DISEASE DIAGNOSER – A DOCTOR'S REFERENCE

A Project

Presented to the faculty of the Department of Computer Science

California State University, Sacramento

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

by

Pragathi Narendra

FALL
2018

DISEASE DIAGNOSER – A DOCTOR'S REFERENCE

A Project

by

Pragathi Narendra

Approved by:

_____, Committee Chair
Dr.Anna Baynes

_____, Second Reader
Dr. Haiquan Chen

_____
Date

Student: Pragathi Narendra

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

_____, Graduate Coordinator _____
Dr. Jinsong Ouyang Date

Department of Computer Science

Abstract

of

DISEASE DIAGNOSER – A DOCTOR'S REFERENCE

by

Pragathi Narendra

The diagnosis of a disease is the most critical and vital job in medicine and it mostly depends on a doctor's intuition based on past experiences. The unfortunate case of recognizing the incorrect symptoms results in a misdiagnosis. To avoid such medical misdiagnoses, my project shows it is beneficial to utilize large datasets collected by healthcare industries to automate the diagnosis of diseases. Such a tool can assist doctors to avoid the unwanted biases in diagnosis. In addition, an automated medical diagnosing system would be useful if the symptoms are ambiguous because including large datasets to the currently known symptoms may illuminate the case.

This project aims to develop machine-learning models to automate the diagnosis of disease. Apart from existing base algorithms and hybrid algorithm (Naïve Bayes with decision tree) in literature, my project implements a new hybrid algorithm with Naïve Bayes and random forest for classification of diseases. Hybrid algorithms are used to overcome the limitations of basic algorithms thereby producing a better machine learning model, which improves accuracy of classification.

The tool allows the user to select the category of disease and enter the symptoms of the selected category. Among all the machine-learning models developed using the preprocessed datasets, the model that achieves highest accuracy with the test dataset is used to analyze the symptoms entered by the user. The result of the diagnosis along with the probability of its occurrence and accuracy of the developed model are displayed to the user.

_____, Committee Chair
Dr. Anna Baynes

_____
Date

ACKNOWLEDGEMENTS

I would like to thank Dr. Anna Baynes for her guidance, continuous support, encouragement and patience during my Master's project implementation. I would also like to thank Dr. Haiquan Chen for his contributions to this project as a second reader. I am very grateful to the California State University, Sacramento and the Computer Science Department of the university.

I would like to thank specially to my parents, family who were my strength, support at all times, and for helping me achieve my goals, which would have otherwise been only a dream.

TABLE OF CONTENTS

Page

LIST OF TABLES

Tables Page

LIST OF FIGURES

Figures Page

1. INTRODUCTION

1.1 INTRODUCTION

Diagnosis of a disease is based on a doctor's knowledge and experience. But under some circumstances the prediction can be wrong which leads to incorrect treatment to the patient. Hence to confirm if the diagnosis is right, it would be useful if there could be a tool which uses data from several patients to diagnose the disease and its probability of occurrence.

In this project, diseases such as breast cancer, cervical cancer, diabetes, heart disease, hepatitis, liver disorders, lung diseases, kidney diseases, Parkinson's disease, skin diseases, thyroid and vertebral column disease are automated for diagnosis. The datasets are downloaded, preprocessed and patterns of disease are analyzed using several machine learning algorithms. A User Interface (UI) is designed to collect the symptoms of patient which are to be diagnosed. Among the different machine learning algorithms implemented in the tool, the best algorithm is selected to diagnose the symptoms of the disease entered by the user in the UI and the result of diagnosis is displayed to the user.

1.2 BACKGROUND WORK

In [1], the authors attempted to detect heart disease using Coronary Heart Disease Dataset from Cleveland Clinical Foundation in two ways. The first approach is using a single data mining algorithm for the heart disease dataset and benchmark the baseline accuracy. Further in the second approach they tried using hybrid algorithms such as J4.8 decision tree, and bagging algorithms. The authors propose that the application of voting

techniques, different data discretization levels and reduced error can increase the accuracy of data mining algorithms. They achieved a maximum of 84.1% accuracy using their techniques and claim that further research is in progress.

Also in [2], heart disease dataset from UCI machine learning repository was analyzed. The authors used an ensemble learning method – adaptive boosting algorithm for classification on the Hungarian Institute of Cardiology (HIC) dataset and the highest accuracy obtained after various experiments done with an ensemble of classifiers is 96%.

The researchers in [3], used real world datasets such as UCI repository and SMBBMU (Shaheed Muhtarma Benazir Bhutto Medical University) for diagnosing various thyroid diseases such as Euthyroid, Hypothyroid, Hyperthyroid, Sub-clinical-hypothyroid and sub-clinical-Hyperthyroid. Two classifiers such as multi and binary SVM are used. The accuracy of the system is 95.7% with 10-k fold cross validation.

Apart from heart disease and thyroid disease, there has been work on other diseases such as diabetes, breast cancer, dengue, hepatitis, liver disorders, and more. To identify liver diseases, in [4] SVM and Naïve Bayes algorithms are used individually and a highest accuracy of 79% is obtained. In [5], the same classification is performed using various algorithms such as J48, MLP, SVM, Bayesian network and random forest classifiers individually. The highest accuracy obtained is 71% for the same.

For detection of diabetes in [6] the authors use decision tree and Naïve Bayes algorithm individually and reach a maximum of 79.68% accuracy. In [7] the CART algorithm, adaboosting and logiboosting techniques are used and the accuracy result is only

78.65% maximum. But in [8] the authors use hybrid decision tree algorithm with Naïve Bayes algorithm for heart disease prediction and achieve 95% accuracy. In same work the hybrid algorithm is tested for different datasets such as contact lens, iris plant, breast cancer, soybean, glass, image segmentation, and tic-tac-toe win. The results of hybrid algorithms have better performance than using an individual decision tree or Naïve Bayes algorithm.

