Classes: N/A
3. Sub Classes: Admin, Lab Technician
4. Purpose: this class is the main class holds all functionality for other classes
5. Collaborations: userType
6. Attributes: Name, Username, password, user type.
7. Operations: Login, Log Out, Uploads sample, view results, print Results.

3.2.3 UserType

1. Type: concrete.
2. List of super classes: None.
3. List of sub classes: None.
4. Purpose: To allow scalability to add new user types.
5. Collaboration: This class is aggregated by class User, UserTypeAttribute and Report.

6. Attributes: id, TypeName, options[].

7. Operations: None.

3.2.4 Admin

1. Class Name: Admin

2. Super Classes: User

3. Sub Classes:N/A

4. Purpose: this class is the holds all functionalities for Admin

5. Collaborations: N/A

6. Attributes:N/A

7. Operations:CRUD Lab Technician

3.2.5 Reseracher

1. Class Name: Reseracher

2. Super Classes: User

3. Sub Classes:N/A

4. Purpose: this class is the holds all functionalities for Lab Technician

5. Collaborations: N/A

6. Attributes: specualization, SSN,gender.

7. Operations:none.

3.2.6 DNA Sample

1. Class Name: DNA Sample

2. Super Classes: N/A

3. Sub Classes:N/A

4. Purpose: this class is the holds all information about a DNA Sample.

5. Collaborations: patient,Report.

6. Attributes: sample id , sample date ,sample File.

7. Operations:none.

3.2.7 patient

1. Class Name: patient
2. Super Classes: N/A
3. Sub Classes:N/A
4. Purpose: this class is the holds all information about a any patient.
5. Collaborations: gender Type.
6. Attributes:id ,name, SSN, age,Gender.
7. Operations:none.

3.2.8 Report

1. Class Name: Report
2. Super Classes: N/A
3. Sub Classes:N/A
4. Purpose: this class is the holds all information about DNA sample report .
5. Collaborations: DNA Sample,Stage,patient.
6. Attributes:id ,date.
7. Operations:none.

3.2.9 Preprocessing

1. Class Name: Preprocessing
2. Super Classes: N/A
3. Sub Classes:N/A
4. Purpose: this class is responsible for all the processing that will be done before clustering.
5. Collaborations: DNA Sample
6. Attributes:none.
7. Operations:Searcher, Removing,ToCsv,Filter,MergToCsv,Convert

3.2.10 ICluster

1. Class Name: ICluster
2. Super Classes: N/A
3. Sub Classes:K-means, Mini Batch
4. Purpose: This interface initiates the cluster function.
5. Collaborations: DNA Sample
6. Attributes:id, Stage .example stage A or B.
7. Operations: none.

3.2.11 ISearch

1. Class Name: ISearch
2. Super Classes: N/A
3. Sub Classes:none
4. Purpose:To allow searching with different Criteria.
5. Collaborations:classes (Admin, Reseracher) implements this class
6. Attributes:name,id
7. Operations:Search (String userName).

3.2.12 Classify

1. Class Name: Classify
2. Super Classes: N/A
3. Sub Classes:N/A
4. Purpose: To allow classification with different classifiers strategies
5. Collaborations: Class SVR
6. Attributes: none.
7. Operations: Classify(Parameters []).

3.2.13 LoginController

1. Class Name: LoginController
2. Super Classes: N/A
3. Sub Classes:N/A
4. Purpose: to control the Login view.
5. Collaborations:this class aggregated by LoginView
6. Attributes: none.
7. Operations:ControlUserLogin (String username, String pass)

3.2.14 Stage

1. Class Name: Stage
2. Super Classes: N/A
3. Sub Classes:N/A
4. Purpose: This class responsible for storing the stages types
5. Collaborations: report
6. Attributes: id, stage.
7. Operations: none.

