Doctor Finder: Find doctors on the Go

R Jagadeesh Kannan¹, Harsh Tamakuwala², Siddharth Kale³ and Harsh Rohan Bhowmick⁴

¹School of Computer Science & Engineering, Vellore Institute of Technology Chennai Campus, India
²,³,⁴ School of Computing Science and Engineering, VIT Chennai, India,
1jagadeeshkannan.r@vit.ac.in,²tamakuwalaharsh42774@outlook.com,
³siddharthkale97@gmail.com

Abstract. Currently most people don’t have access to world-class health care. People tend to go to their local doctors and get their treatment delayed in case of serious diseases. These cases require a specialist to tend to the needs of the patients. A system is created that helps a person find a doctor based on the symptoms the user is facing. The system helps the user to get redirected to the nearest specialist that is appropriate to the special case of each user. This article discusses how the system was designed and implemented and what exactly was the need to implement such a system.

Keywords: Doctor Recommendation System, Disease Prediction, Intelligent Healthcare Management System, Applications of technology in Healthcare

1. Introduction

The world is ridden with various diseases which can be hard to get diagnosed. And what makes it even worse is that it even harder to get these diseases treated even if the disease is correctly diagnosed. In many third world countries, an unprecedented number of people die due to undetected diseases. The most prominent problem rattling the healthcare sector is to identify accurately a disease that a patient might be contacted with as early as possible. It is even more crucial in the case of terminal diseases like Cancer, HIV, COVID-19, etc.

People observe various cases that the families and friends were a late diagnosis of a terminal disease ended up in the demise of our loved ones. Only if there was any way of knowing that we can test ourselves at home just by seeing initial symptoms, that could have saved some of our loved one’s life. That gave us the initial idea of creating such a system, our main inspiration is drawn from such cases only.

Various studies and articles have pointed out and further explained in their sources [35][36] that how serious can a late diagnosis be. The source in [36] states about 10000 people die from a late cancer diagnosis. This study depicts that we urgently need a system to readily diagnose our patients.

At the time of writing this paper, there is an ongoing corona pandemic in the world. As of now, South Korea has done a laudable job of controlling the spread of the virus. As per the source [37], it was all possible due to the massive scale testing that the Koreans were able to conduct. In the article [37] from
the University of New South Wales, Sydney, Raina MacIntyre, an infectious disease scholar stated, that the key to controlling an epidemic is to diagnosis at scale.

All of the above sources point out the need for early diagnosis in the case of terminal diseases. And more often than not people do not approach specialists and wait for the ailment to get worse before consulting a specialist. This poses a serious threat to any patient.

The other problem that came to our mind that most countries like India face a lack of awareness of different kinds of specialists. People tend to visit local doctors or just general doctors. This mentality of not approaching specialists results in even more delayed diagnosis and ultimately late treatment. It is very crucial in many cases that the patient visits a competent medical professional that can properly detect the disease and provide efficient and cost-effective treatment. The source in [33], states that there is less than one doctor per 1000 people in India, which is well below the WHO standard. This even results in even fewer specialists per 1000 people. Just by looking at these statistics, one can assume that people surely struggle with finding an appropriate competent specialist in India. Even a recent UK based study [34], suggested that in almost 50% of the hospitals there have at least one place vacant for respiratory consultant post.

The above studies and statistics suggest that how hard can it be for a normal person to just get an appointment with a local doctor. The empty posts in hospitals might also translate to increased workloads of doctor and that could stall the early and rapid testing and diagnosis of the patients. Surely government and health care professionals can create laws and systems to ensure that people get more easily diagnosed or there is enough medical staff available for rapid treatment. But this can take years to achieve. And this is not possible instantly for many countries. That’s why there is a need for a system that can correctly guide patients to the nearest available specialist that can treat their ailment and is perfectly reliable. The new system discussed here will also decrease the load on the hospitals as it will guide the patients to the correct specialist. Thus, balancing the load from the general doctors and other specialists whom the patients might have wrongly concerned.

With this new system, a pipeline is created that enables anyone to just put what they are suffering through in just a sentence to the app. From there the analysis of the symptoms and conditions takes place. Based on the symptoms, a probabilistic model predicts the disease that is more likely to happen with the particular case of the patient. Information about the disease is also provided along with possible treatment. Various doctors are listed that can effectively treat the possible disease and are near to the user’s location. There is also a method to book an appointment with doctors and have the prescription noted to the customer through the app.

