****

**Disease Prediction Model using Machine Learning Algorithms**

**CSE445**

**Section - 3**

**Spring 2021**

**Prepared by:**

1. **Name: Sumit Kumar Das  
   ID: 1731847642**
2. **Tasneem Mahmud  
   ID: 1731893642**
3. **Durdana Kamal  
   ID: 1813355042**

**Abstract** Every human illness is usually preceded with the display of certain symptoms. The symptoms play a critical role in the eventual onset of a disease. There have been growing efforts to predict disease status, in order to initiate treatment before it gets untreatable. Machine learning has been applied successfully in several biological domains, usually predicting a specific disease. Here, we review the prediction model which will be able to predict a number of diseases, based on their symptoms. We shall review some of the methods and the algorithms that they are based on. We also perform an analysis of symptoms and disease datasets that have given us improved results, using a variety of machine learning and feature extraction methods. The disease prediction model with its code, generated features and datasets are available via GitHub: <https://github.com/durdanakamal/Durdana-Kamal-2-1813355042_CSE445_3_Spring-21>

1. **Introduction**Supervised machine learning algorithms are heavily used in the data mining field. These algorithms are also useful for prediction models. Medical informatics and disease prediction have been popular topics of discussion among the data science community in recent years, mainly because machine learning has been integrated into the health sector in various forms, such as electronic health records, administrative data, etc.

In this project, we performed One Hot Encoding on the symptoms dataset to “binarize” the category and included it as a feature, to help train the model. We have applied multiple algorithms to train our data, in order to improve our accuracy and predictive prowess. Two databases were taken from [www.kaggle.com](http://www.kaggle.com), which contained disease data and their specific symptoms, and the symptom severity.

Both the Random Forest (RF) algorithm and the Decision tree algorithm showed superior accuracy, precision and F1 scores of 100%.

This project compares between two supervised machine learning algorithms, Decision tree and Random forest, and provides an overview of the predictive performance, precision and accuracy of the model, for disease prediction. This information can provide some insight into the biological field, as it utilises the available information at hand to predict a disease that can be treated as soon as possible.

1. **Related Work**

The disease Prediction system is based on predictive modeling which will predict the disease of the accused on the assumption of the symptoms that the accuser provides as an entry to the system. The system will then analyze the symptoms given by the user as entry and will give the probability of the disease as an output. The process is carried out using the Decision tree classifier. Decision tree classifier estimates the probability of the disease. With the growth of big data in the health care sector, accurate analysis of medical data leads to early detection of certain diseases, which can then be treated. This system can be used to predict an illness by analyzing the symptoms. A decision tree classifier is used to evaluate the model. This system is used by end-users.This system uses Machine Learning Technology for predicting diseases, the decision tree classifier algorithm is used.To improve the accuracy from massive data, the existing work will be done on unstructured data. For the prediction of diseases, the existing will be done on linear, KNN, Decision Tree algorithm.The decision tree type used in this research is the gain ratio decision tree. The gain ratio decision tree is based on the information gained approach,which selects the splitting attribute that will not maximize the value of the information gained approach, therefore maximizing the information gain. Information gain is the link between the original information content and the amount of information needed. The features are ranked by the information gained, and then the top-ranked features are chosen as the potential attributes used in the classifier.

To distinguish the splitting attribute of the decision tree, one must calculate the information gain for each attribute and then select the attribute that will maximize the information gain. [1]

In the case of general diseases prediction using ML where the classification task is used for prediction of working cases dependent on past information. Many data mining techniques such as numpy, neural network, decision tree have been applied by researchers to have a precision diagnosis in heart disease. The accuracy given by different algorithms will vary depending on the number of attributes. This research provides diagnostic accuracy score for improvement of better health results, as the accused will be informed beforehand.In the particular Heart disease prediction project they have used WEKA tool in this research for pre-processing the dataset. Just 14 attributes out of all 76 different attributes have been considered for analysis to get precise results. By comparison and analysis using different algorithms with WEKA tool heart disease can be predicted and treated early thus increasing the chance of survival of the accused.[2]

As there is big data progress in biomedical and healthcare societies, accurate studies of medical data shows that it benefits early disease recognition, patient care and society services. When the quality of medical data is incomplete the precision of study is reduced. Additionally, many regional diseases display symptoms differently across the world, which might reduce the prediction ability of the model. In the particular system, it provides machine learning algorithms for effective prediction of various disease occurrences in disease-frequent societies.

It uses the altered estimate models generated from real-life hospital data. It uses a latent factor model to rebuild the missing data, to deal with the problem of incomplete data. This deals with a regional chronic illness, known as cerebral infarction. It uses Map Reduce algorithm and Machine Learning Decision Tree algorithm on structured and unstructured data from hospitals. As far as we are concerned, none of the existing models or work in the field of medical big data analytics, focused on both structured and unstructured data. As opposed to many other estimate algorithms, the accuracy of the proposed algorithm reaches 94.8% with a faster convergence speed than the CNNbased unimodal disease risk prediction (CNN-UDRP) algorithm.[3]

1. **Methodology**In this section, we will first introduce the concept of One Hot Encoding in Section 3.1, where we will elaborate the manner of implementation in our project. Then we will introduce the Random Forest algorithm in Section 3.2, and at last discuss the Decision Tree Algorithm in Section 3.3.  
     
   **3.1. One Hot Encoding**Sometimes in datasets, there are columns which represent categorical data, and sometimes the data in the column is label encoded. To avoid confusion in the machine learning model, this data should be one hot encoded, which means splitting the column into many columns. It depends on how many categories are present in the specific column. Each column will have either “0” or “1” corresponding to where it has been placed.

One hot encoder only works with numerical categorical values, which is why any string value should be label encoded before it is one-hot encoded.

