CAL STATE FULLERTON

Department of Computer Science

This project has been satisfactorily demonstrated and is of suitable form.

This project report is acceptable in partial completion of the requirements for the Master of Science degree in Software Engineering.

Diabetes Foreteller

Project Title (type)

Deepak Prasanna Mayanattanmy Yagneswaran

Student Name (type)

Christopher Ryu, Ph.D.

Advisor's Name (type)

Advisor's signature Date

Bin Cong, Ph.D.

Reviewer's Name (type)

Reviewer's signature Date

Abstract

Diabetes[13] is a disease in which your blood glucose, or blood sugar, levels are too high. There are several different types of diabetes, some of which are more prevalent than others. The most common form of diabetes in the general population is type 2 diabetes, which often develops from pre-diabetes. The majority have type 2 diabetes, but an important minority have type 1 diabetes (~5%). Contrary to popular belief, type 1 diabetes is not a childhood disease. Early detection is the key to prevent diabetes. Hence, apart from other types of diabetes needs lots of attention. Many research teams from biomedical and bio informatics field have started to employ machine learning (ML) techniques to detect the diabetes at the early level.

In my project, I have developed a mobile application to calculate the probability of diabetes for a person based on the key symptoms that influence diabetes[14]. I have performed a research on all the symptoms that influence diabetes and built a machine learning model which predicts the probability of diabetes for a person with 88% accuracy. In addition to the probability of diabetes, my application also provides suggestive measures[15] for preventing diabetes. This application mainly aims to spread the awareness about diabetes and will be very useful to many people all around the world to know their chances of having diabetes, thus providing a sub-ordinate medical assistance from preventing diabetes.

Table of contents

1 INTRODUCTION

1.1 Background

Diabetes[13] and diabetes-related illnesses place an enormous burden on the health care systems of countries throughout the world. Over time diabetes can damage the heart, blood vessels, eyes, kidneys and nerves. The disease is a considerable cause of premature mortality. Due to increases in life expectancy, urbanization, rates of overweight and obesity, and the prevalence of diabetes. The disease burden for diabetes is likely to worsen, particularly in low- and middle-income countries.

There are primarily two types of diabetes[14]. Type 1 diabetes is an autoimmune disease in which the pancreas can no longer produce insulin. As a result, the body cannot control blood sugar levels. However, Type 2 diabetes (previously called adult onset) is a metabolic disorder in which the body gradually becomes insensitive to the action of insulin with decreased beta cell mass and progressive beta cell failure so that blood sugar control is also compromised. In developed countries, most people with diabetes are aged over 60 years, while in developing countries the disease mainly affects people of working age (40 to 60 years)[14].

1.2 Motivation

I find that, diabetes produces detrimental health effects to the human beings, as there are no precise symptoms for the disease and most often diabetes will be diagnosed at the late stages, producing no remedy to the medication. In this project, I have developed an android application named ‘Diabetes Foreteller’ which helps the user to assess his/her probability of having diabetes by analyzing the various symptoms of diabetes and the lifestyle of the user. Suggesting the probability of having diabetes to the user, creates an awareness and cautions him about the detrimental health effects caused by the diabetes. In addition to this, my application also recommends the preventive measures[15] for avoiding the diabetes in one’s lifetime.

1.3 Goals and Objectives

The primary goal of this project is to build an android application that estimates the probability of diabetes for a person in his lifetime. More specifically, the project has the following objectives,

- Research and collect all the symptoms related to diabetes
- Build a machine learning model that generates the probability of diabetes with respect to the user data.
- Convert the machine learning model as RESTful web service and deploy the model in cloud environment
- Build an android application that helps the user to access their probability of diabetes

1.4 Related Work

There are many medical assistances android application available in the market, most particularly WebMD[15], which provides an insightful sub-ordinate medical assistance to the people. WebMD primarily helps a user to check the symptoms of a disease, to find a doctor nearby and to identify the medicine for a specific disease etc. In addition to this, 'Breast Cancer Risk Assessment' android application is an interactive tool to help estimate a women's risk of developing breast cancer. Both these application mainly focuses on specific disease such as general health assessment and breast cancer risk assessment[16] but my application 'Diabetes Foreteller' mainly focuses on diabetes.

1.5 Resources and environment

Resource	Version	Purpose	Cost
ASUS Q301LA (Intel i5 core CPU @ 2.3 GHz, 8 GB of RAM)	1.0	Development/Test environment	Owned
One plus 5T android mobile phone	1.0	Development/Test equipment	Owned
Android Studio	3.2	Android API	Free
Jupyter Notebook	5.7.8	Tool for building machine learning model	Free
TensorFlow	1.9	Framework for building machine learning model	Free
Flask	1.0.2	Converting machine learning model into API	Free
Postman	7.0	Testing REST API	Free
Amazon S3	N/A	Data Storage	Free
Heroku	N/A	REST API deployment	Free

2 DATASET

To estimate the probability of the diabetes for a person, I must collect lots of real-time data for training my machine learning model. Due to time constraints, I was not able to collect the required data to train my machine learning model. To solve this problem, I have made use of PIMA Indians diabetes dataset[5], provided by PIMA group of hospitals.

PIMA is a top-class medical institute and multi-specialty hospital which has released a diabetes dataset with the key factors that influence diabetes. The dataset provides accurate information of all the patients and provides reliability for the students/professionals to use this data in real time projects. In addition to this, PIMA diabetes dataset[6] includes all the proven features to classify diabetes for a person in his lifetime.

2.1 ABOUT THE DATASET

The dataset[6] includes data from 768 women with 8 characteristics, in particular:

- **Pregnancy**

  Pregnancy is the major factor that influence diabetes with women. As per the statistical data, 72% of patients for diabetes are women. In general, High blood sugar levels are high during the pregnancy can cause birth defects. They also can increase the risks of miscarriage and diabetes-related complications. But many women don't know they're pregnant until the baby has been growing for 2 to 4 weeks. That's why most of the women should have good control of blood sugar before women start trying to conceive.

- **Plasma glucose concentration**

  The fasting plasma glucose test (FPG) is the preferred method for diagnosing diabetes, because it is easy to do, convenient, and less expensive than other tests, according to the American Diabetes Association.

  Normal fasting blood glucose -- or blood sugar -- is between 70 and 100 milligrams per deciliter or mg/dL for people who do not have diabetes. The standard diagnosis of diabetes is made when two separate blood tests show that your fasting blood glucose level is greater than or equal to 126 mg/dL. Some people have a normal fasting blood sugar reading, but their blood sugar rapidly rises as they eat. These people may have impaired glucose tolerance. If their blood sugar levels are high enough, they may be diagnosed with diabetes

- **Diastolic blood pressure (mm Hg)**

  High blood pressure (hypertension) can lead to many complications of diabetes, including diabetic eye disease and kidney disease, or make them worse. Most people with diabetes will eventually have high blood pressure, along with other heart and circulation problems.

  Diabetes damages arteries and makes them targets for hardening, called atherosclerosis. That can cause high blood pressure, which if not treated, can lead to trouble including blood vessel damage, heart attack, and kidney failure. If the blood pressure is greater than 140 (mm Hg) than we could classify that, the patient has diabetes. The blood pressure for a normal person is 120 (mm Hg). All the records are mentioned for adults.

- **Triceps skin fold thickness (mm)**

  Skin thickness is primarily determined by collagen content and is increased in insulin-dependent diabetes mellitus (IDDM). The IDDM measured skin thickness correlates

with long-term glycemic control and the presence of certain diabetic complications. Based on these proofs, we could infer that skin thickness is one of the primary diabetes influencers.

- **2-Hour serum insulin (mu U/ml)**

  Insulin is a hormone that your pancreas makes to allow cells to use glucose. When our body isn't making or using insulin correctly, we can take man-made insulin to help control our blood sugar.

- **Body mass index (weight in kg/(height in m)^2)**

  Obesity is one of the major symptoms for diabetes. Obesity is caused due to increase body mass or weight gain. Through weight gain, our body generates or stores lots of unsolvable fatty acids, which leads to a lot of harmful effects. With increased obesity, our body shows insulin resistance, which in-turn influences diabetes. To measure obesity, we use body mass index, which is formulated as follows:

  Body Mass Index (BMI) = (Weight in lbs.) / (Height in inches)

- **Diabetes pedigree function**

  Diabetes pedigree function is the measurement of genetical based diabetes factor. For example, if a person's father is suffering from diabetes, then we have 15% chance that same person could expect diabetes.

- **Age (years)**

  We can expect diabetes to develop at any age, even during childhood. However, type 2 diabetes occurs most often in middle-aged and older people. As per the statistical records, type 2 diabetes could develop after 45 years for majority of the persons who have a family history of diabetes or are overweight or obese.

2.2 STUDY RELATED TO DATASET

The diabetes dataset has been analyzed using statistical methods[10] and performed exploratory data analysis.

Heatmap

Through heatmap we could infer the following,

- The greater the age or the BMI of a patient is, the greater probabilities are the patient can develop diabetes.
- Age and Pregnancy seems to have high correlation with each other.