When comparing work in [8] with other related research, the usage of the hybrid algorithm has better accuracy than usage of individual algorithms. From literature we know that the Naïve Bayes algorithm is computationally expensive if used with many attributes. Hence the fewer the attributes, the better is its performance. However, the attributes to be considered for Naïve Bayes could be a question. As an answer to this question, a hybrid algorithm of Naïve Bayes algorithm is used with a decision tree in [8]. The top K-most important attributes involved in classification for a decision tree can be used for Naïve Bayes algorithm thereby reducing the cost of computation.

1.3 PROPOSED SYSTEM`

From the literature, we see that there has been a lot of effort to improve existing algorithms in different ways. In my project, I have developed machine learning models from native algorithms and hybrid Naïve Bayes with decision tree algorithm discussed in literature. In addition, I have also implemented hybrid Naïve Bayes and random forest algorithm. I chose to implement the new hybrid algorithm because in hybrid Naïve Bayes decision tree, the decision tree algorithm has a disadvantage of overfitting. Using the most

important attributes from an over fitted decision tree can lead to a performance issues in the Naïve Bayes algorithm too. Hence by utilizing the strength of ensemble learning technique, I tried to overcome the limitation of decision tree by using random forest.

Also in literature, efforts are made to improve efficiency of particular algorithms with a specific dataset. Whereas in my project, I have developed a tool which uses datasets belonging to category “diseases”. Datasets analyzed in this project include breast cancer, cervical cancer, diabetes, dermatology, heart diseases, hepatitis, liver diseases, lung disorders, kidney diseases, Parkinson’s disease, thyroid disease, and vertebral column disorders. I have tried optimizing multiple algorithms to achieve high accuracy in the datasets used.

2. TOOLS AND TECHNOLOGY STACK

2.1 R LANGUAGE

R is an open source programming language that is mainly used for data mining and statistical computations. R is runnable on Windows, Mac and Linux distributions. The R language supports a variety of statistical modelling, statistical tests, classification and clustering based modelling and time-series analysis [9]. It also has many built-in packages that helps in preprocessing datasets. According to Wikipedia as of May 2018, apart from core packages included with installation of R, around 12500 additional packages are available with Comprehensive R Archive Network (CRAN) [10]. In this project, I have used the R programming language for cleansing and pre-processing the dataset by utilizing its simple yet powerful preprocessing packages.

2.2 R STUDIO

RStudio is an open source software tool which provides a computing environment for R scripting language. RStudio runs on the desktop computer with Windows, Linux or Mac operating system and on other hand runs in a browser connected to RStudio Server [11]. RStudio supports visualizations, GIT access and authoring PDF's, word documents, presentations and web pages. The following is a screenshot of RStudio environment.

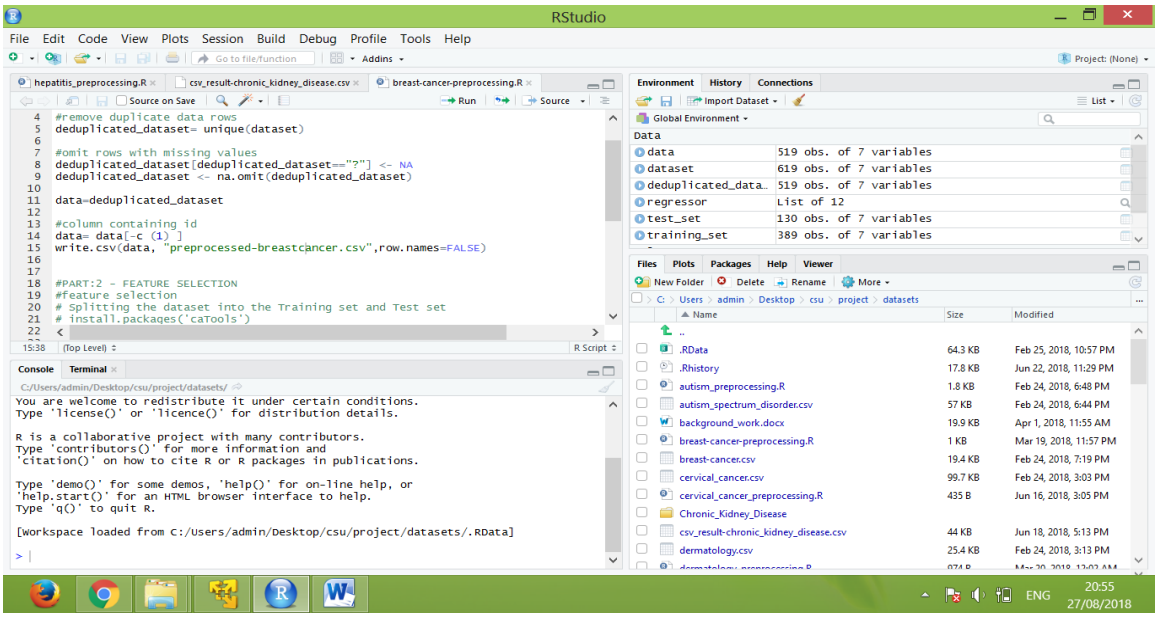

Figure 1 – Rstudio environment screenshot

2.3 PYTHON

Python is an open source programming language which has several uses and applications. It supports a wide variety of machine learning and data mining algorithms in the scikit-learn package. The machine learning models built with scikit-learn package are tunable for high levels of robustness and consistency and that is the main reason I chose to build my project models using Python over R.