Note that the one hot encoder does not accept a 1-dimensional array or a pandas series as input; it must always be a two-dimensional array. Strings should not be present in the data given to the encoder.

|  |
| --- |
| **3.2. Random Forest** Random forest is a [machine learning algorithm](https://builtin.com/data-science/introduction-to-machine-learning) that gives efficient results even without hyper-parameter tuning. It is a simple, easy-to-use algorithm and it is diverse because it can be used for both regression and classification tasks.  Random forest is a supervised learning algorithm. It builds a “forest” of decision trees, which are usually trained with the “bagging” method. It produces a better result as it is a combination of learning models.  Random forest is a [supervised learning algorithm](https://builtin.com/data-science/supervised-learning-python). The "forest" is actually a collection of decision trees, which are trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models improves the accuracy of the overall result.  A random forest builds multiple decision trees and merges them to increase the accuracy in its prediction.  Random forest is diverse as it can be used for both classification and regression tasks.  two tree random forest  Figure 1. This diagram shows how a random forest would look like with two trees.  Random forest has almost the same hyperparameters as a bagging classifier or a decision tree. Usually there's no need to combine a decision tree with a bagging classifier because you can always use the classifier-class in random forest. By using the algorithm's regressor, we can carry out regression tasks.  Random forest searches for the best feature among a random subset of features, instead of looking for the best feature. This increases the diversity in the results.  To increase the randomness of the model, we can additionally use random thresholds corresponding to each feature rather than search for the best possible thresholds.  While predicting, it is possible to measure the relative importance of each feature. Sklearn provides a tool that measures a feature's importance by looking at how much impurity is reduced across all trees, by the tree nodes that are using that specific feature. The score for each feature is calculated after training and the results are scaled so that the sum of importance equals 1.  Feature importance assists us to avoid overfitting. When there are too many features present, it is easy for the model to pick up on the noise in the data. Feature importance can help eliminate some features that have little to no importance.  The hyperparameters of sklearns built-in random forest function, can increase the predictive power and speed of the model.  The n\_estimators hyperparameter is the number of trees that the algorithm will build before taking the average prediction. The higher the number of trees, the better the accuracy. However, the computation time will also increase.  Secondly, another hyperparameter is max\_features, which determines the maximum number of features that random forest will consider in order to split a node.  Thirdly, another hyperparameter is min\_sample\_leaf, which is the minimum number of leafs that are required to split an internal node.  The n\_jobs hyperparameter gives the engine the information of the processor usage limit. A value of 1 means it is allowed to use only one processor. A value of -1 means it is allowed to use infinite processors, i.e. there is no limit.  The random state hyperparameter ensures that the output of the model is repeatable. When a random state is set to a fixed value and the model is given the same hyperparameters and training data, the model will always deliver the same results.  Finally, there's the random forest cross-validation method oob score (also known as oob sampling). About one-third of the data in this sample is not used to train the model but can be utilized to assess its performance. They are called out-of-bag samples. It's comparable to the leave-one-out cross-validation method, except there's essentially no extra computational work involved.  **3.3 Decision Tree**  One of the first and most well-known machine learning techniques is the decision tree (DT). A decision tree represents the decision logics for classifying data objects into a tree-like structure, i.e. tests and outcomes. A DT tree's nodes usually have numerous layers, with the root node being the first or top-most node.  All internal nodes (those with at least one child) reflect input variable or attribute testing. The classification algorithm branches towards the appropriate child node based on the test result, and the process of testing and branching repeats until it reaches the leaf node.  The choice outcomes are represented by the leaf or terminal nodes. DTs are a common component of many medical diagnostic regimens since they are simple to understand and learn.  When traversing the tree for a sample's classification, the results of all tests at each node along the path will provide enough information to make a guess about the sample's class.    Figure 2. An example of a decision tree. A circle represents each variable (C1, C2, and C3), whereas rectangles reflect the choice outcomes (Class A and Class B). Each branch is labeled with either ‘True' or ‘False' based on the outcome value from the test of its respective parent node in order to successfully classify a sample to a class.  **3.4. Difference between random forest and decision trees**  When you give a decision tree a training dataset with features and labels, it will generate a set of rules that will be used to make predictions.  The random forest algorithm, on the other hand, chooses observations and features at random to create numerous decision trees and then produces an average of the results.  Another distinction is that "deep" decision trees may be prone to overfitting. Random forest usually prevents this by generating random subsets of the different features and using those selections to build smaller trees. However, the greater the number of trees, the greater the computational time and lesser the chances of the predictive model to work. |

1. **Results**  
     
   We used a database from Kaggle for this project. Here is the link <https://www.kaggle.com/itachi9604/disease-symptom-description-dataset?select=symptom_Description.csv&fbclid=IwAR2DAnvYMce_GAkzxtJtb8HOb-4EuexTvluPh_N3ovk-_tnINajaMVNlDaA>. Here, we used two datasets one called “dataset.csv” and the other is “Symtom-severity.csv”. We mainly used the libraries numpy and pandas. Firstly, we used .head() to see what the dataset looks like. It showed the first five rows as default. After taking a look at the dataset we saw that there were a lot of null values starting from Symptom7 to Symptom17. Since, there were so many null values we decided to get rid of those columns and ended up with six symptom columns. Then we used .describe() to check different attributes of the dataset. Afterwards we check for null values in the remaining columns. We saw that symptom4, symptom5 and symptom6 still had some null values. So we filled the null blocks with 0 later on in the code. Next we found the count of unique items in each column.

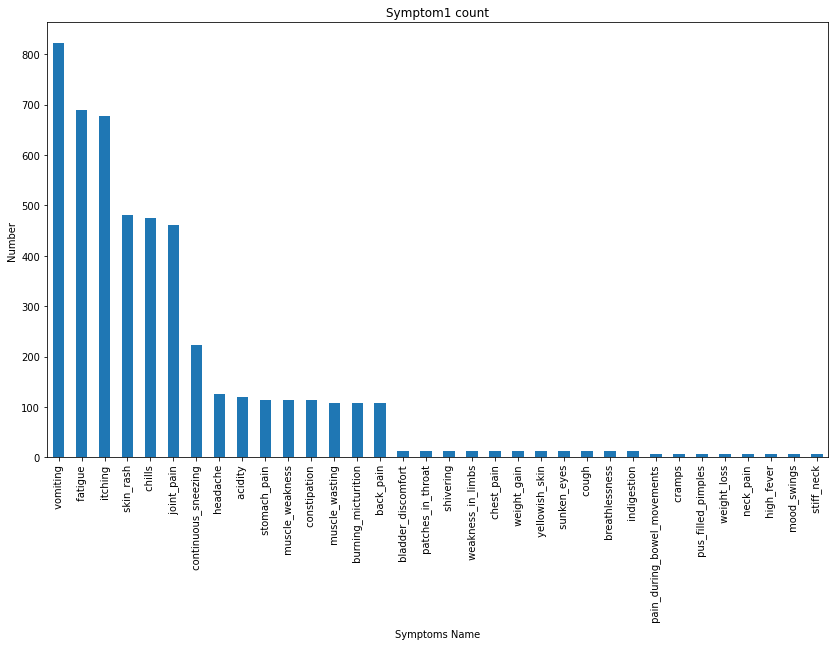
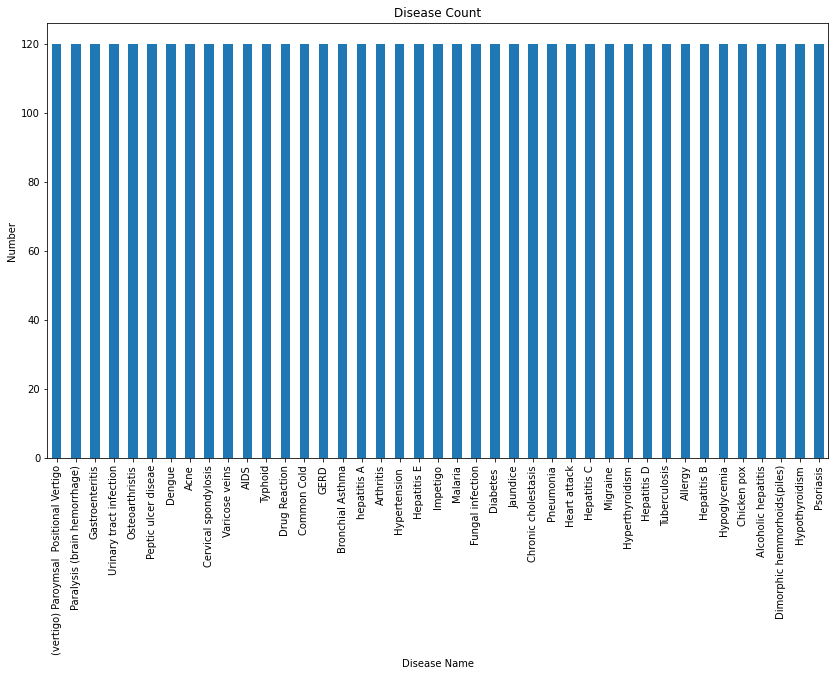