3.2.15 Activity Diagram

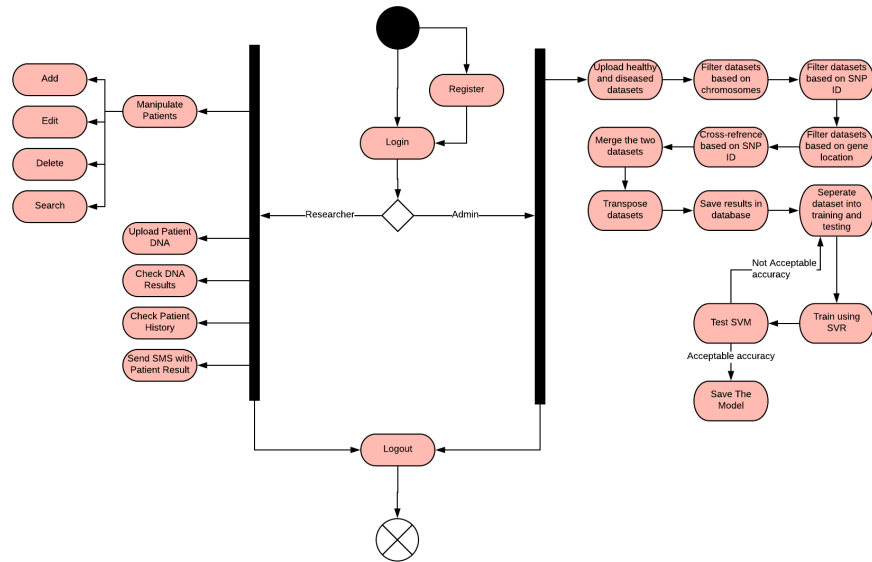


Figure 5: Activity Diagram

3.2.16 Admin Sequence Diagram

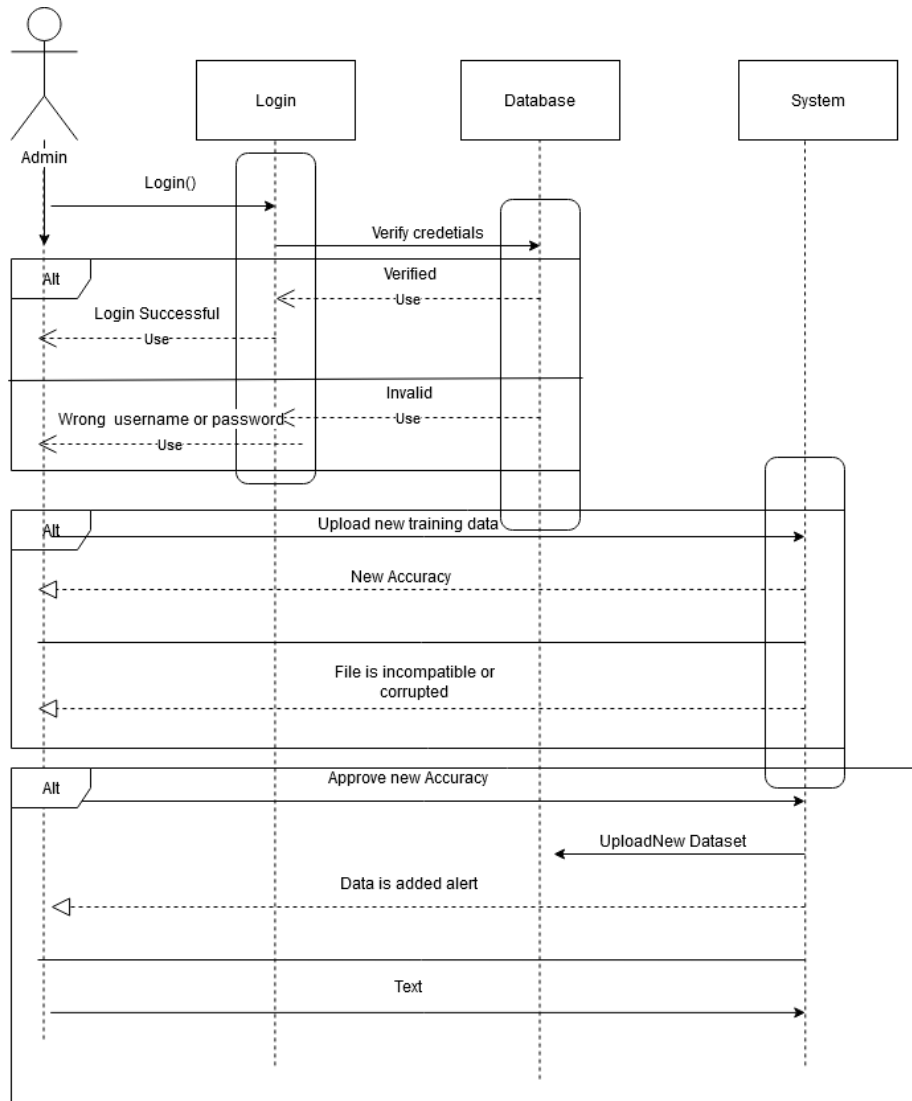


Figure 6: Retrain Model Diagram

Figure 6 displays the sequence of the process of how the system admin can retrain the model in order to further improve the accuracy of the existing model the process start with the admin logging in then the admin will upload the new data that will be appended to the existing dataset so the system will check first if the file is in the same format before appending it then it'll retrain the model and compare its accuracy with the existing one if the accuracy is indeed higher

a model will be created to replace the current one if not the new model and dataset will be discarded.