2. Technologies used

2.1. REST API

To create an actual system that intends to serve the customers, there was a need to bring the functionality of the models to them. For creating such a pipeline, there had to be a system in place that can communicate between the client and the model. That’s why it was decided to use and create APIs to handle communications.

System has specifically incorporated the REST API. APIs are as described in [1], Application Programming Interface, facilitate communication between the client and the webserver. Through APIs, various programs and devices can transfer data, which is made possible by exposing the devices to a set of data and functions. And REST is a common architectural style that is applied to the system to create API for our webservers.

As proposed by Roy Fielding [2] REST is essentially a subset of WWW and based on HTTP. The messages in the REST are not dependent on the state of the conversation going on between various clients and web services, essential it is termed as stateless. The REST agents give uniform interface semantics - basically make, recover, update and delete - as opposed to discretionary or application-explicit interfaces, and control assets just by the trading of representations. [2]

The method of creating the REST API can be subdivided into the following phases in which we tacked different problems, found solutions, and implemented as best suited to the system’s needs.
2.2. Selecting the framework and language
Initially, it was decided to go with NodeJS, but as things moved on, it was observed that the core machine learning pipeline was being developed in python. Moreover, NodeJS being single-threaded will not efficiently process CPU the oncoming requests even for a second. That’s results in creating an asynchronous process, which is hard to get used to. [3]
Keeping the above points in mind, it was decided to go with a framework based in python. Which helped in two ways-
To have a single language for our system that will make it much easier to design and manage.
The ability to host the API as well as the system handling the ML model on the same server, thus saving resources and the hassle of communicating between different resources.
As for the framework it was decided to go with Flask. Flask is a popular framework designed for creating web services and is written in python. Flask is developed in such a way that it doesn’t require any special libraries or tools making it a microframework. Flask is considered as a quick and easy to use framework, which was decided to use as rapid prototyping was used to create this system. [4][5].

2.3. Development of Routes for Website and App
Here API helped to create various methods with which any device can communicate.
Routes, as defined by Flask documentation [5], is a meaningful URI to help locate users to a page. Here we have created several routes with specific URI to handle various oncoming requests. These routes also include the methods with which the oncoming data is manipulated or data is retrieved from a Database server and send back to a client. The system is incorporated to use both of these functionalities in various routes.

2.4. Handling the website
It was noticed that a website interface is certainly important to reach out to as many people as possible. The Website Interface doesn’t have a separate backend, instead, configuration was need so that the API can handle and serve the website as well. There are specific API calls that are specifically modeled to handle the website and its various functionalities.

2.5. Database
The reasons and arguments why a special kind of database is needed to suit the needs of this system have to be mentioned. As to quote the source [10], NoSQL is the current emerging alternative to the traditional database system which chiefly includes the RDBMS. NoSQL is considered to be a database that does not completely replace SQL but indeed compliments the SQL to a point that there is a coexisting scenario in place. [10]
As evident from various sources [6][7], it is very evident that the data in the Healthcare sectors tend to be much more unstructured and unformatted. It is very difficult to handle unstructured data through the traditional method of RDBMS.
As there is a lot of data in Healthcare that is just arbitrary and cannot be structured [7] and modeled specifically to suit a rigid relational schema [8], that’s why a system like NoSQL is needed for our system. Another reason is that the NoSQL database is needed for the system is that it was expected that if this system is implemented in the healthcare sector, it would see that the data will continue to grow, as mentioned above it will be unstructured, but more specifically their would-be occurrence of problems of RDBMS that are related to huge data size. That will include lower management and operational costs, low writing and reading speeds, low scalability (various DBMS tends to get slower after a certain data amount is reached). [9]
Hence, it was decided that RDBMS would stand in the way of data management of unstructured data and we would very well like our system to be scalable, that’s why, the system has used Firebase, a NoSQL database for the project.
Firebase Database is a Google product, that is a very fast, NoSQL based, cloud-hosted database. It let its users store and sync data in real-time all across the globe. [11]
The database comprises of the doctors that the system is suggesting to our user. Every doctor is categorized and stored based on the specialty of the doctor. Other than the private information of the doctors and the users the database also holds all the patient appointments, records of the app queries of the user, user medical history, and user test reports. Out of this data, only Doctor information and appointment detail are very well structured. Rest can be very unstructured and sometimes even some records may be present for one user or one not. For example, a user with the treatment of a broken leg might have an X-Ray report to his name in the database, and the other patient could have the sonographic images of her baby. In this way, there is very ambiguity about what may be uploaded to the database.

