Minor Report

Project Title: Medical Search Engine

(Predicting Disease based on Analysis of Symptoms)

Submitted By-

Shresth Singh (16103266)

Palak Arora(16103046)

Submitted To -

Megha Rathi

Abstract:

Sometimes people feel dubious about the things that happened to their physical health and they are not sure what disease affects them. Then, usually, they end up ignoring and underestimating the symptoms they are experiencing. Most people believe that minor illnesses, such as a cold or diarrhea, do not require special examination or complex treatment. The problem is that a minor illness could be an indicator of a serious illness. Therefore, one must know whether or not they show the symptoms of a harmful disease and what exactly they should do about it. To make it easier for someone to diagnose the symptoms they are experiencing, we developed an Android-based application which uses a neural network trained on disease - symptom database to predict the disease afflicting the user, that can provide solutions for what they should do when they experience these symptoms, whether they can do their own treatment at home or they should be reviewed immediately by a doctor. By using this application, the diagnosis results of the diseases can be detected through the consultation process or answering the questions provided by the system quickly and effectively with an Android smartphone as a means.

Objectives:

The current technology has been used to improve the quality of human life in different ways. One of the areas that most benefit is medicine. However, despite the great progress that the field has made, problems such as the full coverage of the general population, which is presented particularly in developing countries, still remain unresolved. Although the technology has contributed substantially in the way health services are provided, they have not been able to eliminate some problems such as the barriers that a patient has to access the medical service, wait times or the management of all cases presented in the emergency department. One of the solutions that technology proposes is the use of automatic learning methods, that is, applications computing that are trained with data and are capable to perform the same activities as a doctor. This type of

systems has been proposed since the eighties using different methods such as the use of artificial intelligence.

Our project is focused on reducing the time it takes to find the ideal specialist in organizations that provide medical services, with the purpose of focusing the valuable time of the specialists to determine and solve the affliction, leaving the first phase of identification or diagnosis to a computer system.

The system developed by us consists of an Android application which asks the user to enter the symptoms they are experiencing and then sends that data to our web API which predicts the disease based on the symptoms and also returns possible treatments and care tips for the user to employ against the disease.

Division of Work:

Data collection , data cleaning , creation of neural network models was done by both project members separately on two different data sets .

Two different apps were created by the respective group members to implement these neural networks.

Team Member	*Work*
Shresth Singh	● Scraping of diseases and their symptoms and hosting this data on a live server
Palak Arora	● Data Visualization of the prominent symptom ● Real time Heart rate Calculator

Background Study and Findings:

Mani Shankar et. al.[1] propose a method to predict diseases and their cure time based on symptoms reported and severity of these symptoms.They do this by assigning different coefficients to each symptom of a disease, and filtering the dataset with the severity score assigned to each symptom by the user. The diseases are identified based on a numerical value calculated in the fashion mentioned above. For predicting the cure time of a disease, they use reinforcement learning. Their algorithm takes into account the similarity between the condition of the current user and other users who have suffered from the same disease, and uses the similarity scores as weights in prediction of cure time. They also predict the current medical condition of user relative to people who have suffered from same disease.The dataset used was created by collecting medical information about students in their college. They claim that their approach is better than other proposed approaches as they ask for symptoms and severity rating for each symptom which is similar to the interaction between a doctor and a patient.

A. Piñango and R. Dorado[2] in their paper address the problem of prediction of diseases based on specific symptoms in order to improve medical attention given to patients.They propose a flexible Bayesian framework for modeling symptom association with disease in population-based studies. They employ a Bayesian probabilistic model to describe the correlation between specific symptoms such as fever and its cause. The effectiveness of the model is tested with a similarity based measure and training data.The dataset was collected from the openMRS free software project which contained anonymized data of 5000 people and 5,00,000 observations.They claimed to get high accuracy with this dataset.

Imam M Shofi et. al.[3] propose to build an Android based application which asks the user some questions and then predicts their diseases based on their answers. They used a forward chaining inference trained on a dataset extracted from a book "Doctor in Your Home" by Dr. Tony Smith. They claim that predictions of their model were highly accurate as validated by experts.

Sellappan Palaniappan and Rafiah Awang [4] in their paper discuss the efficiency of data mining techniques namely: Naive Bayes Classifier , Decision Trees and Neural Networks in predicting the occurrence of a heart disease in a patient using categorical data from patients medical history. The dataset was extracted from the Cleveland Heart Disease database and all numerical data was converted to categorical data. The results show that the neural network and the naive bayes classifier were able to predict the occurence of heart disease more accurately than the decision tree.

Dharavath Ramesh et. al. [5] in their paper summarize the role of Big Data Analysis in healthcare and the various shortcomings of traditional machine learning algorithms. They argue that the best way to extract useful information from big data is to use a fusion of two or more data mining techniques as they give the best possible result in least amount of time.

Min Chen et. al.[6] in their paper propose a new convolutional neural network based multimodal disease risk prediction (CNN-MDRP) algorithm using structured and unstructured data from hospitals. Their model was tested on a local disease of cerebral infarction which yielded a an accuracy of 94.8 percent.

