**Title - DiagnoseMe: Personal Health Decoder**

**Introduction:** People frequently struggle to recognize their health problems rapidly and accurately in today's fast-paced society. This may delay receiving the right medical attention and, in some situations, result in inaccurate self-diagnosis. Furthermore, it can be challenging for people to distinguish reliable sources from the rest due to the overwhelming amount of health information that is readily available online, some of which may be inaccurate.1

This project seeks to provide a Personal Health Decoder application in recognition of these difficulties. For anyone looking to comprehend their health symptoms and possible underlying illnesses, this tool will act as an accessible, user-friendly resource.

Motivation : Globally, there is a significant unmet demand for accurate illness diagnosis. The intricacy of the patient population's illness processes and underlying symptoms creates enormous obstacles in establishing an early diagnosis tool and successful therapy. Millions of individuals around the world suffer from diseases, making it a global health concern as well. Furthermore, it can be difficult for people to ask for help and support because mental health is frequently stigmatized and misunderstood. By enhancing early detection, offering individualized treatment, and lowering stigma, using Python to predict diagnosis outcomes can assist in resolving these problems.

The primary objective of this project is to identify individuals who, based on information in their medical records. We can recognize trends and risk factors related to mental health concerns by applying Python data analytics tools and machine learning algorithms, enabling personalized care. This may close the gap between health research and clinical practice by giving clinicians easily available tools to help them make better decisions and plan better therapies.

**BACKGROUND RESEARCH:**

There has been a study explores the utilization of NLP techniques for processing patient symptom descriptions to provide accurate diagnoses. The system analysed textual inputs from patients and matched them against a database of symptoms and conditions.

They have used machine learning algorithms like Decision Tree, Support Vector Machine, K-Nearest Neighbor, Naïve Bayes , Logistic Regression. The performance of the study is measured with respect to accuracy, sensitivity, specificity, precision, F-measure, Area under curve (AUC).**2**

There was also one more research investigates the integration of symptom checker applications with Electronic Health Records (EHR) to enhance the accuracy of disease diagnosis. By utilizing comprehensive patient records, the system aims to provide more precise predictions.**3**

The other study evaluates the accuracy and reliability of various web-based symptom checker platforms. It provides insights into the strengths and weaknesses of existing systems, which can be valuable for developing more robust applications.**4**

**METHOD:**

The data was obtained from the Kaggle website. I selected data that would be useful for analysis, was consistent, and showed discernible trends.

The difficulties I experienced while working on the project were selecting the right data set and utilising Python to visualise the data, but with a little effort and knowledge, I was able to overcome them over time.

DiagnoseME helps people by predicting the disease from the signs and symptoms of their illness. The Dataset used in this project is Symptom2Diagnose, which was obtained from Kaggle. There are two columns in this dataset: text and label. The text contains the disease's symptoms, and the label lists several possible diagnoses for the symptoms. To achieve this we are using the K-nearest neighbors algorithm, sometimes referred to as KNN or k-NN, is a non-parametric supervised learning classifier that groups individual data points based on closeness in order to classify or predict data.

The project is divided to three modules:

1. Data Pre-Processing
2. Data Visualization
3. Disease Prediction

**1. Data Pre-Processing**

Data Pre-Processing is performed on the dataset in order to verify, check and make sure there are no missing values, noisy data, or inconsistent data before it is fed into the algorithm. For this we are importing python libraries numpy, pandas as shown in Figure 1.1

A close up of words

Description automatically generated

Figure 1.1

We upload our dataset from our local as shown in Figure 1.2

A screen shot of a computer

Description automatically generated

Figure 1.2

Displaying the dataset as shown in Figure 1.3, gives us the complete data from our uploaded file.