Also, there are many Python based Graphical User Interface (GUI) such as PyQt, GTK, PyForms, Kivy which helps in building a frontend graphical tool for the project [12]. From the python based frontend tool, data can easily be collected for processing using machine learning models. In the backend, the collected data is analyzed with the machine

learning models built with scikit learn to draw an inference. After analysis in the backend, data can easily be displayed in the GUI for user to view the inference. The integration of frontend and backend is much easier this way and that's another reason for choosing Python for processing in this project.

In addition to above reasons, Python can be easily embedded as scripting language in software and web applications [13] which can make my project scalable to a larger audience in the future.

2.4 PyQt

PyQT is a free software that acts as Python plugin and can be used as a cross platform GUI toolkit [14]. According to Wikipedia: "PyQt implements around 440 classes and over 6,000 functions and methods including:

- a substantial set of GUI widgets
- classes for accessing SQL databases (ODBC, MySQL, PostgreSQL, Oracle, SQLite
- QScintilla, Scintilla-based rich text editor widget
- data aware widgets that are automatically populated from a database
- an XML parser
- SVG support
- classes for embedding ActiveX controls on Windows (only in commercial version)" [14]

Among the above strengths, the main reason to choose PyQt is the availability of substantial set of GUI widgets with drag and drop capabilities and its seamless usability. The following is a screenshot of the PyQt GUI window.

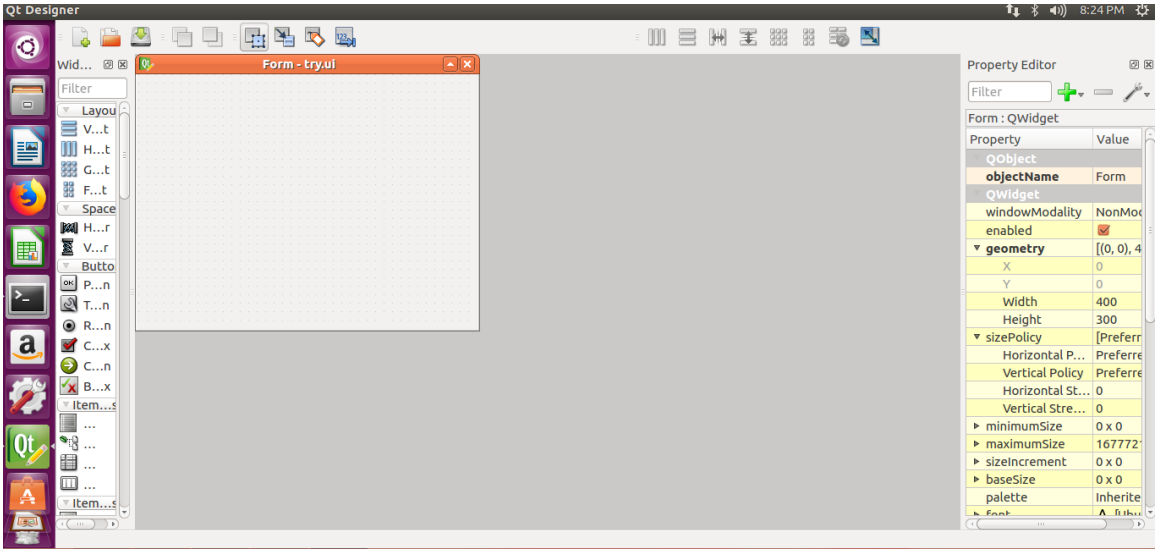

Figure 2 – PyQt designer window screenshot

3. PROJECT DESIGN

The project involves a frontend GUI which collects and displays data and a backend machine learning model which analyzes data from frontend. The frontend is a GUI which contains three set of screens.

The first screen, also known as disease selector screen, allows the user to select the category of the disease. Upon selecting a disease category, the second screen which is the symptom collector screen, pops up and prompts the user to enter the symptoms of the selected disease. After submitting the symptoms, the third screen pops up and displays the diagnosis made by the machine learning model, accuracy of machine learning model used and the probability of the prediction to occur. The first screen and third screen are the same for all diseases.

In the backend, multiple machine learning models are built for every disease from pre-processed datasets. For example to diagnose diabetes, the preprocessed diabetes dataset is used to develop machine learning models using different algorithms. The model that generalizes well by performing classification with highest accuracy is selected to diagnose the disease symptoms entered by user. Upon classification, the diagnosis, probability of diagnosis, and the accuracy of the model are passed to frontend to display the results to the user.

For the GUI design and implementation, PyQt was used. For the pre-processing dataset, R was used. In the backend, Python was used. The block diagram below gives a diagrammatic representation of the project design.