Figure 3. Disease Count Figure 4. Symptom 1 Count

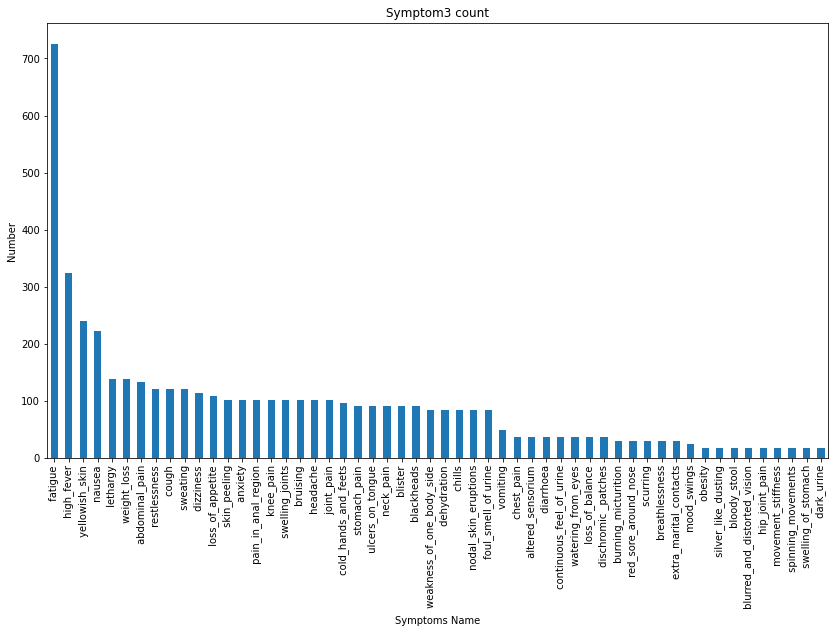
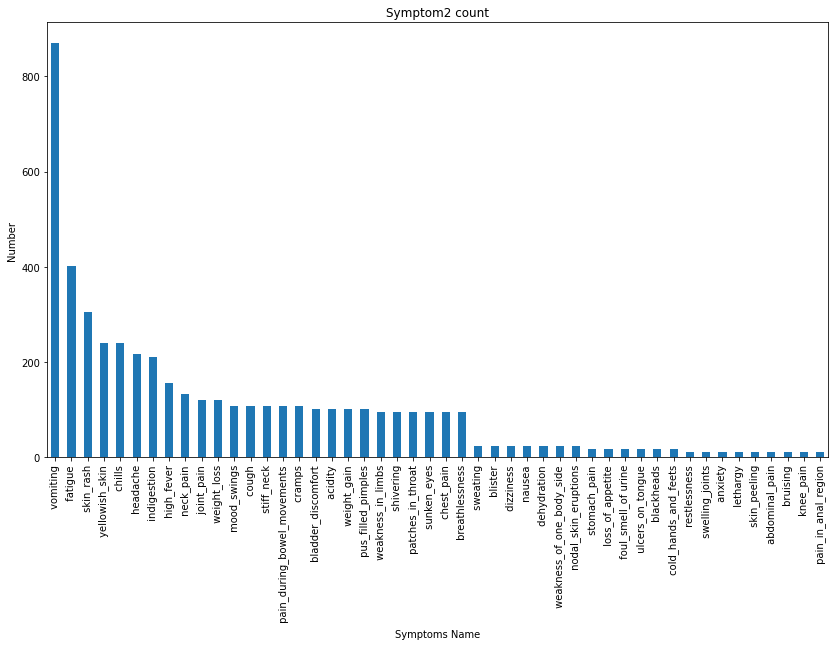