3.2.17 Researcher Sequence Diagram

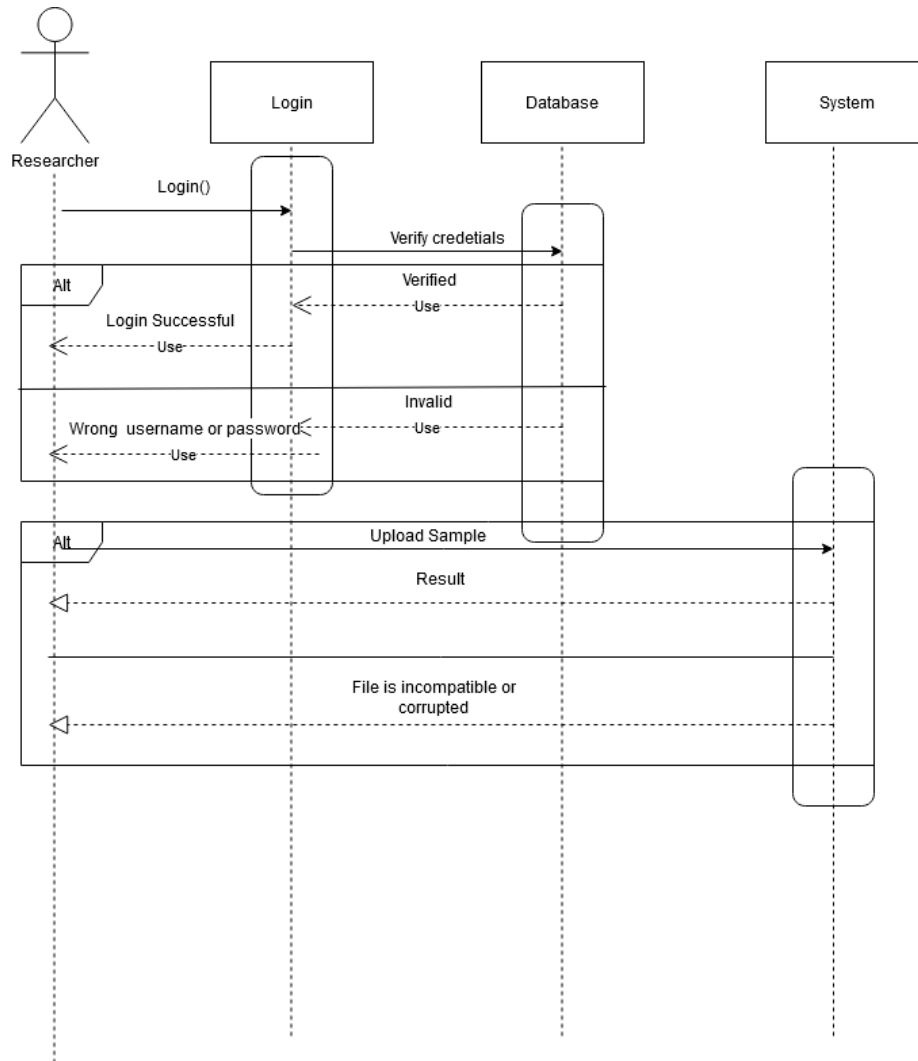


Figure 7: Check DNA Sample Diagram

In figure 7 the processes of how a researcher can check if a patient has AD or not by uploading into the system after logging in the researcher will upload the sample the system will then use the existing model to get a result that is after the system checks the file to make sure its compatible.

3.3 Design Rationale

Concerning the architecture of the system the model view controller architecture was chosen due to its many advantages over other architectures like Development of the application becomes fast and simple ,simplifies for multiple developers to collaborate and work together on the same project since the system is divided into separate parts, furthermore it makes it easier to Update the application since only the component or the part that requires updating is accessed.