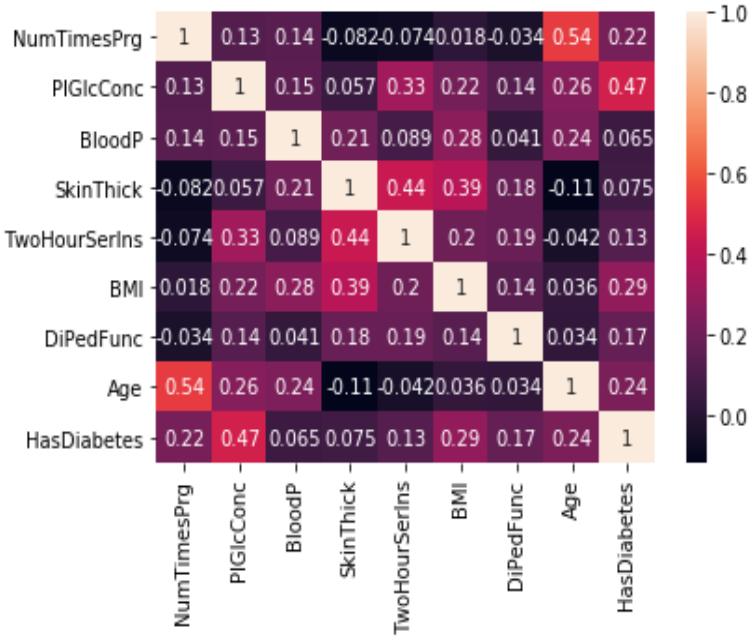


Figure 1Heatmap for PIMA diabetes

Histogram[10]:

Through histogram we could observe that the distribution of data is very uneven, need to standardize the dataset for faster training of machine learning model.

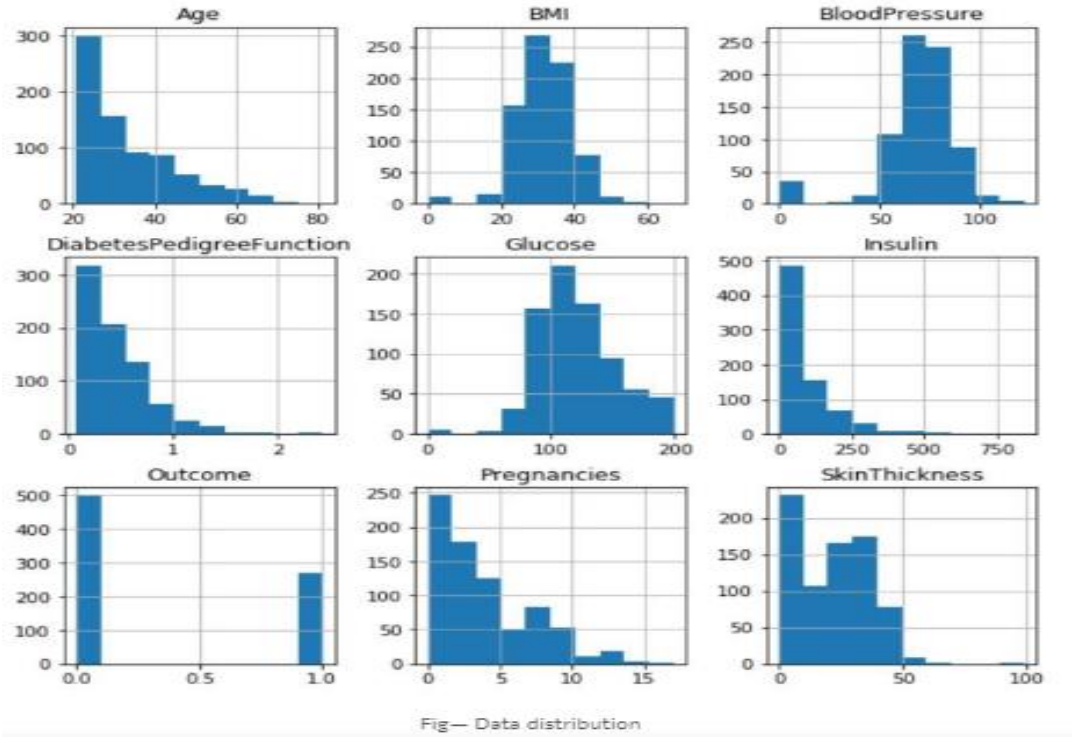


Figure 2 Histogram of diabetes dataset

Other inferences from the dataset:

- Average Insulin value102.5 (mu U/ml) for a healthy person
- The Skin Thickness for most diabetic persons is 32.0 (mm)
- Average Insulin value169.5 (mu U/ml) for a diabetic person
- Blood pressure for diabetic persons always ranges above 74.5 (mm Hg)
- Average Glucose value107.9 for a healthy person and 140.0 for a diabetic person

3 SOFTWARE DEVELOPMENT MODEL

3.1 Introduction

For this project I chose to use the Waterfall Software Development Model. The Waterfall software development model comprises of five phases - requirements, design, implementation, verification, and maintenance. These phases are completed sequentially, you can go to next phase only when previous phase is complete. As a result, this model often brings significant problem with duration of software development, but it is appropriate for small- sized and individual project.

3.2 Waterfall Software Development Model

The Waterfall software development model comprises of five phases - requirements, design, implementation, verification, and maintenance. These phases are completed sequentially, you can go to next phase only when previous phase is complete. As a result, this model often brings significant problem with duration of software development, but it is appropriate for small- sized and individual project.

3.3 Requirements

It assumes that all the requirements can be gathered during the requirements phase and once all requirements are gathered, it goes down to next stage. The detailed information on requirements will be described in Section 3.

3.4 Design

The design phase consists of logical and physical design. A logical describes conceptual components of the system and specify behavior. The physical design comes after the logical design. Hardware and software technologies are selected in this phase.

3.5 Implementation

Based on the requirements and designed determined in previous phases actual coding is implemented in this stage. The coding files will be enclosed with this document.

3.6 Verification

It is also referred as testing in other development models. All software requirements and customer expectations are verified in this phase. Details of testing results will be explained in Section 5.

3.7 Maintenance

In this phase, the software is released to customer and customers give feedback for the delivered product. If any corrections or updates are received, updated are made in this phase.

4 SOFTWARE REQUIREMENT SPECIFICATION

4.1 Introduction

4.1.1 Purpose

This Software Requirement Specification (SRS) document will provide a comprehensive description of the Diabetes Foreteller. It will provide a description of the features, operating environment, as well as design and implementation constraints to drive the implementation of a functioning Android application and restful web service.

4.1.2 Scope

The scope of the Diabetes Foreteller is to predict the probability of diabetes disease for a user in his lifetime and suggest preventive measure to avoid diabetes for the users.

4.2 Overall Description

4.2.1 Product Perspective

The product consists of a REST API that holds the machine learning model and an android API for the end users to access the REST API that outputs the probability of diabetes for a person in his/her lifetime.

4.2.2 Product Features

FR1 Predict the probability of diabetes disease for a user with a incredible accuracy

FR2 Generate suggestive measures based on probability of diabetes for the user

FR3 Provide a detailed description about the diabetes disease to the user

FR4 The application should be available to user at all time

FR5 The application should serve multiple users

FR6 Supports multiple users

FR7 Easy user interface

4.2.3 User Classes and Characteristics

4.2.3.1 General Users

Diabetes foreteller application is mainly developed to serve general class users aging from 20 to 70 years old. The application generates the probability of diabetes for the all the users based on the key symptoms of the diabetes.

4.2.3.2 Administrator

The administrator maintains the restful web service deployed on cloud environment. The admin is also responsible for improving the accuracy of the machine learning model.

4.2.4 Operating Environment

OE-1: The diabetes foreteller restful web service is platform independent and can be used in all type of operating system. The restful web service is deployed on Heroku which is a Platform-as-a-Service (PaaS) supporting several programming languages that are used as a web application or a mobile application

OE-2: The Diabetes Foreteller application runs on Android Operating system 9.0.1 (Android Pie) which serves as an API for the end users.

4.2.5 Design and Implementation Constraints

IC-1: Due to security concerns Android API version 9.0.1 and later do not allow third party users to modify the application level data and serves only as an android API to access the RESTful web service.

4.2.6 User Documentation

UD-1: The diabetes foreteller android application provides an in-application Help and a sample tutorial to access the application and its features, promoting a better user-friendly environment.

UD-2: In case of any discrepancies, the users can shoot a feedback message to the administrator by clicking on the 'Report Issue' button.

4.2.7 Assumptions and Dependencies

AS-1: The diabetes foreteller application assumes the user has a basic understanding about the diabetes disease.

DE-1: The diabetes foreteller application operates only with the help of internet connectivity.

AS-2: The diabetes foreteller application requests users to turn on WiFi connectivity or Mobile Data on their device to access the application.

AS-3: The diabetes foreteller application assumes the user is has basic English fluency.

4.3 Functional Requirements

4.3.1 FR-1 Predict the probability of diabetes for a user

Introduction	The application shall allow the user to predict his/her probability of suffering diabetes in his/her lifetime.
Inputs	The user answers his diabetes related questions such as age, pregnancies, BMI, Insulin, etc.
Processing	The application communicates with the restful web service deployed on Heroku and fetches the probability of diabetes based on user's input

Output	The application shall display the probability of diabetes based on user's input on the screen.
Error Handling	The application outputs an error if the user enters an invalid input.

4.3.2 FR-2 Providing suggestive measures to prevent diabetes

Introduction	The application shall allow the user to display suggestive measures upon request
Inputs	The user knows his probability of suffering diabetes in his lifetime and requests to know suggestive measures to prevent diabetes
Processing	The application analyzes the user's probability of diabetes and displays preventive measures.
Output	The application displays the individual preventive measure as a report based on user's input
Error Handling	Not applicable

4.3.3 FR-3 Display detailed information about diabetes

Introduction	Upon user's interest, the application has an inbuilt feature and reading articles, Videos that provides detailed information about diabetes.
Inputs	The user knows his probability of suffering diabetes in his lifetime and requests to know suggestive measures to prevent diabetes
Processing	The application displays various famous articles and blog sites and diabetes medicines for user's in-depth

	understanding about diabetes
Output	The application displays famous articles related to diabetes for user's reading.
Error Handling	Not applicable

4.4 External Interface Requirements

4.4.1 User Interface Requirements:

US-1: The application design shall be responsive for all android device screen size.
US- 2: The background shall be white.
US- 3: The text color shall be black.
US- 4: The menu options shall be in top left side of header.
US- 5: User's inputs alongside with the probability shall be displayed on the diabetes foreteller screen
US- 6: The application shall highlight all form errors in red color.