Figure 5. Symptom 2 Count Figure 6. Symptom 3 Count

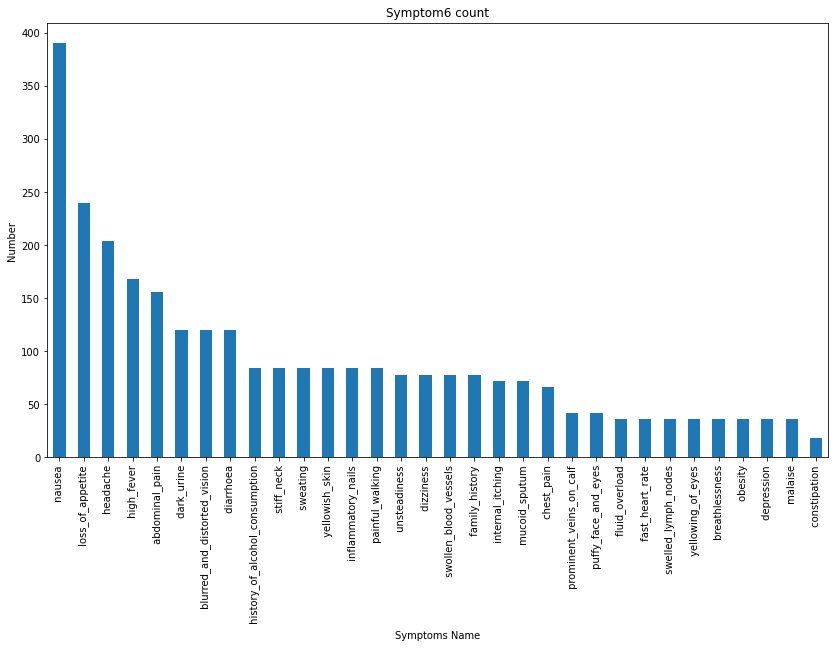
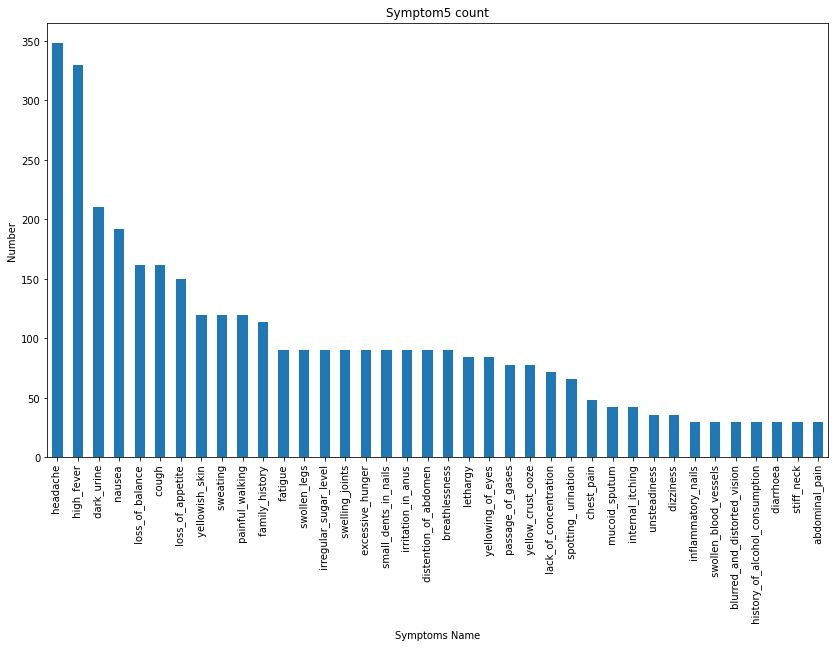


Figure 7. Symptom 5 Count Figure 8. Symptom 6 Count

Then we removed extra spaces from the data in the columns to make the data consistent. Then we created relplot diagrams to check the matching symptoms against the diseases.