A screenshot of a computer

Description automatically generated

Figure 1.3

Using the method “drop” we are removing the unwanted columns from the dataset as shown in Figure 1.4

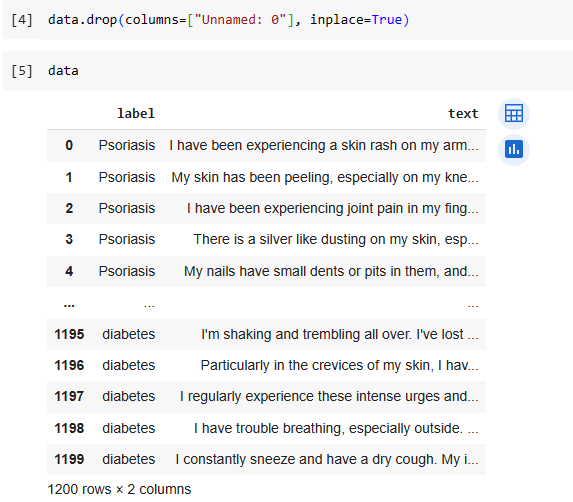


Figure 1.4

Using the methods isnull() and sum(), we are finding if our dataset has any null values present as shown in Figure 1.5

A screenshot of a computer code

Description automatically generated

Figure 1.5

To get some more information about our dataset, we use the “info()” function of pandas which displays various information about our data such as the column names, no. of non-null values in each column(feature), type of each column, memory usage, etc. as shown in Figure 1.6

A screenshot of a computer code

Description automatically generated

Figure 1.6

The function "value\_counts()" is utilized to determine the number of values in each category and the various categories inside a feature as shown in Figure 1.7

A screenshot of a computer program

Description automatically generated

Figure 1.7

From our dataset, we are eliminating stop words and non-alphabetic letters, tokenizing the text, and changing it to lowercase. For this we are importing stopwords and word tokenizers from nltk package as shown in Figure 1.8.

A computer screen shot of words

Description automatically generated

Figure 1.8

using the stopwords that is downloaded from nltk we define a method and apply that to the required columns as shown in Figure 1.9

A screenshot of a computer code

Description automatically generated

Figure 1.9

**2. Data Visualization**

Visualising the text and label columns by generating a word cloud which shows us the frequency of  the words in that column, as shown in Figure 2.2 and Figure 2.4 with Figure 2.1 and Figure 2.3 showing the code to generate them.



Figure 2.1

A close-up of words

Description automatically generated

Figure 2.2

A screen shot of a computer code

Description automatically generated

Figure 2.3

A close-up of words

Description automatically generated

Figure 2.4

Plotting interactive pie chart which displays the Distribution of Diseases provided in the dataset. To achieve this we need to import python libraries like plotly and matplotlib. Shown in Figure 2.5

A close up of words

Description automatically generated

Figure 2.5

Using plotly.express we can plot a pie chart by the method pie() which has parameters as the data frame which is the diseasesCount, values- data from this column is used to set values associated to sectors, names- data from this column is used as labels for sectors, title- gives a description about the chart as shown in Figure 2.6

A screen shot of a computer

Description automatically generated

Figure 2.6

**3. Disease Prediction**

To anticipate the disease based on the signs and symptoms that have been reported, we need to develop a model to which the dataset is fed. We are developing the model using the K-Nearest Neighbor algorithm and are feeding it the dataset and testing it by providing sample user signs/symptoms to a disease.

Every time an algorithm is applied to textual data, the text must be transformed into a numeric format. One pre-processing method that can convert an input word into a numeric form is called TFIDF vectorization. The parameter max\_features is for using only the 'n' most frequent terms as features rather than all of the words made possible by this parameter. This argument may be given with an integer, as shown in Figure 3.1. After this the dataset is splitted into training and testing sets.