4.4.2 Hardware Requirements:

HI- 1: The application shall be accessible on all android devices through internet connection.

4.4.3 Software Requirements:

SR-1: The user-end application runs on android devices
SR-2: The RESTful web service is deployed on Heroku Platform-as-a-Service (PaaS)
SR-3: Amazon S3 is used as cloud storage, where the dataset is stored.

4.5 Non-functional Requirements

4.5.1 Performance requirements

- PR- 1: The application shall be available 24/7.
- PR- 2: The application shall generate the probability of diabetes for the user in less than 3 seconds.
- PR- 3: The restful web service shall be online 24/7.

4.5.2 Security Requirements

- SR- 1: The application shall allow all types of users to access the application

- SR- 2: The application shall allow only the admin to manage the restful web service and to increase the efficiency of the machine learning model.

4.5.3 Quality Attributes

- *SQA- 1: Availability*

  The application shall be available 100% of time.
- *SQA- 2: Modularity*
  The application shall be modularized for future modification.
- *SQA- 3: Testability*
  Both RESTful web service and the android application shall be testable, and application shall define all errors and isolate them.
- *SQA- 4: Performance*
  The application shall provide fast response to the user's inputs.
- *SQA- 5: Usability*
  SQA-5: The application shall have a simple user interface.

4.6 Context Diagram

The context diagram shown in Figure 3 illustrates the entire application as a single process. It briefly represents how the application communicates with end other end points such as user.

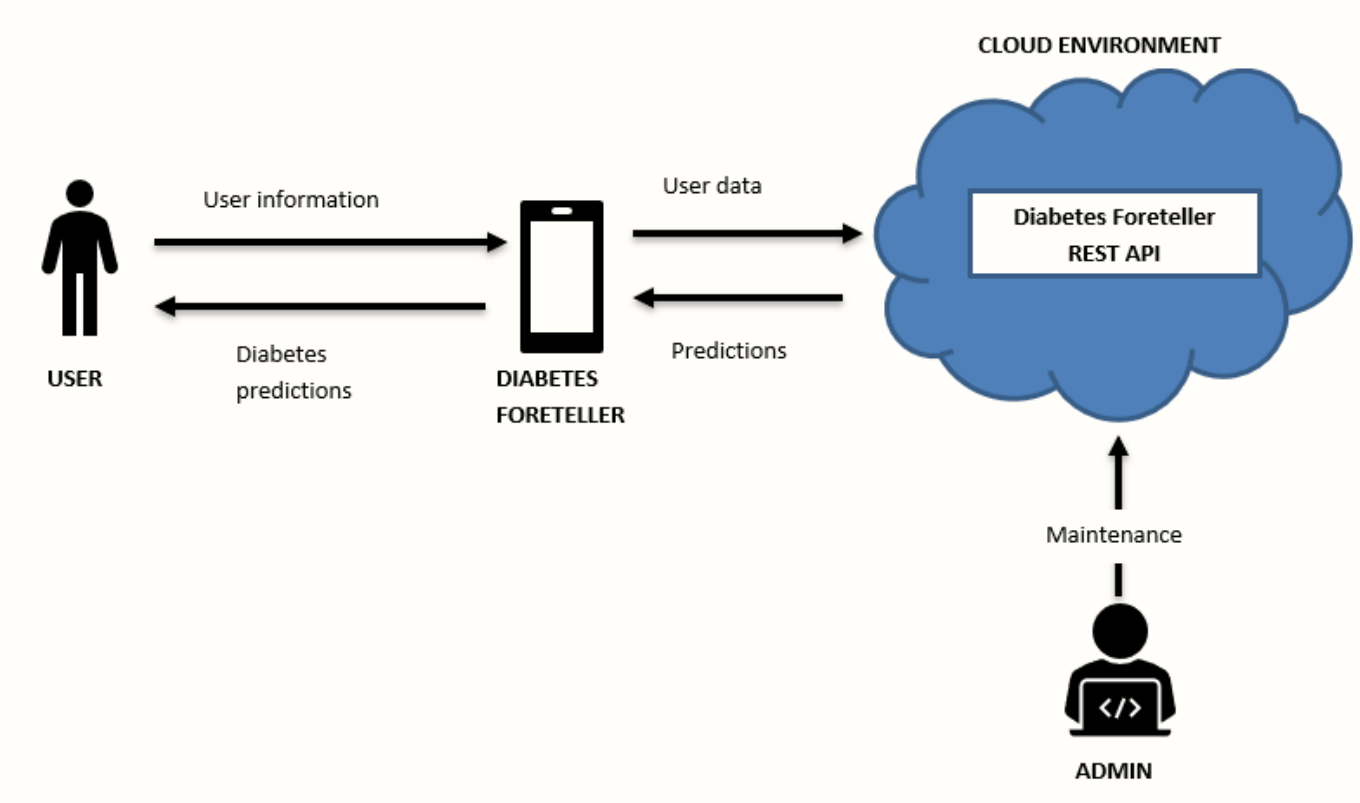


Figure 3 Context Diagram

4.7 Control Flow Diagram

Figure 4 is control flow diagram and it represents a flow of process in the entire application.

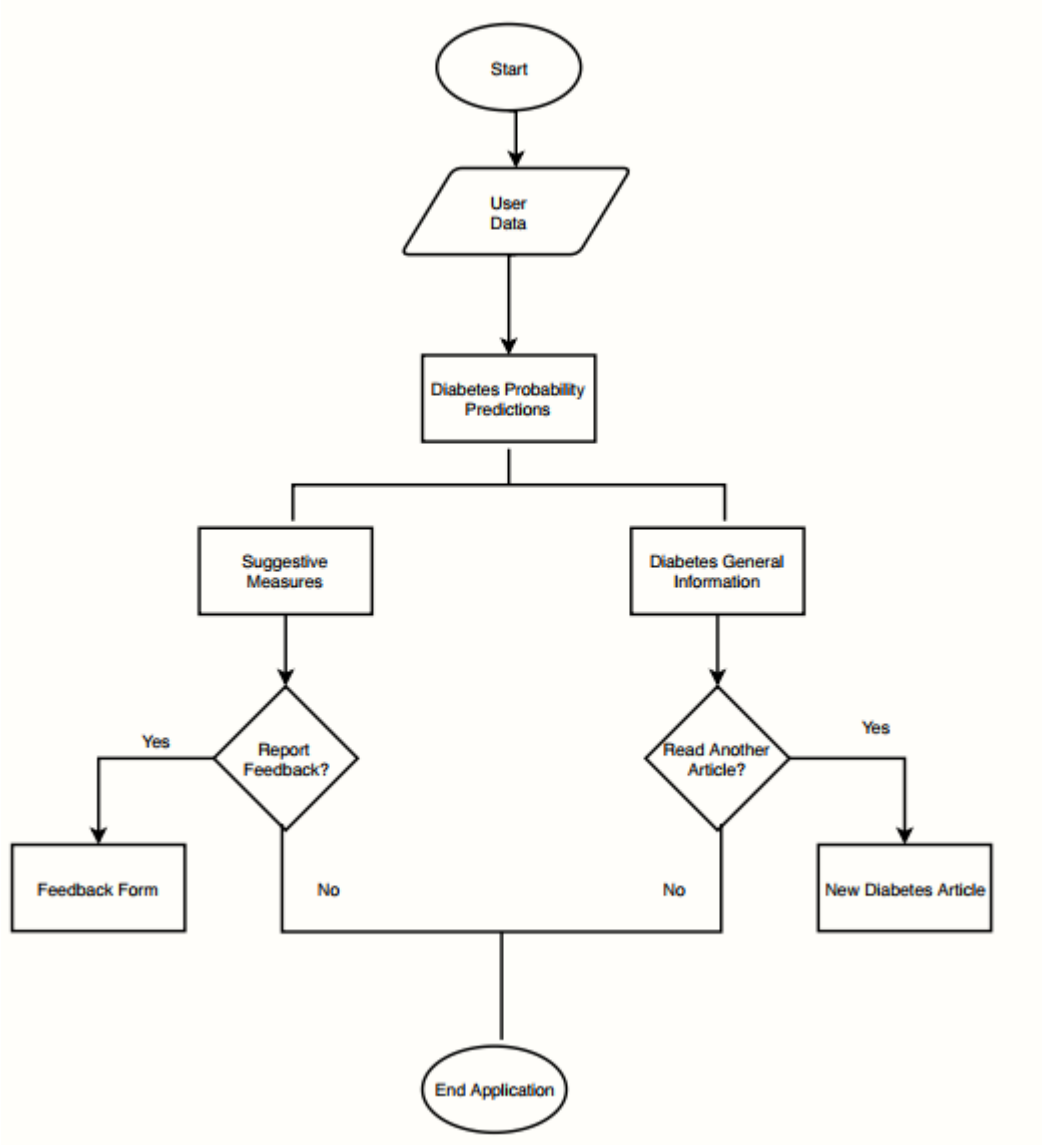


Figure 4 Control Flow Diagram

4.8 Use Case Diagram

Use case descriptions created based on the functional requirements are given in this section.

