International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 06 | June 2020 www.irjet.net p-ISSN: 2395-0072

Assessment for Hyperbilirubinemia Health Care using Django

K Kavinder Roghit[1], G Jayachandran[2]

[1]Student, Dept of Electronics and Communication Engineering, Meenakshi Sundararajan Engineering College, Tamil Nadu, India
[2]Student, Dept of Electronics and Communication Engineering, Meenakshi Sundararajan Engineering College, Tamil Nadu, India

---***---

Abstract -*In 21st century everything is getting digitized, even the consultation of doctors is getting digital like online consultation, skype calls. This project will step towards the modernization of humans through digital consultation, management and treatment of the disease jaundice. The Jaundice is not a disease but rather a sign that it can occur in many different diseases. This project is a savior of jaundice patients in the name of icterus health management; this will save the patient life based on the information of the patient itself. It will put an end to that drastic tragedy since it will go through the patient details and save them from jaundice. Prevention is better than cure, likewise this project will tell the patient's result beforehand, so that one cannot wait until the doctor's confirmation. Simply we need data from an authorized person to enter the data of the patient and it will analyze the data and show them if the patient is suspected to be in danger or not or in an intermediate stage. It also suggests treatment ways to the patient based on input data enlisted. Jaundice Smart health management will create a major impact on the society and cause uproar. This web-based diagnosis method will decrease the number of casualties by predicting the outcomes beforehand. This web-based solution for jaundice Management is a big leap towards the digital world, technological advancement, fast and secure world of human beings.*

Key Words: **Jaundice disease, Django framework, Python language, SQLite query language**

1. INTRODUCTION

Recently, the individuals with jaundice disease have found increasingly difficult to treat. So, a web application framework using Django written in Python programming language is used here. It is based on MVT (Model View Template) design pattern. The Django is very demanding due to its rapid development feature. It takes less time to build application after collecting client requirement. By using Django, we can build web applications in very less time. Django is designed in such a manner that it handles much of configure things automatically, so we can focus on application development only. The normal consultation of doctor during an epidemic jaundice will take a lot of time and may worsen up the situation. This major lapse has been modified by a web-app incorporated with machine learning

Machine learning is the ability to recognize future events based on past events for e.g. In face book ,if we tag our friends in a photo more often and the next time when we are tagging the same set of friends the photo itself provides correct names of our friends in the picture and this is achieved using machine learning.

Supervised learning is a process in machine learning which infers a product from labelled training data. This is similar to a teacher-student scenario. There is a teacher who guides the student to learn from books and other materials. The student is then tested and if correct, the student passes. Else, the teacher tunes the student and makes the student learn from the mistakes that he or she had made in the past. That is the basic principle of Supervised Learning

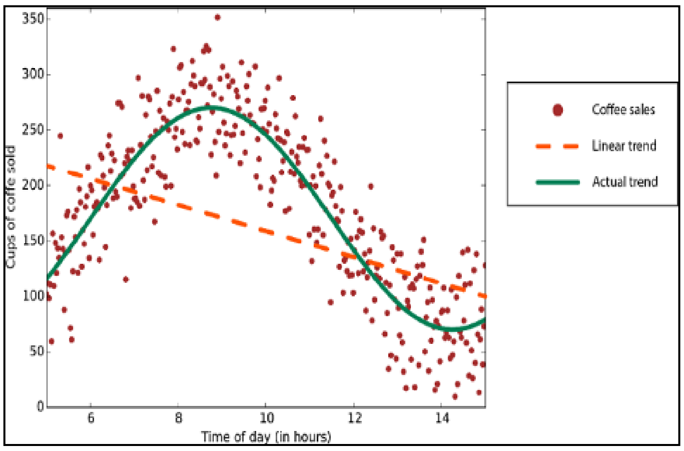


Fig-1: Supervised Learning

Unsupervised learning is also a process in machine learning which infers a product based its own suggestions.
To illustrate unsupervised learning,

- You have an album of 6 persons but **without information who is which one** and want to **divide** them into 6 separate photos, each with photos of each individual
- You have medicines, part of them is medicines and part are not, **but you do not know which are which** and you want to propose an algorithm to discover the medicines or drugs.

So, as a major lapse in unsupervised learning we do not use it and we follow supervised learning in implementing the project.