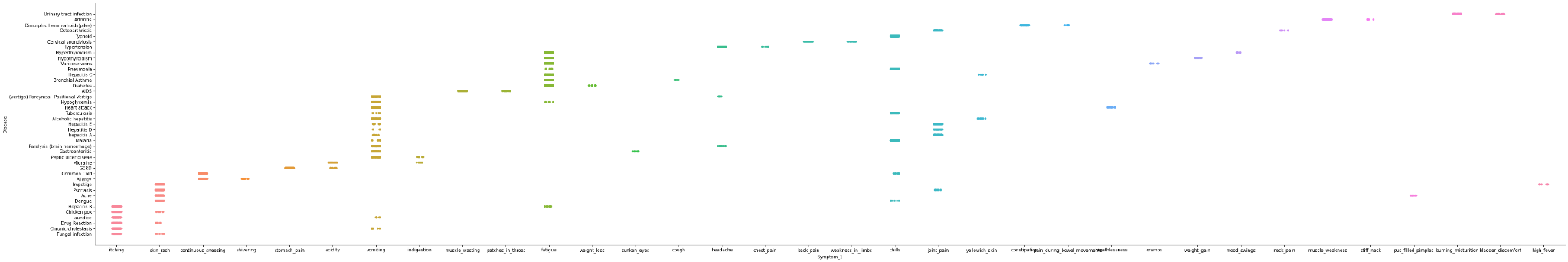


Figure 9. Disease against Symptom 1

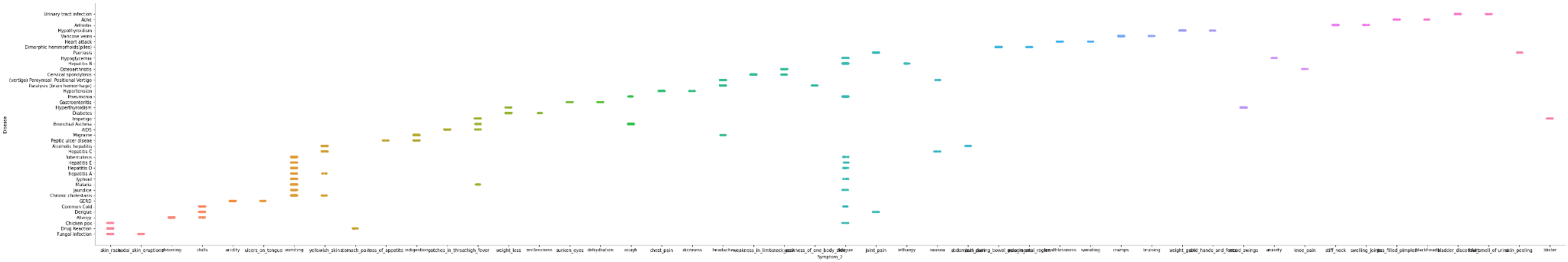
****

Figure 10. Disease against Symptom 2

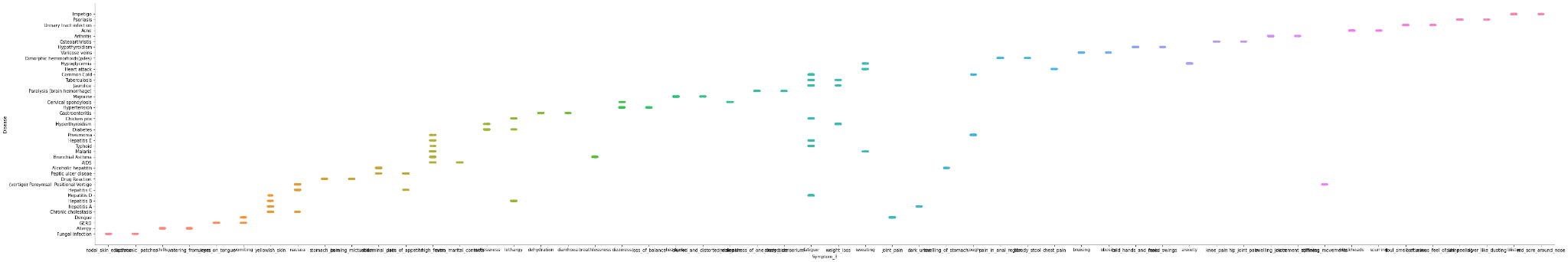
****

Figure 11. Disease against Symptom 3

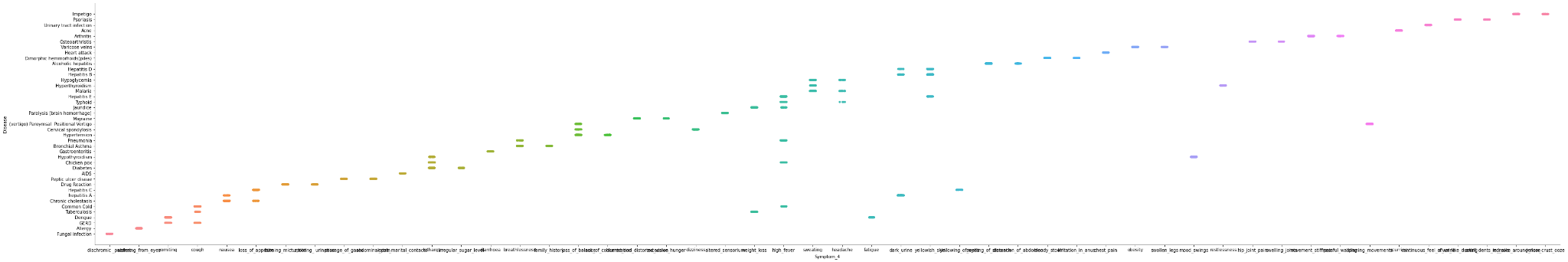
****

Figure 12. Disease against Symptom 4

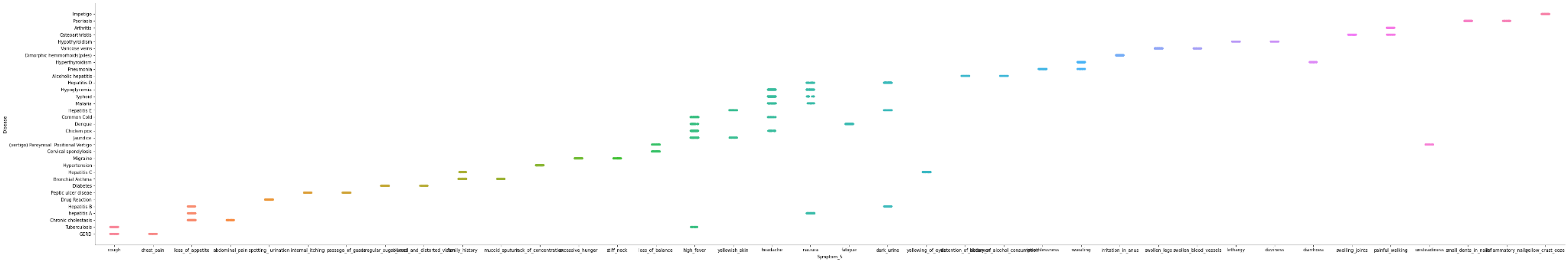
****

Figure 13. Disease against Symptom 5

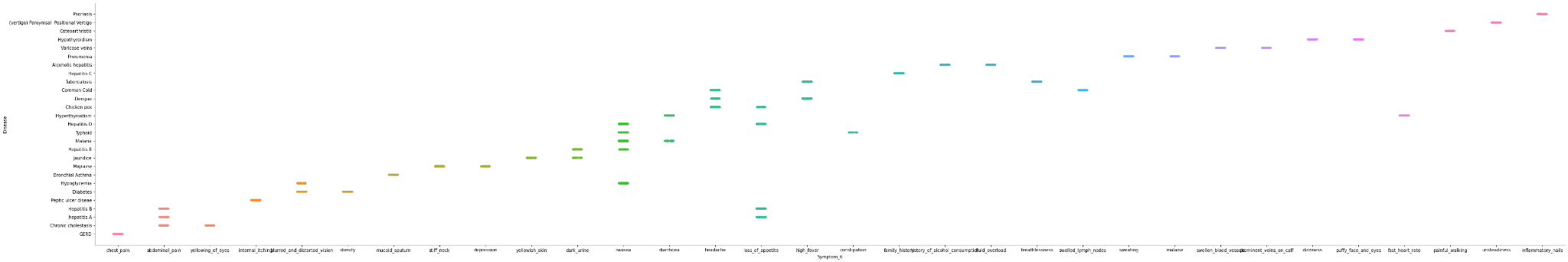
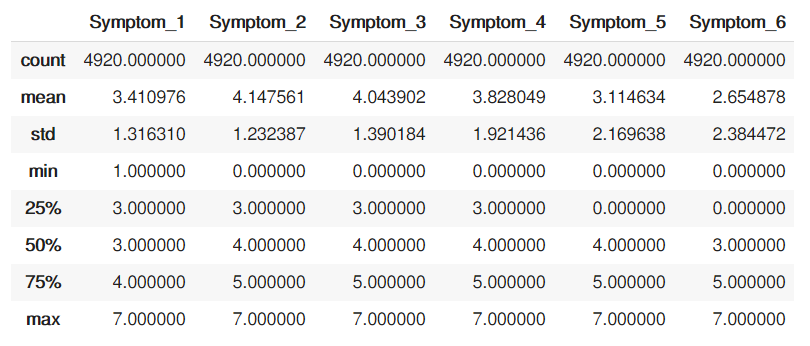
****

Figure 14. Disease against Symptom 6

Then we integrated the two datasets and replaced the symptoms by their severity. After replacing the symptoms we saw that there were some symptoms that did not have any weight. So we replaced those symptoms with 0. And after replacing the symptoms with numerical data we used .describe() to check the mean, mode, etc.



**Table 1. Data description**

Then we checked for matching symptoms in each column using .pairplot().