2.6. ML Models

Below are brief discussions and a bit of explanation of the ML models that have been tested out or used in the system at any point in time in the development of the system.

2.7. Decision Tree

A decision tree has a binary tree-like structure where each node represents a question kind of information upon which the output at the leaf nodes is reached. [22]

2.8. Additive decision tree

A classical decision tree has some weak branches and some strong branches. These divisions are created due to the information obtained from the data. Some test cases may be less of input; some may have very diverse symptoms that the model gets confused about what the disease might be. Hence, to overcome these weaknesses, one such method devised was additive decision trees. In this method, the weak classifiers were trained recursively to convert them to strong classifiers [23].

2.9. Incremental decision tree

This is an algorithm that gives a decision tree as an output. It trains the current model with the newly obtained data, without re-training the model on the previous data. Besides the increasing data problem, this model had no such major change inaccuracy [24].

2.10. Neural Network for multi-label classification

Neural networks are often used by data scientists as an approach to possibly define a simple model into a complex model to gain more accuracy. Neural networks are inspired by the working of neurons in the brain. It is a circuit of artificial neurons meant to solve much more complex problems which moreover has a huge amount of data. [25] Neural nets are generally used for computer vision purposes or Natural language processing where the data to be used as an input is much more complex. [26][27] Various industries have devised their libraries for the implementation of neural networks.

2.11. Incremental decision tree

This is a classical probabilistic classification approach. This method calculates the probability of occurrence of an event given the input. The formula for calculating the probability is shown in figure.1.

\[
P(c | X) = \frac{P(x_1 | c) \times P(x_2 | c) \times \cdots \times P(x_n | c)}{P(c)}
\]

**Figure 1.** Formula for calculating posterior Probability
Naïve Bayes classification naively assumes that each class is independent of one another and predicts one output. [30]

2.12. Multinomial Naïve Bayes Classifier
This classifier is used when the data is multinomially distributed. It uses the same approach as naïve Bayes but for each of the instances of features. This way probability for class can be calculated without any independence and the top three outputs can be used by our system with probability for each of them to give the prediction some meaning [31].

2.13. Cloud
Cloud computing can be simply defined as the delivery of a wide range of services through the internet. These services or resources include data servers, storage of data, networking services, databases as well as software [12]. The main aspect of cloud computing is to remove the need for the local hard drive for storage and instead, use cloud-based storage solutions. As long as the device is connected to the web, it can be used to access the data stored in the database or the applications that can be used to do the same [13]. Companies that use this technology enable the users to store applications and data on remote databases and also give them the ability to retrieve the files back via the web [14]. Cloud computing does all the heavy work lifting which involves a lot of processing and crunching of data and allows the user to access them from a remote location from anywhere in the world.

Cloud computing services can be deployed in various ways, also termed as models. The deployment models are as follows [15]:

2.13.1. Public cloud
It is a type of cloud hosting model where all the services are deployed over a network that is open for public usage.

2.13.2. Private Cloud
it is also known as an internal cloud where the platform is implemented or deployed on a secure network only accessible by the client who has the permission to access it
On the technical side, both cloud types, Public and Private, may have little to no difference at all in the structural side of the cloud but are much different in terms of security offered by the service providers to the client [16].

2.13.3. Hybrid Cloud
It is a type of cloud hosting that is integrated i.e., the arrangement of one or more cloud servers, public, private, or community cloud services bound together but function as separate entities. It helps to move workloads between private and public clouds according to computing and cost needs which also gives the businesses greater flexibility and more data deployment options too.