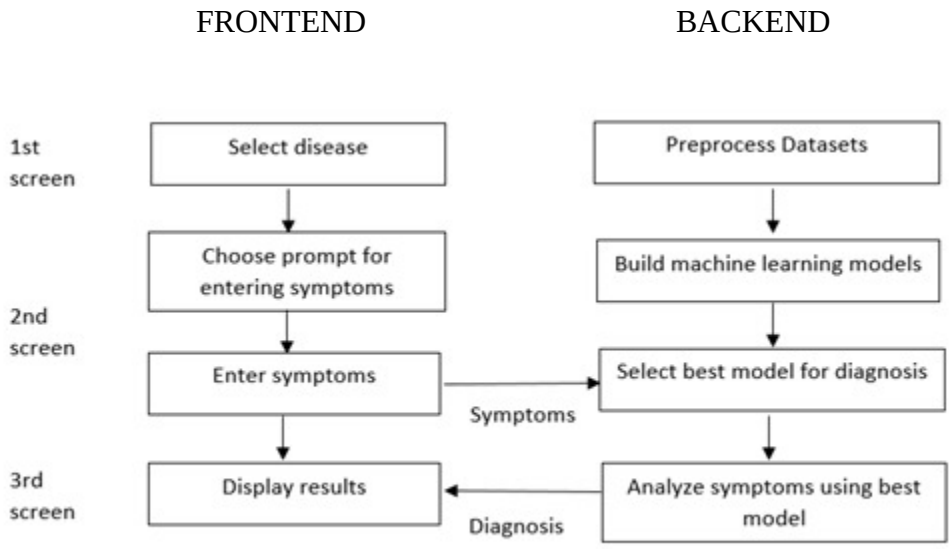

Figure 3 – Block diagram representing project design

4. PROJECT IMPLEMENTATION

4.1 DATA SOURCE

Automated diagnosis of disease involves analyzing symptom patterns associated with the disease. The table below lists diseases automated with the dataset's source website and corresponding links in reference section. UCI refers to University of California, Irvine

Disease	**Website Name**
Breast cancer	UCI Machine learning Repository [15]
Cervical cancer	UCI Machine learning Repository [16]
Diabetes	Kaggle [17]
Heart disease	UCI Machine learning Repository [18]
Hepatitis	UCI Machine learning Repository [19]
Kidney disease	UCI Machine learning Repository [20]
Liver disorders	Kaggle [21]
Lung disease	UCI Machine learning Repository [22]
Parkinson's disease	UCI Machine learning Repository [23]
Skin diseases	UCI Machine learning Repository [24]
Thyroid disease	UCI Machine learning Repository [25]
Vertebral column disorders	UCI Machine learning Repository [26]

Table 1 – List of datasets and website references

4.2 PREPROCESSING DATASET

Every dataset contains duplicate entries, entries with missing values, entries with categorical values, multivalued attributes, attributes with irrelevant entries. All these types of entries and attributes are considered as barriers against achieving accuracy and generalization of the built machine learning model.

Duplicate entries may create unwanted bias towards a set of the final output from the machine learning model. Hence in this project, all the duplicate entries are removed if they belong to the same class. All datasets were de-duplicated and this is done using the code snippet below written in R:

```
#import dataset
dataset = read.csv('hepatitis.csv')
#remove duplicate data rows
deduplicated_dataset= unique(dataset)
```

Missing values can be handled by either eliminating the complete entry or by replacing the missing value with the average value of the attribute across the dataset or by analyzing the pattern and replacing the missed value with similar entries attribute value. In this project for all datasets, missing entries were eliminated using following code snippet in R:

```
#omit rows with missing values
deduplicated_dataset[deduplicated_dataset=="?"] <- NA
deduplicated_dataset <- na.omit(deduplicated_dataset)
data=deduplicated_dataset
```

Some attributes may have categorical attributes for example: male and female. These categorical values need to be converted to discrete entries such as 0 and 1. In this project any categorical attributes were converted into numeric codes using following code snippet in R.

```
data$X.rbc. = factor(data$X.rbc.,
                     levels = c('normal','abnormal'),
                     labels = c(1, 2))
```

Multivalued attributes involve more than one value for an entry in an attribute. It is better to separate the values into different attributes for achieving better generalization. In this project across the datasets described above, I did not find any multivalued attributes. The following table lists categorical attributes mapped and attributes removed from all datasets.

Dataset	**Categorical attributes mapped**	**Irrelevant attributes removed**
Breast cancer	None	Sample code number
Cervical cancer	None	Number of diagnosis, Time since first diagnosis, Time since last diagnosis, Dx.CIN
Diabetes	None	None
Heart disease	None	None
Hepatitis	None	None
Kidney disease	X.rbc, X.pc, X.pcc, X.ba,X.htn, X.dm, X.cad, X.appet, X.pe,X.ane,X.class	None
Liver disorders	Gender	None
Lung cancer	DGN, PRE6, PRE7, PRE8,PRE9,PRE10, PRE11, PRE14, PRE17,PRE19,PRE25, PRE30,PRE32, RISK	None
Parkinson's disease	None	None
Skin disease	None	None
Thyroid disease	None	None
Vertebral column disorders	Results	None

Table 2 – Categorical attributes mapped and attributes removed

4.3 DISEASE SELECTOR SCREEN

The User Interface (UI) was designed using PyQt designer and its functionalities are implemented in Python. UI contains a home screen which shows a list of diseases automated for diagnosis in this project. The following is a screenshot showing the disease selector screen.