2. THE PROPOSED ARCHITECTURE

The architecture has shown below has many parts and there are mainly few parts to note. They are modules, Django framework design, data storage part,

rules engine. [3]
The module which is the main part which is also associated with the views part to display the user interface panel

The framework design will keep the friendly design to interact with the authorized persons that he/she can enter the defined values of the disease parameters.
Django is a web application framework written in python programming; it is designed in such a manner that it handles much of configure things automatically. The data's which are given by the authorized person are saved in the data base which is based on the SQLite 3, which is basically a query language that executes the queries that are typed in a specific format that stored the value in a table format.[6]

Rules engine is the main part in the system where the data's are processed and based on the input data's given to the user interface the result will be predicted, that is the disease which have the parameter that are mentioned in the user interface

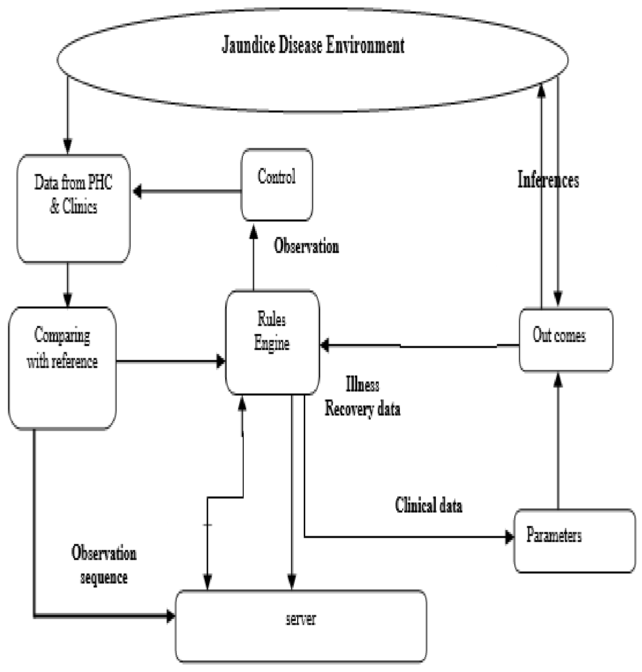


Fig-2: Overall Architecture

Software used:

- Python programming
- HTML
- SQL query language
- Visual studio code

2.1 DJANGO ARCHITECTURE

Django follows a Model-View-Controller (MVC) architecture, which is split up into three different parts:

- The **Model** is the logical data structure behind the entire application and is represented by a database (generally relational databases such as MySQL, Postgres).
- The **View** is the user interface — what you see in your browser when you visit a website. These are represented by HTML/CSS/JavaScript files.
- The **Controller** is the middleman that connects the view and model together, meaning that it is the one passing data from the model to the view.

With MVC, our application will revolve around the model—either displaying it or manipulating it. So, say a user will enter a URL in their browser, that request will go through the internet protocols, to our server, which will call Django. Django will then process the given URL path, and if it matches an URL path you have explicitly stated, it will call the **Controller**, which will then perform a certain action, such as get an entry from our **Model**(database) and then render a **View**(i.e.: JSON text, HTML/CSS/JavaScript Web page).[6]

2.2 Django Apps and Core Files

Any Django project will have at least one Django app. A Django project encompasses the application and all its components, while a Django app is a sub-container of the application with its own functionality that, in theory, may be reusable in another application without much modification. We will create one application (app) called as the main under a Django project called *jaundice.*

Jaundice Management project's MVC architecture is depicted below:

```
jaundice/
  manage.py
jaundice/
    __init__.py
    settings.py
    urls.py
    wsgi.py
  main/
    views.py
    models.py
    __init__.py
    admin.py
    tests.py
    urls.py
    templates/
homepage.html
contactus.html
register.html
      image.jpg
    migrations/
__init__.py
 0001_initial.py
          0002_parameters.py
          0003_medicines.py
          0004_patient details.py
```

All the files under the extension.py under main describe the Django *app* .Let us see the MVC in this project:

- *main/models.py* is the **Model**, or where we define our database. Here we define the parameters of patient, medicines to be give and patients details. It is the place where we feed the inputs of the patient details. When we open database to check the fed data, these models will only appear in the database. Under these categories our input value will be stored and stored

- Everything under *main/templates/main/* are templates or HTML files that defines our View. When a function in views.py renders an HTML file, it can pass objects such as a list of comments in which you can use special syntax to display those comments. Within each template, you can get static files like CSS, JavaScript files, or images to give the webpage life
- *main/views.py* is the **Controller**. Inside views.py we will define different functions/classes (Django offers support for both but most people still use functions) a function defines what happens when a certain URL is accessed and an HTTP request is made the server. Common actions would be to query the database for records and render a certain HTML template file

- *urls.py* is the URL configuration file. This is the file that allows us to map a certain URL to a certain function in views.py.