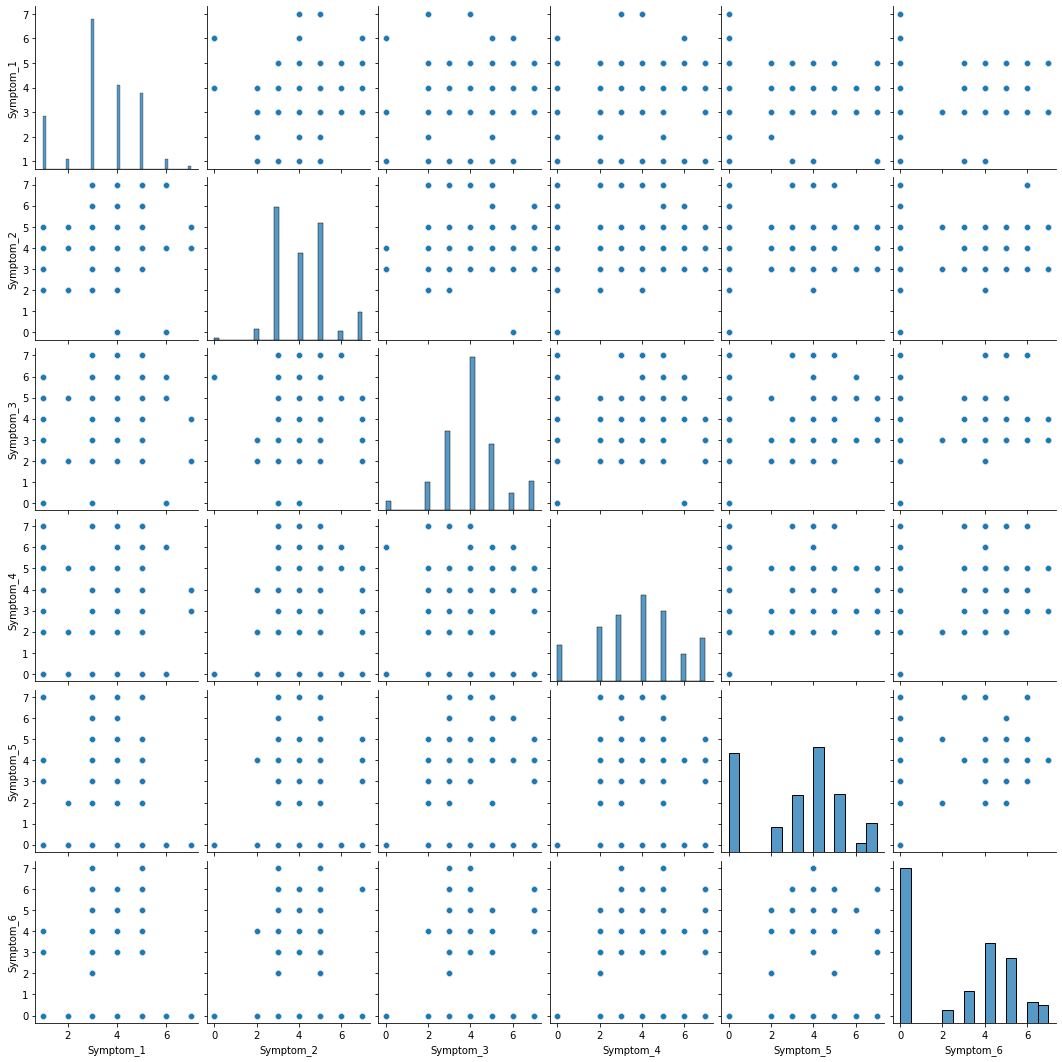


Figure 15. Checking matching symptoms in each Symptom column

We then created a correlation matrix using sns.heatmap()

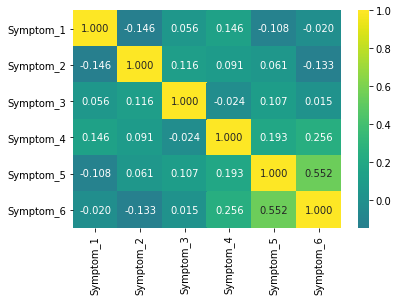
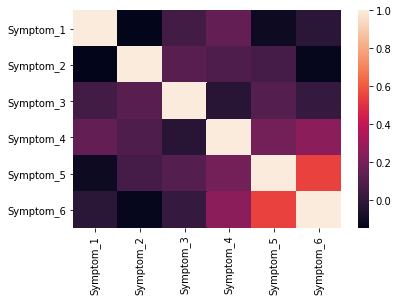
****

Figure 16. Correlation matrix

Then we split the data taking the disease column in a different variable. After that we performed one hot encoding since the disease column was completely categorical. Then we used the SVM model to train and check accuracy. We got a f1 score of 96.99%, Accuracy of 96.99%, Precision 97.07% and Recall 97.17%. Then we created a confusion matrix for that model.

