Jordan University of Science and Technology

Faculty of Computer and Information Technology

Department of Computer Engineering

CPE 592: Graduation Project Final Report Disease Diagnosis Using Machine Learning (DiagnoseMe)

Amal Qazaq

Ahmad Drabkah

Abd Alnaser Alshare

Osama Shatat

Supervised By

Dr. Inad Aljarrah

Table of Contents

I. DECLARATION

We declare that this work has been done by ourselves and no portion of the work contained in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

II. ACKNOWLEDGMENT

The praises are due to Allah, The Lord of the Worlds, and may the prayers of blessing of Allah be upon Prophet Muhammad, the chosen, the trustworthy and upon his family and all his companions.

We thank all faculty members in the department of computer engineering. We know that all words won’t be enough to thank you for all your efforts, but this is the least we can do. You gave us the knowledge that we need to become dependable engineers through the past 5 years, without you we couldn’t reach this far and we are lucky to have you. We want to give a special thanks to our supervisor Dr. Inad Aljarrah who guided us through the entire process of this project. We wish you all the best in your amazing career and we are going to miss you.

III. Abstract

Our web application is used as a platform for patients to get their diagnosis by navigating through our provided list of illnesses that they think or may have, then they enter their medical info like age, gender, weight, calories, blood pressure,..etc. after that the application predict if they have the disease or not by using Machine Learning (ML) and show the result on the screen.
Basically, it helps patients to get their diagnosis in an elegant and easy way without the need to go to a hospital or doctor’s clinic.

IV. Introduction

This web application, designed with a goal to make people life easier by making the diagnosis process faster, easier, more elegant and modern. The application is used as a platform to host patients who need to diagnose their illness when they have any concerns or doubts about their health situation. The application is developed to be used on any mobile or computer as long as it has a web browser. It should, and will be available anytime, anywhere in the user's hands. This project presents the main steps that led to the creation of the DiagnoseMe application, starting from the professional practice constraints, then the design of system architecture. The last step was implementing the system.

Machine learning advancement and online or electronic solutions has proven to be affective and easy to learn. Many aspects has already been studied and published fixing major problems such as transportation, eLearning, and more. This application here aimed at fixing as many problems it can handle, to help the community, people in need, and to share a positive vibe with everyone. In our daily lives, we intend to go to hospitals to get diagnosis and prescriptions from doctors, but it is not as easy as that. You need to wait in line all day just to schedule an appointment; it may take from 1 week to months.

The motivations of creating this project were many, as it intended to use the old and traditional methods to make better and more customizable solution for a variety of reasons.

What has been a slow, repetitive and tiering actions and featured is going to be a better and more elegant solution for all previous problems. The goal here is to design, build and create a web application, which is designed with the ability to diagnose patients' illness in a modern, fast and elegant way.

V. Professional Practice Constraints

1. **Engineering standards [1]:**
   The International Standard defines a consistent framework for software life cycle procedures, completed with well-defined language that the software industry may relate to. It describes the procedures, activities, and duties that must be followed during the purchase of a software system, product, or service, as well as the supply, development, operation, maintenance, and disposal of software products. This is performed through involving stakeholders in the process, with the ultimate goal of ensuring customer happiness. This International Standard covers the purchase of software systems, goods, and services, as well as the supply, development, operation, maintenance, and disposal of software products and the software portion of any system, whether performed internally or externally. The software element of firmware is included in software. Included are the parts of system definition that are required to give context for software goods and services. This International Standard also outlines techniques for defining, controlling, and enhancing software life cycle processes within a company or project. This International Standard's procedures, actions, and tasks may be used alone or in conjunction with ISO/IEC/IEEE 15288, Systems and software engineering—System life cycle processes, during the acquisition of a system, that comprises software.

2. **Project Management [2]:**
   To implement any program professionally, project management systems must be applied to get the best results in the fastest time and with the least error rate. For any project to be implemented, a clear action plan must be developed from the outset based on clear management foundations that facilitate the implementation of the program and clarify the costs and

benefits of this project. Every system must go through a developmental life cycle, whether it's hardware, software, or a combination of the two. The goal of the lifecycle is to guide complex processes in order to speed up development while avoiding costly mistakes. A large-scale systems project's organizational restrictions are defined by the system development life cycle, which is effectively a phased project model. Planning, creating, testing, deploying, and sustaining information systems are all clearly defined work stages in the systems development lifecycle strategy. And the best system that can be followed to speed up the work and reduce the error rate is by following the system of Agile for software development by certifying this system, many firms have had success meeting client objectives. This iterative approach to program design and development accepts ongoing change and allows teams to break down stages into smaller components in order to deliver work programs in phases. The Agile Software Development Project Manager oversees the development team's work and assists them in staying on schedule to release software iterations on a regular basis. The development team chooses at the start of the sprint what they can do in the time allotted in order to deliver a working software that can be installed at the conclusion. The following are some best practices for Agile software development:

1. Hold daily meetings on a regular basis to sustain communication, hold team members accountable, and keep iterations moving forward.
2. To demonstrate progress to important stakeholders, provide live demos of the final product for each iteration.
3. Before beginning the next iteration, share input from stakeholders and customers with the entire development team.
4. Improve your process on a continuous basis based on input to ensure that each iteration of the next process is better.

3. **Manufacturing limitations:**
   When implementing the project on the ground, it first needs to develop a comprehensive work plan for it that includes:
   1- Determining the appropriate project management method to implement the project in the fastest time and with the least error rate during implementation. In the second stage and so on.
   2- Determine the appropriate work environment that can withstand this amount of data, and develop the system on it without a failure, and provide a system to preserve the data in the event of any failure or any defect to prevent a major loss in the project, which may be cloudy or unhosted.
   3- Determining the work team we need from developers, interface designers, network engineers, administrators to organize the work and others.
   4- Determine the platforms on which the system will be deployed to allow the system to run on different operating systems if necessary.
   5- Testing the system several times to ensure that it guarantees safety standards and does not cause any problems for users, and to ensure the quality of its work based on IEEE Standards for Software Engineering.
   6- Determining the design that this project will adopt in order to take its own identity in front of the world.

4. **Economic Obstacles:**
   Our project to implement it on the ground requires a large budget that covers the costs of its manufacture at first, then the project can cover its expenses, the most important of which are:
   1- The labor costs of those who will assist in the development, preservation and protection of the project.

2- Initial costs include the computer hardware on which the system will be developed, providing a suitable work environment for the team to communicate and organize work among them, whether it is a regular office or a virtual office, the servers on which the system will be upgraded and the provision of support servers on which copies of work are made to ensure that the work is not lost work, and others
3- Determining the time period for each stage of project implementation, such as defining the period of preparation, testing and initial launch, and then launching it completely.
4- The costs of the marketing plan that will be developed for the project to disseminate it and make people trust it, use it and deliver it to the largest number of users in order to benefit from it, and the costs of implementing this plan.

5. Sustainability:

The sustainability of any project depends on several basic factors needed, for example, the meat sale project has existed since the foundation of the community, and our project is a diagnostic system for diseases and can be described as giving advance warning to the person of the possibility of contracting a certain disease, and this may help him to preserve his life in treatment the disease before its penetration, and yes, the issues of death and life are in the hands of God alone, but we have to take into account the causes. The answer is yes, with the continuous development of lifestyles in all courses, unfortunately bad habits develop with them, including movement habits that have become very few, eating regimen, electromagnetic waves that have become surrounding us, etc..... The possibility of a person suffering from heart disease, cancer, diabetes and others has increased, the existence of such a system may greatly help society and is evidence of its sustainability, and with its continuous development, it may become a strong supporter for doctors.

6. **Environmental constraints:**
   Our project does not have any damage to the environment. It is a technical project. On the contrary, it can preserve it. If it is developed in the required manner, it may reduce the medical tools used in the detection of diseases, reducing waste and thus preserving the environment.

