Disease Prediction using EEG

Shaurya Kohli, Shreyas Chaudhry, Nehul Jindal, Santhi K

Introduction

For decades, researchers have been struggling to predict diseases using patients' treatment histories and health data by applying data mining and machine learning techniques. Previous works have used data mining techniques to predict the reoccurrence of specific diseases, control and progression of diseases. However, recent advancements in deep learning have led to a shift towards machine learning models that can learn complex patterns from raw data with minimal preprocessing and produce more accurate results. Big data technology has also contributed to disease prediction by automatically selecting relevant features from large amounts of data to improve the accuracy of risk classification. The main goal is to use machine learning to supplement patient care in healthcare and improve disease diagnosis and prediction. Predictive analysis with the help of multiple efficient machine learning algorithms helps in accurate disease prediction and patient treatment.

The use of algorithms to predict diseases based on patient symptoms can provide accurate and cost-effective treatment. With the healthcare system overloaded and becoming more expensive due to an increasing number of patients and diseases annually, disease prediction through algorithms can be a useful tool. In a project that used four different algorithms to predict diseases based on patient symptoms, an accuracy of 92-95% was achieved. Such a system has great potential for future medical treatments and an intelligently designed interface encourages interaction with the framework. The results of the study and project were visualized and presented. Nowadays, doctors are using various scientific technologies and methods to diagnose not only common illnesses but also fatal diseases. The success of treatment is often attributed to accurate and proper diagnosis. The results of the study and project were visualized and presented. Nowadays, doctors are using various scientific technologies and methods to diagnose not only common illnesses but also fatal diseases. The success of treatment is often attributed to accurate and proper diagnosis. The project of disease prediction using machine learning has been developed to detect general diseases in the early stages. In today's competitive economic environment, people have become so busy that they tend to ignore their health, which can result in harmful diseases later on. Research shows that 40% of people ignore common diseases, leading to severe health problems. This ignorance is mainly due to laziness in consulting a doctor, as people have very busy schedules and no time to take appointments and visit a doctor. The statistics also show that 70% of people in India suffer from general diseases, and 25% of people face death due to early ignorance.

The primary goal of developing this project is to allow users to sit at their convenience and have a health check-up. The user interface has been designed in such a way that it is straightforward to operate and use for everyone.

The healthcare system is overloaded due to the increasing number of patients and diseases, leading to high medical costs in many countries. Most diseases require a consultation with a doctor for proper diagnosis and treatment. With sufficient data, disease prediction using an algorithm can be very easy and cost-effective. Predicting diseases by observing the symptoms is an integral part of the treatment process. In this project, we have tried to accurately predict diseases by analyzing the patient's symptoms. We have used four different algorithms for this purpose and achieved an accuracy rate of 92-95%. Such a system has great potential for future medical treatments.

Our study and project results have been visualized and presented to highlight the importance of accurate disease prediction and patient treatment. Currently, doctors are adopting various scientific technologies and methodologies for accurate diagnosis and treatment of common and fatal diseases. Successful treatment is often attributed to correct and timely identification of diseases.

The project's main motive is to make healthcare more accessible and convenient for people, especially those who tend to ignore their health due to their busy schedules. With the help of this system, users can have a check-up of their health from anywhere at any time. The user interface has been designed to be simple and user-friendly, allowing everyone to use it with ease.The healthcare industry produces large amounts of health- care data daily that can be used to extract information for predicting disease that can happen to a patient in future while using the treatment history and health data. This hidden information in the healthcare data will be later used for affective decision making for patient’s health. Also, this areas need improvement by using the informative data in healthcare.

One such implementation of machine learning algorithms is in the field of healthcare. Medical facilities need to be advanced so that better decisions for patient diagnosis and treatment options can be made. Machine learning in healthcare aids the humans to process huge and complex medical datasets and then analyze them into clinical insights. This then can further be used by physicians in providing medical care. Hence machine learning when implemented in healthcare can leads to increased patient satisfaction. The k-mean algorithm is used to predict diseases using patient treatment history and health data.

With big data growth in biomedical and healthcare communities, accurate analysis of medical data benefits early disease detection, patient care, and community services. However, the analysis accuracy is reduced when the quality of medical data is incomplete. Moreover, different regions exhibit unique characteristics of certain regional diseases, which may weaken the prediction of disease outbreaks. In this paper, we streamline machine learning algorithms for effective prediction of chronic disease outbreak in disease-frequent communities.