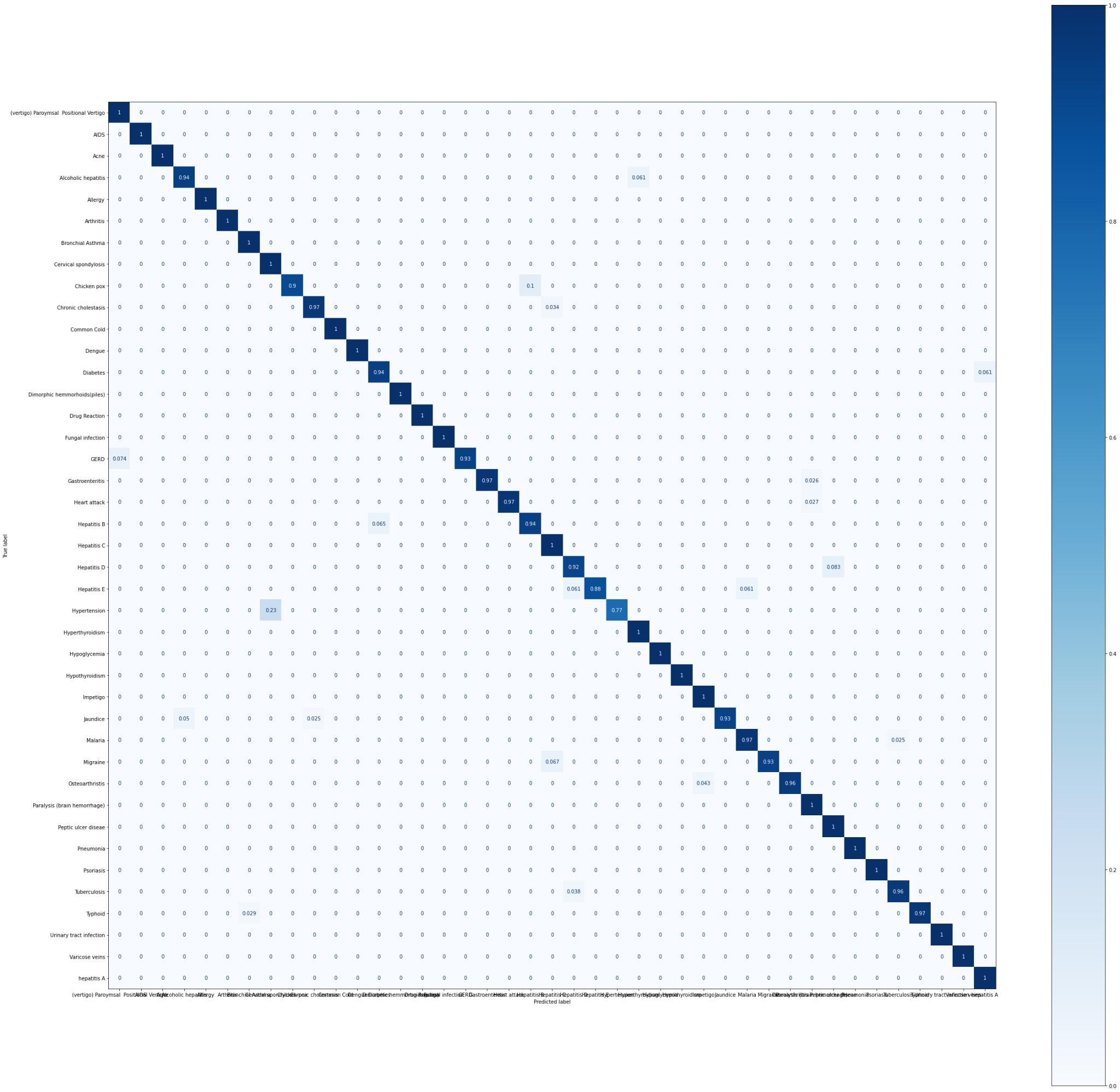
****

Figure 17. Confusion matrix for SVM

Then we used the Random Forest Classifier. For this model F1,Accuracy,Recall and Precision all were 100%. Here is the confusion matrix.

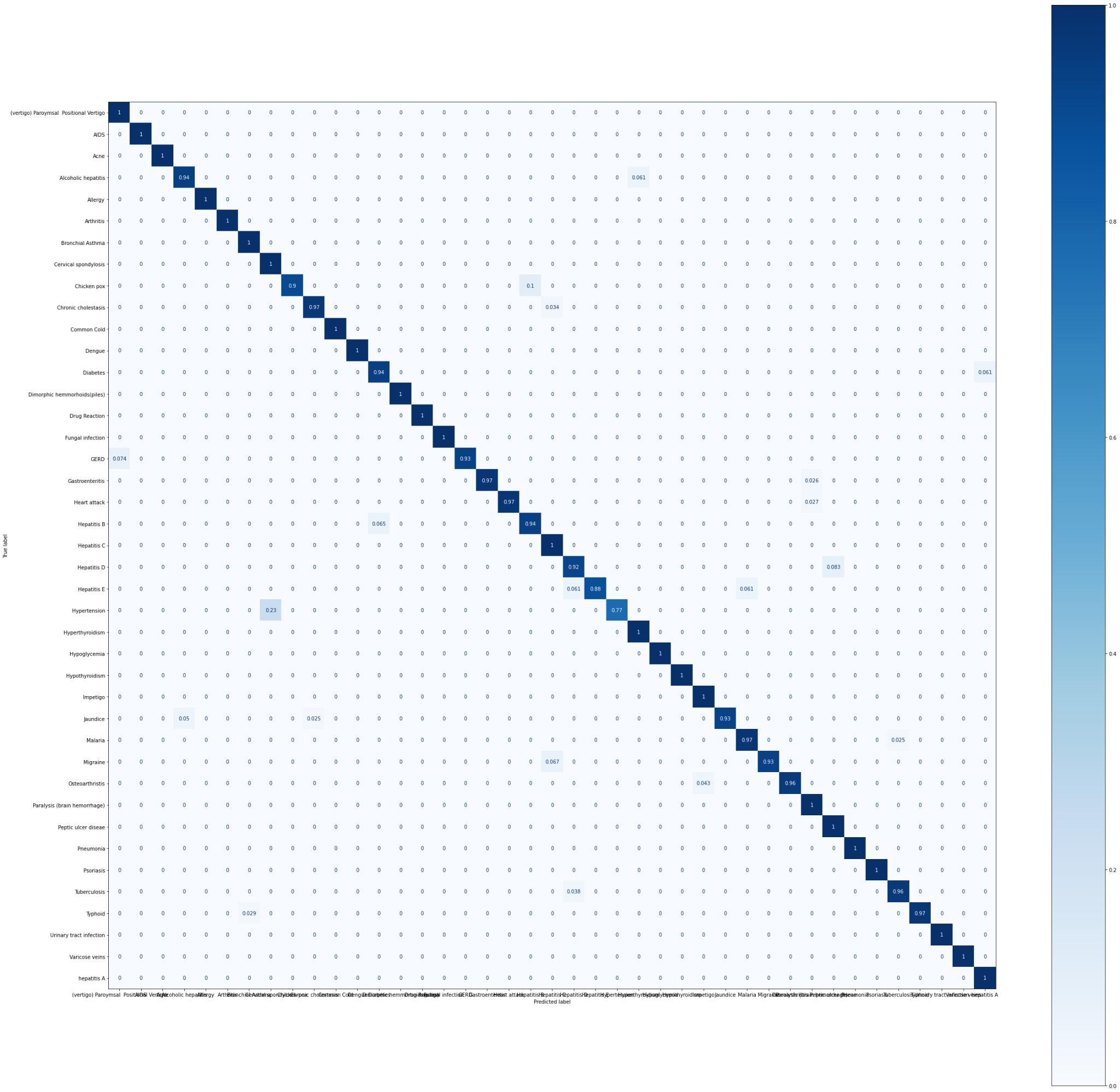


Figure 18. Confusion matrix for Random Forest Classifier

Lastly, we used a Decision Tree. For this F1,Accuracy, Precision and Recall all were 100%. Here is the confusion matrix.