7. **Health and Safety Restrictions:**
   We have already mentioned that our project is based on paying attention to human health and warning him against catching diseases at an early stage, so that he alerts the patient to urgently treat them before they worsen, and the system may develop until it becomes capable of diagnosing and determining the appropriate treatment method. It has seemed at the present time to rely on technology completely in several areas, but with regard to human life, we have not reached absolute confidence in handing over human life to a machine. For example, all Tesla cars are capable of fully automated driving, but they are not authorized to do so, because even a small error may occur and threaten the life of any person. However, semi-automated driving is permitted, which are the driving systems that help the driver to drive on the main roads for his convenience and Alert when danger is expected. Therefore, the project, in order for it to be authorized to work, must be tested and ensured that the error rate is almost not until it is approved.

8. **Ethical limitations:**
   The largest and most important human right is the right to privacy based on protecting the privacy of the person from any exposure, and in our case here protecting the person by preserving the privacy of his illness and

symptoms in his own and not publishing, publicizing and accessing it in the first place, except for the persons authorized to do so. Such as the doctor responsible for him or a member of his family of the first degree.

9. **Limitations of Social Standards:**
Society's culture and educational level depend on the speed of its acceptance of change and acceptance of everything that is new, especially in the field of health. The world has suffered from the Corona pandemic, and when the vaccine was created for this disease, the topic took a long time to convince society to take it, especially in our society, but in the age of the Internet, the world has become as a small village, the concept of globalization has had a great impact on the speed of society's acceptance of everything new. In our case we do not have any violation of social or religious values or norms, as our application helps people and society by preventing the spread of diseases and maintaining the safety of society.

10. **Political restrictions:**
It differs from one country to another in that each country has its own laws that impose them on its territory. For example, the automated driving which concerns cars that are entertaining and their ability to drive fully automated, in some regions of the world this system is allowed so that this area is suitable for such this technology, the ease of licensing a system like this varies from one country to another. In our case it is healthcare application so it will have to match several political conditions as it is work with very sensitive term in the whole world not only country which is the person privacy so it need to be reliable and take many approvals from international organization like World Health Organization and it will be placed under government supervision to ensure its safety and readiness for use by the public.

VI. System Architecture and Design

Our software contains three main parts ML models, web application and the user interface (UI). Each part has been developed and implemented independently of the others parts using a programing language.

The using of the software and dataflow in the application is like the following

1. Client open the website and navigate to disease he/she want to diagnose then start fill diagnose form with his/her medical data.
2. After client submit the diagnose form the UI will request the web app for the result which in turn call another app with client request data.
3. The app that called by web app will select the model that the client want to diagnose its disease and return the result to web app which will send it to the UI.

Fig. 1 will illustrate this work more:

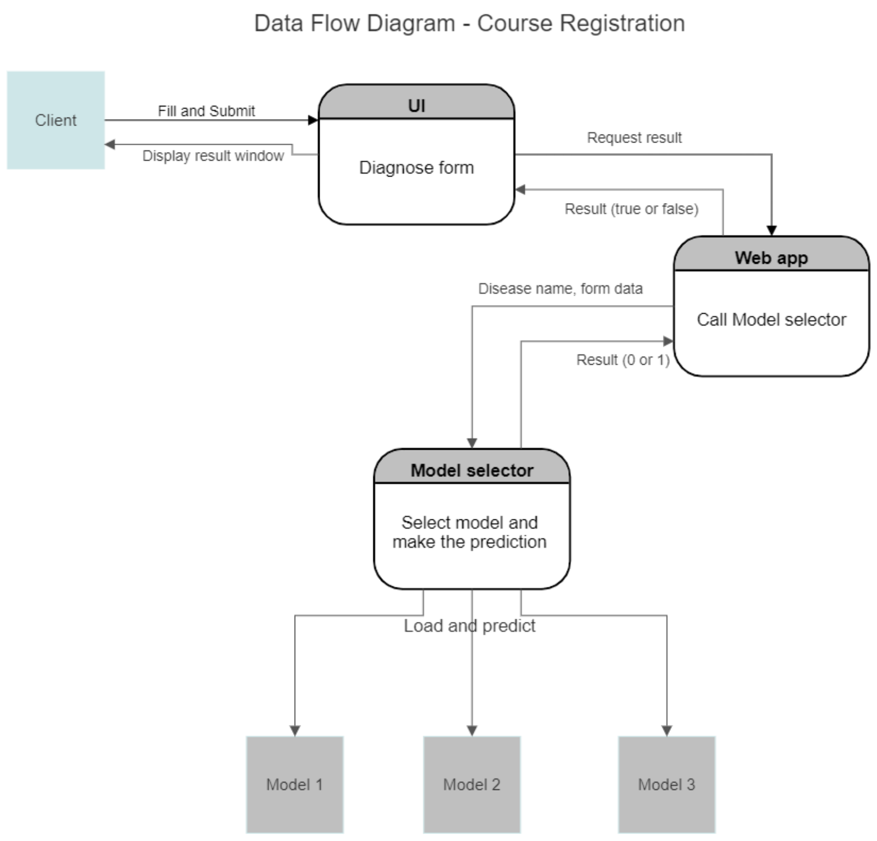
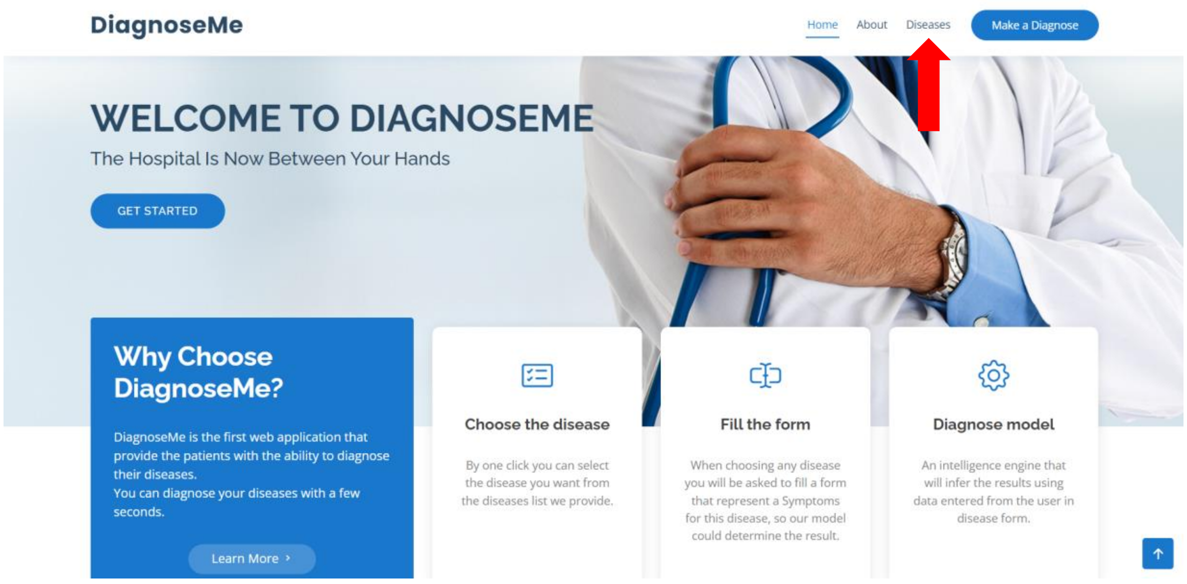


Fig. 1 – Dataflow diagram

Like we see in the above figure each part of the application work independent of the other then when the client request a diagnose the UI delegate the request to the web app which in turn delegate it to the model selector. This design called Chain of Responsibility which is one of the most popular design pattern in software engineering but in our case we applied it to application level not class level.