Prabakaran.N , and Kannadasan.R[7] in their paper propose a reliable multi process method combining decision tree techniques and clustering to build a cardiac arrest risk prediction system.

Aditi Gavhane et. al.[8] propose to develop an application which can predict the vulnerability of a heart disease given basic symptoms like age, sex, pulse rate etc. They used the Cleveland database from UCI library to collect their data set and trained a multi-layered perceptron on it which yielded a high accuracy in predicting the heart disease.

Project Design:

Our project consists of two different Android applications which use different neural network models trained on two different data sets which communicate with a common web server to deliver results to the user.

Design of first model/dataset:

The first dataset is taken from Disease-Symptom Knowledge Database[9] which contains 149 diseases with their symptoms and the count of their occurrences which was collected from New York Presbyterian Hospital.

The data was imported to a dataframe then after appropriate data cleaning and preparing methods it was converted to a dataframe with 149 rows and 404 columns where the rows contain observation for each disease and the columns contain the information about presence of a symptom in every row. If for an observation the symptom contains one then that shows presence of that symptom and if it contains zero then that shows absence of that symptom for that particular observation.

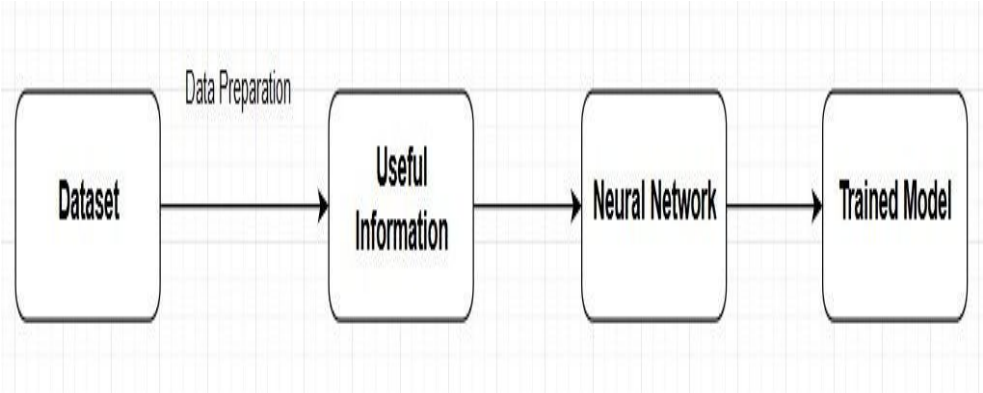
The neural network was implemented using the keras library and trained on the environment provided by Google Colab(Python 3 Google Compute Engine). The neural network consists of a total of three fully connected layers consisting of 12, 404 and 149 layers respectively. The first two layers use relu as activation function while the final layer uses softmax function to give a probabilistic output. To compile the model binary crossenthropy was used as a loss function and adam was used as optimizer while accuracy was chosen for the metric. The model was trained for 500 epochs with a batch size of 16.

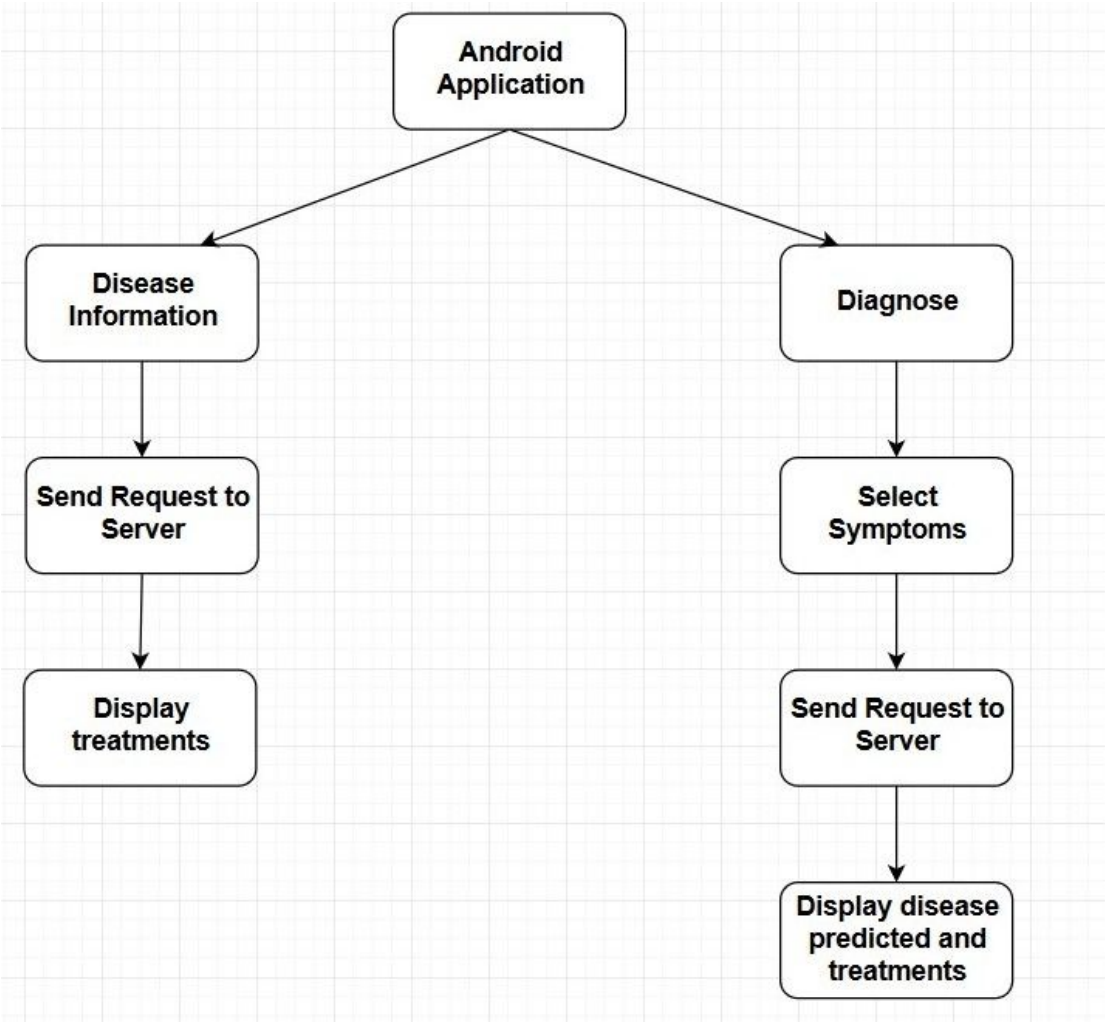
The weights of the trained model were stored in a .h5 file to be used later predict the disease at the server side.