Disease prediction using patient treatment history and health data by applying data mining and machine learning techniques is ongoing struggle for the past decades. Many works have been applied data mining techniques to pathological data or medical profiles for prediction of specific diseases. These approaches tried to predict the reoccurrence of disease. Also, some approaches try to do prediction on control and progression of disease. The recent success of deep learning in disparate areas of machine learning has driven a shift towards machine learning models that can learn rich, hierarchical representations of raw data with little preprocessing and produce more accurate results. Numbers of papers have been published on several data mining techniques for diagnosis of heart disease such as Decision Tree, Naive Bayes, neural network, kernel density, automatically defined groups, bagging algorithm and support vector machine showing different levels of accuracies in diseases prediction. In this type of research generally used tool is Waikato Environment for Knowledge Analysis (WEKA).

Due to big data progress in biomedical and healthcare communities, accurate study of medical data benefits early disease recognition, patient care and community services. When the quality of medical data is incomplete the exactness of study is reduced. Moreover, different regions exhibit unique appearances of certain regional diseases, which may results in weakening the prediction of disease outbreaks.

In this project, it bid a Machine learning Decision tree map, Navie Bayes, Random forest algorithm by using structured and unstructured data from hospital. It also uses Machine learning algorithm for partitioning the data. To the highest of gen, none of the current work attentive on together data types in the zone of remedial big data analytics. Compared to several typical calculating algorithms, the scheming accuracyof our proposed algorithm reaches 94.8% with an regular speed which is quicker than that of the unimodel disease risk prediction algorithm and produces report As per the Centres for Medicare and Medicaid services, 50% of Americans have multiple chronic diseases with a total US health care expenditure in 2016 to be about $3.3trillion, which amounts to $10,348 per person in the US. With the growth in medical data collecting electronic health records (EHR) is

increasingly convenient Besides, first presented a bio- inspired high-performance Heterogeneous vehicular telematics paradigm, such that the Collection of mobile users'health- related real-time big data can be achieved with the deployment of advanced heterogeneous vehicular networks. Chen et al. proposed a healthcare system using smart clothing for sustainable health monitoring. Qiu et al. had thoroughly studied the heterogeneous systems and achieved the best results for cost minimization on tree and simple path cases for heterogeneous systems. Patients' statistical information, test results and disease history are recorded in the EHR, enabling us to identify potential data-centric solutions to reduce the costs of medical case studies.

With the development of big data analytics technology, more attention has been paid to disease prediction from the perspective of big data analysis, various researches have been conducted by selecting the characteristics automatically from a large number of data to improve the accuracy of risk classification rather than the previously selected characteristics. However, those existing work mostly considered structured data.

EXISTING SYSTEM

Prediction using traditional disease risk model usually involves a machine learning and supervised learning algorithm which uses training data with the labels for the training of the models. High-risk and Low-risk patient classification is done in groups test sets. But these models are only valuable in clinical situations and are widely studied. A system for sustainable health monitoring using smart clothing by Chen et.al. He thoroughly studied heterogeneous systems and was able to achieve the best results for cost minimization on the tree and simple path cases for heterogeneous systems.

The information of patient's statistics, test results, and disease history is recorded in EHR which enables to identify potential data-centric solutions which reduce the cost of medical case studies. Bates et al. propose six applications of big data in the healthcare field. Existing systems can predict the diseases but not the subtype of diseases. It fails to predict the condition of people.

The predictions of diseases have been non-specific and indefinite

PROPOSED SYSTEM

In this paper, we have combined the structure and unstructured data in healthcare fields that let us assess the risk of disease. The approach of the latent factor model for reconstructing the missing data in medical records which are collected from the hospital. And by using statistical knowledge, we could determine the major chronic diseases in a particular region and in particular community. To handle structured data, we consult hospital experts to know useful features.

Decision Tree Analysis is a general, predictive modelling tool that has applications spanning a number of different areas. In general, decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

The decision rules are generally in form of if-then-else statements. The deeper the tree, the more complex the rules and fitter the model.

Before we dive deep, let's get familiar with some of the terminologies:

- Instances: Refer to the vector of features or attributes that define the input space
- Attribute: A quantity describing an instance
- Concept: The function that maps input to output
- Target Concept: The function that we are trying to find, i.e., the actual answer
- Hypothesis Class: Set of all the possible functions
- Sample: A set of inputs paired with a label, which is the correct output (also known as the Training Set)

- Candidate Concept: A concept which we think is the target concept
- Testing Set: Similar to the training set and is used to test the candidate concept and determine its performance

Introduction

A decision tree is a tree-like graph with nodes representing the place where we pick an attribute and ask a question; edges represent the answers the to the question; and the leaves represent the actual output or class label. They are used in non-linear decision making with simple linear decision surface.