1. User interface :

   This is the part which user interact with; we implemented our UI to be user friendly that can be used from any person and any age. We implemented it using the most common language used in this field HTML, CSS and JavaScript. The UI (website) entry point in the home page from it you can navigate for about and diseases pages. The important page is the diseases page which you can select the disease you want from it then it will take you to the diagnose form page. You can access it by typing diagnoseme.team in the browser.



WELCOME TO DIAGNOSEME

The Hospital Is Now Between Your Hands

Why Choose DiagnoseMe?

DiagnoseMe is the first web application that provide the patients with the ability to diagnose their diseases.
You can diagnose your diseases with a few seconds.

Choose the disease

By one click you can select the disease you want from the diseases list we provide.

Fill the form

When choosing any disease you will be asked to fill a form that represent a Symptoms for this disease, so our model could determine the result.

Diagnose model

An intelligence engine that will infer the results using data entered from the user in disease form.

Fig. 2 - Home page

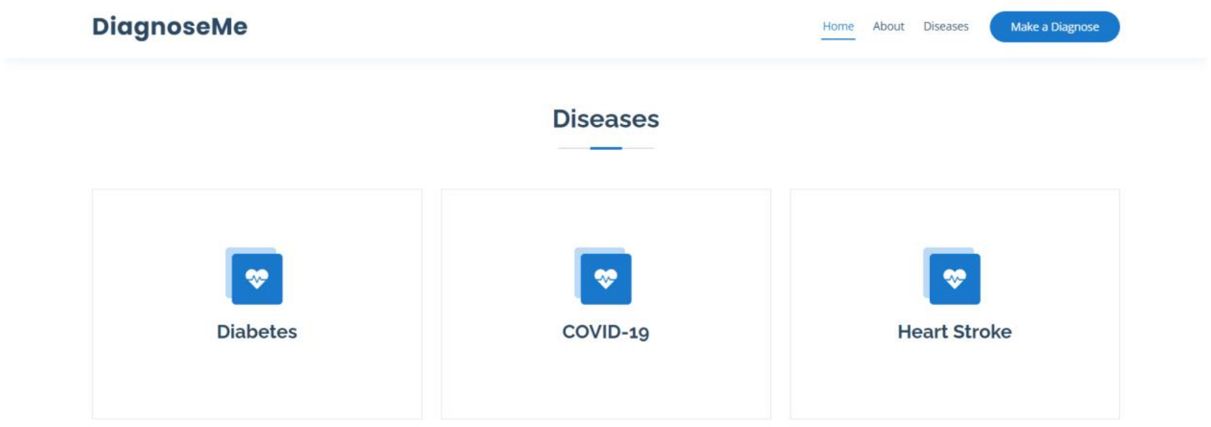
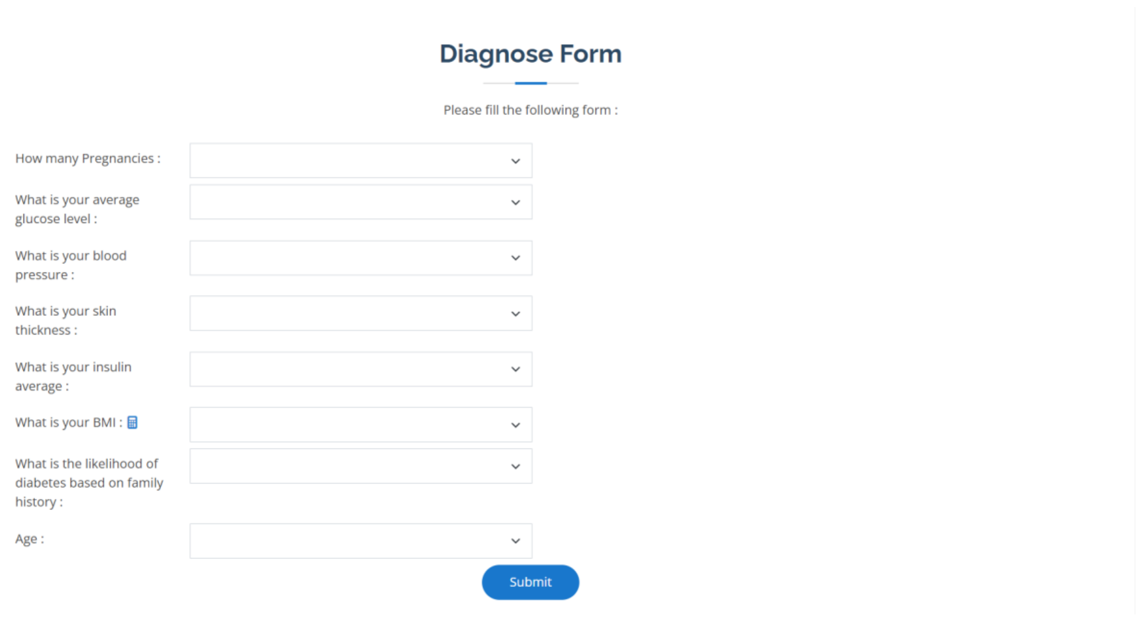


Fig. 3 - Diseases page



Diagnose Form

Please fill the following form :

How many Pregnancies :	
What is your average glucose level :	
What is your blood pressure :	
What is your skin thickness :	
What is your insulin average :	
What is your BMI :	
What is the likelihood of diabetes based on family history :	
Age :	

Fig. 4 – Diabetes diagnose form page

When the client submit its information JavaScript will send a request to the web application to get the diagnose result and then display it in a window tell the client his/her result is positive or negative.

2. Web application :

   We implemented this part using Spring Boot, which is a Java framework, that enables to build RestAPI that can be used from a third party, in our case the third party is the UI especially Ajax request that send by the JavaScript to get the diagnose result.

For each ML model in our system, we implement a Controller, which represent the RestAPI that handle the request and trigger the model to make the prediction and then respond by the predicted result.

In order to be able to trigger the model from inside the controller we need interface as web app in implement in Java and the ML model in python. We implement this interface by create RestAPI using Flask which is a python framework for web, to load the desired model and make the prediction then respond by the result to the web app.

3. ML models :

   This part has been implemented using TesorFlow, which is python platform for creating the models. All of the models is Deep Neural Network (DNN), for each disease in our system there is a model we train it on a prober dataset and validate its accuracy, precision, recall and F1 score to make sure from the correctness of the prediction. Before create the model we implement some of data preprocessing steps like remove features that have bad correlation, encode the values (convert values from words to numbers) …etc.

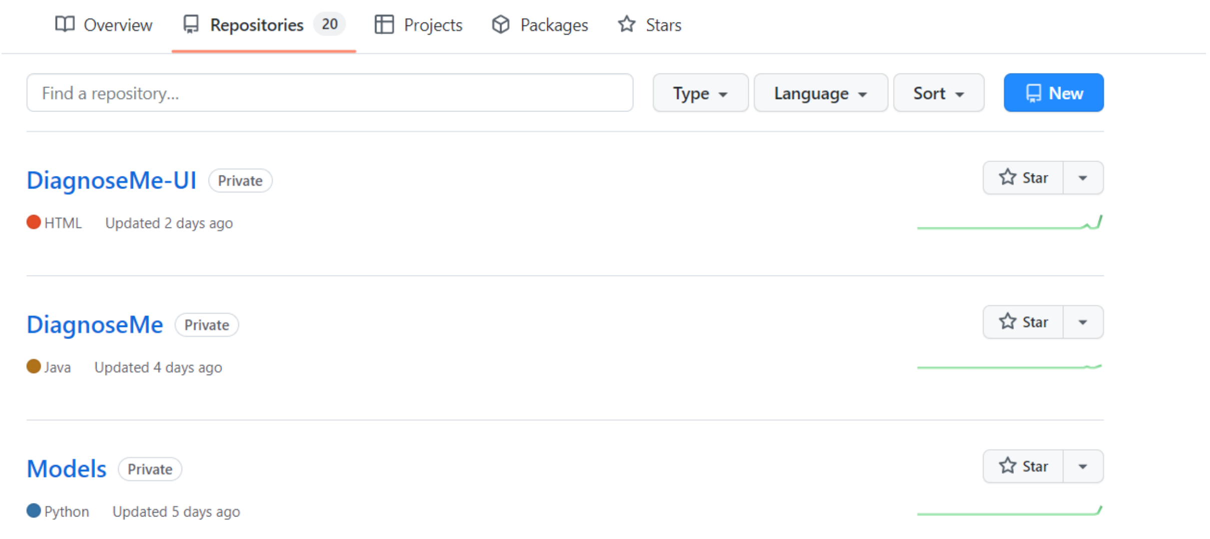
   After train the model and validate its correctness we save it as .h5 file so model selector (Python RestAPI) can load and use it.