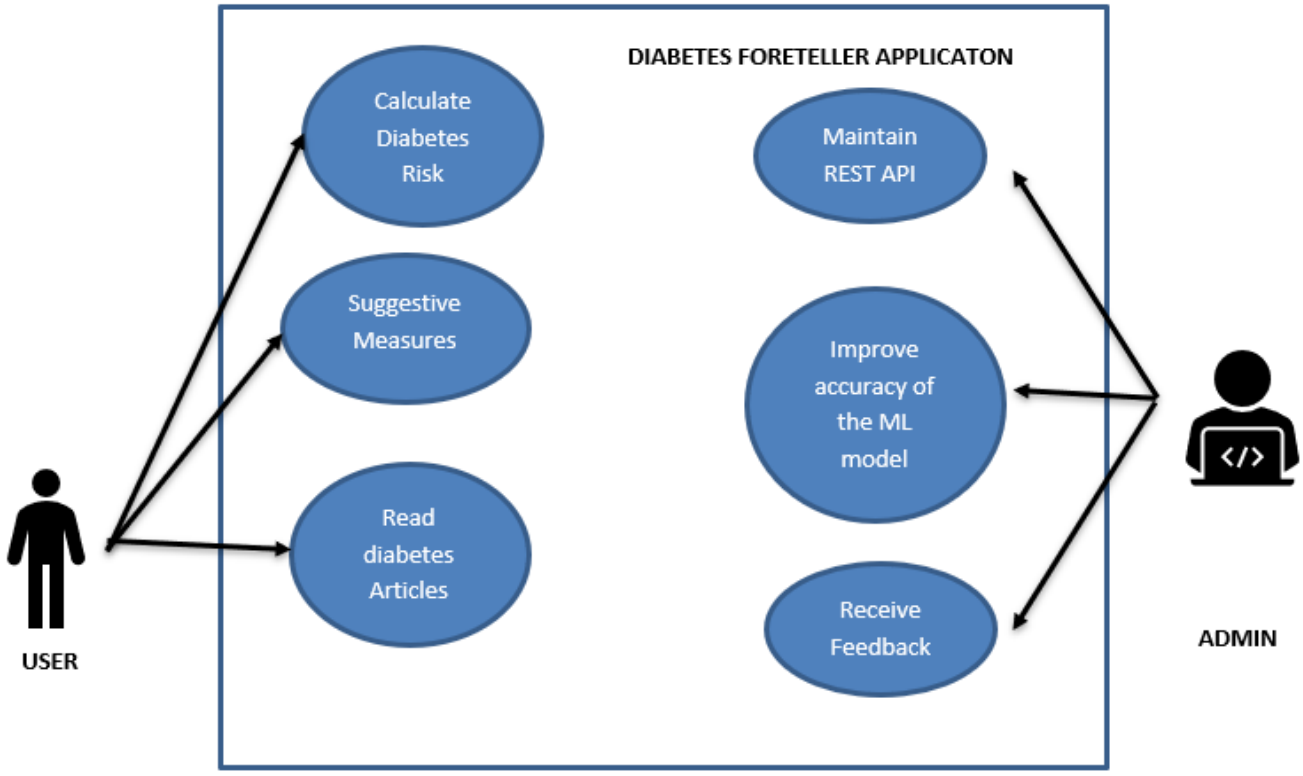


Figure 5 Use Case Diagram

4.8.1 UC-1: Assess the probability of diabetes

Use Case ID:	UC-1
Use Case Name:	Calculate Diabetes Risk
Actors:	User
Description:	This use case helps the user to access the probability of diabetes in his/her lifetime
Pre-conditions:	The user has installed diabetes foreteller application in his mobile device
Scenario:	The user enters his details – BMI, Insulin, Pregnancy, Age etc.
Post-conditions:	The user accesses his probability of suffering diabetes in his lifetime
Exceptions:	The application throws an error if the user does not enter valid data.
Includes:	The user must be aware of his insulin, glucose, skin thickness, blood pressure before calculating his risk.

4.8.2 UC-2: Assess suggestive measures

Use Case ID:	UC-2
Use Case Name:	Assess the suggestive measures
Actors:	User
Description:	This use case helps the user to assess the suggestive measures to prevent diabetes in his/her lifetime
Pre-conditions:	1. The user has installed diabetes foreteller application in his mobile device. 2. The user has assessed his probability of suffering diabetes in his lifetime
Scenario:	1. The user has assessed his probability of suffering diabetes in his lifetime 2. The user wishes to know the suggested precautionary measures that helps him to stay fit from diabetes
Post-conditions:	The user assesses his suggestive measures to prevent diabetes in his/her lifetime.
Exceptions:	N/A

4.8.3 UC-3: Read diabetes articles

Use Case ID:	UC-3
Use Case Name:	Read diabetes articles
Actors:	User
Description:	This use case helps the user to access to understand the recently trending medicines and treatments in diabetes.
Pre-conditions:	1. The user has installed diabetes foreteller application in his mobile device. 2. The user has assessed his probability of suffering diabetes in his lifetime
Scenario:	1. The user has assessed his probability of suffering diabetes in his lifetime 2. The user wishes to know more information about diabetes.
Post-conditions:	The application displays various articles related to diabetes for user's reading.
Alternative Flows:	N/A

5 DESIGN AND ARCHIECTURE

I have categorized the design and architecture for this project into two divisions,

1. Machine Learning Model
2. Android User Interface

5.1 Machine Learning Model

Rather than relying on popular machine learning frameworks such as Keras, TensorFlow, I wanted to challenge myself by building a neural network[2] completely on my own. Based on my experience with Deep Learning, I have followed a methodology I learnt from 'Deep Learning Specialization'[1] by Andrew Ng.

5.1.1 Neural Networks Definition

Neural networks[2] are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated.

Neural networks help us cluster and classify. We can think of them as a clustering and classification layer on top of the data us to store and manage. They help to group unlabeled data according to similarities among the example inputs, and they classify data when they have a labeled dataset to train on. Neural networks can also extract features that are fed to other algorithms for clustering and classification; so we can think of deep neural networks as components of larger machine-learning applications involving algorithms for reinforcement learning, classification and regression.

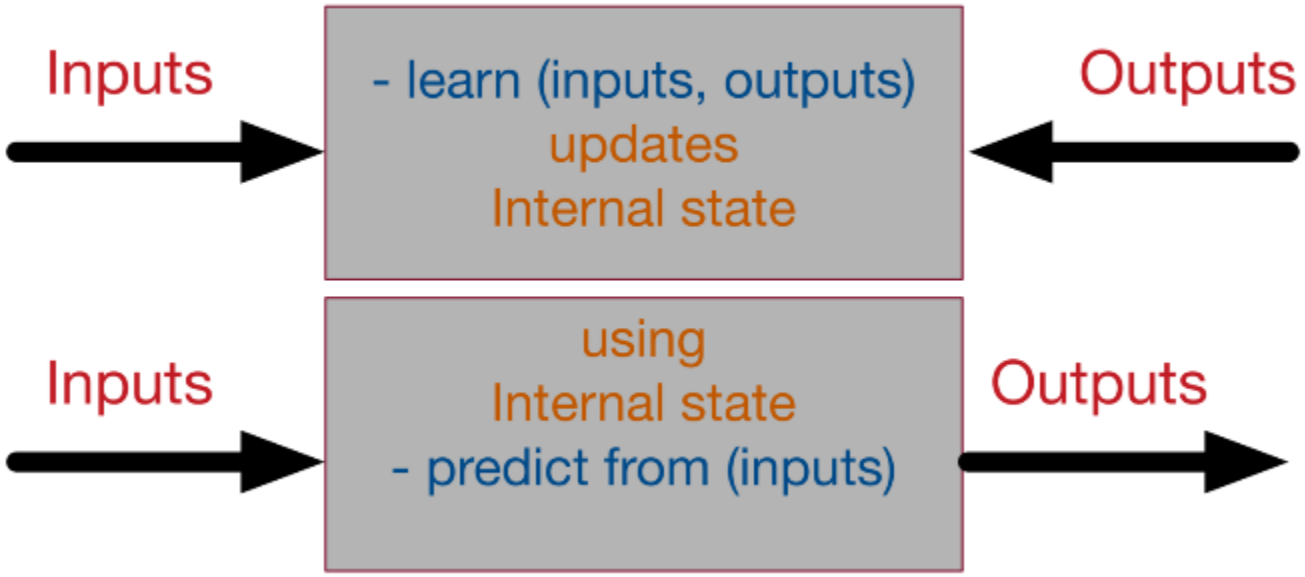


Figure 6 Neural Networks as a black box

5.1.2 Methodology for implementing neural networks[1]

To construct neural networks[2], I have followed a 3-step methodology that helps me to implement neural networks without any confusions,

1. Initialize parameters / Define hyperparameters

We'll first initialize the weight matrices and the bias vectors. It's important to note that we shouldn't initialize all the parameters to zero because doing so will lead the gradients to be equal and on each iteration the output would be the same and the learning algorithm won't learn anything. Initialize the following three parameters,

1. The size of the input layer
2. The size of the hidden layer
3. The size of the output layer

2. Loop for number of iterations:

Set the number of iterations before training our models. Number of iterations are based on the problem set

a. Forward propagation

The input provides the initial information that then propagates to the hidden units at each layer and finally produce the output y^. The architecture of the network entails determining its depth, width, and activation functions used on each layer. Depth is the number of hidden layers. Width is the number of units (nodes) on each hidden layer since we don't control neither input layer nor output layer dimensions. There are quite a few sets of activation functions such Rectified Linear Unit, Sigmoid, Hyperbolic tangent, etc. Research has proven that deeper networks outperform networks with more hidden units. Therefore, it's always better and won't hurt to train a deeper network (with diminishing returns).

