SMT. PARVATIBAI CHOWGULE OF ARTS AND SCIENCE

GOA-403601, GOA(INDIA)

Computer Science and Engineering

Clustered Machine Learning for Predicting Diabetes

Submitted By:
Alexander Roque Rodrigues
SU170331
TYBSC

Submitted To:
Mrs. Ashweta Fondekar
Asst. Professor
Dept. of CSE

June-February
2019-2020
Graduation Dissertation

1 Acknowledgements

Contents

Abstract

Your abstract.

2 Introduction

In recent days, there has been a sharp increase in the cases of diabetes mellitus. Diabetes mellitus is on the rise amongst many people and the rate of contracting this lifestyle disease could be reduced significantly if proper measures and precautions were to be instilled amongst people the number of people can be reduced.

Machine learning is a growing field in computer science. With the development and introduction of many algorithms the prediction and accuracy of the predictions itself has improved substantially. Machine learning and healthcare systems are also becoming increasingly popular in the healthcare sector.

The project encompasses the qualities of Remote Patient Monitoring (RPM) and Clinical Decision Support (CDS). RPM provides medical facilities that have the ability to transmit patient data to healthcare professionals who might very well be halfway around the world. RPM can monitor blood glucose levels and blood pressure. It is particularly helpful for patients with chronic conditions such as type 2 diabetes, hypertension, or cardiac disease. Data collected and transmitted via PRM can be used by a healthcare professional or a healthcare team to detect medical events such as stroke or heart attack that require immediate and aggressive medical intervention. Data collected may be used as part of a research project or health study. RPM is a life-saving system for patients in remote areas who cannot access face-to-face health care. CDS analyzes data from clinical and administrative systems. The aim is to assist healthcare providers in making informed clinical decisions. Data available can provide information to medical professions who are preparing diagnoses or predicting medical conditions like drug interactions and reactions. CDS tools filter information to assist healthcare professionals in caring for individual clients.

The objective of this project is to create a system that is able to use the machine learning algorithms and predict the outcome of the parameters entered into the algorithm and help the patient draw a conclusion whether or not he/she has the same traits exibhited by simillar patients that have diabetes. Also the system should have a UI that is capable of displaying the data of the patients to the doctor and to the patients themselves for further interpretation.

3 Software Requirements

The main function of this project is to enable the doctors to advise their patients with the help of the prediction software. The system should be accessible to the patient as weel as the doctor, therefore it should have a web interface for the two parties to interact with.

3.1 Functionalities for Doctors

As an owner of a doctors account a doctor should be able to:

- Add new patients.
- Add new observations for the machine learning algorithm to predict.
- Analyse the patients previous records.
- Leave notes for patients to act on.

3.2 Functionalities for Patients

As the patient, one should be able to:

- Should be able to see the predicted risk of developing diabetes.
- Should be able to view historic data.
- Should be able to view notes or suggestions left by doctor.

3.3 Functionalities for Master Nodes

As the patient, one should be able to:

- Should be able to see the predicted risk of developing diabetes.
- Should be able to view historic data.
- Should be able to view notes or suggestions left by doctor.

3.4 Functionalities for Slave Nodes

As the patient, one should be able to:

- Should be able to see the predicted risk of developing diabetes.
- Should be able to view historic data.
- Should be able to view notes or suggestions left by doctor.

4 Hardware Requirements

4.1 Raspberry Pi

According to raspberrypi.org, the Raspberry Pi 3 Model B is the earliest model of the third-generation Raspberry Pi. It replaced the Raspberry Pi 2 Model B in February 2016. Some of the key features of this single board computer or SBC are:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU.
- 1GB RAM.
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board.
- 100 Base Ethernet.
- 40-pin extended GPIO.
- 4 USB 2 ports.
- Full size HDMI.
- Micro SD port for loading your operating system and storing data.

In this project I will be using 3 Raspberry Pi's to implement a cluster and deploy the machine learning algorithm on.

4.2 Switch

A network switch was used in the project to make up for the lack of ethernet ports available on the router. The dumb network switch was able to connect upto 3 Raspberry Pi and one cable back to the network router itself to connect the switch to the main network.

4.3 Router

4.4 Master Node

4.5 Miscalleneous Components

4.5.1 WiFi Enabled Router

5 Technology Stack

5.1 Python

5.1.1 Pandas

5.1.2 Sklearn

5.1.3 Numpy

5.1.4 Itertools

5.2 MYSQL

5.3 Apache Web Server

5.4 PHP

5.5 AJAX

5.6 GitHub

6 Finding the Right Algorithm

For our predictions we need to use algorithms that give us the maximum accuracy and once we find that algorithm we will further need to tune the selected algorithm from the algorithms we selected to optimise the performance of the predictions generated by the machine learning algorithm. For this purpose we turn to the EDA or exploratory data analysis stage of any machine learning project.

7 Writing Code

```python
import itertools
import mysql.connector
from colorama import init
from colorama import Fore, Back, Style
from mysql.connector import Error
import masterenvironment as en
from os import system
from os import chdir
init()

def divide_chunks(l, n):
    for i in range(0, len(l), n):
        yield l[i:i + n]


def readNodes():
    global lineList
    with open(en.fileName) as f:
        lineList = f.readlines()
    lineList = [line.rstrip(’\n’) for line in open(en.fileName)]


def db_connect():
    curr_node = 0
    curr_chunk = 0
    print("Connecting to DB ", end="")
    try:
        mydb = mysql.connector.connect(
            host=en.host, user=en.user, passwd=en.passwd, database=en.database)
        mycursor = mydb.cursor()
        print(Fore.GREEN+"[SUCCESS]"+Style.RESET_ALL)
        pending = []
        query = "select * from {} where PredictedOutcome {}".format("diagnosis", "IS NULL")
        mycursor.execute(query)
        myresult = mycursor.fetchall()
        for x in myresult:
            pending.append(x[0])

        x = list(divide_chunks(pending, en.chunkSize))
        executeStatus = 0
        nodeCount = len(lineList)
        nodeCount = nodeCount - 1
        chunkCount = len(x)
        while(executeStatus != 1):
            temp = x[curr_chunk]
            temp = str(temp)
            temp = temp[1:-1]
            temp = temp.replace(" ", "")
            query = "ssh pi@{} python3 hive-ml/slave.py {}".format(lineList[curr_node], temp)
            query = str(query)
            system(query)
            curr_chunk = curr_chunk + 1
            if(curr_node == nodeCount):
                curr_node = 0
            else:
                curr_node = curr_node+1
            if(curr_chunk == chunkCount):
                executeStatus = 1

    except Error as e:
        print(Fore.RED+"[FAILED]"+Style.RESET_ALL)
        print(e)
        exit(0)

if __name__ == "__main__":
    readNodes()
    db_connect()
```

8 Conclusion