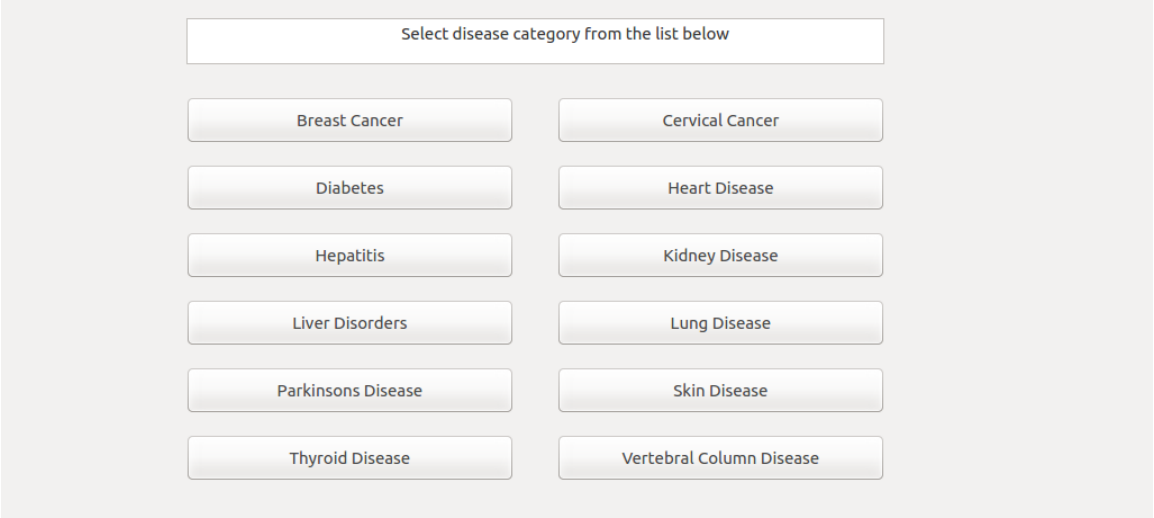

Figure 4 – Screenshot of disease selector screen

4.4 SYMPTOM COLLECTOR SCREEN

The second screen, the symptom collector, is based on the disease selected in first screen. The user is prompted to enter the symptoms based on the disease selected. Upon submission, the entries are validated. If there are any missing entries in a text box, it is replaced with 0.

The following is the code snippet in Python for the same.

```
text=(str(self.plainTextEdit_2.toPlainText()))
if text=="":
     myfile.write("0"+"\n")
else:
     myfile.write(str(self.plainTextEdit_2.toPlainText())+"
     \n")
```

If there is no option selected from the radio button then it is replaced with the most common entry from dataset. The following is the code snippet in Python for the same.

```
if self.radioButton_10.isChecked() == True:
     myfile.write("1"+"\n")
else:
     myfile.write("0"+"\n")
```

The validated entries are stored in a file and applied as test data to the best machine learning model developed for that corresponding dataset. The results of the analysis are stored in another file. The following is a sample screenshot of the symptom collector from the project.

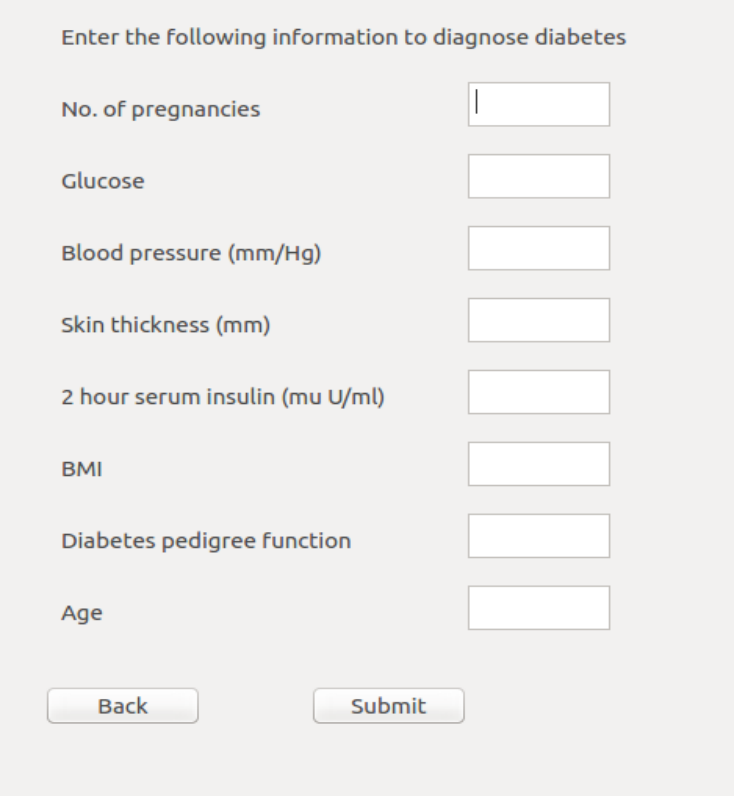

Figure 5 – Screenshot of symptom collector for diabetes

4.5 DIAGNOSIS USING MACHINE LEARNING ALGORITHMS

Each of the pre-processed datasets is split into a training dataset and a testing dataset using the following Python code:

```
X=data[:,0:26]
Y=data[:,26]
# Splitting training and testing dataset
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size = 0.3, random_state = 100)
```

The training dataset is used to train the machine learning model against several patterns. The trained machine learning model is tested for its generalization capabilities using the test dataset. Several machine learning algorithms are used in this project for this purpose. Among all the algorithms used, the algorithm that achieves highest accuracy with the test dataset is used for diagnosing the symptoms entered in UI prompt by the user. In the sections 4.5.1 and 4.5.2, the algorithms used in this project are discussed further.

