CHAPTER 3
RESEARCH METHODOLOGY

The dataset of this research is taken from the klikdokter.com website. This website has a collection of disease and its symptoms. To make the process faster, we scrape this website using Python 3 and Beautifull Soup. First, we have to get the disease list from this website and the disease article URL. Right-click one of the disease name and inspect the element to know what tag and id or class is used to store the disease name, URL, and know what the looping logic is used by the website to show the data. Then we make the scraping code to extract disease name and URL and store it in the array. Print the array and compare it with the website to make sure we have all the data displayed on the website. If the array of data is correct, the next thing to do is open at least 5 disease articles from this website to know where is the symptoms are located. Inspect element of the article, and take note of every tag that stores the symptoms. The symptoms is using <ul> and <li> tag or using a HTML list. Once we got the list to move up and find <h1> tag that can be used as a target. The logic, if the <h1> tag has "Gejala" inside the string, then use it to grab the list inside compare every article so we know the difference of every article and how much different that it has. If the 5 articles don't have the same pattern, open 5 more articles and compare it until the pattern is clear. Find how the system could detect which pattern to used to scrape the article, it can be the start tag, id, class or name. Then make the code to loop array of disease and get URL to open every disease article. After the program gets the website structure, the program has to pick one of the patterns to get the disease symptoms. Every symptom then collected and store the disease and symptoms inside the new array. To analyze the dataset, make a CSV of the array. This step needs some time to finish and requires a stable internet connection. For your information, try to open one of the disease articles and calculate how long your browser loading until the website is fully loaded. That is the time this program needs to run, but the

difference is, we did not scrape 1 article, but at least 300 articles from this website and program need to extract article data until finish than can continue to the next article.

Analyze the CSV dataset using any sheet editor. This is required to make the database design. What we know after analyzing this is there are only 2 components, disease and it's symptoms. With that info, the database should have a table to store the disease name and symptoms name and make 1 table that will store the relation between disease and symptoms. Disease and symptoms table only store the unique data and every row have its id. This ID then used in the relation table to store their relation.

Before typing the code to store CSV data in the database, try to separate disease and symptoms to 3 tables using sheet editor to get the logic to store unique data only. After we get the logic, and the sheet has 3 tables, then we can start to make the database and store the CSV data in it. Make the code to store it in the database.

Naive Bayes Algorithm is used in this research. Before we code for the system, we need to know how the Naive Bayes works. Use the last sheet that we make to calculate Naive Bayes Algorithm manually. Pick at least 5 diseases and their symptoms and make sure 5 of them have 1 or more the same symptoms because this is the real cause of why the problem needs an algorithm. Choose 3-5 symptoms as the user input then start to calculate the score. Naive Bayes has 3 steps, the first step is to get the NC. Compare the input symptoms with each disease. Give the symptoms 1 if the disease has the input symptoms, and 0 if it doesn't have. Then calculate the PV to know input score based on disease appearance and store it. Finally, calculate the final score using all input score times appearance probability. The score should between 0 and 1 and it will have a lot of 0 behind the comma. If it did not like that, your calculation may be wrong, just try again. Try to sort by the highest score to the lowest. The highest is the most correct and vice versa.

The system that this research build is based on the web. Django framework is used to develop the system because python is much faster to manage thousand of data. Django has 3 important parts. Template to store the view, URLs to store web addresses that return specific templates, and models that store our database. First, write the process that this system has. Page 1 is where a user could see the list of supported disease and symptoms, store that in the table tag. Page 2 has to do with user input, this page is where the user chooses every symptom that they feel. For the user convenience, make search feature, a column where user can see the disease that they choose, and reset form to clear user input. Why do we need to search? Because there are about 1500 symptoms, it is hard to search using the scroll bar. A column to store user input is needed so the user can review their symptoms. The last is a reset button, this is needed so the user can quickly clear the input if they manage to fresh start without refreshing the page. The last page is for the result of user input or the Naive Bayes result. This page will show the user what disease related to user symptoms. There is a maximum of 3 highest disease to show to the user. Why there is only 3? Because the system will make users confuse and the user will not focus on the disease, but in the data. The result is stored in the table view, it shows disease name, naive Bayes score, total symptoms in the disease and total same symptoms from user input. Naive Bayes score column is colored, green for the highest one, yellow for the second-best, and the last is gray for the lowest one. Styling for every page is done using bootstrap framework.

Naive Bayes algorithm could be stored in the models.py but we not do that. This system has to be easy to maintain. Therefore, the algorithm code is stored in other file and model have to import that file and call the function. The algorithm is function-based, and for making the testing process easier, the model has to send the database with the function call and algorithm function is not collecting any database data, it just processing data that is sent to the function.

Because of function-based, there is 4 important function. The first one is where the data is stored. Not just getting the data, but this function is loop the data and send it to other function. Another function that mentioned is the rest 3 or the Naive Bayes Algorithm itself. We have analyzed the data and make calculation manual, therefore we know there is 3 step. This 3 step is the next 3 functions. The second function is where NC Score is collected by checking the input is in an array of symptoms from one disease or not. Then the result is a return to the first function and this function sends the returned value to the fourth step. The fourth step is to calculate the PV Score. This score is calculate using NC score and stored in the array too then returned to the first function. After that, it is sent again to the fifth function that calculates the final score. The score then stored in the array just like other functions and returned to the first function. The last function is where the filtering begins. This function is sorting the array from highest to lowest score. When the sorting is done, make a new array and then loop the array result of the sorting. Insert the result in the new array and stop inserting if the new array has 3 data in it. To make it more real, dived total disease with the total correct input disease. The result is used to sort the disease. The system has suggested a disease that has fewer symptoms but the highest selected symptoms.

What it needs to do next is to make sure that the calculation score is correct. To do this, we need to make a manual calculation. We have done manual calculation before, but we can not use it because that calculation only uses some of the data. Right now, we have to try entering some of the diseases in the page diagnosis, take note of the selected disease and submit the data. Then take note again for the 3 diseases from the result page and the score. Get the symptoms of the resulting disease and write in the sheet application. Next, write the input too, after that we can use it to make a manual calculation. The information that we needed for this calculation is total disease and symptoms. We can get that information by count row data of the database. If all the information needed is collected, we can start the manual calculation with 3 naive Bayes steps. When

done, get that naive Bayes score, and compare it with the system and make sure it is the same. If not, try to correct the algorithm or move back to step about the algorithm code.

The next thing to do is to test the system. Try to get several diseases and input their symptoms in the diagnosis page. The system should suggest that disease than others. If not try to capture why it is not recommended the targeted disease. Some case has to do with the sorting using total symptoms divided by total input that appear in the disease.

The last step for this research is to make a research report. The report is needed to share any step we did to make this research so that anyone can see and do the same research or improve the available research. The research has to include all the steps and information about the research as detail as it should.

$$P(a_i|a_j) = \frac{N_c + mp}{n + m}$$

Illustration 1: Naive Bayes Formula