   Further implementation details of the above parts will be mention and declare will be descript later.

4. Deployment :

   This phase of our development for the application to make it available for anyone from anywhere. We deploy our app using Microsoft Azure [3] and GitHub [4] services. First we use GitHub as version control system to manage the shared work and then benefit from GitHub Actions which is a

CI/CD environment that enable us to write a workflow that automatically work when any change push to the repository and deploy the changes to the Azure server.

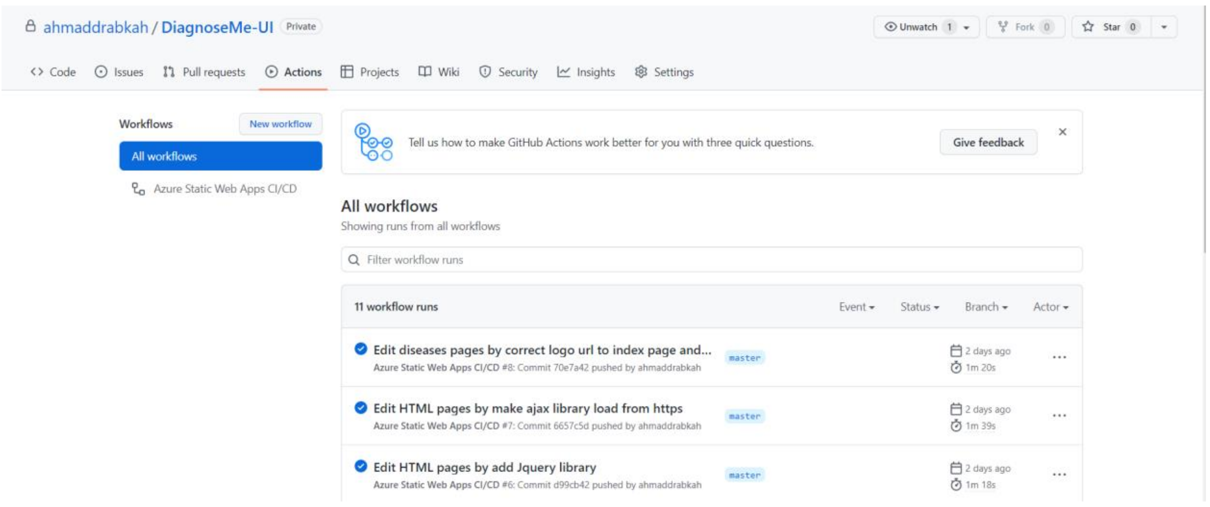


DiagnoseMe-UI Private
HTML Updated 2 days ago

DiagnoseMe Private
Java Updated 4 days ago

Models Private
Python Updated 5 days ago

Fig. 5 – GitHub repositories



Tell us how to make GitHub Actions work better for you with three quick questions.

All workflows
Showing runs from all workflows

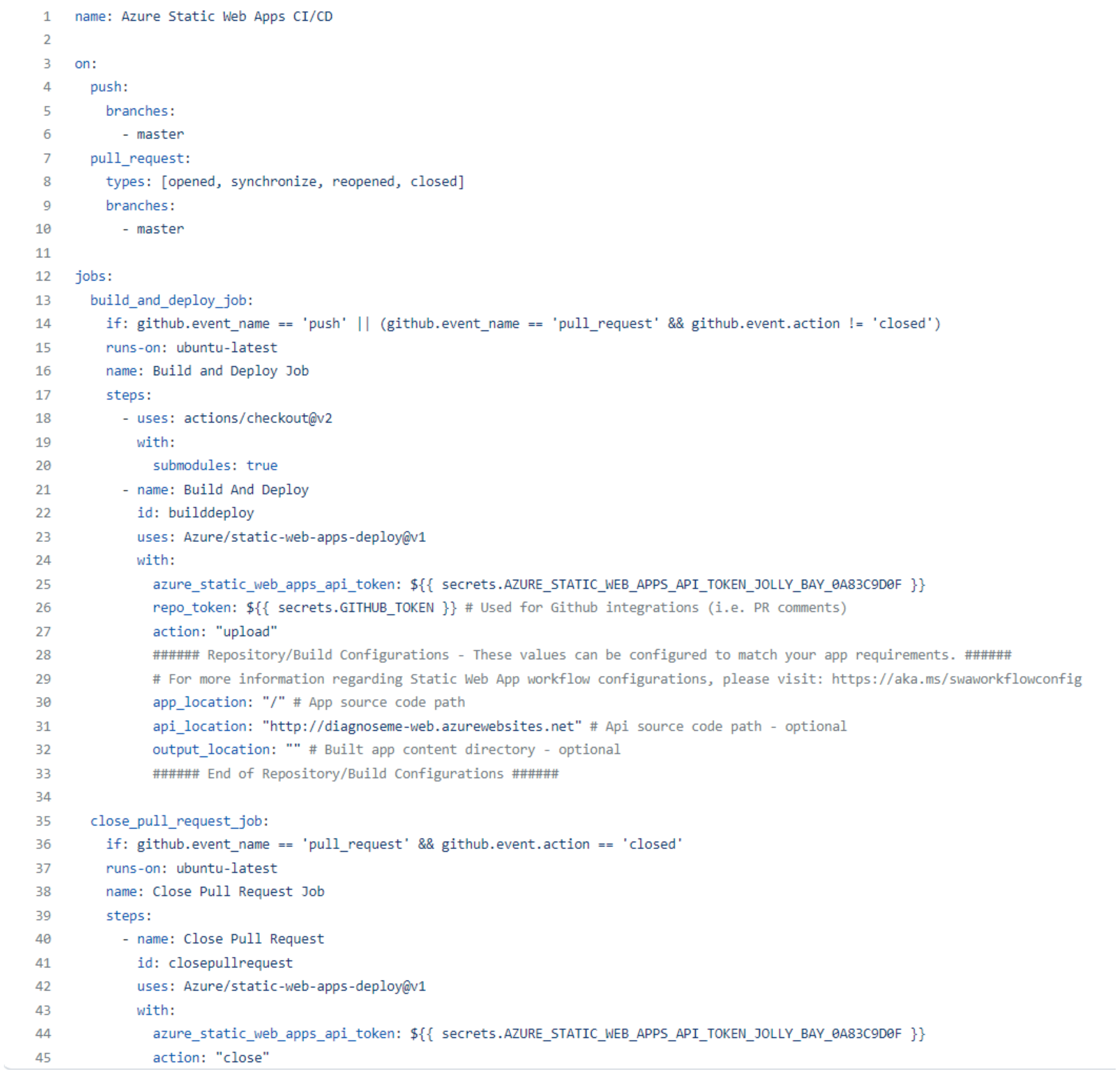
11 workflow runs

Edit diseases pages by correct logo url to index page and... master 2 days ago 1m 20s
Azure Static Web Apps CI/CD #8: Commit 70e7a42 pushed by ahmaddrabkah