4.5.1 BASIC ALGORITHMS

The basic machine learning algorithms used for diagnosis in this project are Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Artificial Neural Networks (ANN), Naïve Bayes algorithm, Decision Tree (DT), and Random Forest (RF). The implementation details are as follows:

```
#SVM
svm_clf = svm.SVC()
svm_clf.fit(X_train,Y_train)
y_pred= svm_clf.predict(X_test)

#KNN
knn_clf=KNeighborsClassifier(n_neighbors=3)
knn_clf.fit(X_train,Y_train)
y_pred= knn_clf.predict(X_test)

#ANN
ann_clf = MLPClassifier()
ann_clf.fit(X_train,Y_train)
```

```
y_pred= ann_clf.predict(X_test)
#RF
rf_clf= RandomForestClassifier()
rf_clf.fit(X_train,Y_train)
y_pred= rf_clf.predict(X_test)



#DT
dt_clf = DecisionTreeClassifier(criterion = "entropy",
random_state = 100)
dt_clf.fit(X_train, Y_train)
y_pred = dt_clf.predict(X_test)



#Naive Bayes
nb_clf = GaussianNB()
nb_clf.fit(X_train, Y_train)
y_pred = nb_clf.predict(X_test)
```

4.5.2 HYBRID ALGORITHMS

The purpose of using hybrid algorithm is to overcome the disadvantage of one algorithm with the strength of another algorithm. Two hybrid Naïve Bayes algorithms are implemented in this project. One algorithm involves Naïve Bayes and random forest. The other algorithm involves Naïve Bayes and decision tree.

4.5.2.1 PURPOSE OF HYBRID ALGORITHM

The usage of Naïve Bayes classifier can often be computationally expensive when there are too many attributes to be considered for mining. Hence having fewer attributes leads to better performance. To achieve this, the top K most important attributes are selected from decision tree or random forest for Naïve Bayes algorithm and only the top K most important attributes are used for training the hybrid model. The important attributes involved in classification can be determined from the tree generated in random forest and decision tree as the tree split is based on information gain. Hence the first K attributes based on which split is made on decision tree/random forest are used. Based on size of dataset, the k value is decided for each dataset.

4.5.2.2 IMPLEMENTATION OF HYBRID ALGORITHM

The top K attributes are extracted from the random forest and decision tree using the n-largest method in Python. The method returns the attributes in an array in the order of importance and the array is reduced to K number of important attributes. The following is the code snippet that does this.

```
dimportance=dt_clf.feature_importances_
N=len(dimportance)
from operator import itemgetter
from heapq import nlargest
result=nlargest(15,enumerate(dimportance),itemgetter(1))
dimportance2=[item[0]for item in result]
```

Choosing the value of K depends on the size of dataset. The K-value should be selected such that, it is greater than the minimum number of attributes required for analysis and lesser than total number of attributes. If K-value is less than minimum number of attributes required for analysis, the accuracy of hybrid algorithm will be less. On the other hand, K-value should be lesser than total number of attributes to avoid losing accuracy for considering irrelevant attributes. Hence it is necessary to find an optimum K-value to achieve improved accuracy. The following figure represents relation between K-value and accuracy for a dataset with 20 attributes.

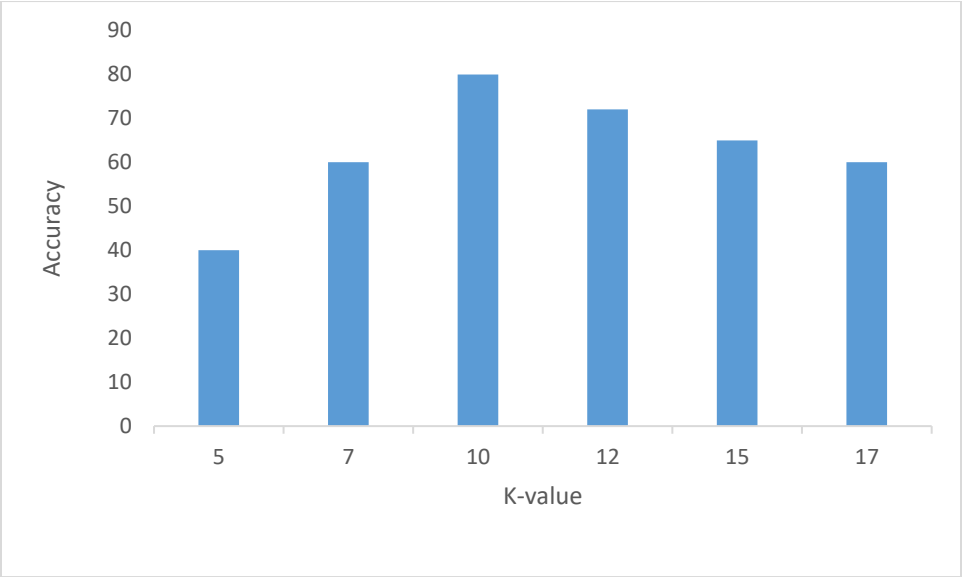

Figure 5- Relation between K-value and accuracy

From the above figure, we see that if K-value is high or low the accuracy is less. Hence it is essential to select K-value carefully for the implementation of hybrid algorithm.

For datasets with less than 10 attributes, the hybrid algorithm is not effective. On the other hand, for datasets with 10+ attributes, the K value was selected based on trial and error.

The K-value used for different datasets along with the total number of attributes for each dataset is listed below. These K-values are optimum as it enables the database to accommodate attributes that deliver best performance.

Dataset	**Number of attributes**	**K-value**
Breast cancer	9	5
Cervical cancer	29	15
Diabetes	8	5
Heart disease	13	10
Hepatitis	19	12
Kidney disease	24	5
Liver disorders	10	7
Lung cancer	16	10
Parkinson’s disease	22	5
Skin disease	35	15
Thyroid disease	5	5
Vertebral column disorders	6	6

Table 3 – Number of attributes vs K- value for all datasets

4.5.3 ALGORITHM FOR DIAGNOSIS

Among all the models developed for the dataset (6 base models and 2 hybrid models), the model that has highest accuracy is selected for diagnosing the symptoms entered by user in the prompt. The best algorithm's accuracy, prediction, and prediction probability are written to the output file.