After the selection of the right type of cloud, the type of services has to be selected. The services of the cloud include SaaS- Software as a service, PaaS- Platform as a Service, IaaS- Infrastructure as a Service. The cloud service model SaaS is a type of software distribution model where a third-party provider hosts the services and makes them available to the clients over the network. PaaS is a computing model where a third-party provider provides or delivers necessary hardware and software tools that are generally required for the development process, to the client deployed over the network. Here, the PaaS provider hosts the software and hardware on its infrastructure which helps the users avoiding stalling hardware and software to develop or run a new application. IaaS is a form of model that provisions virtualized computing resources deployed over the network. Here, a cloud provider hosts the infrastructure components present in an on-premises data center like storage, networking hardware, and also servers. The customers using IaaS access the resources or the infrastructure services through the internet and use the provider's services to install the elements required of an application stack.
In the project model, we deploy our SaaS software application on the public cloud using GCP Firebase cloud services [17].

3. System architecture
The main objective of the application is to process the patient's input and display the details of the respective doctors and hospitals nearby with their previous statistics and location marked on the map. The patient inputs the symptoms and the model detects the type of disease corresponding to the input and the user henceforth, can consult the required doctor and track the details and location of the doctor or the hospital. The interface also provides the user with information about different types of diseases and gets required details regarding their symptoms, required treatments, and steps for prevention.

![Use Case for the whole system](image)

**Figure 2. Use Case for the whole system**

The model system application as shown in figure.2 is a two-part system containing the front-end part which acts as an interface for the user and the back-end part which stores the details of the users and connects with the ML model hosted in a web server using required APIs. Specifically, in order to serve this purpose, the REST API [18] is used for this system. To avoid a separate backend process requiring to store the inputs of the user and to preserve the privacy of the users, the API was configured to handle and serve the application by using specific API calls that are specifically modeled to handle the application and its various functionalities [19].

For the database part, GCP cloud storage service namely, Firebase storage [20] is used to store the details of the user as well as the doctors. Apart from individual details, it also includes the location of the hospitals and doctors as well as treatment information and prevention details too. The flow chart of the system is represented in figure.3.

- The architecture of the application involves four modules namely, collection, organization, recommendation, and updation.
- Collection phase: In this phase, the data from the user is collected in the form of symptoms and then submitted to the server.
- Organization phase: In this phase, the input data is then converted into respective JSON format or HTTP format to make it easier to process it in the server [21]. The server here, hosting the ML model is used to process the data of the user. After the data organization, the data is sent to the server using the REST APIs.
Recommendation phase: The data from the application after the organization phase, goes to the web server where the data is processed and provides the details required from the database.

![Figure 3. System Architecture explained by the flowchart](image)

The architecture of the system is described as follows,

3.1. Data Collection phase
According to the system architecture given above, apart from the application having the option of storing various details of the user and doctors including the location, the application also enables the user to input symptoms to detect the corresponding disease. The data is given by the user in the form of symptoms. The user can either manually input the diseases in the text box given in the application or can use the search box and select the diseases from the list.

3.2. Data Organization Phase
After the selection of the symptoms, the input is then organized in a form acceptable by the web server and then sent to the server for further processing. This is the data organization phase. After this phase, the data is sent to the web server using required APIs, here, REST APIs. As it is the core process of the system, a specific route is created that handles this process [11]. The route is supposed to get Symptom data as an HTTP post method from the client-side device. The data is then sent to the ML model pipeline handling process hosted in the Heroku web server.

3.3. Data Recommendation System phase
Beacon, next comes the data recommendation phase, where the ML model hosted in the web server is used to process the input and detect the corresponding diseases according to the probabilities and the related various information and treatment info is then fetched from the database and then sent to the mobile application.

The architecture of the data recommendation phase is as follows
According to the architecture figure 4 given above, it is divided into 5 modules,

3.4. User input
In this module, the input from the application via the APIs is sent to the server where the ML pipeline is hosted.

3.5. Input translation and output format
The UMLS defined codes were used as input and output of the machine learning model. Hence, after receiving the symptoms as input from the user, the basic English terminology was converted to a UMLS code and fed into the model after appropriate pre-processing which includes pivoting the whole input to convert it into a two-dimensional matrix containing binary values. The output given by the model was the UMLS code for the disease which was then to be translated into the English language so that it is understandable to the user. But, before converting it to the English language, the same UMLS code was used to retrieve disease information/diagnosis, treatment info, reference links, and doctor specialization info. These UMLS codes were used so that no syntactical errors may occur.