Edit HTML pages by make ajax library load from https master 2 days ago 1m 39s
Azure Static Web Apps CI/CD #7: Commit 6657c5d pushed by ahmaddrabkah

Edit HTML pages by add Jquery library master 2 days ago 1m 18s
Azure Static Web Apps CI/CD #6: Commit d99cb42 pushed by ahmaddrabkah

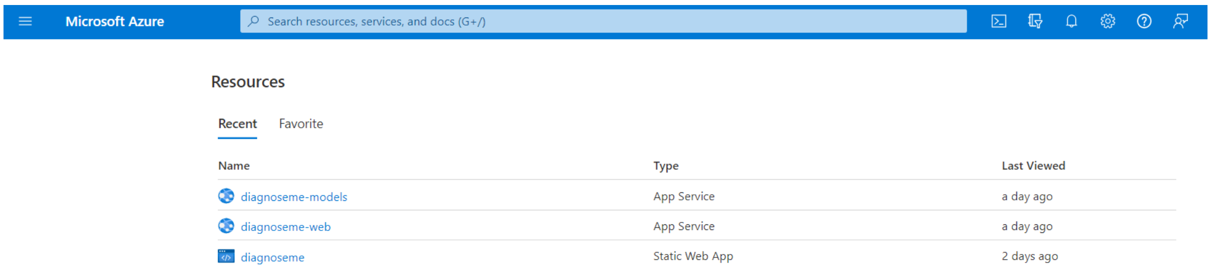
Fig. 6 – GitHub Actions for UI



```yaml
name: Azure Static Web Apps CI/CD

on:
  push:
    branches:
      - master
  pull_request:
    types: [opened, synchronize, reopened, closed]
    branches:
      - master

jobs:
  build_and_deploy_job:
    if: github.event_name == 'push' || (github.event_name == 'pull_request' && github.event.action != 'closed')
    runs-on: ubuntu-latest
    name: Build and Deploy Job
    steps:
      - uses: actions/checkout@v2
        with:
          submodules: true
      - name: Build And Deploy
        id: builddeploy
        uses: Azure/static-web-apps-deploy@v1
        with:
          azure_static_web_apps_api_token: ${{ secrets.AZURE_STATIC_WEB_APPS_API_TOKEN_JOLLY_BAY_0A83C9D0F }}
          repo_token: ${{ secrets.GITHUB_TOKEN }} # Used for Github integrations (i.e. PR comments)
          action: "upload"
          ###### Repository/Build Configurations - These values can be configured to match your app requirements. ######
          # For more information regarding Static Web App workflow configurations, please visit: https://aka.ms/swaworkflowconfig
          app_location: "/" # App source code path
          api_location: "http://diagnoseme-web.azurewebsites.net" # Api source code path - optional
          output_location: "" # Built app content directory - optional
          ###### End of Repository/Build Configurations ######

  close_pull_request_job:
    if: github.event_name == 'pull_request' && github.event.action == 'closed'
    runs-on: ubuntu-latest
    name: Close Pull Request Job
    steps:
      - name: Close Pull Request
        id: closepullrequest
        uses: Azure/static-web-apps-deploy@v1
        with:
          azure_static_web_apps_api_token: ${{ secrets.AZURE_STATIC_WEB_APPS_API_TOKEN_JOLLY_BAY_0A83C9D0F }}
          action: "close"
```

Fig. 7 – Workflow scripts for UI

In Azure side, we use its services to deploy our app, we deploy each part into a server and they communicate throw Http request as mentioned before.


Microsoft Azure

Search resources, services, and docs (G+/)

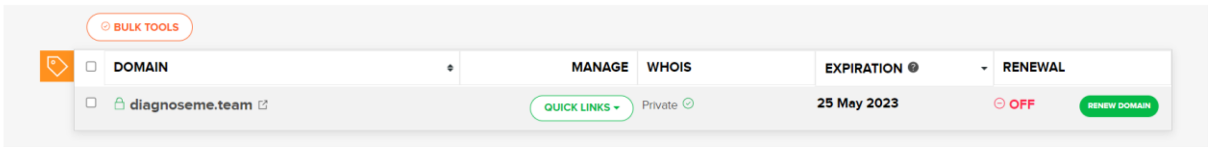
Resources

Recent Favorite

Name	Type	Last Viewed
diagnoseme-models	App Service	a day ago
diagnoseme-web	App Service	a day ago
diagnoseme	Static Web App	2 days ago

Fig. 8 – Azure service

For the website to be accessed with easy URL we need domain name as Azure provide a random domain name that is so long with no meaning, so we got our domain name from Name.com [5] and liked it to our web site in Azure using CNAME.


BULK TOOLS

DOMAIN	MANAGE	WHOIS	EXPIRATION	RENEWAL	
diagnoseme.team	QUICK LINKS	Private	25 May 2023	OFF	RENEW DOMAIN

Fig. 9 – Domain Name

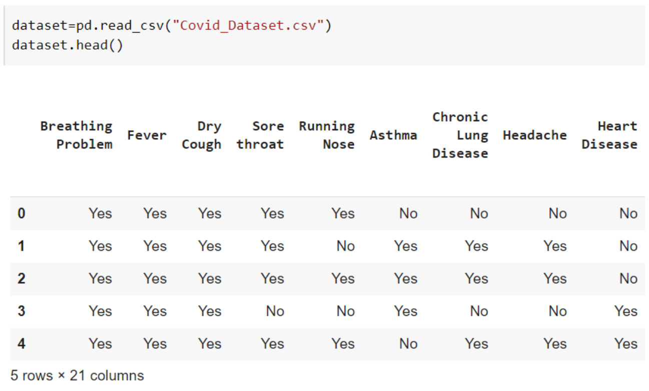
VII. Software Implementation

In this part, we will discuss the implementation of our software starting from ML parts to UI part.

1. ML models :

In our implementation, there are three models each one for a disease COVID-19, Diabetes and Heart stork. All the models are implemented as mentioned before using TensorFlow [6] platform to create a DNN [7], before create DNN there was a preprocessing steps to prepare data to be used in model training. We will discuss COVID-19 model to show the implementation as all model are the same but the differ in preprocessing steps:

This model is for diagnosis "Coronavirus", which is the most recent disease these days. First, this dataset consist of 5434 rows with 21 features.



```
dataset=pd.read_csv("Covid_Dataset.csv")
dataset.head()
```

	Breathing Problem	Fever	Dry Cough	Sore throat	Running Nose	Asthma	Chronic Lung Disease	Headache	Heart Disease
0	Yes	Yes	Yes	Yes	Yes	No	No	No	No
1	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No
2	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
3	Yes	Yes	Yes	No	No	Yes	No	No	Yes
4	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes

5 rows × 21 columns

Fig. 10 – COVID-19 Dataset

The values is “Yes” or “No” so the first step to be done is to encode these values for numbers, then we study the correlation to select the best features.