Regarding the algorithm used in order to classify the patients support vector regression (SVR) was used [1]. This algorithm works by parsing through the given dataset and mapping data to a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable. This algorithm was chosen after trying out numerous algorithms that may fit the needs of our project however SVR turned out to be the best one overall regarding its complexity and its accuracy furthermore there were a handful of other algorithms that were experimented with and those are Decision Tree, Random Forest , Grid search (Linear , RBF), Random Search (Linear , RBF) , Naïve Bayes , Logistic Regression , Deep Learning , Generalized Linear Model, Fast Large Margin , and Gradient Boosted Trees.

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches. Leaf node represents a classification or decision.

Random forest is similar to the Decision tree model since the random forest is comprised of a number of Decision trees instead of just one When training, each tree in a random forest learns from a random sample of the data points. With each tree training slightly different on different observation therefore producing a well-balanced prediction derived from the average of the all the trees that were trained however the abundance of trained trees may produce noise.

Grid Search and Random Search are algorithms that are given models and data and they each parse through the parameters that can be passed to the models in order to produce the best possible results that can be produced given that particular model however each algorithm parse through the data in a different way. Grid searching works by trying all the different combinations of parameters by iterating through them thus producing great results however can be computationally expensive while random search works by trying out sets of random combinations of parameters in order to find the best possible model while optimizing the parameters by evaluating the model at random configuration points. Random search can produce better results because of its random method of trying out random parameters and can have a lower complexity when compared to Grid Search.

Naïve Bayes

Logistic regression does not try to predict the value of a numeric variable given a set of inputs. Instead, the output is a probability that the given input point belongs to a certain class. It is primarily used in binary classification.

Deep learning works by training an AI that if given a set of inputs can predict the output the AI consists of 3 types of layers (input, hidden, output) with each layer having a number of neurons inside it the connections between neurons represents the weight of each input and based on this weight the neuron decides what to do with the given output through an activation function inside each neuron.

Generalized Linear Models are an extension of traditional linear models. This algorithm fits generalized linear models to the data by maximizing the log-likelihood. The elastic net penalty can be used for parameter regularization. The model fitting computation is parallel, extremely fast, and scales extremely well for models with a limited number of predictors with non-zero coefficients.

Fast Large Margin (FLM) is similar to normal linear SVM algorithms in the sense that it tries to map the data and inputs onto a linear plane in order to properly classify the inputs however FLM is designed to work on huge datasets with millions of records.

Gradient Boosted Trees is a tree-based learning model that trains many models in a gradual, additive and sequential manner. Predictions of the final ensemble model is therefore the weighted sum of the predictions made by the previous tree models that were previously built therefore producing a model based on the best fitted trees trained during the run time.

However due to the nature of the dataset the models built with the previously mentioned classifiers ended up overfitting and produced unrealistic accuracy in almost all attempts with results ranging from 98-100 % therefore the SVR model was chosen since it had the most realistic accuracy of almost 92% and it was also one of the most computationally efficient algorithms that were tested.

4 Data Design

4.1 Data Description

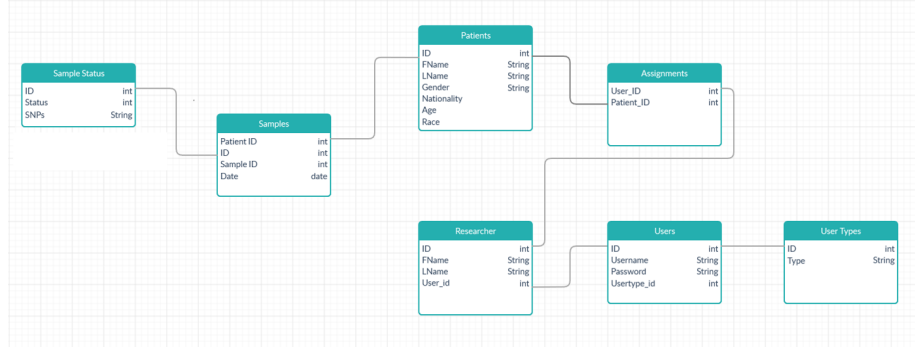


Figure 8: Database Diagram

4.2 Data Dictionary

4.2.1 Security

Security is a very important factor for the project so no one has the access to the patient's data unless he has a profile and his profile is allowed to access the data. The password of every system user is hashed by using sha324 function.

4.2.2 Reliability

The system is reliable enough to handle all failure events. The time needed to diagnose a patient on the system has an average speed to check since the data is large.

4.2.3 Portability

The system is written by Python so it is an executable file that can be deployed on Windows operating system and Mac OS.