Decision trees classify the examples by sorting them down the tree from the root to some leaf node, with the leaf node providing the classification to the example. Each node in the tree acts as a test case for some attribute, and each edge descending from that node corresponds to one of the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new nodes.

Let's illustrate this with help of an example. Let's assume we want to play badminton on a particular day — say Saturday — how will you decide whether to play or not. Let's say you go out and check if it's hot or cold, check the speed of the wind and humidity, how the weather is, i.e. is it sunny, cloudy, or rainy. You take all these factors into account to decide if you want to play or not.

So, you calculate all these factors for the last ten days and form a lookup table like the one below.

Day	**Weather**	**Temperature**	**Humidity**	**Wind**	**Play?**
1	Sunny	Hot	High	Weak	No
2	Cloudy	Hot	High	Weak	Yes
3	Sunny	Mild	Normal	Strong	Yes
4	Cloudy	Mild	High	Strong	Yes

5	Rainy	Mild	High	Strong	No
6	Rainy	Cool	Normal	Strong	No
7	Rainy	Mild	High	Weak	Yes
8	Sunny	Hot	High	Strong	No
9	Cloudy	Hot	Normal	Weak	Yes
10	Rainy	Mild	High	Strong	No

Table 1. Obeservations of the last ten days.

Now, you may use this table to decide whether to play or not. But, what if the weather pattern on Saturday does not match with any of rows in the table? This may be a problem. A decision tree would be a great way to represent data like this because it takes into account all the possible paths that can lead to the final decision by following a tree-like structure.

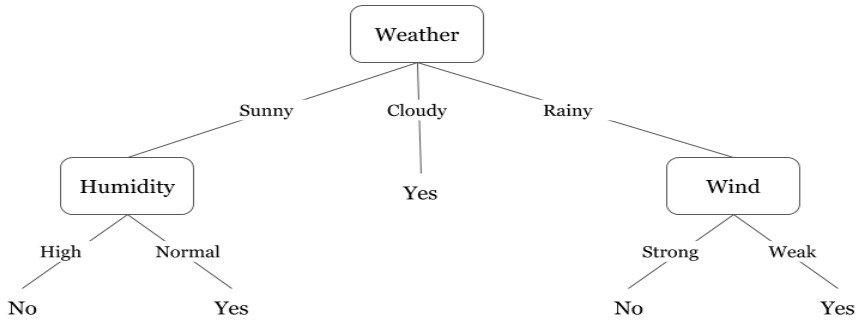


Fig 1. A decision tree for the concept Play Badminton

Fig 1. illustrates a learned decision tree. We can see that each node represents an attribute or feature and the branch from each node represents the outcome of that node. Finally, its the leaves of the tree where the final decision is made. If features are continuous, internal nodes can test the value of a feature against a threshold (see Fig. 2).

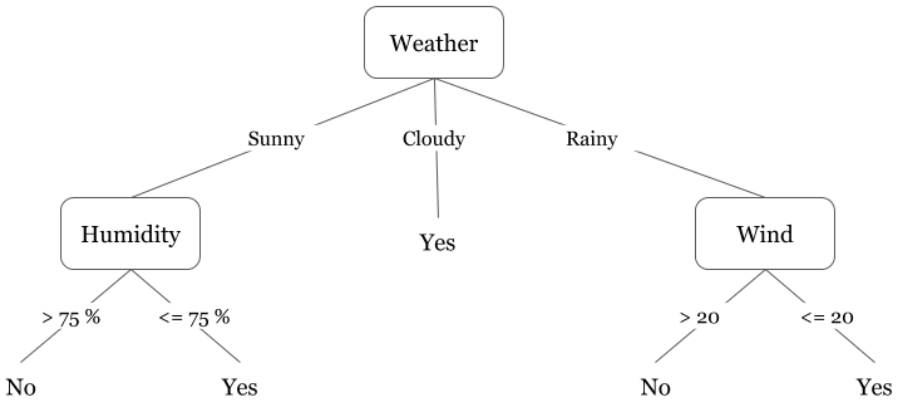


Fig 2. A decision tree for the concept Play Badminton (when attributes are continuous)

A general algorithm for a decision tree can be described as follows:

1. Pick the best attribute/feature. The best attribute is one which best splits or separates the data.
2. Ask the relevant question.
3. Follow the answer path.
4. Go to step 1 until you arrive to the answer.