A computer code with text

Description automatically generated with medium confidence

Figure 3.1

To determine the value of K for a good accuracy, we are taking a range of k values and are testing it with KNNClassifier which is then visually displayed by a line graph with the accuracies of training and testing datasets for each value of k as shown below in Figure 3.2 and Figure 3.3

A screenshot of a computer

Description automatically generated

Figure 3.2

A graph of a number of neighbors

Description automatically generated

Figure 3.3

From the above graph, we understood that the highest accuracy is obtained when the value of k is equal to 1. So setting the k value to 1  as shown in Figure 3.4, we train our model with our dataset and the classification report is generated and is shown in Figure 3.5

A screenshot of a computer code

Description automatically generated

Figure 3.4

A screenshot of a computer

Description automatically generated

Figure 3.5

Below in Figure 3.6, is the bar chart displaying the accuracies of testing and training datasets for the KNNClassifier when the value of K is 1.

A blue and green bar graph

Description automatically generated

Figure 3.6

To know the prediction to your illness, we need to provide our signs or symptoms which will be preprocessed by removing the stopwords and non-alphabetical characters, which is then vectorized and is fed to the model that we developed i.e., knn\_classifier\_for\_diagnoseME and will give us the predicted disease. as shown in Figure 3.7

A screenshot of a computer program

Description automatically generated

Figure 3.7

RESULTS :

DISCUSSION :

Python is a well-known programming language for developing machine learning models, and it includes a variety of tools and frameworks for predicting mental health. The first stage in developing a mental health prediction model is data gathering and preparation. The dataset from the Kaggle website is used for this project. The dataset is imported and text preparation is performed using pandas, pyplot, and numpy. The preprocessed data is then visualised using text patterns. The dataset is then used to train and test a machine learning classifier for text classification. Each model's performance is assessed.

The trained and proven model can be used to make disease diagnosis for a person. I attempted to keep the programme as basic as possible so that it could be understood and used as a reference in the future. I did face some challenges at the beginning, but I learned a lot from this class that I could use to this project, which made it easier for me to move forward.

CONCLUSION :

To summarise, creating an automated symptoms checker and diagnostic system with Python is a significant and valuable endeavour. With the advent of healthcare digitalization and a doctor scarcity in many locations, such a system can increase access and give preliminary support to patients.

However, developing an accurate system is difficult, necessitating a huge dataset of symptoms and illnesses as well as the use of techniques such as machine learning and natural language processing. To minimise potentially hazardous misdiagnoses, thorough testing and validation are required prior to implementation.

An automated symptoms checker, if intelligently built with medical practitioner involvement, can assist triage patients, identify those in need of urgent care, and decrease load on healthcare systems. It may also encourage patients to seek timely in-person therapy when necessary. Such preventive care and early intervention.

REFLECTION :

I went through a wonderful course on Introduction to Programming offered by Dr. Felesia Stukes, who is an efficient, motivating, and imaginative educator. The course was interesting since I learnt to develop numerous programmes using the command prompt, ranging from simple "Hello World" programmes to complicated jobs including regular expressions and data visualisation. Dr. Stukes gave us with several exercises and examinations to practise and develop our programming abilities in working with files, dictionaries, tuples, data visualisation, regular expressions, and strings. This course provided a fantastic chance for me to understand how programming languages may be used to create applications and tools that improve healthcare outcomes.

**References:**

1. Semigran, H. L., Linder, J. A., Gidengil, C., & Mehrotra, A. (2015). Evaluation of symptom checkers for self diagnosis and triage: audit study.
2. A Comprehensive Review on Symptom Checker Systems: From Medical Data Analysis to Machine Learning Techniques by Gopinath Ganapathy, Ramesh Kesavan, and Renganathan R, published in the Journal of Medical Systems, 2020.
3. Application of Machine Learning Techniques for Clinical Diagnosis of Chronic Kidney Disease by Prajnashree Panda, Manas Ranjan Patra, and Ruchi D. Raval, published in the Journal of King Saud University - Computer and Information Sciences, 2021.
4. Evaluating Symptom Checkers for Self Diagnosis and Triage: Audit Study by Semigran, Hannah L., et al., published in the British Medical Journal (BMJ), 2015.