3.6. Disease Prediction
Various methodologies were tried for the disease prediction from the symptoms. Each of them along with its pros and cons is mentioned below:

3.6.1. Decision Tree
For classification on a small amount of data, decision trees are very accurate. These models were implemented which gave pretty accurate output if enough input was fed to them. A basic decision tree was used since the decision tree formed by it was not complex enough and accuracy change between different variations was not noticeable. The decision tree was implemented using the sklearn library.

\[
E(S) = \sum_{i=1}^{c} -p_1 \log_2 p_1
\]

(1)

3.6.2. Additive Decision tree
A classical decision tree has some weak branches and some strong branches. These divisions are created due to the information obtained from the data. Some test cases may have less input, some may have very diverse symptoms that the model gets confused about what the disease might be. Hence, to overcome these weaknesses in the models, various methods are used. One such method devised was additive decision trees. In this method, the weak classifiers were trained recursively to convert them to strong classifiers. Various pre-processing methods like interpolation can also be used. Since the dataset had an equal number of cases, this method did not affect the accuracy that much but this was more of a future endeavor when the data might become more varied. [23] This variation of the decision tree was implemented from scratch on a small amount of data which was then compared with the standard decision tree approach on the same data. The main concern at the moment was for accuracy which this method did not solve.

3.6.3. Incremental decision tree
This was also a future endeavor for the model. When new diseases are added to the dataset, currently the model has to be trained completely from scratch to accommodate the newly obtained data which can be very resource-consuming and the system has to be down for maintenance more often to update the machine learning model. To tackle this problem an incremental decision tree was implemented. This is an algorithm that gives a decision tree as an output. It trains the current model with the newly obtained data, without re-training the model on the previous data. Upon researching more, this model might be dangerous in the medical domain. There might occur an instance where a disease’s symptoms are completely changed due to some previous human error or even due to the adoption of the microorganisms. In such cases, the whole model needs to be trained again else, the model might predict the old outputs which are false. Besides the increasing data problem, this model had no such major change inaccuracy. [24] This variation of the decision tree was also not focused on accuracy since its main task was to reduce the workload required to accommodate the new data into the current environment. Hence, this approach also met the dead end.

3.6.4. Neural Network for multi-label classification
Neural networks are often used by data scientists as an approach to possibly define a simple model into a complex model to gain more accuracy. Neural networks are inspired by the working of neurons in the brain. It is a circuit of artificial neurons meant to solve much more complex problems which moreover has a huge amount of data. [25] Neural nets are generally used for computer vision purposes or Natural language processing where the data to be used as an input is much more complex. [26][27] Various industries have devised their libraries for the implementation of neural networks. The accuracy of the model could still be increased; hence, this method was tried. Also, there was one more problem in the current approach. The current model predicted only one disease. Hence, in some scenarios, when two or more diseases have a quite similar probability of occurring, the current approach might fail to give correct predictions. Hence, after that, approaches were more probability-based. Since having some actual number related to the disease the prediction might make more meaning. One such approach was using a neural network. Upon testing the model, scores for each possibility were calculated and the top three were predicted as output. This approach was very resource-consuming for practical purposes. And the approach was even more resource consuming during training, hence when some new data is obtained, retraining the whole model could end up being more costly. Also, concurrent access to the model could drastically increase the waiting time for the users since the computation of the output is more time-consuming. [28]

3.6.5. Hybrid model
For the multi-label classification problem, various custom prototypes were tried. One approach was based on the theory of random forest implementation. Random forest is a collection of various decision trees which are fed different data upon input, and it gives an output based on the most voted output by the trees. [29] This approach was modified by giving each tree the same input but in an incremental manner. After the prediction from the first tree, that case could be hypothetically scraped out from the
input and the rest of the input could be fed into the second possible tree. This way 3 diseases could be predicted but without any actual numbers to rely on. This method was also implemented from scratch on a small amount of data but no appropriate scoring mechanism was found to get concrete proof about the result. Another method was tried using the same decision tree but gradually feeding the input. A buffer of the outputs could be created at various intervals to give the top three outputs. A score-based mechanism has to be accommodated to get the top three outputs based on the score. This method is not too reliable since it might give various false results since the score calculated is relative to the other inputs fed at that interval. This method might fail also when the symptoms of the first disease in the dataset are matched completely with the input. Hence, for every iteration, it will give the same output. This can be solved by running the above method more times, each time feeding randomness into the input. This method could be more time-consuming.

