Medi-App: A Python Application for Self-Diagnosis of Illnesses

Faculty of Engineering and Applied Science, University of Regina

Ali Rizvi, Alwin Baby, Kelly Holtzman, Mansi Patel

Introduction

Medi-App provides the symptom-to-illness investigation of online clinical databases without the time consumption of searching online. Symptoms and illness prognosis data are from online machine learning database Kaggle (Anonymous, 2020) under the Database Contents License v1.0, and are fit for our purpose of academic use. The data are used to train a Bayesian Network (Bayes's Net) and an Artificial Neural Network (ANN) with Multi-Layer Perceptron (MLP). The resulting probabilistic models provide Medi-App with the capability to compare user symptoms with past observed symptoms, and report the probability that the user is observing symptoms that indicate the presence of a particular illness or disease.

Objective

We want the application to be desirable to use over manual investigation of illnesses for the average, non-software or medical person: the application should not take more time than manual investigation of the same illnesses using arbitrary means not including the application or its underlying concepts.

Conclusions & Future Work

From the results in the above section we determine that the Bayes's Net implementation was most effective for our purpose: Bayes's Net was the preferred choice for the criteria outlined, including runtime, understandability, and implementation complexity. We can conclude that future work on the Medi-App application should expand further on the Bayesian Network implementation portion.

Acknowledgments

Dr. Kin-Choong Yow, Professor, Faculty of Engineering and Applied Science, University of Regina

Further Information

https://github.com/holtzmak/Medi-App

Reference

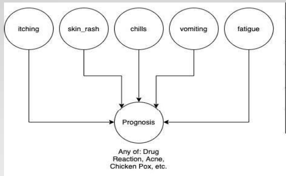
Anonymous (Nirma University) (2020). Disease Prediction Using Machine Learning (Version 1) [Data set]. Kaggle. https://www.kaggle.com/kaushil268/disease-prediction-using-machine-learning

Materials and Methods

The Medi-App is implemented in two different ways. The different approaches were used :

1. Bayesian Network
2. Artificial Neural Network

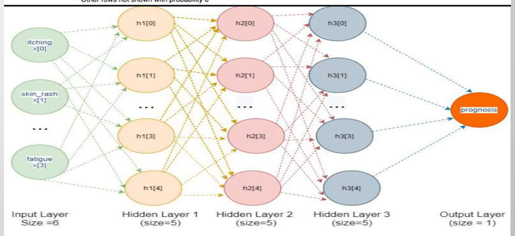
To implement application using Bayesian Network, the `pomegranate` library were used.
To implement application using ANN method, the `sk-learn` libraries were used.



Symptom I Probability			
itching	0.137804878	not_itching	0.862195122
skin_rash	0.1597560976	not_skin_rash	0.8402439024
chills	0.162195122	not_chills	0.837804878
vomiting	0.3890243902	not_vomiting	0.6109756098
fatigue	0.3926829268	not_fatigue	0.6073170732

Chicken Pox Prognosis I Conditional Probability					
not_itching	skin_rash	not_chills	not_vomiting	fatigue	0.04545454545
itching	not_skin_rash	not_chills	not_vomiting	fatigue	0.04545454545
itching	skin_rash	not_chills	not_vomiting	not_fatigue	0.04545454545
itching	skin_rash	not_chills	not_vomiting	fatigue	0.8636363636

Other rows not shown with probability 0



```
hidden_layer1 = g1(W1*x+b1)    g1 is the  activation function, W1 are the weights and b1 is the bias term
hidden_layer2 = g2(W2*h1+b2)
hidden_layer3 = g3(W3*h2+b3)
output_layer= g3(W3*g2(W2*g1(W1*x))) + [b3+...]
```

Result

Criterion for Comparison	Preferred Algorithm Given Criteria	Bayesian Network (Using `pomegranate` package)	ANN (Using `sk-learn` packages)
Full runtime (See Appendix A)	Bayes's Net, faster time	0.1044455678 seconds (from samples)	3.2411391276 seconds
Runtime to generate model (See Appendix A)	Bayes's Net, faster time	0.0436754752 seconds (from samples)	2.685259133 seconds
Understandability	Bayes's Net, because of it's simple, graphical nature	Bayes's Nets are comparatively simple to understand as the graphical format is straightforward and the idea that the distribution tables are tied to nodes made it easier to translate to a program	ANN was complex and confusing: choosing how many layers, nodes, and iterations would be needed was not straightforward. Not sure if our implementation is even the optimal choice for these features
Implementation Complexity	Bayes's Net, because examples online could be used almost directly for our purposes despite the difficulty looking at deeper package methods	Fairly simple to start with using the online examples, however the `pomegranate` package is not very well defined and the input/output provided by package objects and their functions was difficult to understand and use	Finding the right package modules for training was difficult because no online examples were using the latest python version. `sk-learn` had very good documentation once the right modules were identified
Dataset Complexity	ANN, because the structure of the ANN can be taken directly from dataset header data	It is difficult to prepare the dataset manually for the algorithm, we must know the format of data and program for it. It is possible for the package to figure out it's own pattern but it is usually not correct, hence we have to explicitly define the graph nodes and edges	Have to explicitly know the structure of the ANN in order to program it from a known dataset. Similar to Bayes's Net
Accuracy (of package used)	ANN, because it is easier in our case to obtain more data than it is to obtain known, accurate data	Depending on data used, if the dataset is accurate the Bayes's Net's predictive accuracy will be on par at least	More data means more accuracy: the number of data attributes is directly proportional to accuracy