Importance: Compute the predicted output

b. **Compute cost function**
   Cost function in neural network is used to calculate the loss between the actual value and the expected value. In my problem, I have used the entropy loss cost function. Post computing the cost function, we move to the backward propagation.
   Importance: Compute loss
c. **Backward propagation**
   We have the starting point of errors, which is the loss function, and we know how to derivate it, and if we know how to derivate each function from the composition, we can propagate back the error from the end to the start. Backward propagation mainly uses chain rule for computing the derivates.
   Importance: Calculate the derivates and bias at each layer of neural network
d. **Update parameters (using parameters, and grads from backprop)**
   After gathering all the weights at each subsequent layers, we will be updating the weights. With these updated weights our loop ends, and we will be re-implementing the forward propagation again to compute the predicted output

3. **Use trained parameters to predict labels**
   Once our neural network has completed training by learning the weights and bias values, we will be passing the training and test sets to determine the quality of our neural network.

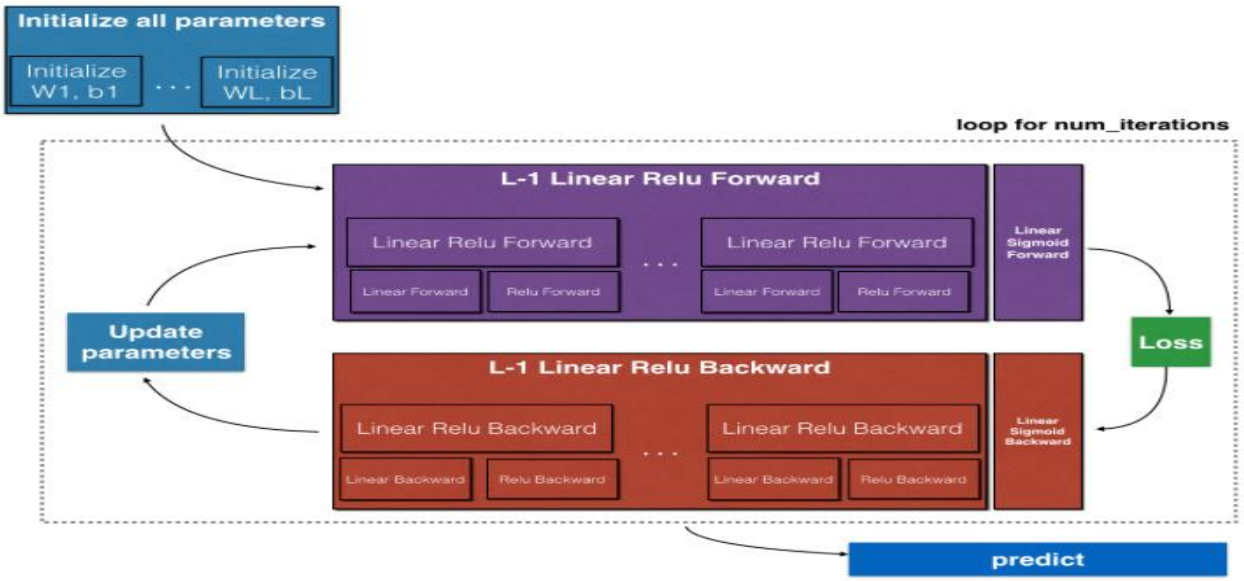


Figure 7 Methodology for implementing neural nets

5.1.3 Architecture of neural network for diabetes foreteller[2]

- I have built a deep neural network with 6 layers.
- Hence, my model has 5 hidden layers and one output layer
- My model is mentioned as,

[LINEAR -> RELU] × (L-1) -> LINEAR -> SIGMOID

RELU activation function or Rectifier:

I have used RELU activation function for removing the negative values

- RELU activation function is used on 5 hidden layers as the activation function
- RELU activation is given by,

$R(Z) = max(0,Z)$

Sigmoid Function:

Sigmoid function is used to return the values ranging from 0 to 1, which is a best function for calculating the probability. At the output layer (layer 6), I have used Sigmoid function as the activation function for fetching the probability values at the output.

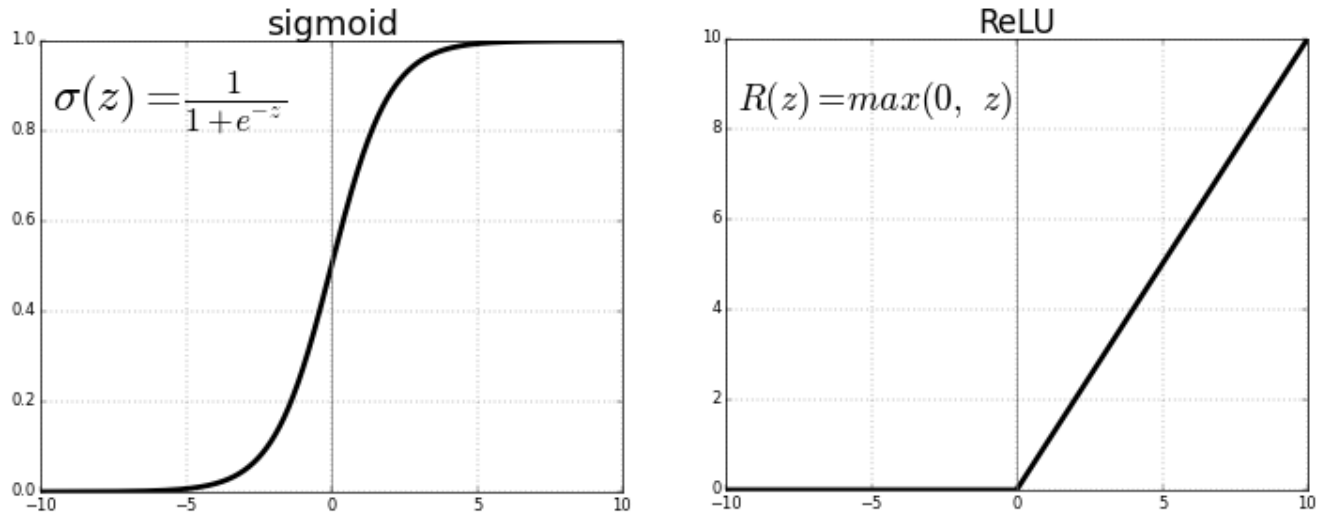


Figure 8 Sigmoid and ReLu

Figure 9 Overview of Diabetes Foreteller Neural Nets

Model Training details:

To train the model, I have set the following parameters,
1. Learning Rate = 0.0075
2. Num of iterations = 8000
3. Regularization = 'Dropout Regularization'

Regularization helps my neural network model to avoid overfitting. In my model, I have chosen dropout regularization. Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel. During training, some number of layer outputs are randomly ignored or "dropped out." This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the

prior layer. In effect, each update to a layer during training is performed with a different "view" of the configured layer.

By dropping a unit out, we mean temporarily removing it from the network, along with all its incoming and outgoing connections

— *Dropout: A Simple Way to Prevent Neural Networks from Overfitting, 2014.*

Dropout has the effect of making the training process noisy, forcing nodes within a layer to probabilistically take on responsibility for the inputs.

4. **Early stopping[1] = True**
   Early stopping is a form of regularization used to avoid overfitting when training a learner with an iterative method, such as gradient descent. Such methods update the learner so as to make it better fit the training data with each iteration. Up to a point, this improves the learner's performance on data outside of the training set. Past that point, however, improving the learner's fit to the training data comes at the expense of increased generalization error. Early stopping rules provide guidance as to how many iterations can be run before the learner begins to over-fit.

5. **Optimization = 'Stochastic Gradient Descent'[9]**
   The word 'stochastic' means a system or a process that is linked with a random probability. Hence, in Stochastic Gradient Descent, a few samples are selected randomly instead of the whole data set for each iteration. We use stochastic gradient descent instead of normal gradient descent to train our model.

6. **Cost function used: Entropy loss function**

   The cross-entropy compares the model's prediction with the label which is the true probability distribution. The cross-entropy goes down as the prediction gets more and more accurate. It becomes zero if the prediction is perfect.

$$-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\left(a^{[L](i)}\right)+\left(1-y^{(i)}\right)\log\left(1-a^{[L](i)}\right)\right)$$

Figure 10 Entropy cross function

5.2 Android User Interface architecture[4]

I have chosen Model, View and Presenter (MVP) design pattern for building diabetes foreteller application, mainly due to its modularity feature. I observed that, amongst other architectural patterns MVP offers better code readability and I can make new changes easily without

disturbing other features. The model-view-presenter (MVP) pattern separates application functionality into three kinds of components:

Model:
The model component contains the business logic and stores the application data. The model component communicates with the Diabetes Foreteller REST API through JSON request.

View:
The view component contains activities, fragments and other android views come into this layer, the general idea is that this layer has be very dumb, it shouldn't know or be concerned about things like business logic or data layer logic of the application, the responsibility of this layer then is to just display the data to the user.

Presenter:
Presenter is the middleman or mediator between View and Model which hold responsibilities of everything which must deal with presentation logic in our application. Presenter does the job of querying your Model, updating the View while responding to the user's interactions. It monitors Model and talks to View so that they can handle when a View needs to be updated and when to not.