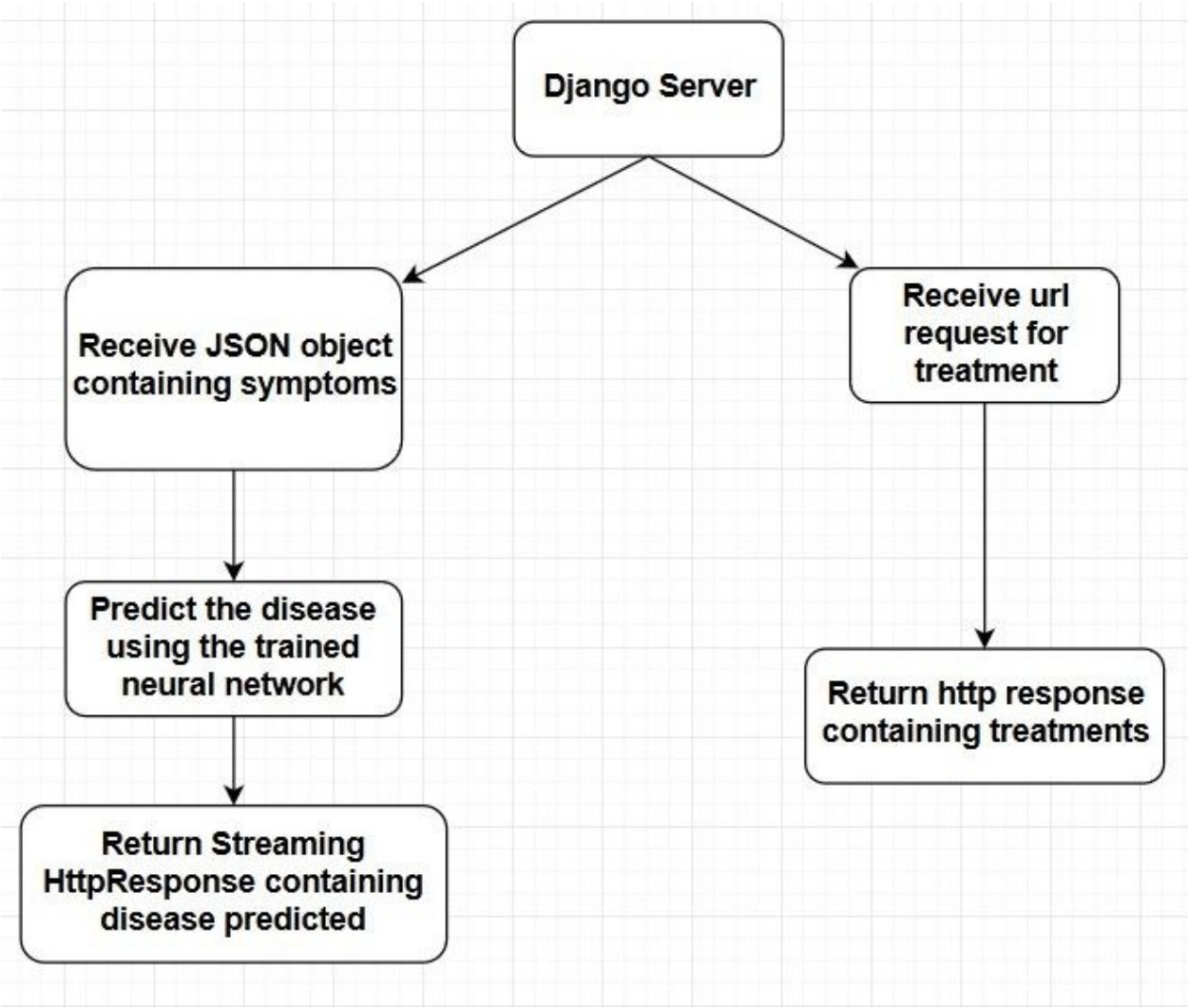
The Android application to be used as an interface was developed on Android Studio which gives the user an option to search for treatments of diseases present in our database and an option diagnose themselves. In order to diagnose