The best split is one which separates two different labels into two sets.

Expressiveness of decision trees

Decision trees can represent any boolean function of the input attributes. Let's use decision trees to perform the function of three boolean gates AND, OR and XOR.

Boolean Function: AND

A	B	A AND B
F	F	F
F	T	F
T	F	F
T	T	T

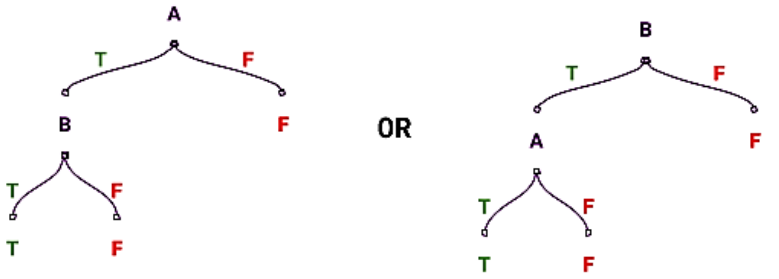


Fig 3. Decision tree for an AND operation.

In Fig 3., we can see that there are two candidate concepts for producing the decision tree that performs the AND operation. Similarly, we can also produce a decision tree that performs the boolean OR operation.

Boolean Function: OR

A	B	A OR B
F	F	F
F	T	T
T	F	T
T	T	T

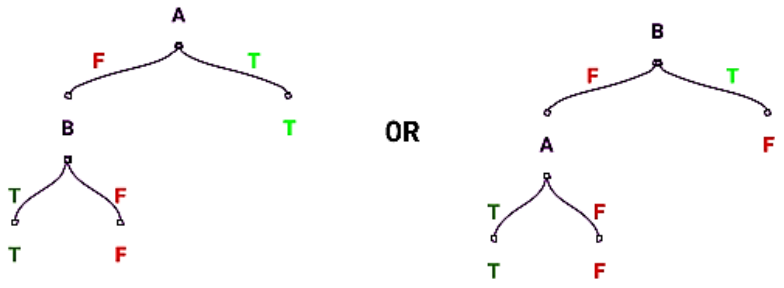


Fig 4. Decision tree for an OR operation

Boolean Function: XOR

A	B	A XOR B
F	F	F
F	T	T
T	F	T
T	T	F

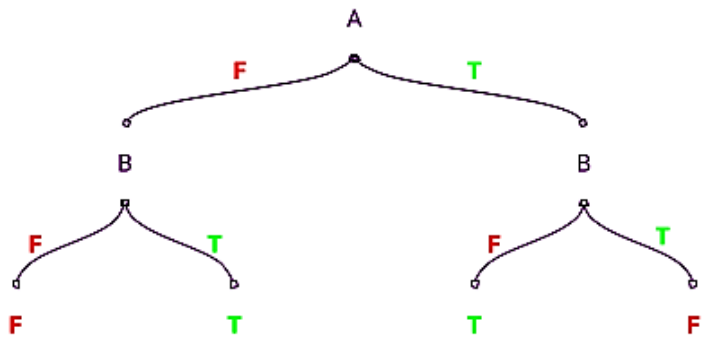


Fig 5. Decision tree for an XOR operation.

Let's produce a decision tree performing XOR functionality using 3 attributes:

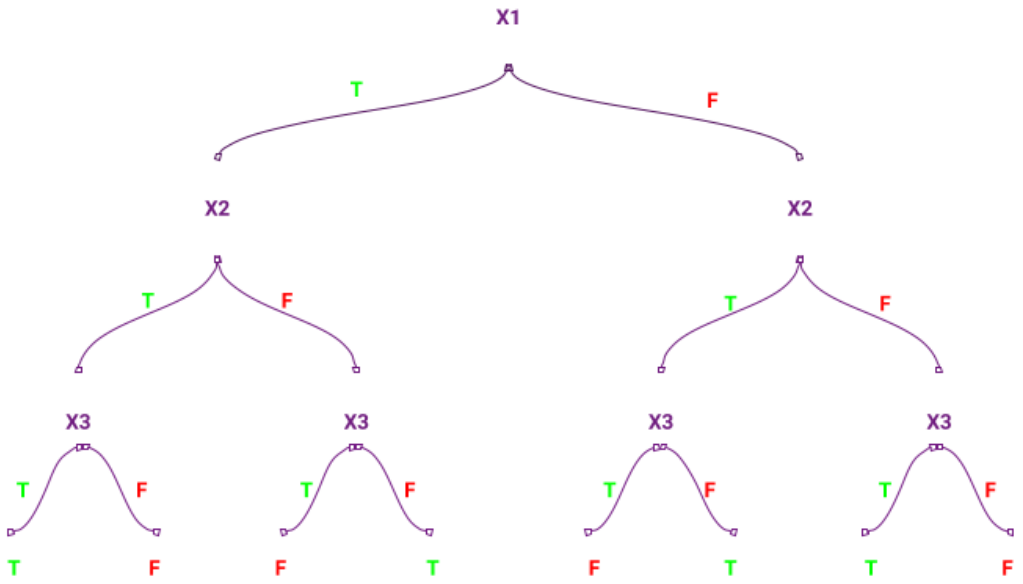


Fig 6. Decision tree for an XOR operation involving three operands

In the decision tree, shown above (Fig 6.), for three attributes there are 7 nodes in the tree, i.e., for $n = 3$, number of nodes = 2^3-1. Similarly, if we have n attributes, there are 2^n nodes (approx.) in the decision tree. So, the tree requires exponential number of nodes in the worst case.

We can represent boolean operations using decision trees. But, what other kind of functions can we represent and if we search over the various possible decision trees to find the right one, how many decision trees do we have to worry about. Let's answer this question by finding out the possible number of decision

trees we can generate given N different attributes (assuming the attributes are boolean). Since a truth table can be transformed into a decision tree, we will form a truth table of N attributes as input.

X1	X2	X3	....	XN	OUTPUT
T	T	T	...	T	
T	T	T	...	F	
...	...	...	...	...	
...	...	...	...	...	
...	...	...	...	...	
F	F	F	...	F	

The above truth table has 2^n rows (i.e. the number of nodes in the decision tree), which represents the possible combinations of the input attributes, and since each node can a hold a binary value, the number of ways to fill the values in the decision tree is ${2^{2^n}}$. Thus, the space of decision trees, i.e, the hypothesis space of the decision tree is very expressive because there are a lot of different functions it can represent. But, it also means one needs to have a clever way to search the best tree among them.

Decision tree boundary

Decision trees divide the feature space into axis-parallel rectangles or hyperplanes. Let's demonstrate this with help of an example. Let's consider a simple AND operation on two variables (see Fig 3.). Assume X and Y to be the coordinates on the x and y axes, respectively, and plot the possible values of X and Y (as seen the table below). Fig 7. represents the formation of the decision boundary as each decision is taken.

We can see that as each decision is made, the feature space gets divided into smaller rectangles and more data points get correctly classified.

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

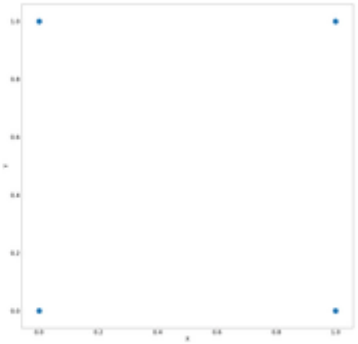


Fig 7. Animation showing the formation of the decision tree boundary for AND operation

The decision tree learning algorithm

The basic algorithm used in decision trees is known as the ID3 (by Quinlan) algorithm. The ID3 algorithm builds decision trees using a top-down, greedy approach. Briefly, the steps to the algorithm are: - Select the best attribute → A - Assign A as the decision attribute (test case) for the **NODE**. - For each value of A, create a new descendant of the **NODE**. - Sort the training examples to the appropriate descendant node leaf. - If examples are perfectly classified, then STOP else iterate over the new leaf nodes.

Now, the next big question is how to choose the best attribute. For ID3, we think of the best attribute in terms of which attribute has the most *information gain*, a measure that expresses how well an attribute splits that data into groups based on classification.

Pseudocode: ID3 is a greedy algorithm that grows the tree top-down, at each node selecting the attribute that best classifies the local training examples. This process continues until the tree perfectly classifies the training examples or until all attributes have been used.