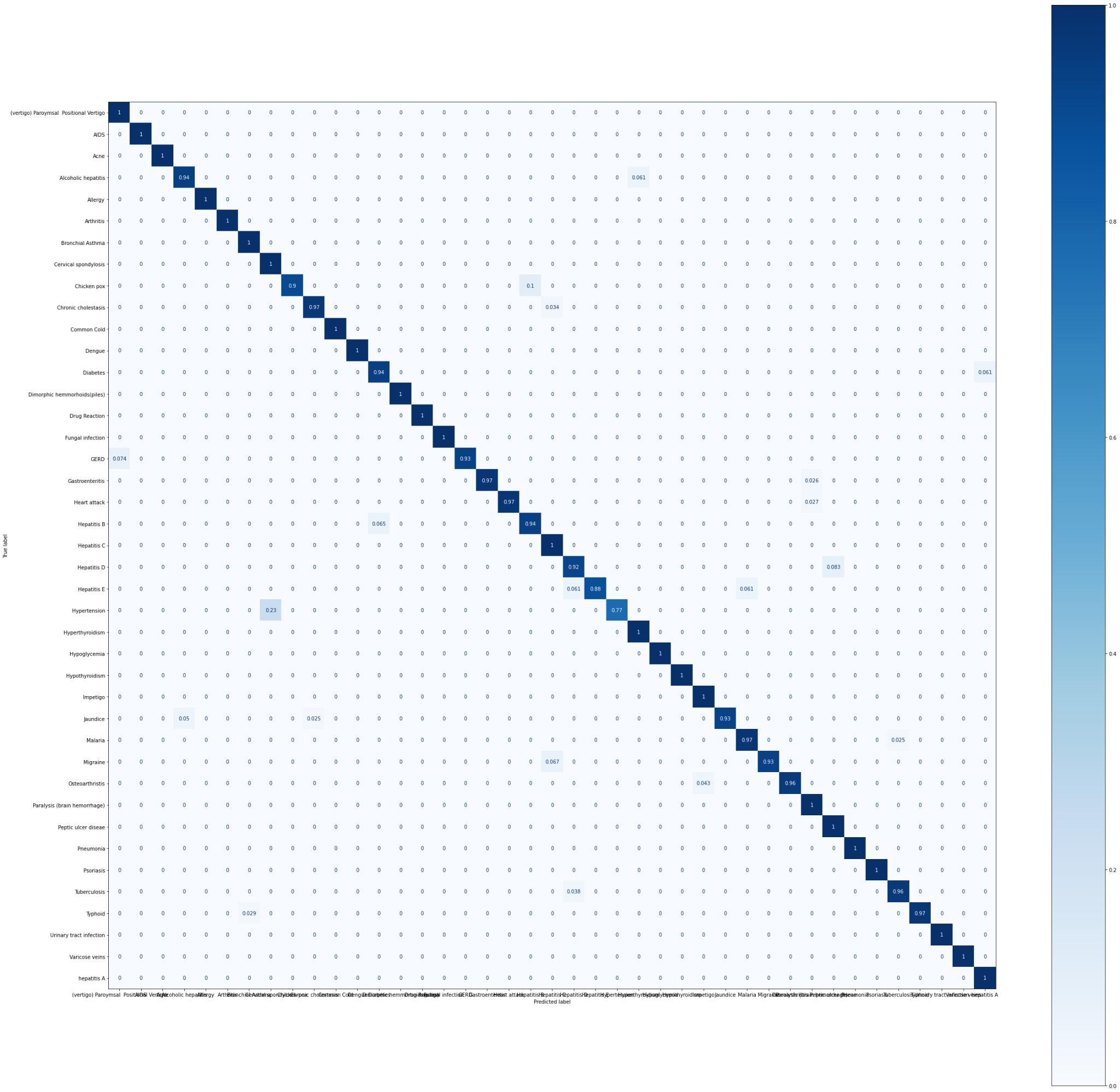
****

Figure 19. Confusion matrix for Decision Tree

1. **Conclusion:**

Illness isn’t something people take lightly. Often the panic is more harmful than the disease itself. Especially when an unknown life threatening disease is spreading worldwide so quickly that it gives us flashbacks of the 1910s, when the destructive wave of the spanish flu created havoc all over the world. This virus, widely known as the coronavirus, has already taken millions of lives and has left the rest in panic. Our goal is to help people understand the disease more, so that they can save their own lives and of those they care about, by taking necessary precautions. And in the future, no one should live their lives oblivious to the diseases out there. A timely prediction of a disease will help to slow down the disease, if not prevent it. The aim of this research is to predict whether or not a patient will develop any kind of disease or not. This research was done on supervised machine learning classification techniques using Numpy, decision tree, random forest. Various experiments using different classifier algorithms were conducted . Research was performed on 6th generation Intel Corei5 having an 6200H processor up to 2.4 GHz CPU and 8 GB ram. Dataset was classified and split into a training set and a test set. Pre-processing of the data is done and supervised classification techniques such as Numpy, decision tree, K-nearest , and random forest are applied to get an accuracy score. The accuracy score results of different classification techniques were noted using Python Programming for training and test data sets. Percentage accuracy scores are depicted in different algorithms.

**References :**

[1] Uddin, S., Khan, A., Hossain, M. *et al.* Comparing different supervised machine learning algorithms for disease prediction. *BMC Med Inform Decis Mak* 19, 281 (2019). <https://doi.org/10.1186/s12911-019-1004-8>  
  
[2] https://www.geeksforgeeks.org/ml-one-hot-encoding-of-datasets-in-python/

[3] Raj H. Chauhan, Daksh N. Naik, Rinal A. Halpati, Sagarkumar J. Patel, Mr. A.D.Prajapati-“Disease Prediction using Machine Learning”,May 2020,

https://www.irjet.net/archives/V7/i5/IRJET-V7I5385.pdf

[4] [Devansh Shah](https://link.springer.com/article/10.1007/s42979-020-00365-y#auth-Devansh-Shah), [Samir Patel](https://link.springer.com/article/10.1007/s42979-020-00365-y#auth-Samir-Patel) & [Santosh Kumar Bharti](https://link.springer.com/article/10.1007/s42979-020-00365-y#auth-Santosh_Kumar-Bharti) -“Heart disease prediction.” *SpringLink*, October 2020,<https://link.springer.com/article/10.1007/s42979-020-00365-y>

[5] S, Vinitha and S, Sweetlin and H, Vinusha and S, Sajini, Disease Prediction Using Machine Learning Over Big Data (February 2018). Computer Science & Engineering: An International Journal (CSEIJ), Vol.8, No.1, February 2018- <https://ssrn.com/abstract=3458775> or [http://dx.doi.org/10.2139/ssrn.3458775](https://dx.doi.org/10.2139/ssrn.3458775)