themselves the user has to select the symptoms they are showing and submit them which sends selected symptoms to our Django server hosted on pythonanywhere[10] where the previously created model is used predict most probable disease and return to the user the name of the disease and treatments for the the predicted disease.

The database for treatments of the diseases was made by scraping data from wikipedia and webmd by creating a web crawler using the python library BeautifulSoup4.




Android Application
Disease Information
Diagnose
Send Request to Server
Select Symptoms
Display treatments
Send Request to Server
Display disease predicted and treatments



Design for Second model/dataset:

The second dataset is also taken from Disease-Symptom Knowledge Database[9] which contains 149 diseases with 404 symptoms and the count of their occurrences which was collected from New York Presbyterian Hospital survey in 2004.

The data was imported to a dataframe using pandas library and then after appropriate data cleaning and preparing methods it was converted to a dataframe with 4962 rows and 133 columns where the rows contain observation for various disease with different symptom occurrences and the columns contain the information about presence of a symptom and the prognosis(output class). If

for an observation the symptom contains one then that shows presence of that symptom and if it contains zero then that shows absence of that symptom for that particular observation.

The neural network was implemented using the keras classifier library and the training is done with an 4- layer dense ANN with the input layer having 132 input symptoms.Three hidden layers containing 65,33 and 33 neurons from the first layer respectively.The activation function used in all the layers starting from the first layer is relu, tanh,relu and softmax respectively. The output class i.e. prognosis containing 44 classes of categorical type was converted into binary format by using OneHotEncoder , for better prediction of results and to remove redundancy.The ANN training is done with batch size of 25 and 500 epochs.The accuracy obtained is 0.8769 i.e. approximately 88%

- Overfitting :

Keeping irrelevant attributes in your dataset can result in overfitting.
It is important to remove redundant and irrelevant attributes from your dataset before evaluating algorithms. This task should be tackled in the Prepare Data step of the applied machine learning process.In our project we have tackled this problem by using Dropouts in all the layers.

The trained keras model is then converted to ‘model.h5’ file which is further converted to TensorFlow Lite file ‘model.tflite’ which is easily compatible with Android application and since the model is present in the app the it doesn’t take time for prediction , along with this the probability of occurrence of the predicted disease according to the provided diseases is also shown on the app.

The ‘model.tflite’ is then added in Assets directory of the Android Studio , android application is made with basic spinners taking symptoms information from the user and then passing it into the model then the predicted disease is displayed on the App along with the probability of the actual occurrence of the disease .

These predicted disease is then passed to the Web Scraper which is .ipynb made with with Beautiful Soup in python file hosted on pythonanywhere.The data is scraped from it are the treatment ,management or prevention measured and is displayed on the app as well.

Heart Rate Calculator :

Heart Rate Calculator is an extra feature added in the android app that uses the camera and flash light of the user's phone to calculate heart rate by calculating amount of red in preview frames and calculates average.

Working:

1. Gets the amount of red in each preview frame.
2. Decodes and gets the average amount of red component in image.
3. Calculates the rolling average.
4. Takes data in chunks of 10 seconds.
5. Currently using textureview instead of surfaceview for removing preview of camera from user's eyes.

The user can also see for the condition of this heart i.e. where the heart is working Good , Average ,Poor etc. based upon the out coming Heart Rate by seeing through the reference guide provided in the app.

Implementation :

```
import pandas as pd
import csv
from collections import defaultdict

disease_list = []

def return_list(disease):
    disease_list = []
    match = disease.replace('^','_').replace("\xc2\xa0","").split('_')
    ctr = 1
    for group in match:
        if ctr%2==0:
            disease_list.append(group)
        ctr = ctr + 1

    return disease_list

with open("dataset_uncleaned.csv") as csvfile:
    reader = csv.reader(csvfile)
    disease=""
    #weight = 0
    disease_list = []
    #dict_wt = {}
    dict_=defaultdict(list)
    for row in reader:

        if row[0]!="\xc2\xa0" and row[0]!="":
            disease = row[0]
            disease_list = return_list(disease)
            #weight = row[1]

        if row[2]!="\xc2\xa0" and row[2]!="":
            symptom_list = return_list(row[2])

            for d in disease_list:
                for s in symptom_list:
                    dict_[d].append(s)
```

```
with open("dataset_clean.csv","w") as csvfile:
    writer = csv.writer(csvfile)
    for key,values in dict_.items():
        for v in values:
            #key = str.encode(key)
            #key = str.encode(key).decode('utf-8')
            #.strip()
            key=key.strip()
            #v = v.encode('utf-8').strip()
            #v = str.encode(v)
            writer.writerow([key,v])
```