3.6.6. Naïve Bayes Classifier
Since the main focus currently was the accuracy and then a proper grading mechanism of the output, Naïve Bayes Theorem was used. Naïve Bayes classification naively assumes that each class is independent of one another and predicts one output. [30]

\[
P(A|B) = \frac{p(B|A)P(A)}{P(B)}
\]  

(2)

3.6.7. Multinomial Naïve Bayes Classifier
This classifier is used when the data is multinomial distributed. It uses the same approach as naïve Bayes but for each of the instances of features. This way probability for class can be calculated without any independence and the top three outputs can be used by the system with probability for each of them to give the prediction some meaning. [31] The Naïve Bayes and Multinomial Naïve Bayes approach was implemented using the sklearn library. [32] Instead of directly getting the output, the probabilities were taken as output and they were mapped to each disease in the dataset. The top three disease names were then sent to the flask API along with the disease information.

3.6.8. Final Takeover
Currently, Multinomial Naïve Bayes classifier is used in the system due to its simplicity, less resource-consuming approach. Research is still ongoing on reducing the resource consumption by the neural networks to make it more practical since the outputs obtained by that approach were promising too and can be more scalable and reliable on huge amounts of data. Some hypothetical hybrid models could also be created to reduce the maintenance cost during the training with new data once the scaling problem is solved.

3.7. Retrieving other disease info
After the model was prepared, the next task was to provide some disease information and mapping it to the doctors. No APIs were found for getting the disease-related information. No datasets were also found of the mapping between disease and the disease information. There were two future possibilities: redirecting the user to some other search website/engine using the search query as the disease name but that could be less reliable due to some differences in results among the websites. Another option was to create a dataset with the disease information and the mapping with the doctors. This method was found more reliable since the disease information can be manually checked for information so that correct and relevant information is visible to the users. A csv file was created containing the mapping between disease name and disease information, treatment information, reference links & specialist doctors for more reliable information for the users. The mapping of the disease with the specialization of the doctors was necessary to search for relevant doctors in the local area. There were several mappings in the project.

3.8. Sending Output to the Flask API
Using the UML S code, disease name, disease diagnosis, disease treatment, reference links, specialization of doctor required for worse cases were retrieved by the python script and sent to the flask API in a python dictionary format.

3.9. Data Updation phase

Finally, in the data updating phase, the diseases predicted with necessary information and doctor details are displayed in the application for the user to view. Here, the result includes the name of the detected diseases displayed based on probabilities and also the information of the doctors and treatment details. The user can also track the location of the doctor from his present location by using the map tracking feature present in the application.

4. Results

Most of the algorithm approach changes were due to logical reasons and not due to metrics. In the system architecture section, we have discussed in detail what was the reason for choosing each algorithm. Here we have tried to elucidate what exactly were results we had obtained that lead us to our final model. Naïve Bayes (NB) and Neural Networks were used instead of the Decision Tree because of its multiple output prediction capabilities while consuming fewer resources and maintaining the accuracy of prediction. But some key metrics which were also useful like execution time, accuracy were noted and are shown as figure. 5.

Figure 5. Execution time comparison of ML models

The precision and recall are 1 for 2 of the algorithms which means that this is a hypothetically perfect model which is practically not possible. Here, we get those values due to insufficient training data. For training data, we had 1 test case for each of the disease, and for each disease, symptoms were almost different. Also, during testing, the 6 diseases chosen were exactly predicted with 0.99 probability because they might have almost distinct symptoms compared to other diseases. For testing, all symptoms for that disease except for a random single symptom were used. On further testing, it was also found that the minimum number of symptoms required to predict the disease was different for every disease. Hence, as a common barrier, a threshold of 0.05 probability was set upon which meeting the disease output is shown else the user is asked to enter more symptoms. This way more accurate output can be shown to a user.

As suggested from the above results we need to get more training data which covers more cases and diseases.