The pseudocode assumes that the attributes are discrete and that the classification is binary. Examples are the training example. *Target_attribute* is the attribute whose value is to be predicted by the tree. *Attributes* is a list of other attributes that may be tested by the learned decision tree. Finally, it returns a decision tree that correctly classifies the given *Examples*.

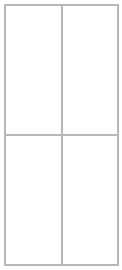
ID3(*Examples, Target_attribute, Attributes*): - Create a *root* node for the tree. - If all *Examples* are positive, return the single-node tree *root*, with positive labels. - If all *Examples* are negative, return the single-node tree *root*, with negative labels. - If *Attributes* is empty, return the single-node tree *root*, with the most common labels of the *Target_attribute* in *Examples*. - Otherwise, begin - A ← the attribute from *Attributes*that best* classifies Examples - The decision attribute for *root* ← A - For each possible value *v_i*, of A, - Add a new tree branch below root, corresponding to the test A = v_i - Let *Examples_vi* be the subset of Examples that have value v_i for A. - If *Examples_vi* is empty - Then, below this new branch add a leaf node with the labels having the most common value of *Target_attribute* in *Examples*. - Else, below this new branch add the subtree(or call the function) - ID3(*Examples_vi, Target_attribute, Attributes-{A}*) - End - Return *root*

* Adopted from Machine Learning by Tom M. Mitchell*

**The best attribute is the one with the highest information gain.*

Calculating information gain

As stated earlier, information gain is a statistical property that measures how well a given attribute separates the training examples according to their target classification. In the figure below, we can see that an attribute with low information gain (right) splits the data relatively evenly and as a result doesn't bring us any closer to a decision. Whereas, an attribute with high information gain (left) splits the data into groups with an uneven number of positives and negatives and as a result helps in separating the two from each other.



To define information gain precisely, we need to define a measure commonly used in information theory called *entropy* that measures the level of *impurity* in a group of examples. Mathematically, it is defined as:

Entropy:$\sum i=1-p*\log2(pi)$

pi=Probabilityofclassi

Since, the basic version of the ID3 algorithm deal with the case where classification are either positive or negative, we can define entropy as :

Entropy(S)=−p+log2p+−p−log2p−

where,

S is a sample of training examples

p_+ is the proportion of positive examples in S

p_- is the proportion of negative examples in S

To illustrate, suppose *S* is a sample containing 14 boolean examples, with 9 positive and 5 negative examples. Then, the entropy of *S* relative to this boolean classification is:

Entropy ([9+, 5-]) = -(9/14)$\cdot log_2$(9/14) - (5/14)$\cdot log_2$(5/14) = 0.940

Note that entropy is 0 if all the members of S belong to the same class. For example, if all members are positive (p_+=1), then p_- is 0, and *Entropy(S)* = -1$\cdot log_2$(1) -0$\cdot log_2$(0) = 0. Entropy is 1 when the sample contains an equal number of positive and negative examples. If the sample contains unequal number of positive and negative examples, entropy is between 0 and 1. The following figure shows the form of the entropy function relative to a boolean classification as p_+ varies between 0 and 1.

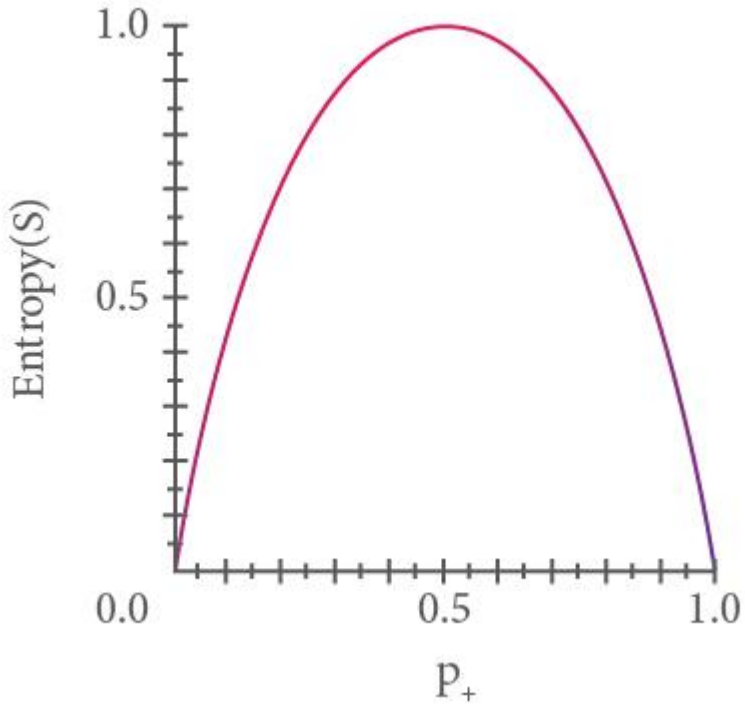


Fig 9. Entropy function to a boolean classification, as the proportion p_+, of positive examples varies between 0 & 1.