4.6 RESULTS DISPLAY

From the output file, the diagnosis results are read and displayed in corresponding test boxes as shown in screenshot below. From here, the user can enter the symptoms of any disease by clicking the "Home" button.

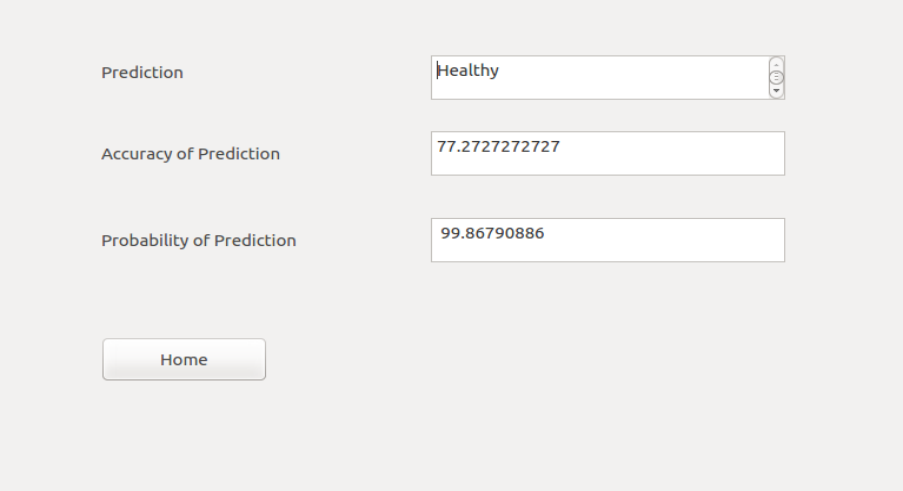

Figure 7 – Screenshot of results screen

5. PERFORMANCE EVALUATION

Every machine learning model is trained with a portion of the dataset (i.e. training dataset). The accuracy of the model depends upon how well the machine learning model is able to learn the training datasets. However even if the model is trained well, there are chances of overfitting. In the case of overfitting, the model memorizes the training data and does not have the ability to generalize well to new unseen test data. Under such a scenario the model fails in maintaining high levels of accuracy in the testing data.

A machine learning model built is said to be performing well only if it generalizes well. Generalization is a capability of the machine learning model to learn the patterns from the training dataset well and apply the learnt pattern effectively to the new unseen testing dataset. The table below is the comparison accuracy for different models involved in different datasets used. The following table shows accuracy of all algorithms across all datasets used.

Dataset	**SVM**	**KNN**	**ANN**	**RF**	**DT**	**NB**	**NB with DT**	**NB with RF**
Breast cancer	95.05	97.03	96.53	96.03	93.56	96.04	95.05	95.54
Cervical cancer	94.44	88.88	94.44	94.44	88.88	88.88	88.88	88.88
Diabetes	62.34	70.12	69.48	74.67	75.32	77.27	74.46	74.46
Heart disease	53.93	51.68	53.93	56.18	47.19	51.68	52.81	52.81
Hepatitis	91.67	79.16	66.67	87.5	91.67	54.17	100	100
Kidney disease	68.75	79.17	68.75	100	97.91	100	100	100
Liver disorders	72.92	71.98	65.74	82.54	80.91	88.54	88.54	88.54
Lung cancer	85.81	80.14	85.82	83.69	73.43	26.95	82.26	82.26
Parkinson’s	81.36	83.05	20.33	84.75	83.05	61.02	69.49	69.49
Skin disease	93.15	91.67	99.07	97.22	93.52	87.04	87.04	95.37
Thyroid	70.77	96.92	70.76	96.92	96.92	100	100	100
Vertebral disorders	13.46	30.77	39.10	23.07	21.80	46.15	46.15	46.15

Table 4 – Accuracy of datasets across different algorithms

From Table 4 we observe that across all datasets, random forest algorithm achieves highest accuracy in 4 datasets (cervical cancer, heart disease, kidney disease and Parkinson's disease). Naïve Bayes algorithm achieves highest accuracy in 5 datasets (hepatitis, liver disease, kidney disease, thyroid disorders and vertebral column disorders).

In addition, we observe that the accuracy of Naïve Bayes can be improved with hybrid Naïve Bayes algorithm involving random forest or decision tree. From Table 3, I infer that the number of attributes play a major role in performance of Naïve Bayes algorithm . For breast cancer, diabetes, liver disorders, thyroid and vertebral column datasets, the number of attributes is very low and hence the attribute reduction done using hybrid algorithm only reduces the accuracy or maintains it. On the other hand for hepatitis, heart disease, lung cancer and skin disease datasets, attribute reduction has helped in obtaining higher accuracy than the basic Naïve Bayes algorithm.

6. CONCLUSION

According to literature, there has been several attempts to deploy machine learning for diagnosing diseases. In my project, besides implementing algorithms from literature (basic algorithms, and hybrid algorithm involving Naïve Bayes and decision tree), I have successfully implemented a new hybrid algorithm which utilizes Naïve Bayes and random forest for classifying diseases. This hybrid algorithm utilizes the strength of ensemble learning and achieves accuracy levels of hybrid Naïve Bayes with decision tree algorithm. In some cases the hybrid Naïve Bayes with random forest outperforms Naïve Bayes and hybrid Naïve Bayes with decision tree. From Table 4 we see that for skin diseases, the hybrid Naïve Bayes with random forest has performed better than Naïve Bayes and hybrid Naïve Bayes with decision tree.