4.2.4 Efficiency

The system is very efficient with the way it handles both system memory and storage. Since the dataset is very large and many operations are done on each file in the dataset the system handles each file and moves the desired portion of the file into a new smaller sized file therefore the dataset's size is reduced significantly, moreover after processing the files we delete them in order to eliminate any wastage of the system resources

4.2.5 Maintainability

The code is very simple so it has the availability to be maintained later.

5 Component Design

In this phase we prepare the DNA sample for the classifier, first we check if the format of the file is csv or not, then we filter the data by locating only the desired locations based on the chromosome numbers and locations furthermore we select only the SNP names that starts with “rs”, drops any rows with null values, drops any other columns that are not important and finally we check if the data is all numeric or not if not. We calculate a score by comparing the ref column with Allele1- forward and Allele2-forward. If both of them are equal to ref column, we represent it by 0 if only one of them matches with the ref column their score will be represented by 1 and if both of them don’t match, the ref column their score will be represented by 2.

5.0.1 SVR

In details, SVR works by using a hyper-plane and two boundary lines also known as thresholds. In order to properly build an accurate model that fits the data properly, the hyper-plane is the line that separates the different classes in the data while the boundaries are the error threshold that is allowed the goal is to create a model that has a hyper-plane and boundaries that definitively covers most of our data points. Furthermore, the boundary lines should be equidistant from the hyper-plane meaning that the model should not favor a class more than the other in other words if represented by linear equation that lines for the boundaries and the hyper-plane should be $wx+b=0$ for the hyper-plane and $Wx+b=+e$, $Wx+b=-e$ for the boundary lines so that the e is the same for both lines creating the margin that’ll become the model that fits the data.

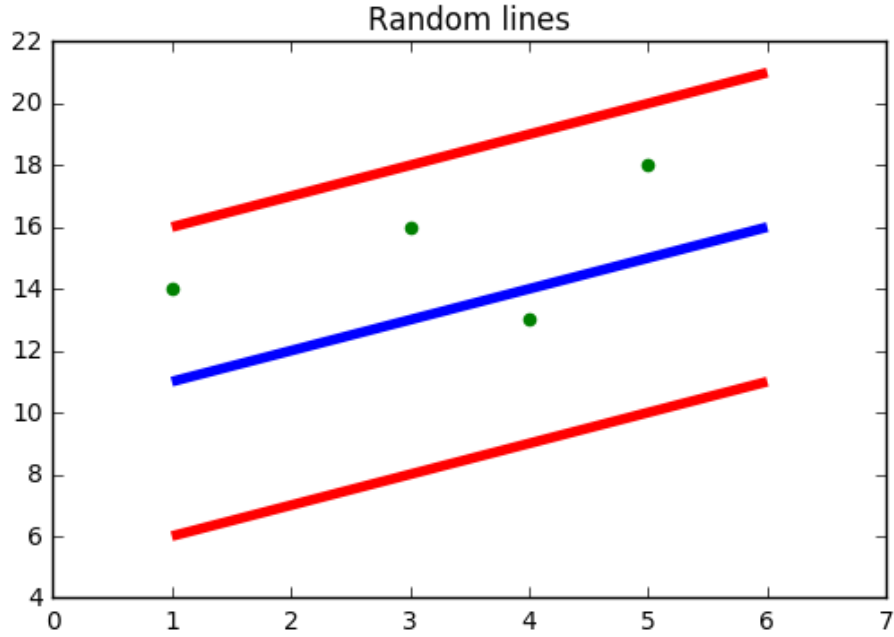


Figure 9: SVR

Where the red lines represent the boundaries and the red line represent the hyper-plane.

6 Human Interface Design

6.1 Overview of User Interface

Our system is a desktop application. It's user interface is very applicable and usable. You can login whether you are an admin or a researcher. The login screen directs you to different screens depends on the user type. System admins have some functions such as uploading data and testing the model. While researchers are responsible for dealing with patients. Illustration for the whole system is shown in Section 6.2.

6.2 Screen Images

First we have the home Screen with two buttons login and register. If login button is pressed, it will direct the user to the login screen in Figure 6. The first field to enter username and second is for the password. If you logged in as an admin you will access the screens in Fig.9 and if you logged in as a researcher you will access screens in Fig.10. Both the admin and the researcher can upload a DNA sample to be tested and both can view the patients history. The admin

can also upload a new dataset that can be appended to the existing one in order to retrain the model and get a higher accuracy than before. If the register button is pressed the user will be directed to the register screen shown in Fig.7. Then, they will have to enter their first name, last name, unique username, password and at last choose a user type either admin or researcher and after the registration will be directed to the login screen shown in Fig.8.