The code shown above extracts the name of diseases and their symptoms from the database and stores them in a .csv file.

```
import io
import pandas as pd
df = pd.read_csv('/content/gdrive/My Drive/dataset_clean.csv')
df
```

	Source	Target
0	hypertensive disease	pain chest
1	hypertensive disease	shortness of breath
2	hypertensive disease	dizziness
3	hypertensive disease	asthenia
4	hypertensive disease	fall
5	hypertensive disease	syncope
6	hypertensive disease	vertigo
7	hypertensive disease	sweat
8	hypertensive disease	sweating increased

```
df_1 = pd.get_dummies(df.Target)
df_1.head()
```

	Heberden's node	Murphy's sign	Stahli's line	abdomen acute	abdominal bloating	abdominal tenderness	abnormal sensation	abnormally hard consistency	abortion	abscess bacterial	...	vision blurred	vomiting	weepiness	weight gain	welt
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

```
df_1 = pd.get_dummies(df.Target)
df_1.head()
```

	Heberden's node	Murphy's sign	Stahli's line	abdomen acute	abdominal bloating	abdominal tenderness	abnormal sensation	abnormally hard consistency	abortion	abscess bacterial	...	vision blurred	vomiting	weepiness	weight gain	welt
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

5 rows × 404 columns

```
df_s = df['Source']
df_pivoted = pd.concat([df_s,df_1], axis=1)
df_pivoted.drop_duplicates(keep='first',inplace=True)
df_pivoted[:5]
```

	Source	Heberden's node	Murphy's sign	Stahli's line	abdomen acute	abdominal bloating	abdominal tenderness	abnormal sensation	abnormally hard consistency	abortion	...	vision blurred	vomiting	weepiness	weight gain	wel
0	hypertensive disease	0	0	0	0	0	0	0	0	0	...	0	0	0	0	(
1	hypertensive disease	0	0	0	0	0	0	0	0	0	...	0	0	0	0	(
2	hypertensive disease	0	0	0	0	0	0	0	0	0	...	0	0	0	0	(

```
cols = df_pivoted.columns
cols = cols[1:]
df_pivoted = df_pivoted.groupby('Source').sum()
df_pivoted = df_pivoted.reset_index()
df_pivoted
```

	Source	Heberden's node	Murphy's sign	Stahli's line	abdomen acute	abdominal bloating	abdominal tenderness	abnormal sensation	abnormally hard consistency	abortion	...	vision blurred	vomiting
0	Alzheimer's disease	0	0	0	0	0	0	0	0	0	...	0	0
1	HIV	0	0	0	0	0	0	0	0	0	...	0	0
2	Pneumocystis carinii pneumonia	0	0	0	0	0	0	0	0	0	...	0	0
3	accident cerebrovascular	0	0	0	0	0	0	0	0	0	...	0	0
4	acquired immuno-deficiency syndrome	0	0	0	0	0	0	0	0	0	...	0	0
5	adenocarcinoma	0	0	0	0	0	0	0	0	0	...	0	0
6	adhesion	0	0	0	0	0	0	0	0	0	...	0	1

The code above shows the process of data preparation.

For Model-1 :

```
from keras.models import Sequential
from keras.layers import Dense
from sklearn.preprocessing import LabelEncoder
from keras.utils import np_utils
import numpy

seed = 7
numpy.random.seed(seed)

dataset=df_pivoted.values
X = dataset[:,1:].astype(float)
Y = dataset[:,0]

encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)

dummy_y = np_utils.to_categorical(encoded_Y)
```

```
Using TensorFlow backend.
```

```
model = Sequential()
model.add(Dense(12, input_dim=404, init='uniform', activation='relu'))
model.add(Dense(404, init='uniform', activation='relu'))
model.add(Dense(149, init='uniform', activation='softmax'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(X, dummy_y, epochs=500, batch_size=16)
```

```
Epoch 3/500
149/149 [==============================] - 0s 209us/step - loss: 0.0402 - acc: 0.9933
Epoch 4/500
149/149 [==============================] - 0s 206us/step - loss: 0.0402 - acc: 0.9933
Epoch 5/500
149/149 [==============================] - 0s 185us/step - loss: 0.0401 - acc: 0.9933
Epoch 6/500
149/149 [==============================] - 0s 192us/step - loss: 0.0399 - acc: 0.9933
Epoch 7/500
149/149 [==============================] - 0s 176us/step - loss: 0.0397 - acc: 0.9933
```