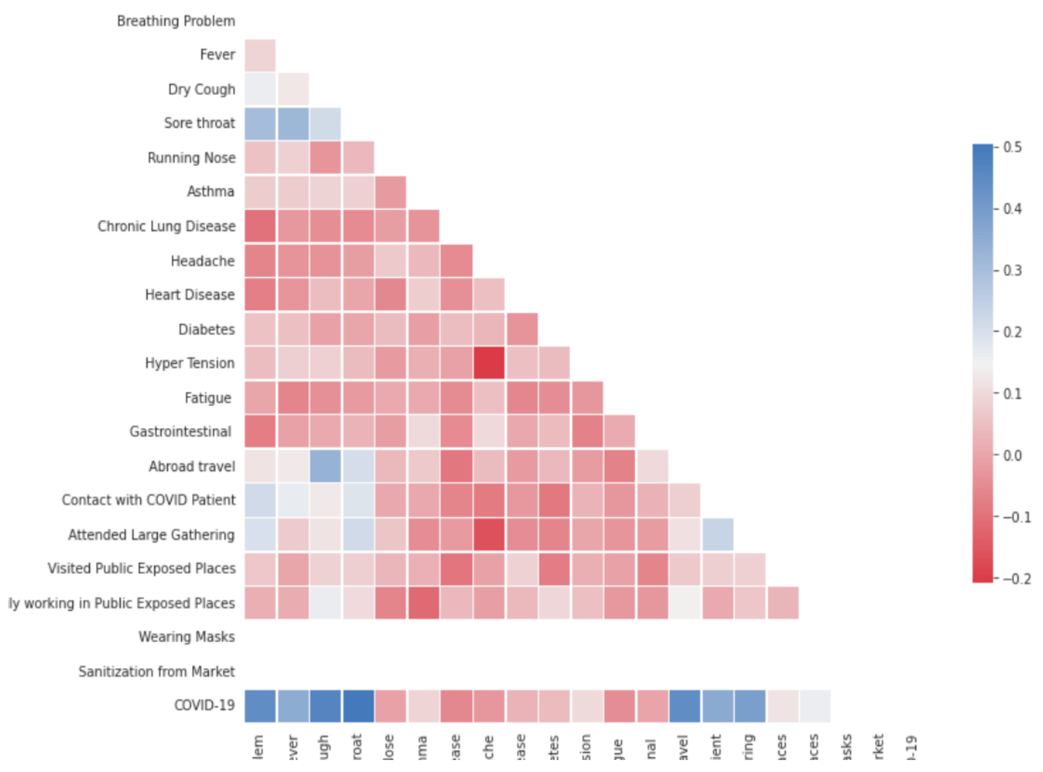
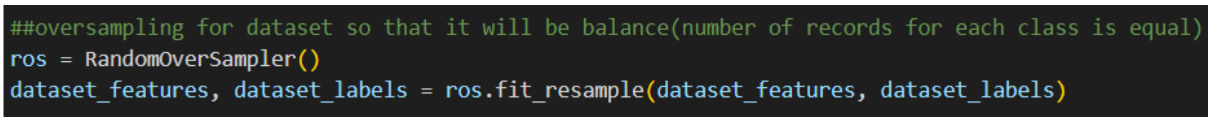
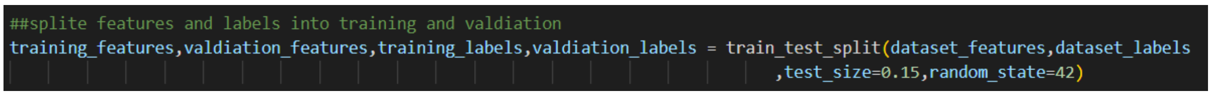


Fig. 11 – Correlation plot for COVID-19 model

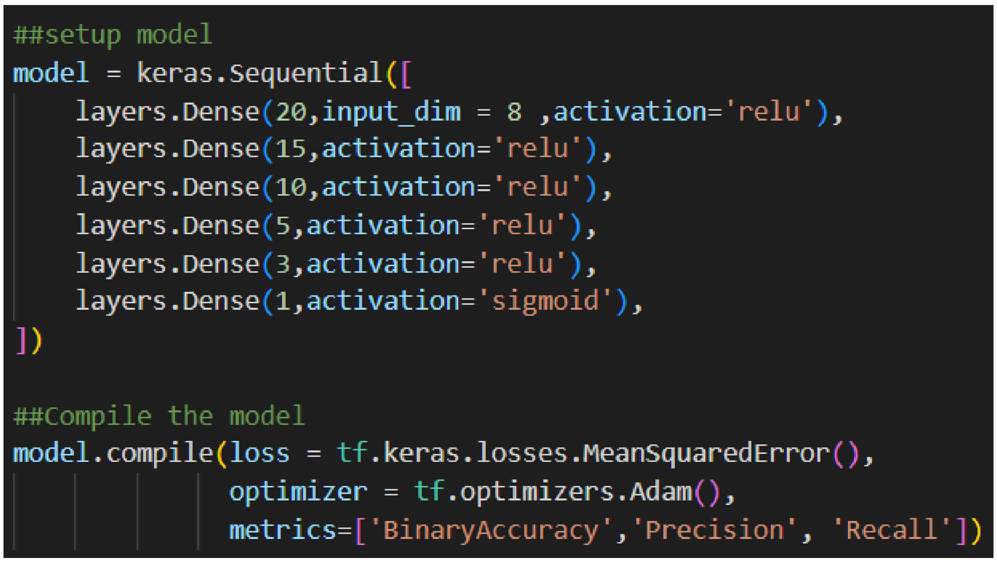
As the dataset is unbalanced (target classes are not equals) the model will have a bad performance so we resolved this by resample the dataset (make number of minor class equal to major class by redundant).


```
##oversampling for dataset so that it will be balance(number of records for each class is equal)
ros = RandomOverSampler()
dataset_features, dataset_labels = ros.fit_resample(dataset_features, dataset_labels)
```
Fig. 12 – Over sampling

Then split dataset into training and validation randomly as we used the validation dataset to check correctness values accuracy, precision, recall and F1 score.


```
##splite features and labels into training and valdiation
training_features,valdiation_features,training_labels,valdiation_labels = train_test_split(dataset_features,dataset_labels
                                                                                            ,test_size=0.15,random_state=42)
```
Fig. 13 – Dataset split

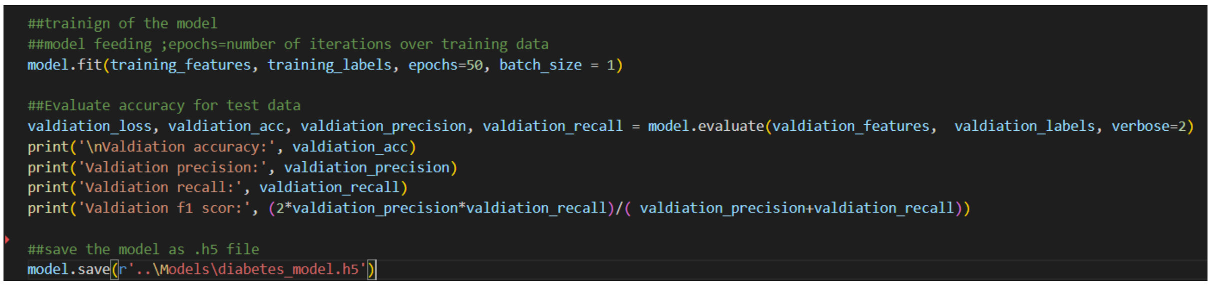
Then we create the model by specifying the number of input neurons (equal to number of feature select from dataset), number of hidden layers and neurons number in each one and activation function of each layer. In the compile part, we specify the loss function, the optimizer and metrics (accuracy, precision and recall).



```python
##setup model
model = keras.Sequential([
    layers.Dense(20,input_dim = 8 ,activation='relu'),
    layers.Dense(15,activation='relu'),
    layers.Dense(10,activation='relu'),
    layers.Dense(5,activation='relu'),
    layers.Dense(3,activation='relu'),
    layers.Dense(1,activation='sigmoid'),
])

##Compile the model
model.compile(loss = tf.keras.losses.MeanSquaredError(),
              optimizer = tf.optimizers.Adam(),
              metrics=['BinaryAccuracy','Precision', 'Recall'])
```