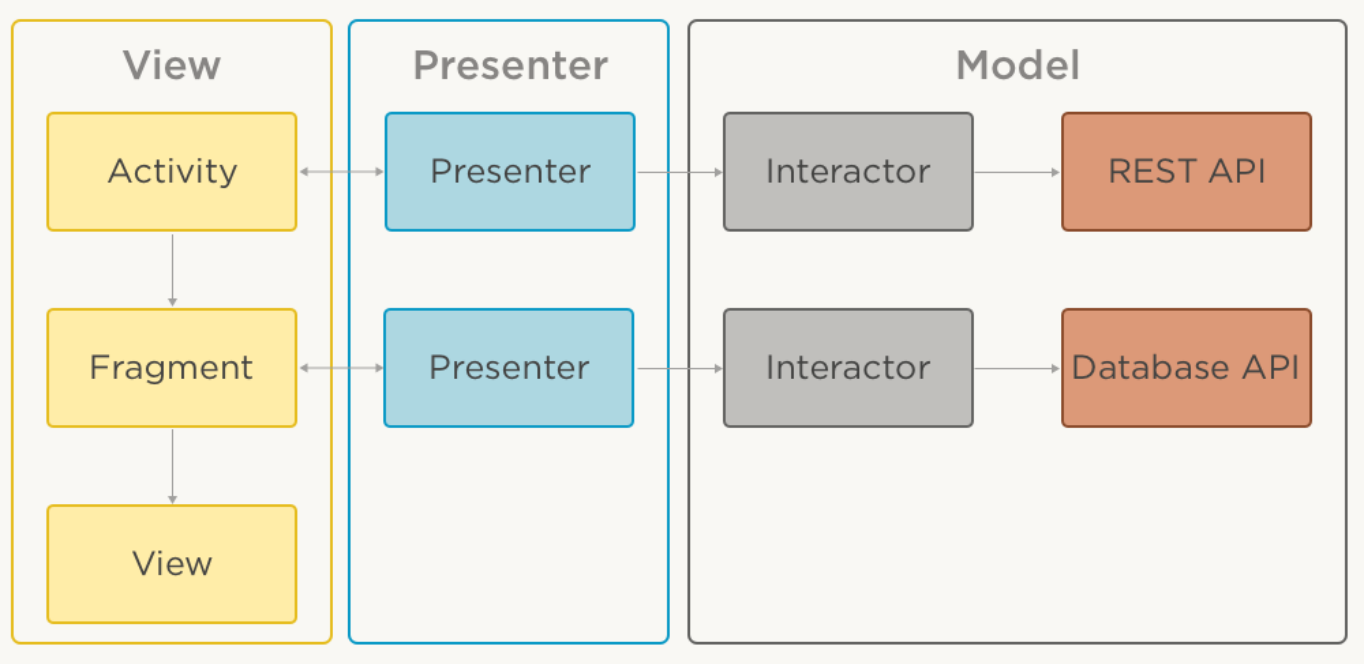


Figure 11 MVP Architecture for android application

Advantages of Model View Presenter pattern[4],

1. **Clear separation of responsibilities between components.**
   This separation allows for an easier understanding and maintenance of the code base.

2. **Modularity:**
   Modularity allows you to e.g. switch to a different implementation of view component in order to completely change application's UI, while all other components remain intact.
3. **Easier testing.**
   MVP pattern offers well defined boundaries between components, it becomes much easier to test each component in isolation.

6 IMPLEMENTATION

I have handled two levels of implementation. The first implementation is deploying my machine learning model as RESTful web service[11] in Heroku. Initially, I have collected the dataset and performed exploratory data analysis correcting the missing values and including additional features to make my dataset more flexible. Once completing the data analysis, I built a logistic regression neural network[1] using neural networks methodology to build my machine learning model with 86% accuracy.

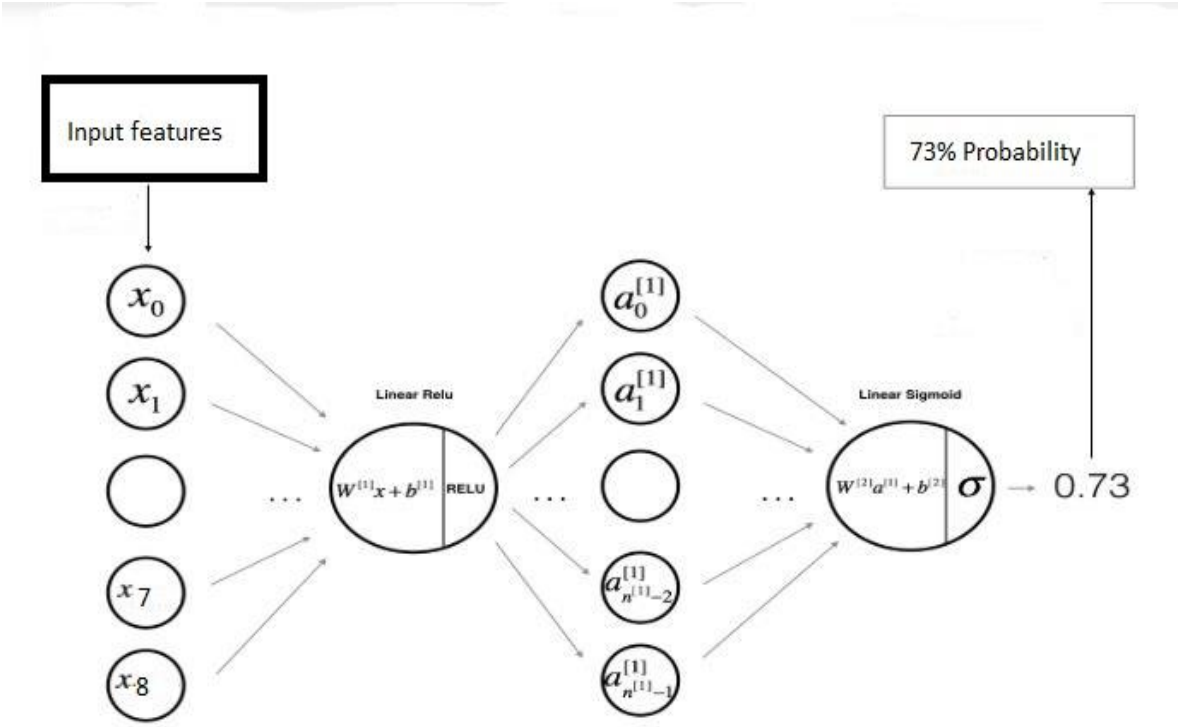


Figure 12 Overview of Diabetes Foreteller Neural Network

After building my model, I converted my machine learning model as RESTful Web service[11] using Flask framework. After converting my machine learning model as RESTful Web service, I will feed all the data related to the key symptoms of diabetes to my diabetes foreteller API and it delivers the probability of diabetes with respect to my input. I tested my RESTful Web Service using Postman software. After testing my model, I deployed my RESTful web service in Heroku cloud environment[12].

Post completing my first level implementation, I built an android application which serves as user interface to the end users to assess their probability of suffering diabetes in their lifetime. I built the application using Model View Presenter design pattern. The model component contains all the application data and communicates with the Diabetes Foreteller RESTful web service. The View component acts as an interface to the user, all the user's actions will be recorded in the interface component. The presenter renders all the android activity to the users upon request. The primary objective of this android application is to fetch the inputs from the user and transfer the inputs to the Restful web service deployed on the Heroku application through a JSON request. The RESTFUL Web service process all the inputs and delivers a response to the application through a JSON response. Post receiving the response, the application displays the probability of diabetes to the user.

7 TESTING

7.1 Results Achieved with my machine learning model:

To measure my machine learning model, I have used F1 score and accuracy to measure the quality of my model.

F1 Score[1]:

The F1 score (also F-score or F-measure) is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score: p is the number of correct positive results divided by the number of all positive results returned by the classifier, and r is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive). The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

	Predicted Positives	Predicted Negatives
Positives	True Positives	False Negatives
Negatives	False Positives	True Negatives

Figure 13 F1 score matrix

7.1.2 Test Cases for machine learning model:

TC-01 Test Machine Learning Model with training data

Description	Test the probability of diabetes for a user
Requirements Tested	FR-1
Related Use Cases	UC-1
Test Setup	Tested machine learning model with test dataset containing 500 training cases
Test Steps	Once the model is built, the machine learning model is tested with training dataset .
Expected Results	Accurate results for 100% of training data Zero false positives and false negatives on the f1 score matrix

After evaluating my model, I have got the following scores,

Accuracy = 86%

An accuracy of 86% in a dataset less than 800 values shows a healthy sign.

```
In [56]: from sklearn.metrics import accuracy_score

deep_network = L_Layered_model(learning_rate = 0.075, **params)

print('Accuracy is {}'.format(accuracy_score(y_test,y_pred )))

train_predict(deep_network, X_train, y_train, X_test, y_test)
```

```
Accuracy is 0.8658008658008658
F1 score for training set is: 1.0000
F1 score for testing set is: 0.7974
```

Figure 14 Accuracy and F1 score screenshot

F1 Score (Training set) = **1.0**

```
In [25]: print(f1_score_matrix(y_train,prediction_final))
```

```
[[349   0]
 [  0 187]]
```

Figure 15 F1 score matrix of train dataset

F1 score (Test set) = **0.79**

TC-02 Test Machine Learning Model with test data

Description	Test the probability of diabetes for a user
Requirements Tested	FR-1
Related Use Cases	UC-1
Test Setup	Tested machine learning model with test dataset containing 100 test cases
Test Steps	Once the model is built, the machine learning model is tested with test dataset.
Expected Results	Accurate results for 100% of testing data Zero false positives and false negatives on the f1 score matrix

F1 score matrices:

```
In [22]: print(f1_score_matrix(y_test,prediction_final))

[[139  12]
 [ 19  61]]
```

Figure 16 F1 score matrix of test dataset

Graphical plot showing the fall in loss:

The plot below shows that, the drop-in loss after 2500 iterations.

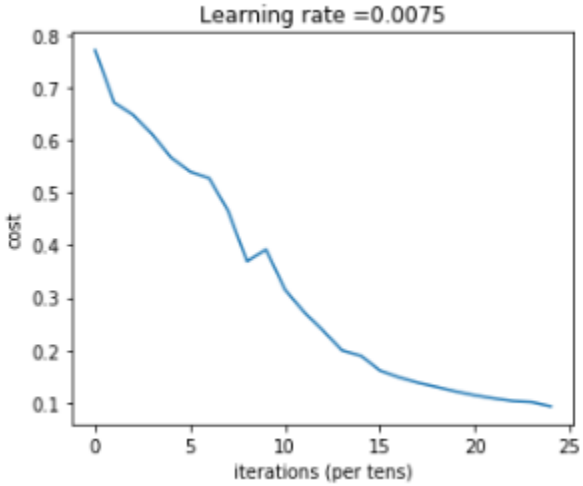


Figure 17 Drop in cost function

7.2 Testing my android application

Manual testing is implemented for this application and results are monitored. For each test case 2 input are designed, one of them expecting to give successful result and the other one to arise fail result.