On the other hand, for the automated diagnosis to happen on a large and reliable scale, we would need large amounts of medical data for each patient across geographies for all diseases. This feat could involve huge amounts of data collection, data entry, and data storage efforts.

7. FUTURE WORK

In this project, two hybrid algorithms (Naïve Bayes with decision tree and Naïve Bayes with random forest) were constructed. Apart from these two hybrid algorithms, it can be possible to construct several other hybrid algorithms to overcome the limitations of a basic algorithm. This path could lead to potential new machine learning algorithms.

In addition, machine learning and data mining being an evolving field can generate new algorithms which can help in developing better models for mining. When there are better models, the accuracy of the diagnosis would be much higher thereby making the tool more reliable.

In this project, almost all of the datasets are from University of California, Irvine's machine learning repository. Although common diseases such as hepatitis, heart diseases, diabetes, etc. were available to study for automated diagnosis in this project, there are still thousands of other diseases yet to be automated for diagnosis this way.

REFERENCES

1. Shouman, Mia, et al. "Using data mining techniques in heart disease diagnosis and treatment," *2012 Japan-Egypt Conference on Electronics, Communications and Computers*, Alexandria, 2012, pp. 173-177. doi: 10.1109/JEC ECC.2012.618697
2. Miao1, Kathleen H., et al. "Diagnosing Coronary Heart Disease Using Ensemble Machine Learning", *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 10, 2016
3. Ahmed, Jamil and Soomrani, Abdul Rehman M., "TDTD: Thyroid disease type diagnostics," *2016 International Conference on Intelligent Systems Engineering*, Islamabad, 2016, pp.44-50. doi: 10.1109/INTELSE.2016.7475160
4. Vijayarani, Dr. S. and Dhayanand, Mr. S.. "Liver Disease Prediction using SVM and Naïve Bayes Algorithms." *International Journal of Science, Engineering and Technology Research*, vol. 4, Issue 4, Apr 2015
5. Gulia, Anju, et al. "Liver Patient Classification Using Intelligent Techniques." *International Journal of Computer Science and Information Technologies*, vol. 5, 2014, 5110-5115.
6. Iyer, Aiswarya, et al. "Diagnosis of Diabetes Using Classification Mining Techniques" *International Journal of Data Mining & Knowledge Management Process*, vol.5, no.1, Jan 2015, pp. 1-14

7. Sen, Sanjay Kumar and Dash, Sujata, "Application of Meta Learning Algorithms for the Prediction of Diabetes Disease", *International Journal of Advance Research in Computer Science and Management Studies*, vol.2, Issue 12 Dec 2014.
8. Farid, Dewan Md., et al, "Hybrid decision tree and Naïve Bayes classifiers for multi-class classification tasks" *Expert Systems with Applications*, vol 41, Issue 4, Part 2, Mar, 2014 pp. 1937-1946
9. Introduction to R, Online material available at: https://www.r-project.org/about.html - last accessed on 14th Aug 2018
10. Wikipedia's page on R (programming language), Online material available at: https://en.wikipedia.org/wiki/R (programming_language) - last accessed on 14th Aug 2018
11. Introduction and Overview of RStudio, Online material available at: https://www.rstudio.com/products/RStudio/ - last accessed on 14th Aug 2018
12. User Interface Design with Python, Online material available at: https://wiki.python.org/moin/GuiProgramming - last accessed on 14th Aug 2018
13. Wikipedia's page on Introduction to Python, Online material available at: https://en.wikipedia.org/wiki/Python_(programming_language)#Uses - last accessed on 14th Aug 2018
14. Introduction about PyQt, Online material available at: https://en.wikipedia.org/wiki/PyQt - last accessed on 14th Aug 2018

15. University of California Irvine, machine learning repository dataset available at: https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data - last accessed on 14th Aug 2018
16. University of California Irvine, machine learning repository dataset available at: https://archive.ics.uci.edu/ml/machine-learning-databases/00383/- last accessed on 14th Aug 2018
17. Kaggle dataset available at: https://www.kaggle.com/uciml/pima-indians-diabetes-database- last accessed on 14th Aug 2018
18. University of California Irvine, machine learning repository dataset available at: http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/heart/heart.dat- last accessed on 14th Aug 2018
19. University of California Irvine, machine learning repository dataset available at: https://archive.ics.uci.edu/ml/machine-learning-databases/hepatitis/hepatitis.data- last accessed on 14th Aug 2018
20. University of California Irvine, machine learning repository dataset available at: https://archive.ics.uci.edu/ml/machine-learning-databases/00336/- last accessed on 14th Aug 2018
21. Kaggle dataset available at: https://www.kaggle.com/uciml/indian-liver-patient-records- last accessed on 14th Aug 2018
22. University of California Irvine, machine learning repository dataset available at: https://archive.ics.uci.edu/ml/machine-learning-databases/00277/ - last accessed on 14th Aug 2018

23. University of California Irvine, machine learning repository dataset available at: https://archive.ics.uci.edu/ml/machine-learning-databases/parkinsons/parkinsons.data- last accessed on 14th Aug 2018
24. University of California Irvine, machine learning repository dataset available at: https://archive.ics.uci.edu/ml/machine-learning-databases/dermatology/dermatology.data - last accessed on 14th Aug 2018
25. University of California Irvine, machine learning repository dataset available at: https://archive.ics.uci.edu/ml/machine-learning-databases/thyroid-disease/new-thyroid.data - last accessed on 14th Aug 2018
26. University of California Irvine, machine learning repository dataset available at: http://archive.ics.uci.edu/ml/machine-learning-databases/00212/- last accessed on 14th Aug 2018