Fig. 14 – Model creation and compilation

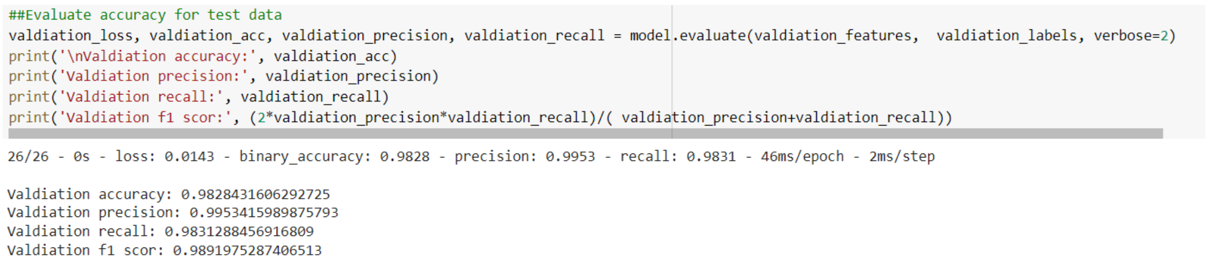
Then train the model after that we validate it using validation dataset then save it as .h5 file to be load and use in model loader (will descript later).



```python
##trainign of the model
##model feeding ;epochs=number of iterations over training data
model.fit(training_features, training_labels, epochs=50, batch_size = 1)

##Evaluate accuracy for test data
valdiation_loss, valdiation_acc, valdiation_precision, valdiation_recall = model.evaluate(valdiation_features,  valdiation_labels, verbose=2)
print('\nValdiation accuracy:', valdiation_acc)
print('Valdiation precision:', valdiation_precision)
print('Valdiation recall:', valdiation_recall)
print('Valdiation f1 scor:', (2*valdiation_precision*valdiation_recall)/( valdiation_precision+valdiation_recall))

##save the model as .h5 file
model.save(r'..\Models\diabetes_model.h5')
```

Fig. 15 – Model train and validation

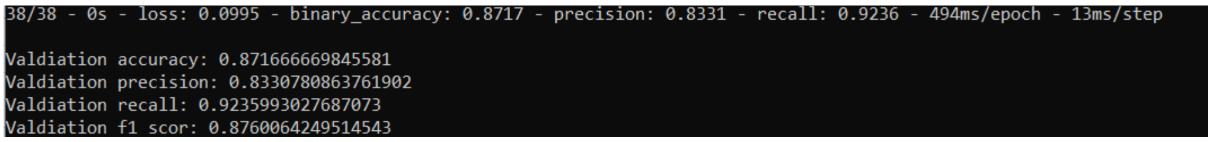
All others model are the same and maybe differ in a small part like preprocessing steps. The accuracy we got for these dataset [8] shown in the following figures.



```
##Evaluate accuracy for test data
valdiation_loss, valdiation_acc, valdiation_precision, valdiation_recall = model.evaluate(valdiation_features,  valdiation_labels, verbose=2)
print('\nValdiation accuracy:', valdiation_acc)
print('Valdiation precision:', valdiation_precision)
print('Valdiation recall:', valdiation_recall)
print('Valdiation f1 scor:', (2*valdiation_precision*valdiation_recall)/( valdiation_precision+valdiation_recall))
```

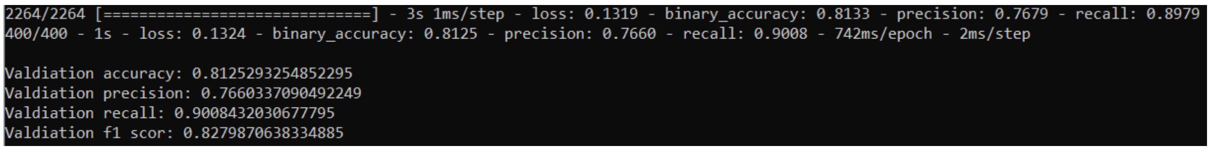
```
26/26 - 0s - loss: 0.0143 - binary_accuracy: 0.9828 - precision: 0.9953 - recall: 0.9831 - 46ms/epoch - 2ms/step

Valdiation accuracy: 0.9828431606292725
Valdiation precision: 0.99534159898875793
Valdiation recall: 0.9831288456916809
Valdiation f1 scor: 0.9891975287406513
```

Fig. 16 – COVID-19 model summary



```
38/38 - 0s - loss: 0.0995 - binary_accuracy: 0.8717 - precision: 0.8331 - recall: 0.9236 - 494ms/epoch - 13ms/step

Valdiation accuracy: 0.871666669845581
Valdiation precision: 0.8330780863761902
Valdiation recall: 0.9235993027687073
Valdiation f1 scor: 0.8760064249514543
```

Fig. 17 – Diabetes model summary



```
2264/2264 [==============================] - 3s 1ms/step - loss: 0.1319 - binary_accuracy: 0.8133 - precision: 0.7679 - recall: 0.8979
400/400 - 1s - loss: 0.1324 - binary_accuracy: 0.8125 - precision: 0.7660 - recall: 0.9008 - 742ms/epoch - 2ms/step

Valdiation accuracy: 0.8125293254852295
Valdiation precision: 0.7660337090492249
Valdiation recall: 0.9008432030677795
Valdiation f1 scor: 0.8279870638334885
```

Fig. 18 – Heart stroke model summary

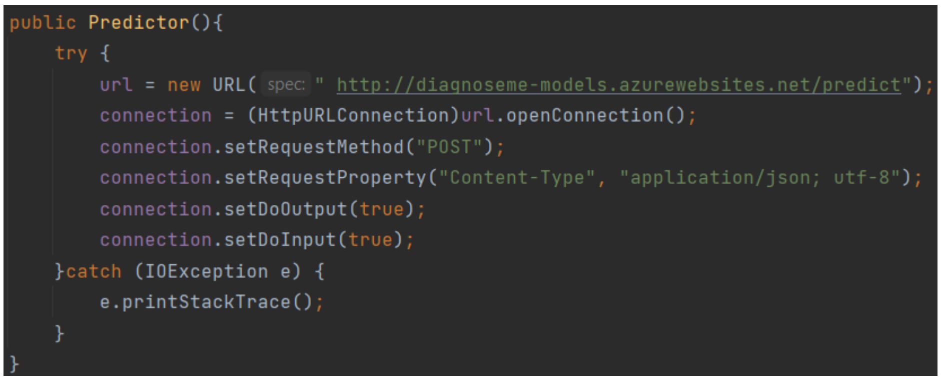
You can access the code in GitHub repository in creating models code folder using this link: https://github.com/ahmaddrabkah/Models.git .

2. **Model Loader :**

   This part is responsible to implement a RestAPI to be an interface between web app and models. This application is implement using Flask framework as web app send a HTTP request with post method the request most contain the model name (disease to be diagnosed) and data input by the user. Then it will load the desired model from Models folder and make the prediction using data from the request and return “Yes” if prediction value is greater than or equal than 0.5, which indicate that the client has the disease otherwise it will return “No”. You can access the code from the link above in model_loader.py file.

