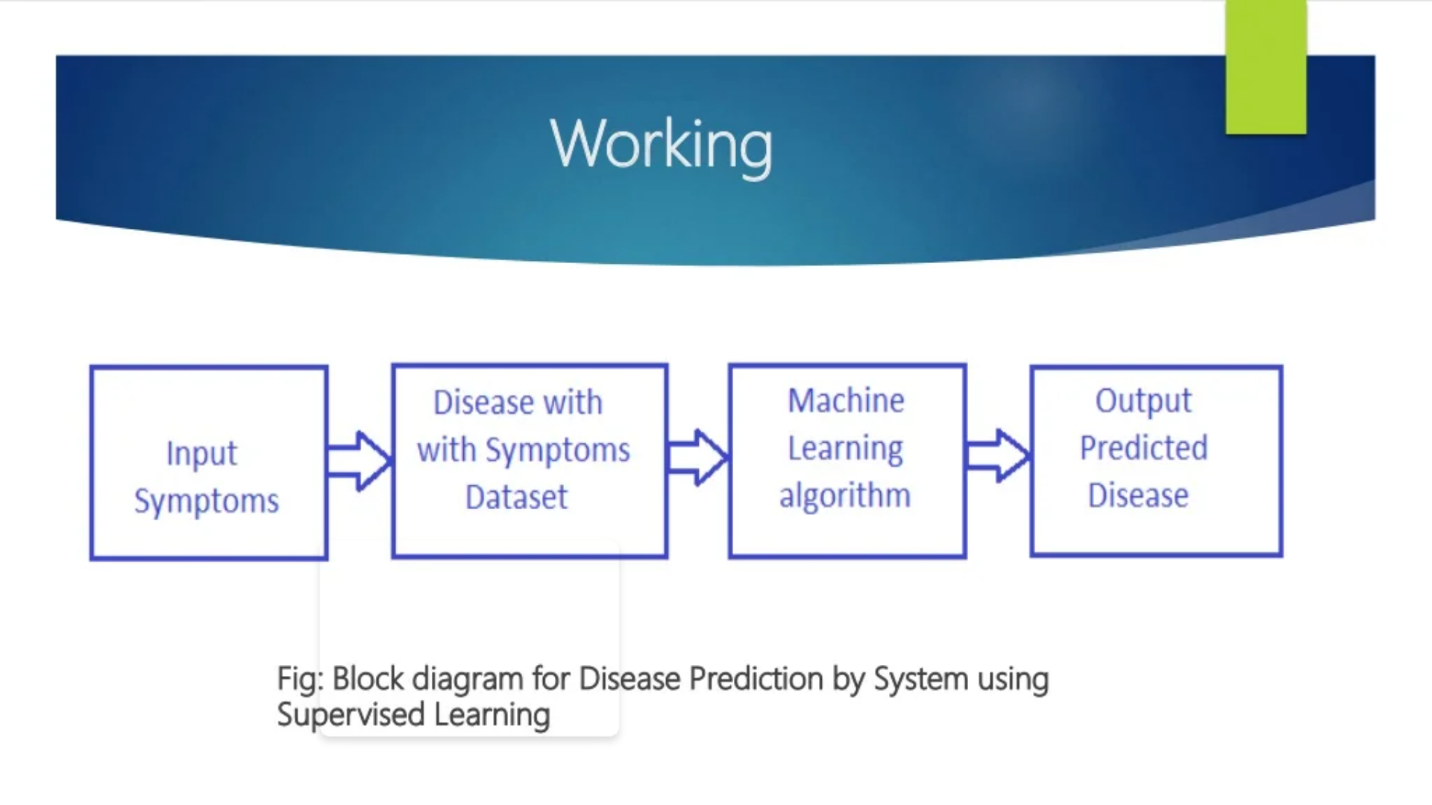
**DISEASE PREDICTION SYSTEM USING MACHINE LEARNING**

***Problem statement: -***

* + Predict disease based on symptoms (in order to improve medical attention given to patients)
  + Classical Diagnosis.
  + Machine Learning algorithms use a computer aided prediction can be made by inputting symptoms

***Working:-***



***Dataset:-***

***Graphical user interface, application, table

Description automatically generated***

***Naïve Based Algorithm: -***

***Graphical user interface, website

Description automatically generated***

***Naïve Based Algorithm:-***

An algorithm is said to be **naive** when it is simple and straightforward but does not exhibit a desirable level of [efficiency](https://wcipeg.com/wiki/index.php?title=Analysis_of_algorithms&action=edit&redlink=1) (usually in terms of time, but also possibly memory) despite finding a correct solution or it does not find an optimal solution to an [optimization problem](https://wcipeg.com/wiki/index.php?title=Optimization_problem&action=edit&redlink=1), and better algorithms can be designed and implemented with more careful thought and clever techniques.