So now we need include necessary codes and import modules in VS Code accordingly to our needs

3. FLOW DESIGN

The flow design of the project will be discussed with the help of rules engine design, webapp design along with data storing

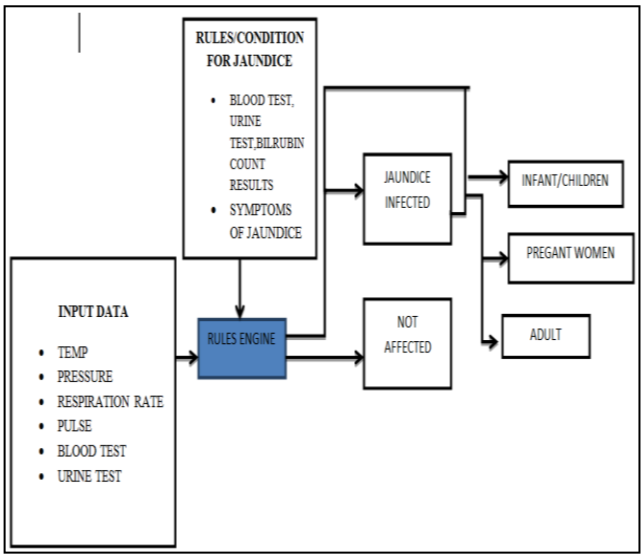


Fig-3.1: Rules Engine

3.1 DESIGN OF RULES ENGINE

First the parameters i.e. the input such the body temperature, blood pressure, platelet count, bilirubin count and various blood test results of a jaundice affected person

1. Now we need to get the above specified parameters of a person affected with disease
2. It is feed in the Rule's engine, it checks whether the conditions are satisfied for jaundice by comparing both the data.
3. If it detects jaundice, then we need to go for treatment according to the patient like infant treatment, adult and pregnant treatment.

3.2 DATA FEEDING IN WEB-APP

1. Open Power Shell in the folder where we have our project, webapp and manage.py
2. Type python manage.py run server in Power Shell window
3. The process will start running and it will display *http://127.0.0.1:8000/*
4. Open local browser and run the *http://127.0.0.1:8000/*it will display the homepage for our project