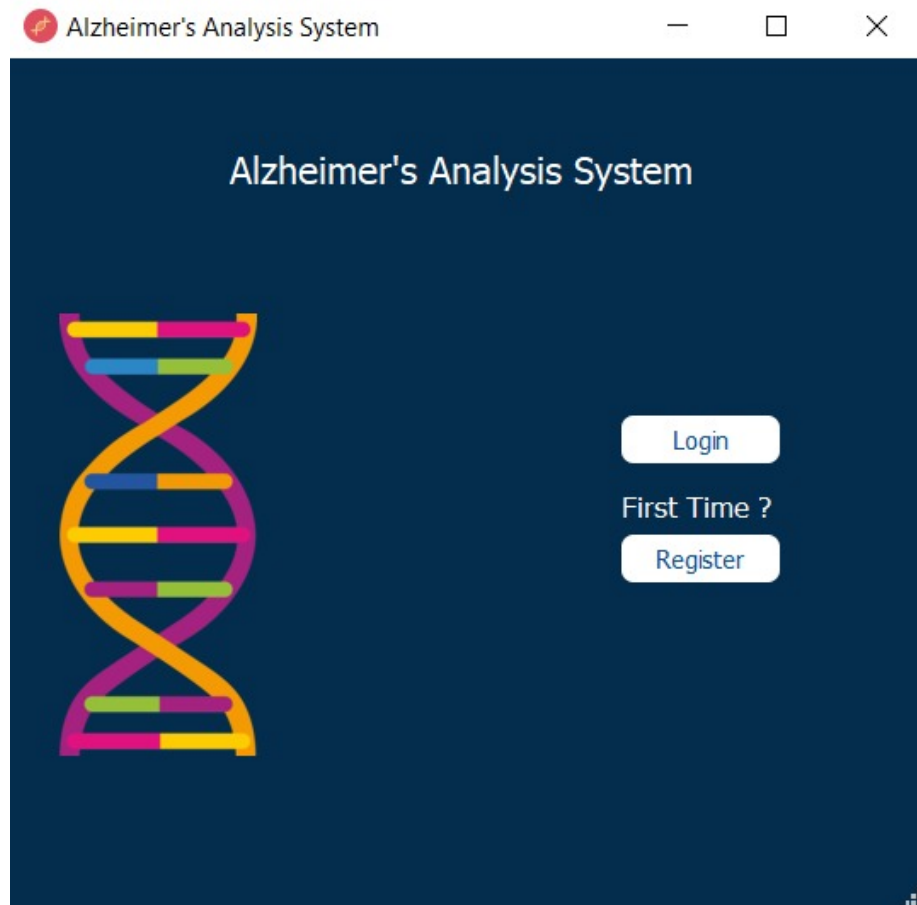


Figure 10: Home Screen

Alzheimer's Analysis System

←

Welcome

First Name

Last Name

Username

Password

Type

☐ Admin

☐ Researcher

[Register](#)