Neural Networks seemed promising but due to high resources consumption and inability to accommodate varying inputs puts its hold on use in this case. For now, based on the results and testing a Bayesian approach seems to be the most suitable for this system.
5. Future work
The ability to improve upon ourselves is the key that has made the human race come so far. And the same applied to our system. We know it is not perfect yet and even in the future will be needing constant changes. Below are some key features and improvements that we are working on bringing in the version of our system or we hope to include these features in our system or similar systems someday.
Neural network models can be created to predict output with more probable outputs. Currently, the probability of the user having a disease is shown as per the number of symptoms he possesses. But this cannot be a proper metric as the disease output may also depend on the severity and also the occurrence of the disease. A neural network model can be created by taking the number of people getting that disease metric into account as well. This way location-based disease output can be predicted which could be more accurate than the current model. Also, the hybrid model could be accommodated for future updating of the models to be less resource consuming and less time-consuming.
As we would like to reach out to more and more people, a voice operation model is what we feel that this system needs. As our main aim is to help people with no knowledge about Healthcare, we notice that many people are unable to read and write, and there are also people with disabilities who cannot operate our system in traditional ways. A voice model can mitigate this problem to some extent.
Moreover, our models will need to adapt to accommodate various local diseases. There are certain diseases common in one area, and most of the time that some symptoms translate to a specific local disease rather than an uncommon foreign one. But it has to be made sure that the uncommon disease that might also have a high probability isn’t ruled out. And that can be easily accommodated by our current system as we predict multiple possible diseases.
Currently, we cover over 200 commonly occurring diseases, and in the future, we would like to expand on that as well. We plan to cover at least 1000 diseases in the next version of the system.
Our ultimate aim is to create a system where a patient can get medical assistance on the go. And along with that, there is a vast pool of disease and treatment data is created so that doctors and medical professionals have something to refer to for anything. If this data is stored, managed, and shared with all the data privacy laws in concern, it could help boost the research in the medical field and can save millions of lives.
Lastly, we want our interface to be even simpler. We believe that incorporating Natural Language Processing into our interface can make our system easier to use. One example could be that patient instead of entering the symptoms, can just say in one-two lines, what exactly is happening to them. And our system can handle the rest.

6. Conclusion
In this work, an application of machine learning, cloud, Web-based APIs were studied and successfully implemented for the cause. The machine learning model used was Naïve Bayes classifier on multinomial distribution.
From the results, it can be concluded that the application can be deployed successfully for the benefit of the common crowd to get pre-treatment information and contact a suitable specialist doctor.
Besides, the current implementation, various future aspects and different use cases were also discussed in future works to make this implementation even more accurate.

References
[1] REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces
[2] Mark M Web Services Architecture World Wide Web Consortium. 2016.
[3] Shah H and Soomro TR 2017 Node Challenges in Implementation Global Journal of Computer Science and Technology
[4] Flask Foreword. Archived from the original on 2017, pp. 11-17.
[5] Welcome to Flask. https://flask.palletsprojects.com/en/1.1.x/
[6] Amir G and Haider M 2015 Beyond the hype: Big data concepts, methods, and analytics International Journal of Information Management 35 137-144
[7] Kale S, Tamakuwala H, Vijayakumar V, Yang L, and Rawal Kshatriya B 2020 Big Data in Healthcare: Challenges and Promise Proceedings of 6th International Conference on Big Data and Cloud Computing Challenges. Smart Innovation, Systems and Technologies 164

[8] Michael S 2010 SQL databases v. NoSQL databases. Commun. ACM 53 10–11.

[9] Jing H, Haihong E, Guan LE and Du J Survey on NoSQL database International Conference on Pervasive Computing and Applications, Port Elizabeth 363-366.

[10] Nayak AP, Poojary A and Dikshay 2013 Type of NoSQL databases and its comparison with relational databases International Journal of Applied Information Systems 5 16-19

[11] Firebase: store and sync data in real-time. https://firebase.google.com/products/realtime-database

[12] Gantz J and Reinsel D The digital universe in 2020: Big data, bigger digital shadows, biggest growth in the far east IDC iView: IDC Analyze the Future.

[13] Pankaj S 2013 Cloud Computing: Types, Architecture, Applications, Concerns, Virtualization and Role of IT Governance in Cloud International Journal of Advanced Research in Computer Science and Software Engineering 3 2013

[14] Buyya R, Chee Y S and Srikumar V 2008 Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities, Department of Computer Science and Software Engineering, University of Melbourne 9

[15] The Revolution of Cloud Computing https://www.trianz.com/insights/revolution-that-is-cloud-computing

[16] “Private vs. Public Cloud: What’s the Difference?” https://www.expedient.com/knowledgebase/blog/2014-06-05-private-vs-public-cloud-whats-difference/. Date accessed: 15 Jan. 2020.