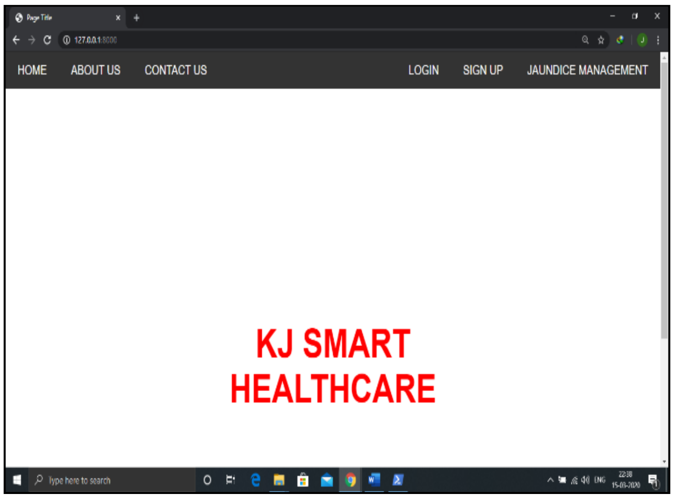


Fig-3.2: Home page of Web-app

5. To open the admin page where we need to enter the username and password, we to enter *http://127.0.0.1:8000/admin*

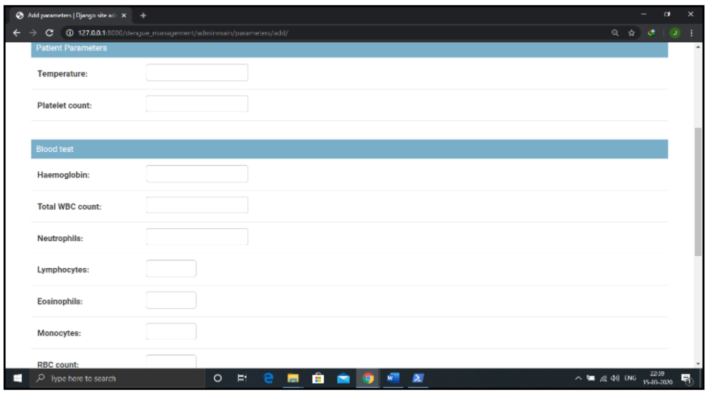


Fig-3.3: Modules present in Web-app

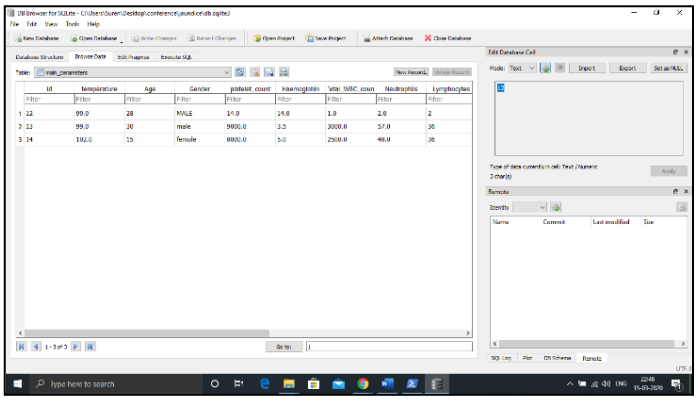
6. After which we need to update the patient's data, patient parameters and also the medicines necessary for them accordingly
   - **PATIENT DETAILS:** It contains the information of patient data

- **PARAMETER MODULE**: It contains all the functionalities related to patient conditions
- **MEDICINE MODULE**: It contains required medicines that are required by the patients based on the condition like age and pregnancy test

3.3 DATA STORING IN MY SQL

- The data that we entered as input in the web app will be stored in the database
- Open the MySQL and check the patient details, patient parameters and medicines of the patient which stores as we update the in the web-app

Fig-3.4: Data stored in MYSQL database



DB Browser for SQLite - C:\Users\Suren\Desktop\conference\jaundice\db.sqlite3

Table: main_parameters

	id	temperature	Age	Gender	platelet_count	Haemoglobin	'otal_WBC_coun	Neutrophils	Lymphocytes
1	12	99.0	28	MALE	14.0	14.0	1.0	2.0	2
2	13	99.0	30	male	9000.0	3.5	3000.0	57.0	36
3	14	102.0	15	female	8000.0	5.0	2500.0	40.0	36

3.4 EXTRACTION OF DATA FROM DATABASE

The data stored in the database can be extracted by using query language commands from which we can provide maximum efficient output that can be displayed.

4. FUTURE WORK

This system collects the user input data's like parameters, platelet count, WBC count etc. Then the collected input data from the authorized person is processed with the help of rules engine, comprised of set of rules that are modified to get or predict the outcomes based on the input data's which are given with the help of the interface.so we can define the set of rules which satisfies the condition so that it can predict the output. Later we can implement the notification system for the emergency patient directly to the doctor by using messaging app, which is half way done. This system can be developed into a full type module which takes input directly from the user itself no need to manually enter it.

4.1 USE OF DIGITAL HEALTH ID AND DEEP LEARNING

We can use digital health ID which is currently growing, where the user has to main the constant health data updated to it, so the extraction of data is much simpler than manual entry. In case of deep learning which can predict the percentage of disease which is occurring to a person, it can be predicted with the huge amount of data

5. CONCLUSION

Jaundice is currently a problematic global disease. Diagnoses via web-based techniques have become a new hope for early diagnosis, but are still limited due to uneducated people and standardization. The possibility of treatment of jaundice via antiviral drugs is still under investigation. This project will bring forth a massive solution for people who are suffering with icterus. It is a boon for the upcoming era. We have proposed to design the rules engine for the project but we have done a framework of the project to get the parameters and the treat them with appropriate medicines. Since this was the ideal solution as of now. We will further implement the machine learning part of the rules engine so that it would become a success in future.

REFERENCES

[1] Y. Yin, Y. Zeng, X. Chen, Y. Fan, "The Internet of Things in healthcare: An overview", J. Ind. Inf. Integr., vol. 1, pp. 3-13, Mar. 2016
[2] D. V. Dimitrov, "Medical Internet of Things and big data in healthcare", Healthcare Inform. Res., vol. 22, no. 3, pp. 156-163, Jul. 2016
[3] C. A. Tokognon, B. Gao, G. Tian, Y. Yan, "Structural health monitoring framework based on Internet of Things: A survey", IEEE Internet Things J., vol. 4, no. 3, pp. 619-635, Jun. 2017.
[4] J. Zhou, Z. Cao, X. Dong, A. V. Vasilakos, "Securityand privacy for cloud-based IoT: Challenges", IEEE Commun. Mag., vol. 55, no. 1, pp. 26-33, Jan. 2017
[5] M. Kovatsch, M. Lanter, S. Duquennoy, "Actinium: A RESTful runtime container for scriptable Internet of Things applications", 2012 3rd International Conference on IoT, pp. 135-142, 2012
[6]Django documentation is available from the following link http://docs.djangoproject.com/en/3.0/