The above code shows the training process of the model.

```
@csrf_exempt
def predict(request):
    if request.method =='POST':

        data=json.loads(request.body.decode("utf-8"))
        symptoms_reported = []
        size = data["size"]
        for i in range(size):
            symptoms_reported.append(data[str(i)])
        loaded_model = Sequential()
        loaded_model.add(Dense(12, input_dim=404, init='uniform', activation='relu'))
        loaded_model.add(Dense(404, init='uniform', activation='relu'))
        loaded_model.add(Dense(149, init='uniform', activation='softmax'))
        loaded_model.load_weights("/home/pythonestpython/mysite/serve/model.h5")
        loaded_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
        disease_df = pd.read_csv('/home/pythonestpython/mysite/serve/diseaselist.csv')
        symptom_df = pd.read_csv('/home/pythonestpython/mysite/serve/symptomlist.csv')
        x=np.zeros(404)
        for i in range(len(symptom_df['0'])):
            for j in symptoms_reported:
                if j==symptom_df['0'].values[i]:
                    x[i]=1
                    break
        arr=np.empty([5,404])
        for i in range(5):
            arr[i]=x
        pred=loaded_model.predict(arr)
        return StreamingHttpResponse(disease_df['Source'][np.argmax(pred,axis=1)[0]])

    #return JsonResponse(disease_df['Source'][np.argmax(pred,axis=1)].tolist(),safe=False)

def show(request,foo):
    df = pd.read_csv('/home/pythonestpython/mysite/serve/treatmentswiki.csv')
    return HttpResponse(df[df['Disease']==foo.replace('_',' ')]['Treatment'])
    #return HttpResponse(foo)

def show2(request,foo):
    df = pd.read_csv('/home/pythonestpython/mysite/serve/treatments2.csv')
    return HttpResponse(df[df['Disease']==foo.replace('_',' ')]['Treatment'])
```

The image above shows the functions which respond to requests made by the application. The predict() function contains the code to predict the disease while the functions show1() and show2() return the treatments of the disease as requested by the two Android Applications.

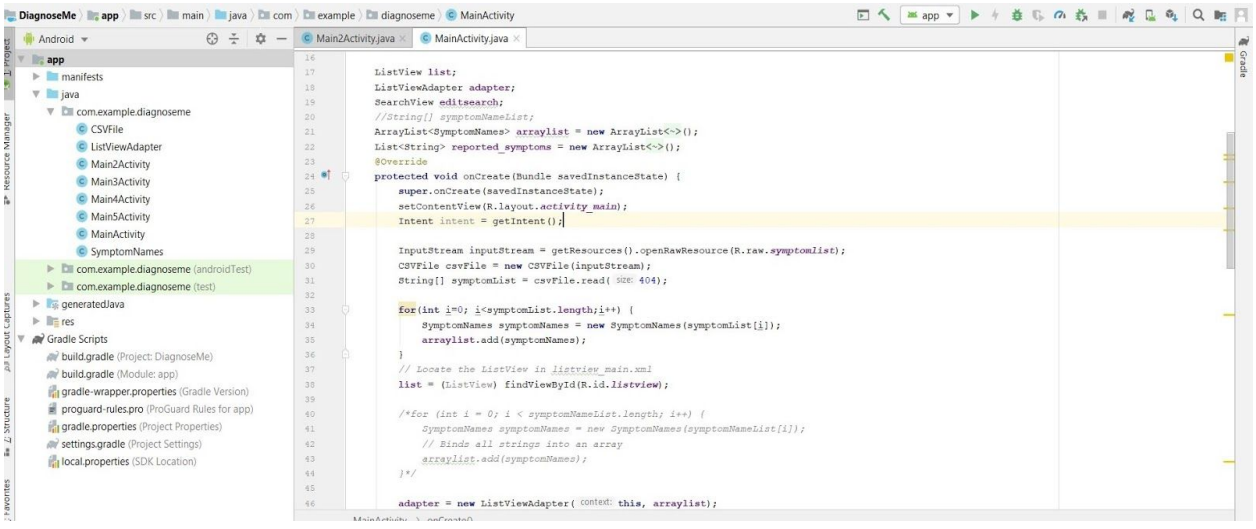
```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import urllib
df = pd.read_csv('diseaselist.csv')
```

```
df
```

	Unnamed: 0	Source
0	0	Alzheimer's disease
1	1	HIV
2	2	Pneumocystis carinii pneumonia
3	3	accident cerebrovascular
4	4	acquired immuno-deficiency syndrome
5	5	adenocarcinoma
6	6	adhesion
7	7	affect labile
8	8	anemia
9	9	anxiety state
10	10	aphasia
11	11	arthritis
12	12	asthma