Now, given entropy as a measure of the impurity in a sample of training examples, we can now define *information gain* as a measure of the effectiveness of an attribute in classifying the training data. Information gain, *Gain (S, A)* of an attribute A, relative to a sample of examples *S*, is defined as:

Gain(S,A)≡Entrpoy(S)−∑v∈Values(A)rac|Sv||S|·Entropy(Sv)

where *Values(A)* is the set of all possible values for attribute the A, and S_v is the subset of *S* for which attribute *A* has value *v*. Note the first term in the equation is just *entropy* of the original sample *S*, and the second term is the expected value of entropy after *S* is partitioned using attribute *A*, i.e. entropy of its children. Expected entropy described by this second term is simply the sum of entropies of each subset S_v, weighted by the fraction of examples $rac{|S_v|}{|S|}$that belong to S_v. *Gain(S, A)* is therefore the expected reduction in *entropy* caused by knowing the value of attribute *A*.

In short :

InformationGain=Entropy(parentnode)−[AverageEntropy(children)]

For example, suppose a sample (S) has 30 instances (14 positive and 16 negative labels) and an attribute A divides the samples into two subsamples of 17 instances (4 negative and 13 positive labels) and 13 instances (1 positive and 12 negative labels) (see Fig. 9).

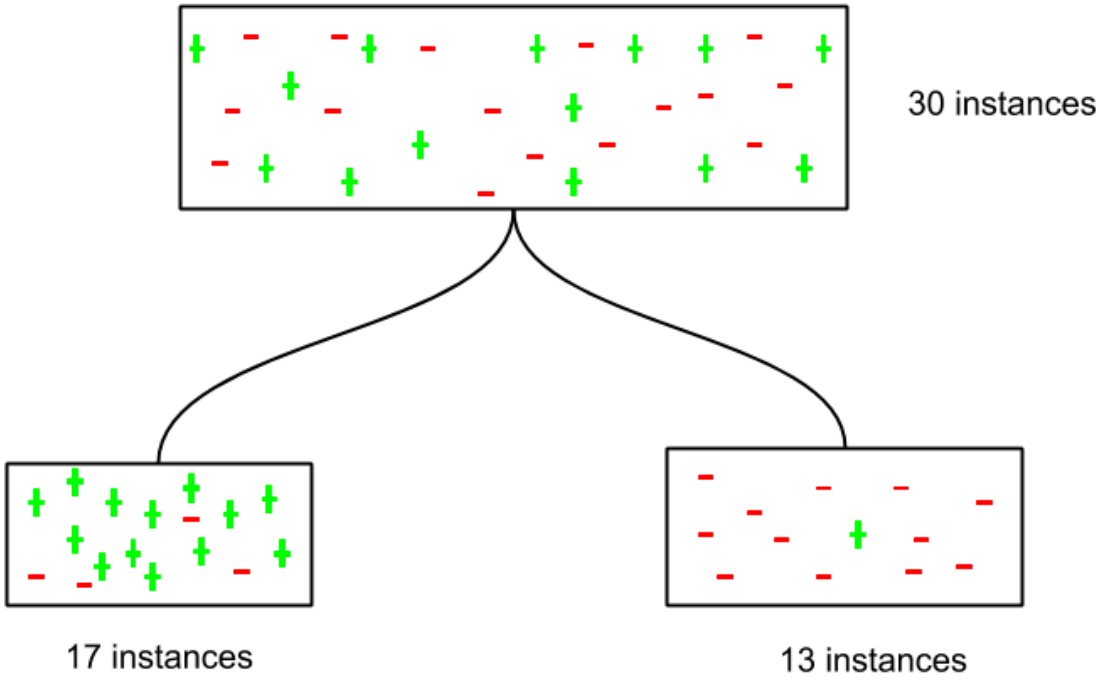


Fig 10. Example of decision tree sorting instances based on information gain.