7.2.1 Test Cases for Android Application

TC-03 Test the REST API using Postman

Description	Test the newly created REST API that includes diabetes predictor machine learning model.
Requirements Tested	FR-1
Related Use Cases	UC-1
Test Setup	Tested machine learning model with test dataset containing 5 test cases
Test Steps	The REST API is fed into Postman and passed

	the input values as JSON files.
Expected Results	Accurate results for 90% of test data

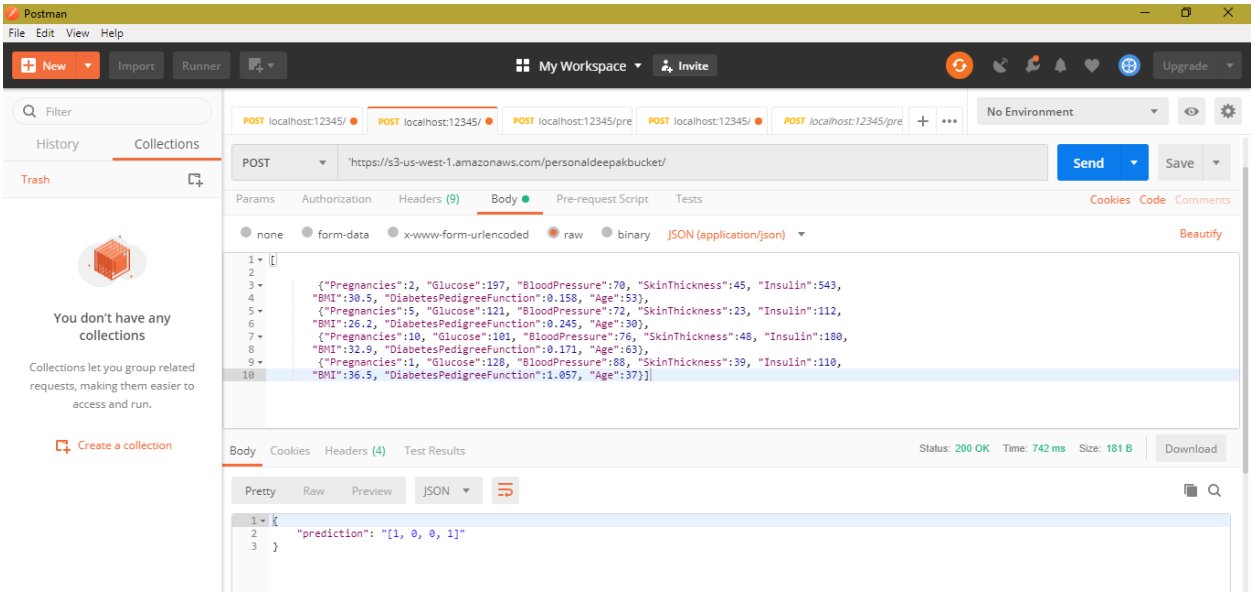


Figure 18 Testing API using Postman

TC-04 Testing the REST API with diabetes data.

Description	Test the probability of a diabetes patient's data in postman
Requirements Tested	FR-1
Related Use Cases	UC-1
Test Setup	Tested machine learning model with test data that contains the current diabetes patient's values as 10 test cases.
Test Steps	The REST API is fed into Postman and passed the input values as JSON files.
Expected Results	Accurate results for 90% of test data

TC-05 Testing the communication between REST API and the mobile application

Description	Testing the communication between the mobile application and REST API by transferring a dummy data.
Requirements Tested	FR-1
Related Use Cases	UC-1
Test Setup	Both the mobile application and the REST API communicates using JSON.

Test Steps	Created a POST request from the mobile application to ensure the connectivity.
Expected Results	Expected '200' return code from the REST API

TC-06 Testing the mobile application

Description	Testing whether the mobile application is delivering the exact values as the test data.
Requirements Tested	FR-1
Related Use Cases	UC-1
Test Setup	Both the mobile application and the REST API communicates using JSON.
Test Steps	Created a POST request from the mobile application to fetch the probability of diabetes for a person's data.
Expected Results	Expected 90% accurate values.

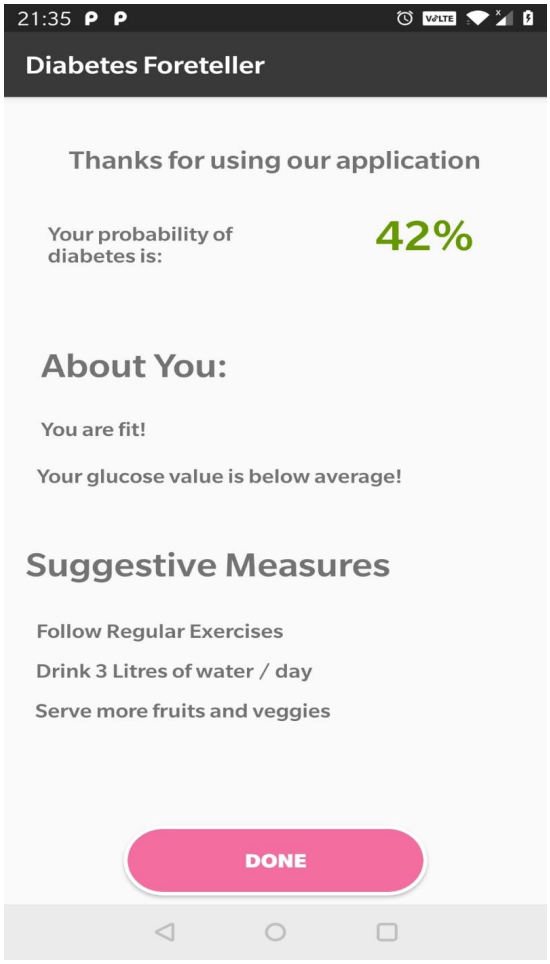


Figure 19 TC-6 Calculate Diabetes Risk and Suggestive measures

- The user has entered all her biometric details
- The output is the probability of diabetes, which is 42% for this user.
- Based on the biometric details the user has entered and probability of diabetes, few suggestive measures has been recommended to the user.

TC-07 Testing the suggestive measures for the user

Description	Testing the suggestive measures for the users
Requirements Tested	FR-2
Related Use Cases	UC-2
Test Setup	The suggestive measures are pre-programmed based on the probability values and they will shown to the users upon request.
Test Steps	1. User enters his personal information 2. Assesses his probability of diabetes 3. Gathers his suggestive measures based

	on the probability values
Expected Results	The application shall display the pre-programmed suggestive measures based on the probability.

TC-08 Test the diabetes information android activity

Description	Testing the diabetes information android activity
Requirements Tested	FR-3
Related Use Cases	UC-3
Test Setup	The diabetes information screen consists of all pre-programmed documents that will be shown to the user upon request.
Test Steps	1. User enters his personal information 2. Assesses his probability of diabetes 3. User clicks on 'Recent trends in diabetes' button on the android screen
Expected Results	The application shall display the pre-programmed documents related to the diabetes to the user.

6.1 TEST CASE TRACEABILITY

Functional Requirements	Test Cases
FR-01	TC-1, TC-2, TC-3, TC-4, TC-5
FR-02	TC-6, TC-7
FR-03	TC-8

8 SUMMARY AND CONCLUSION

At the end of the project, all the goal and objectives are achieved.

- The key goal to predict the probability of diabetes for an individual is successfully achieved using 86% accuracy.
- The project is completely built using Waterfall software development model.
- I have built a machine learning model using deep neural networks with 6 layers. The model produces 86% accurate results.
- The model is converted as RESTful API using Flask framework and deployed in AWS lambda.
- An android user interface is successfully created for the end users to access the probability of diabetes.
- The android application is powered by a RESTful web service at the backend.
- In addition to this, diabetes foreteller application also suggests preventive measures to the user to stay fit and prevent diabetes.

Possible Enhancements

- **Build Reinforcement Learning model:** Convert the supervised machine learning model into Reinforcement model for collecting more data. Through reinforcement model, we can collect more real time user data and improve our model. Due to minimal exposure with reinforcement model, I was not able to implement reinforcement model in this project.
- **Built a website:** Make a responsive web application that runs on all major personal computers, and all devices irrespective of the platform.
- **Improve the accuracy to 99%:** I hope that, once I collect one lakh data, my model would be 99% accurate.