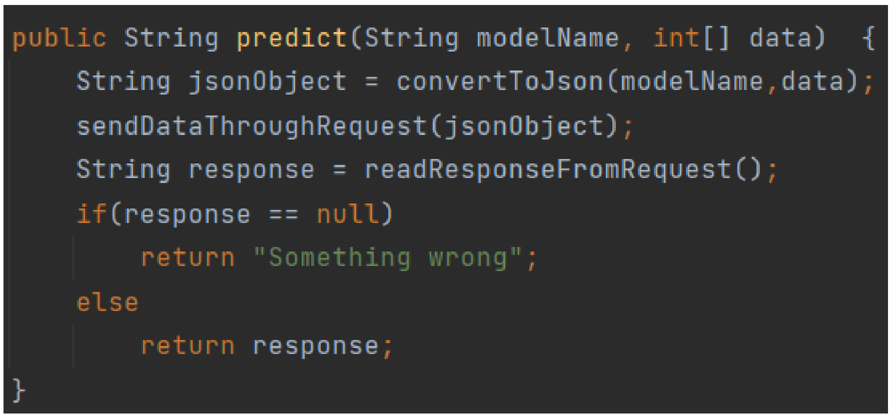
3. **Web Application :**

   This part implemented using Java Spring Boot [9], to create a RestAPI that used by UI to run the desired model. First, a Predictor class is responsible to connect to the API implemented by model loader and get the prediction result. This class establish the connection with model loader API.



```
public Predictor(){
    try {
        url = new URL( spec: " http://diagnoseme-models.azurewebsites.net/predict");
        connection = (HttpURLConnection)url.openConnection();
        connection.setRequestMethod("POST");
        connection.setRequestProperty("Content-Type", "application/json; utf-8");
        connection.setDoOutput(true);
        connection.setDoInput(true);
    }catch (IOException e) {
        e.printStackTrace();
    }
}
```

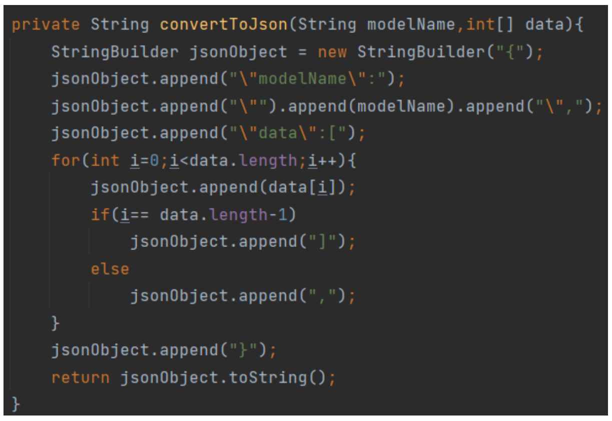
Fig. 19 – Heart stroke model summary



```java
public String predict(String modelName, int[] data)  {
    String jsonObject = convertToJson(modelName,data);
    sendDataThroughRequest(jsonObject);
    String response = readResponseFromRequest();
    if(response == null)
        return "Something wrong";
    else
        return response;
}
```

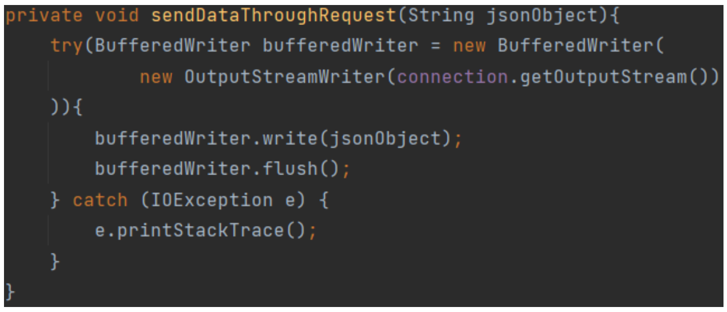
Fig. 20 – Predict method

In the Predict method it receive model name and data entered by the user and convert them to JSON object to be send through the request using convertToJson method (which take model name and data and convert them to string that represent JSON object as show in Fig. 21). Then it send the JSON object through the request to the API and read the response from the request



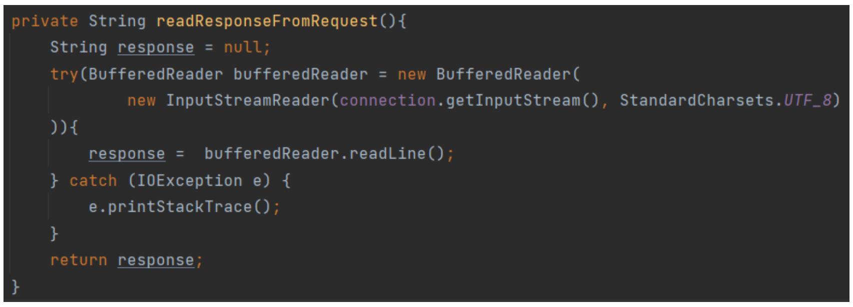
```java
private String convertToJson(String modelName,int[] data){
    StringBuilder jsonObject = new StringBuilder("{");
    jsonObject.append("\"modelName\":");
    jsonObject.append("\"").append(modelName).append("\",");
    jsonObject.append("\"data\":[");
    for(int i=0;i<data.length;i++){
        jsonObject.append(data[i]);
        if(i== data.length-1)
            jsonObject.append("]");
        else
            jsonObject.append(",");
    }
    jsonObject.append("}");
    return jsonObject.toString();
}
```

Fig. 21 – Convert to JSON object method



```java
private void sendDataThroughRequest(String jsonObject){
    try(BufferedWriter bufferedWriter = new BufferedWriter(
            new OutputStreamWriter(connection.getOutputStream())
    )){
        bufferedWriter.write(jsonObject);
        bufferedWriter.flush();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

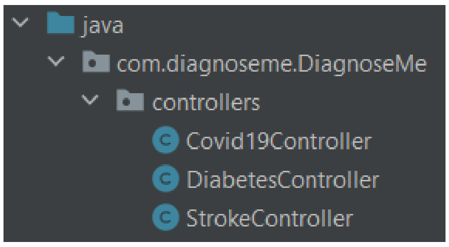
Fig. 22 - Send data through request method



```java
private String readResponseFromRequest(){
    String response = null;
    try(BufferedReader bufferedReader = new BufferedReader(
            new InputStreamReader(connection.getInputStream(), StandardCharsets.UTF_8)
    )){
        response =  bufferedReader.readLine();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return response;
}
```

Fig. 23 – Read response from request method

In addition, this part contain a controller for each disease (model) in the system, which is the entry point of the application that implement a RestAPI that is used by the UI.



```
java
  com.diagnoseme.DiagnoseMe
    controllers
      Covid19Controller
      DiabetesController
      StrokeController
```

Fig. 24 – Web app controllers

Each controller has a makePrediction method, which represent the RestAPI with post method and each controller has different mapping path, which is the name of the disease it diagnose. It receive array of integers in the request body which the data entered by the client. It create a predictor object and call predict method to get the predict value and return true if it is yes otherwise no.

You can access the code in GitHub repository using this link: https://github.com/ahmaddrabkah/DiagnoseMe.git .

4. **User interface (UI) :**

   This is the last part of our software, it is the part that the client's use and interact with. It as mentioned in the previous section it implemented using HTML, CSS and JavaScript. All pages are developed using these languages and Bootstrap for some styling.
   The important file is apiHandler.js, which implement the connection to the API of the web app using Ajax request and then it display the result in a window depends on the prediction result.
   In predict method first it read all values entered by the client inside array, as it receive the controllerName from the disease page it call inside it, it use controllerName to build the request URL and then specify the request method and the content. On success request, it display the result for the client with proper message. Then it clear all filed from the inputs.

   You can access the code in GitHub repository using this link:
   https://github.com/ahmaddrabkah/DiagnoseMe-UI.git .

VIII. Bibliography

1. ISO 12207 : https://www.iso.org/standard/43447.html
2. Agile Software Development :
   https://en.wikipedia.org/wiki/Agile_software_development
3. Microsoft Azure : https://azure.microsoft.com/
4. GitHub : https://github.com/
5. Domain Name : https://name.com/
6. TensorFlow : https://www.tensorflow.org/
7. DNN :
8. https://www.tutorialspoint.com/artificial_neural_network/index.htm
9. Dataset: all datasets are from Kaggle, which is the most popular website for AI & ML dataset.
10. Spring : https://spring.io/