[17] “Firebase Cloud Storage How-to and Use Case” https://expertise.jetruby.com/firebase-cloud-storage-how-to-and-use-case-dd9810fe041b Date accessed: 15 Jan. 2020.

[18] Mark M 2016 Web Services Architecturem, World Wide Web Consortium Relationship to the World Wide Web and REST Architectures

[19] Neumann A, Laranjeiro N, and Jorge B 2020 An Analysis of Public REST Web Service APIs. IEEE Transactions on Services Computing 1-1

[20] Chatterjee N, Chakraborty S, Aakash D and Asoke N 2020 Real-time Communication Application Based on Android Using Google Firebase IJARCSMS, 6

[21] “JSONObject by Android developers” https://developer.android.com/reference/org/json/JSONObject. Date accessed: 5 Feb. 2020.

[22] “Decision trees in machine learning” https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052

[23] Luna JM, Gennatas ED, Ungar LH, Eaton E, Diffenderfer ES, Jensen ST, Simone CB, Friedman JH, Solberg TD, Valdes G 2019 Building more accurate decision trees with the additive tree Proceedings of the National Academy of Sciences 116 19887-19893

[24] Chao S and Fai W 2009 An incremental decision tree learning methodology regarding attributes in medical data mining International Conference on Machine Learning and Cybernetics 3 1694-1699

[25] Hecht N and Robert 1992 Theory of the backpropagation neural network Neural networks for perception. Academic Press 65-93.

[26] Kulkarni AD 2001 Computer vision and fuzzy-neural systems Upper Saddle River NJ: Prentice Hall PTR 2001.

[27] Goldberg Y 2017 Neural network methods for natural language processing Synthesis Lectures on Human Language Technologies 10 1-309

[28] Cerri R, Barros RC and De Carvalho A 2014 Hierarchical multi-label classification using local neural networks Journal of Computer and System Sciences 80 39-56

[29] Liaw A and Wiener M 2002 Classification and regression by random Forest R news 2 18-22

[30] Rish I 2001 An empirical study of the naive Bayes classifier IJCAI 2001 workshop on empirical methods in artificial intelligence 3 22

[31] Schneider KM 2004 On word frequency information and negative evidence in Naive Bayes text classification International Conference on Natural Language Processing 474-485
[32] “Naive Bayes- Scikit Learn” https://scikit-learn.org/stable/modules/naive_bayes.html

[33] Less than one doctor for 1000 population in India: Government to Lok Sabha
https://economictimes.indiatimes.com/industry/healthcare/biotech/healthcare/less-than-one-doctor-for-1000-population-in-india-government-to-lok-sabha/articleshow/59697608.cms?utm_source=contentofinterest&utm_medium=text&utm_campaign=cppst. Article accessed 15 April 2020.

[34] A critical shortage of lung specialists could leave the NHS struggling to cope this winter - new study. https://www.brit-thoracic.org.uk/about-us/pressmedia/2019/a-critical-shortage-of-lung-specialists-could-leave-the-nhs-struggling-to-cope-this-winter-new-study/. Article accessed 10 January 2020.

[35] Driemeier M, de Oliveira PA, Druzian AF, Lopes Brum LF, Pontes ER, Dorval ME, Paniago AM Late diagnosis: a factor associated with death from visceral leishmaniasis in elderly patients,” Pathogens and global health 109 283-9

[36] Late cancer diagnosis kills 10,000 a year according to government tsar, https://www.theguardian.com/society/2009/nov/29/late-cancer-diagnosis-kills-thousands, Article Accessed 28 March 2020.

[37] Dennis Normile Coronavirus cases have dropped sharply in South Korea. What’s the secret to its success? Article accessed 2020.

[38] Shahada S, Abu A, Mohammed Hreiji S and Shamsudheen S 2019 IOT Based Garbage Clearance Alert System with GPS Location Using Arduino International Journal of MC Square Scientific Research 11 1-8

[39] Badawi WA 2019 Underground Pipeline Water Leakage Monitoring Based On IoT International Journal of MC Square Scientific Research 11 01-8.