Figure 11: Register Screen

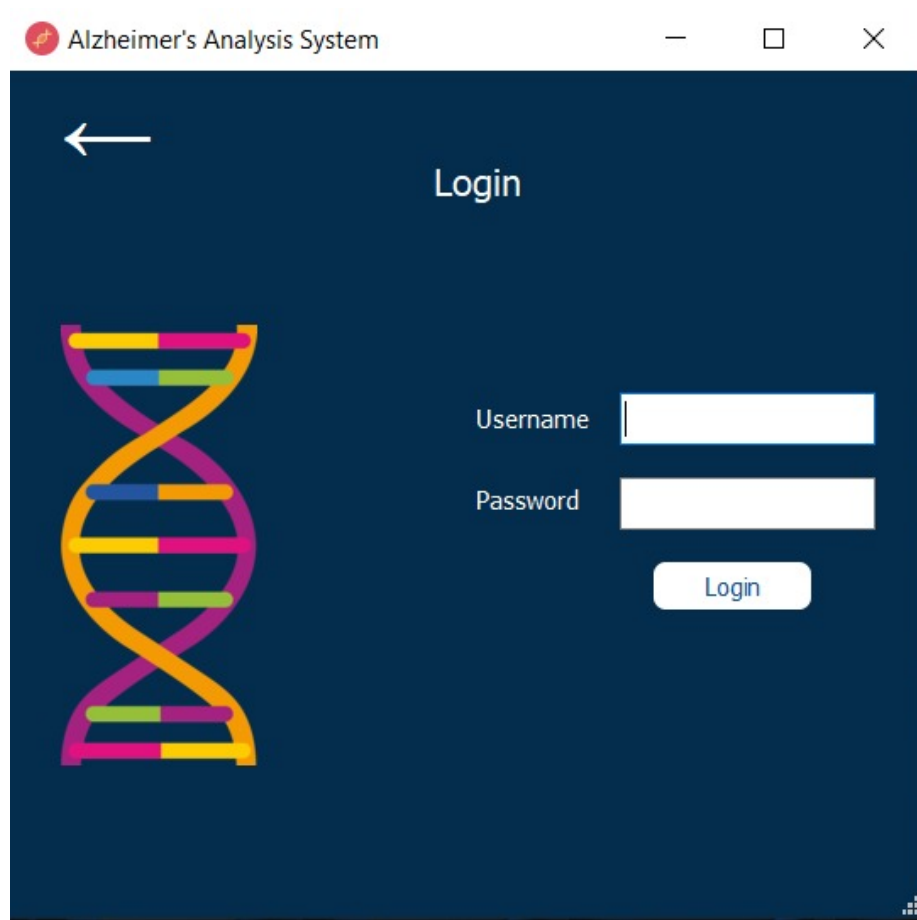


Figure 12: Login Screen

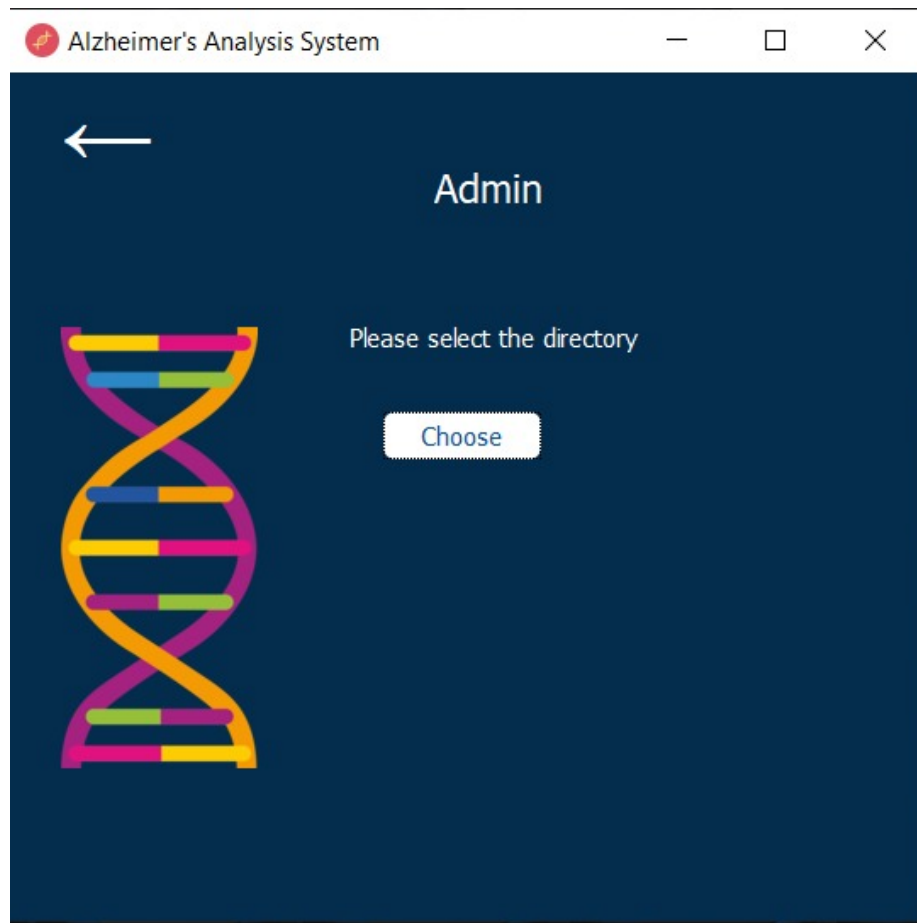


Figure 13: Admin Screen

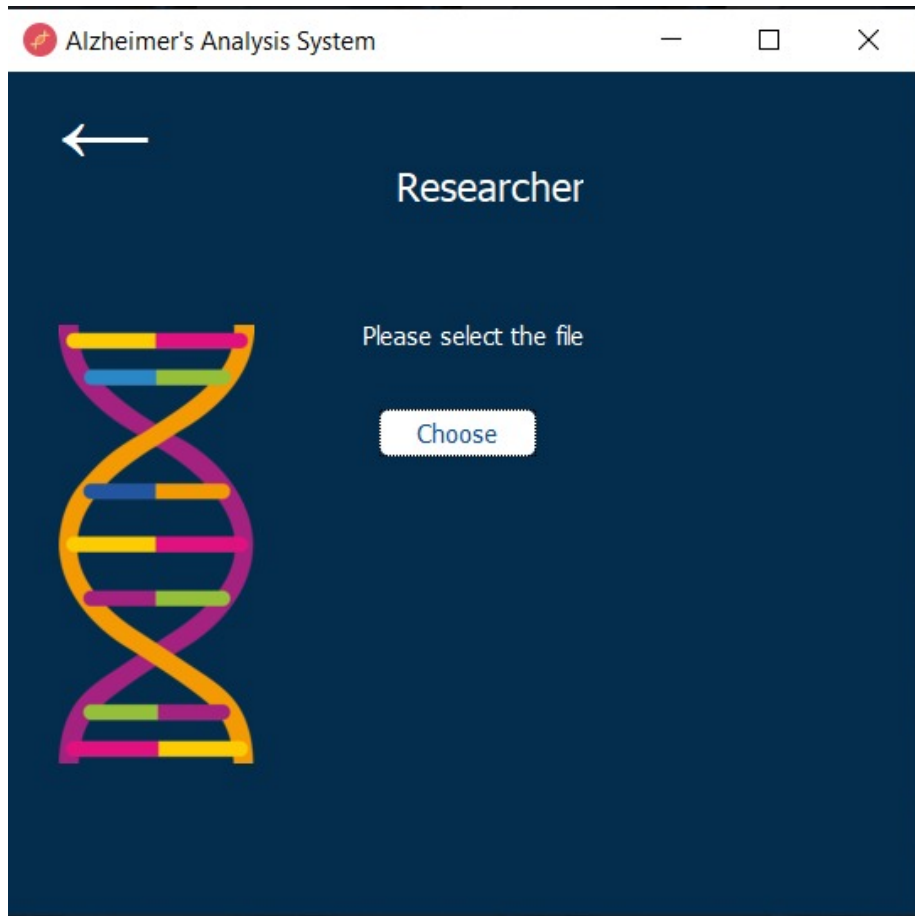


Figure 14: Researcher Screen

7 Requirements Matrix

| Name | Requirement Id | Type | Description | Module | Status |
|-----------------------|----------------|----------|--|-----------------------------|-----------|
| Upload DNA | 3.3 | Required | Lab technicians will be able to upload the sample DNA. | Lab technicians /Researcher | Completed |
| View Result | 3.4 | Required | Outputs the result of the analysis to the user. | Lab technicians /Researcher | Completed |
| Check medical history | 3.6 | Required | By using the patients SSN it will retrieve the medical history of the patient. | Lab technicians /Researcher | Pending |
| Filter | 3.8 | Required | Filters all the unnecessary data that may alter the process | Preprocessing | Completed |
| MergeCsv | 3.9 | Required | Merges all the .csv files in a given directory into one | Preprocessing | Completed |
| Conversion | 3.10 | Required | Converts data into numerals in order to be handled by an algorithm | Preprocessing | Completed |
| Cluster | 3.11 | Required | Clusters a given dataset by using kmeans and minibatch kmeans to cluster the diseased samples into 3 | Lab technicians /Researcher | Completed |
| Searcher | 3.12 | Required | Extracts the four desired chromosomes out of the WGS | N/A | Completed |
| Remover | 3.13 | Required | Cuts the unnecessary parts from the WGS chromosomes | N/A | Completed |
| ToCsv | 3.14 | Required | Converts WGS files into .csv and splits the sequence into threes each in a cell | Preprocessing | Completed |
| CrossReference | 3.15 | New | Cross references the snps in the datasets and outputs a new dataset with common snps | Preprocessing | Completed |
| GetScore | 3.16 | New | Gets the score of the patient in each of the SNPs | Preprocessing | Completed |
| Transpose | 3.17 | New | Reshapes the dataset into the required shape | Preprocessing | Completed |
| SVRClassifier | 3.18 | New | Classifies the given dataset using support vector regression | Classification | Completed |

8 References

References

- [1] Mariette Awad and Rahul Khanna. Support vector regression. In *Efficient Learning Machines*, pages 67–80. Springer, 2015.