Let's calculate the information gain of the attribute *A*. We know that:

Gain(S,A)≡Entrpoy(S)−∑v∈Values(A)rac|Sv||S|·Entropy(Sv)

and,

$Entropy(S) = -p_+log_2p_+ - p_-log_2p_-$

Entropy of parent = *Entropy(S)* = -$rac{14}{30}\cdot log_2rac{14}{30} $ - $rac{16}{30}\cdot log_2rac{16}{30} $ = 0.996

Entropy of child with 17 instances = *Entropy(S_1)* = -$rac{13}{17}\cdot log_2rac{13}{17} $ - $rac{4}{17}\cdot log_2rac{4}{17} $ = 0.787

Entropy of child with 13 instances = *Entropy(S_2)* = -$rac{1}{13}\cdot log_2rac{1}{13} $ - $rac{12}{13}\cdot log_2rac{12}{13} $ = 0.391

(Weighted) Average Entropy of children = $rac{17}{30} \cdot 0.787$ + $rac{13}{30} \cdot 0.391$ = 0.615

Information Gain = G(S, A) = 0.996 - 0.615 = 0.38

Similarly, we can calculate the *information gain* for each attribute (from the set of attributes) and select the attribute with highest *information gain* as the *best* attribute to split upon.

STEPS TO DEVELOP THE PROJECT

A. Analyzing the problem statement & requirements Analyze the problem in terms of what we want to predict and what kind of observation data we have to make those predictions.

B. Collect and clean the data

Identify what kind of historical data we have for prediction modeling, the next step is to collect the data from datasets or from any other data sources.

C. Prepare data for ML application

Transform the data in the form that the Machine Learning system can understand.

D. Prepare the Graphical User Interface (GUI) of the model

Graphical User Interface (GUI) is designed for taking input and displaying output. There are 5 input text boxes which consist of dropdown menu of symptoms and the user can select those one by one. Python Tkinter package is used for designing the GUI. On pressing the 'Result' button, the disease is predicted in the output field. Also, the drugs are described in the specified field.

CONCLUSION

With the proposed system, higher accuracy can be achieved. We not only use structured data, but also the text data of the patient based on the proposed Decision tree algorithm. To find that out, we combine both data, and the accuracy rate can be reached up to 95%. None of the existing system and work is focused on using both the data types in the field of medical big data analytics. We propose a Decision tree algorithm for both structured and unstructured data. The disease risk model is obtained by combining both structured and unstructured features.

REFERENCES

[1] "M. Chen, S. Mao and Y. Liu. Big data: A survey".
[2] "P. B. Jensen, L. J. Jensen and S. Brunak. Mining electronic health records: Towards better research applications and clinical care".
[3] "Yulei wang1, Jun yang2, Viming.Big Health Application System based on Health Internet of Things and Big Data".
[4] "S.-M. Chu,W.-T. Shih,Y.-H. Yang, P.-C. Chen and Y.-H. Chu. Use of traditional Chinese medicine in patients with hyperlipidemia: A population-based study in Taiwan".
[5] "S. Zhai, Chang, R. Zhang and Z. M. Zhang. Deepintent: Learning attentions for online advertising with recurrent neural networks".
[6] "M. Chen, Y. Ma, J. Song, C. Lai, and B. Hu. Smart clothing: Connecting human with clouds and big data for sustainable health monitoring".
[7] "W. Yin and H. Schutze. Convolutional neural network for paraphrase identification".
[8] "Weixing Wang and Shuguang Wu. A Study on Lung Cancer Detection by Image Processing".
[9] "Thanh Trung Giang, Thanh Phuong Nguyen and Dang Hung Tran. Stratifying Cancer Patients based on Multiple Kernel Learning and Dimensionality Reduction, 2017 IEEE 9th International Conference on Knowledge and Systems Engineering (KSE)".
[10] "Saranya P and Satheeskumar B. A Survey on Feature Selection of Cancer Disease Using Data Mining Techniques", International Journal of Computer Science and Mobile Computing, Vol.5 Issue.5, May- 2016, pg. 713-719".
[11] "Dmitry Ignatov and Andrey Ignatov. Decision Stream: Cultivating Deep Decision Trees", 3 Sep 2017 IEEE".
[12] "Kelvin KF Tsoi1, Yong-Hong Kuo and Helen M. Meng. A Data Capturing Platform in the Cloud for Behavioral Analysis Among Smokers An Application Platform for Public Health Research", 2015 IEEE".