9 REFERENCES

[1] Deep Learning Specialization
< https://www.coursera.org/specializations/deep-learning>

[2] IEEE Neural Networks paper
< https://ieeexplore.ieee.org/document/7012785>

[3] INTRODUCTION TO MACHINE LEARNING
< http://robotics.stanford.edu/~nilsson/MLBOOK.pdf>

[4] MVP Architecture for Android Applications
<https://medium.com/@arifnadeem7/mvp-architecture-for-android-applications-3a403355b345>

[5] Diabetes dataset
< https://archive.ics.uci.edu/ml/datasets/diabetes>

[6] Analysis and prediction of diabetes diseases using machine learning algorithm: Ensemble approach.
<https://pdfs.semanticscholar.org/f5e0/290d66e0fc12ba409fda0fc2a2c6a7c1066e.pdf>

[7] Gradient Boosting
< https://en.wikipedia.org/wiki/Gradient_boosting>

[8] Fundamentals of Neural Networks
<https://medium.com/datadriveninvestor/the-basics-of-neural-networks-304364b712dc>

[9] Difference between batch and stochastic gradient descent
<https://towardsdatascience.com/difference-between-batch-gradient-descent-and-stochastic-gradient-descent-1187f1291aa1>

[10] Statistics Fundamentals with Python
<https://www.datacamp.com/tracks/statistics-fundamentals-with-python>

[11] Turning Machine Learning Model into REST API
<https://www.datacamp.com/community/tutorials/machine-learning-models-api-python>

[12] Deploying machine learning model in Heroku
<https://medium.com/@ashiqgiga07/deploying-rest-api-based-flask-app-on-heroku-part-1-cb43a14c50c>

[13] Diabetes
<http://www.diabetes.org/>

[14] Study of diabetes symptoms
<https://ieeexplore.ieee.org/document/7724629/>

[15] Suggestive measures of diabetes
<https://www.betterhealth.vic.gov.au/health/ten-tips/10-tips-to-help-prevent-type-2-diabetes>

10 APPENDIX:

Operating Android Application:

Note: Before testing your own diabetes probability, please take blood test in a laboratory mainly for blood pressure, insulin, skin thickness and glucose level. Without these readings the application won't execute your readings.

A. Open diabetes foreteller android application

Step 1: Open the application

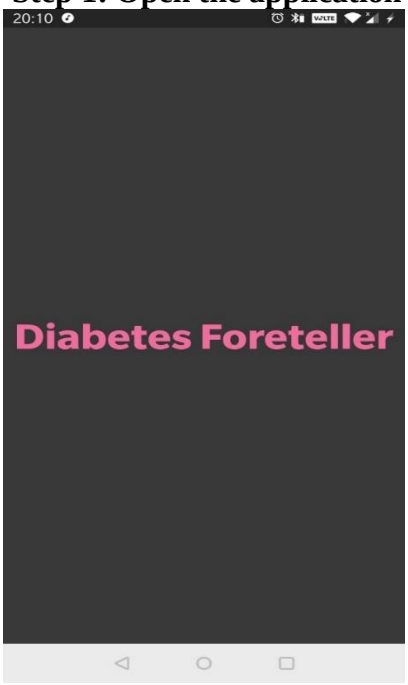


Figure 20 Welcome screen of diabetes foreteller

B. Enter personal details:

1. Enter name
2. Enter age
3. Select your sex
4. Select Continue

Step 2: Enter basic details

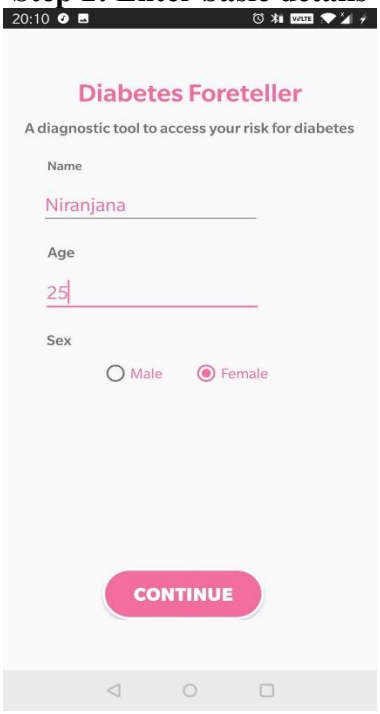
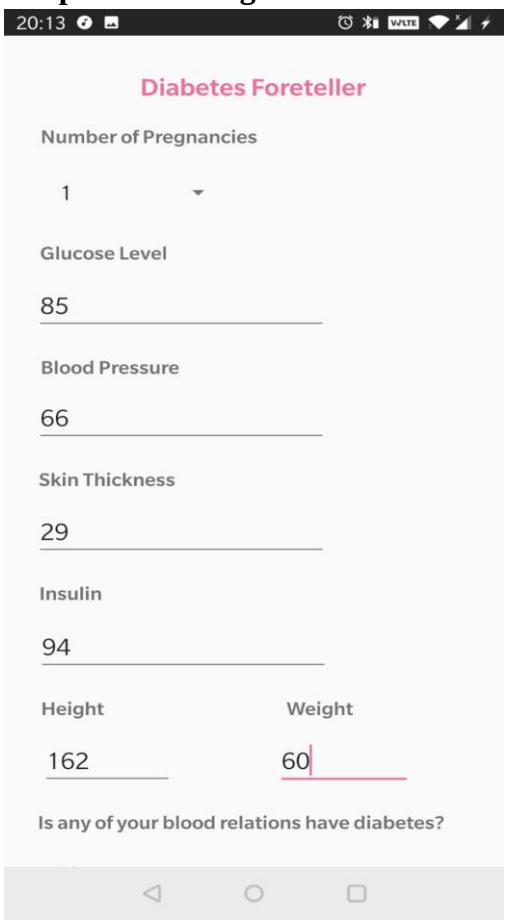


Figure 21 Enter personal details

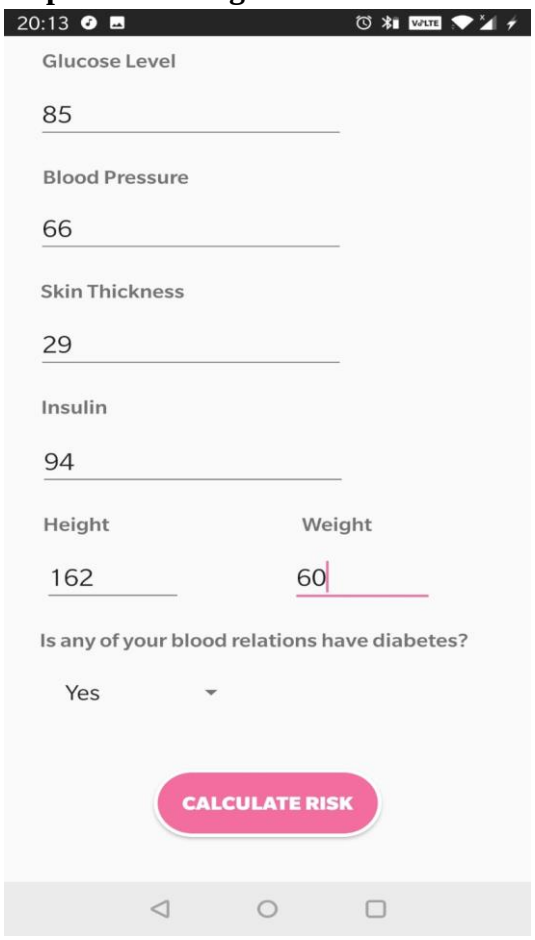
C. Enter biometric details:

1. Enter your number of pregnancies for female users.
   If the user is male, please select '0'
2. Enter your blood glucose level
3. Enter your blood pressure
4. Enter your skin thickness
5. Enter your insulin measurement
6. Enter your height, weight
7. Answer the question, "Is any of your blood relations have diabetes?"
8. Select 'Calculate Risk'

Step 3a: Entering biometric details



Step 3b: Entering biometric details



D. Accessing the probability

1. After selecting 'Calculate Risk', we can access the probability of diabetes
2. We could see whether we are fit or not
3. The application suggests basic preventive measures.

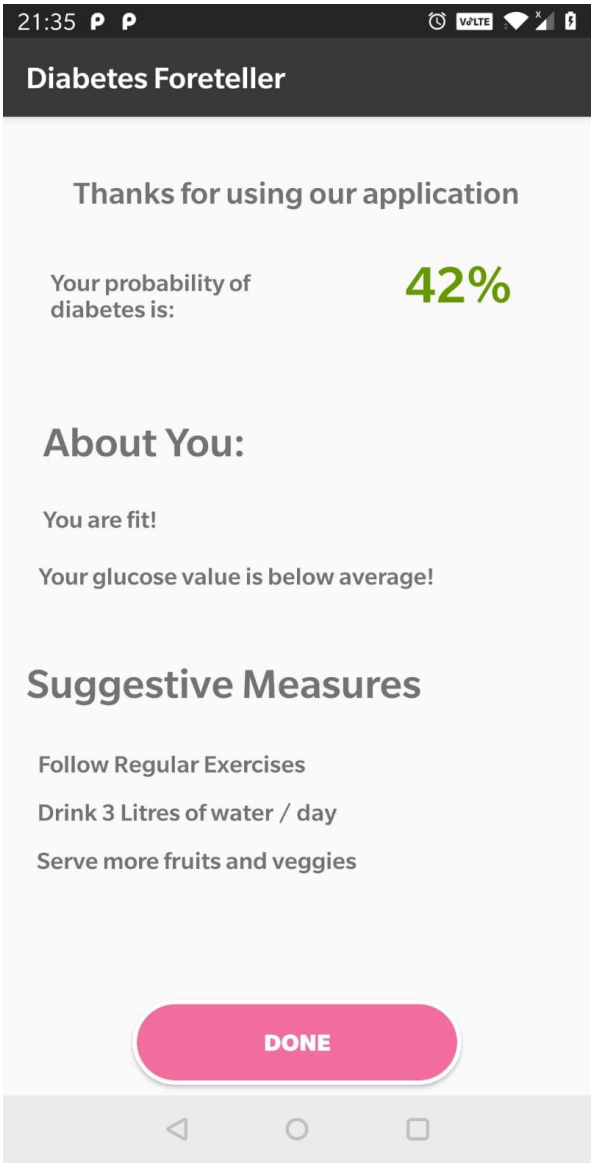


Figure 22 Final Output screen

11 Project Deliverables:

The project deliverables include the following documents:

1. Source code
2. Jupyter Notebooks
3. Android Project files
4. API files
5. PIMA diabetes dataset

Project Deliverables Link:

<https://drive.google.com/drive/folders/1CSHR9YKMPAYbL2n5onUJwIRVdd_2Dy3K?usp=sharing>