Naive algorithms are easy to discover, often easy to prove correct, and often immediately obvious to the problem solver. They are often based on simple [simulation](https://wcipeg.com/wiki/Simulation) or on [brute force](https://wcipeg.com/wiki/index.php?title=Brute_force&action=edit&redlink=1) generation of candidate solutions with little or no attempt at [optimization](https://wcipeg.com/wiki/Optimization). Despite their inefficiency, naive algorithms are often the stepping stone to more efficient, perhaps even asymptotically optimal algorithms, especially when their efficiency can be improved by choosing more appropriate [data structures](https://wcipeg.com/wiki/Data_structure).

***Naïve Bayes***

Bayes Theorem:

P(A|B) = P(B|A) x P(A)

P(B)

Example: (When CSK WON)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr. No | Toss | Venue | Outlook | Result |
| 1 | Won | Mumbai | Overcast | Won |
| 2 | Lost | Chennai | Sunny | Won |
| 3 | Won | Kolkata | Sunny | Won |
| 4 | Won | Chennai | Sunny | Won |
| 5 | Lost | Mumbai | Sunny | Lost |
| 6 | Won | Chennai | Overcast | Lost |
| 7 | Won | Kolkata | Overcast | Lost |
| 8 | Won | Mumbai | Sunny | Won |

Input = Lost, Mumbai, Sunny ( Given By user)

Predict = Yes / No ( Output Class)

Training Peroid :

In Training Phase what Naïve bayes does is create a Frequency Table for each attribute against the target. Then, molding the frequency tables to Likelihood Tables

For Toss:

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Csk | |
| Won | Lost |
| Toss | Won | 4/5 | 2/3 |
| Lost | 1/5 | 1/3 |

For Outlook:

|  |  |  |  |
| --- | --- | --- | --- |
|  | | CSK | |
| Won | Lost |
| Outlook | Overcast | 1/5 | 2/3 |
| Sunny | 4/5 | 1/3 |

For Venue:

|  |  |  |  |
| --- | --- | --- | --- |
|  | | CSK | |
| Won | Lost |
| Venue | Mumbai | 2/5 | 1/3 |
| Chennai | 2/5 | 1/3 |
| Kolkata | 1/5 | 1/3 |
|  |  |  |  |

Now if we use normal Bayes theorem it will try to find out Probability of user inputs in the dataset for particular target column there is possibility that such combination isn’t present in the dataset.

So,

Probablity of Winning:

P(W|Lost , Mumbai, Sunny) = P( Lost, Mumbai, Sunny|W) x P(W) (This Combination isn’t present)

P( Lost, Mumbai, Sunny )

P(W|Lost , Mumbai, Sunny) = 0 X (5/8)

P(W|Lost , Mumbai, Sunny) = 0

Probablity of Losing:

P(L | Lost, Mumbai, Sunny) = P( Lost, Mumbai, Sunny|L) x P(L) (This Combination is Present)

P( Lost, Mumbai, Sunny)

P(L | Lost, Mumbai, Sunny) = (1/3) x(3/8) = 1/8

Assumption:

To overcome this Problem Naïve bayes made one assumption, it assumes that each and every feature made equal and independent contribution in outcome that’s why its called Naïve Bayes.

After Assumptions:

Probablity of Winning:

P(W|Lost , Mumbai, Sunny) = P(Lost|W) X P(Mumbai | W) X P(Sunny | W) X P(W)

= (1/5) x (2/5) x (4/5) x (5/8)

= 0.040

Probablity of Lost:

P(L| Lost, Mumbai, Sunny) = P(Lost|L) X P(Mumbai|L) X P(Sunny|L) X P(L)

= (1/3) x (1/3) x (1/3) x (3/8)

= 0.013

Now,

P(Won) = (0.040 / (0.040 + 0.013)) = 0.754

P(Lost) = (0.013 /(0.040 + 0.013)) = 0.246

= Prediction Class will be Won.

***Working 1: Initial part***

* Import all the packages required i.e., TKinter for GUI, numpy to perform numerical operations and pandas for reading the csv files.
* Create a list which contains all the symptoms which are according to the csv files.
* Create another list which contain the disease.
* Then, create a empty list  
  L1 and L2 have equal length.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| L1 | Sym1 | Sym2 | Sym3 | Sym4 | Sym5 | Sym6 | Sym7 | Sym8 | ……….. |
| L2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

***Working 2: Dataset part:-***

Perform same steps for both testing and training dataset

* 1. Using Pandas reading the csv files.
  2. Replace with index.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | …. | Prognosis |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |  | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |  | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |  | 2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  | 3 |