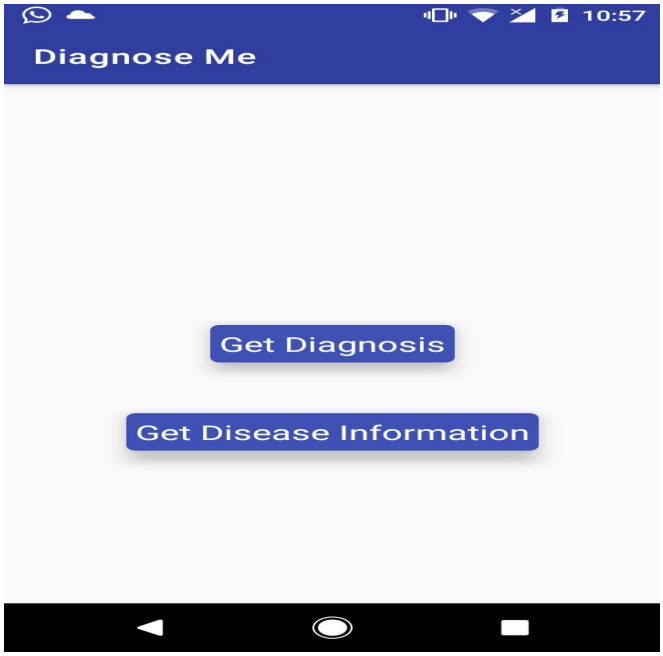
```
from time import sleep
import urllib.parse as urlparse
url1='https://google.com/search?q='
l=[]
for i in range(149):
    dis=df['Source'].loc[i]
    wordlist=dis.split(' ')
    url=url1
    for w in wordlist:
        if w!=' ':
            url=url+w+'+'
    url=url+'wikipedia'
    pages = requests.get(url)
    soup = BeautifulSoup(pages.content,'html.parser')
    url2=''
    for r in soup.find_all(class_='r'):
        newurl = list(r.children)[0]['href']
        if re.search("en.wikipedia.org",newurl):
            data = urlparse.parse_qs(urlparse.urlparse(newurl).query)
            url2 = data['q'][0]
            #print(url2)
            #url2 = newurl[re.search("https:",newurl).start():re.search("&sa",newurl).start()]
            break
    if url2 == '':
        continue
    pages2 = requests.get(url2)
    soup2 = BeautifulSoup(pages2.content,'html.parser')
    sentence = ''
    parent = soup2.find('div',class_='mw-parser-output')
    flag = 0
    count = 0
    for ch in list(parent.children):
        if type(parent) is type(ch):
            if ch.name == 'h2' and (re.search('prevention', ch.span.text, re.IGNORECASE) or re.search('treatment', ch.span.te
                flag = 1
                count = count +1
                #print(ch.text)
            elif ch.name == 'h2':
                flag = 0
        if flag == 1:
            if type(parent) is type(ch):
                sentence = sentence+"<"+ch.name+">"+ch.text+"</"+ch.name+">"
    innerl=[]
    innerl.append(df.iloc[i][1])
    innerl.append(sentence)
    l.append(innerl)
    print(i)
    if count == 0:
        notfound.append(i)
    sleep(2)
```

The code shown in the images above scrapes the treatments of the diseases present in our database from wikipedia.

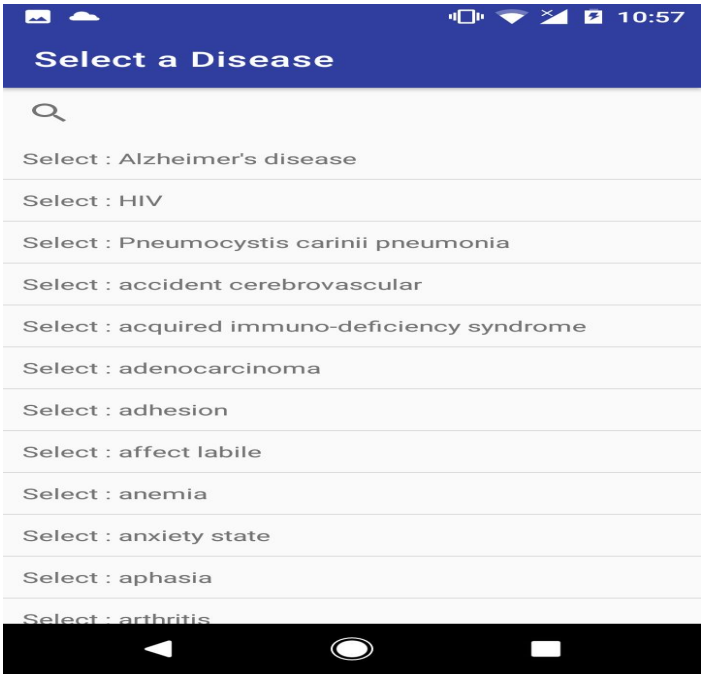


The above image shows a snippet of the code for the Android App.

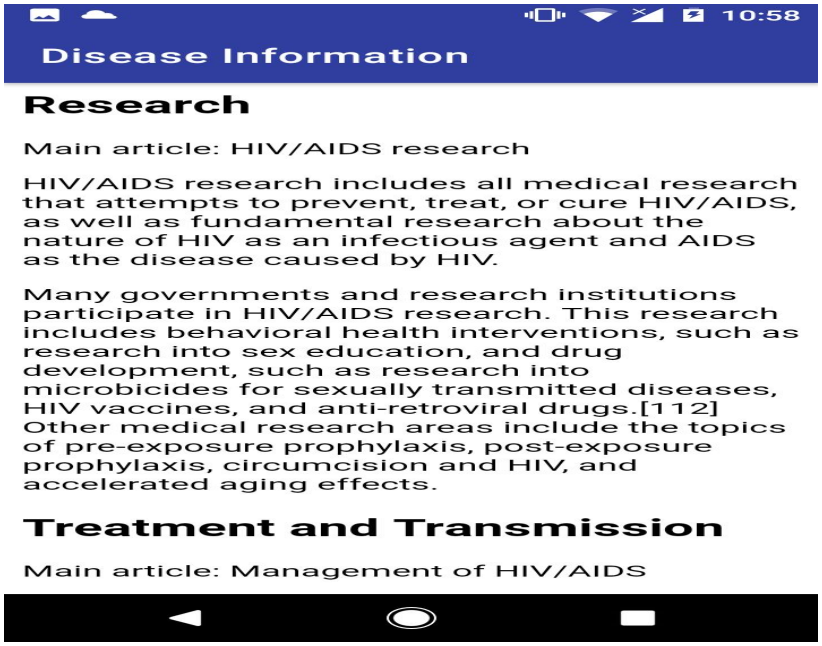
Screenshots:



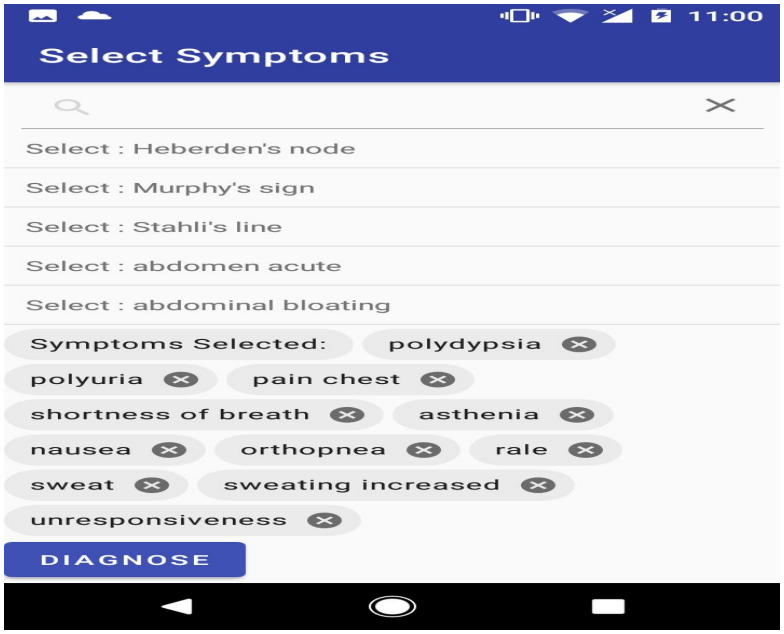
This image shows the first screen of the app.



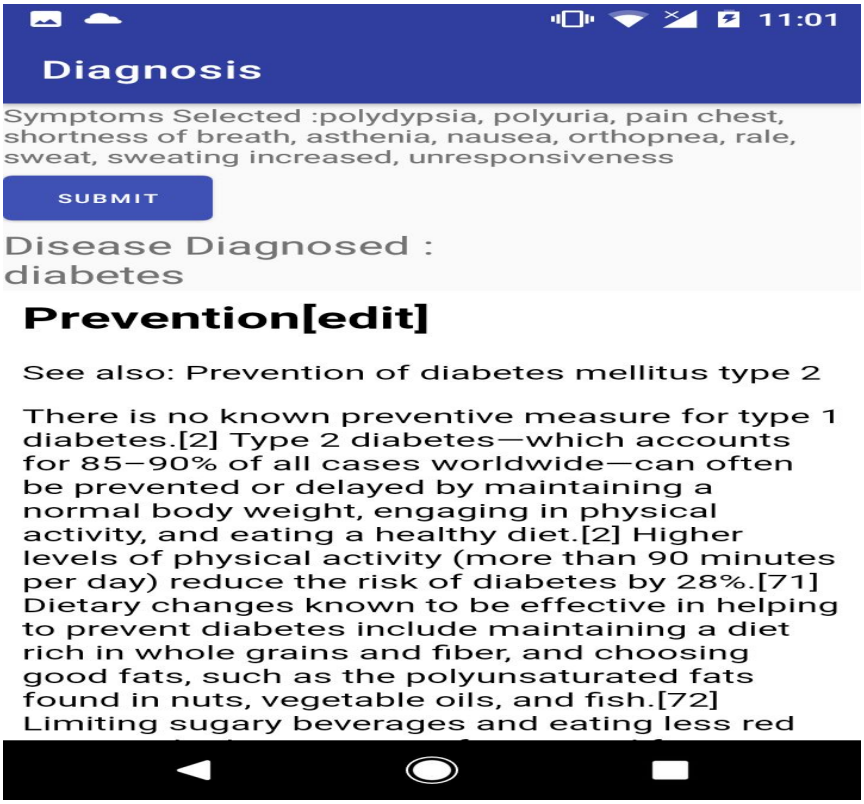
This is the screen displayed if the user clicks on Get Disease Information button.The user can select a disease he wants to know about.



This is the screen displayed if the user chose HIV disease.

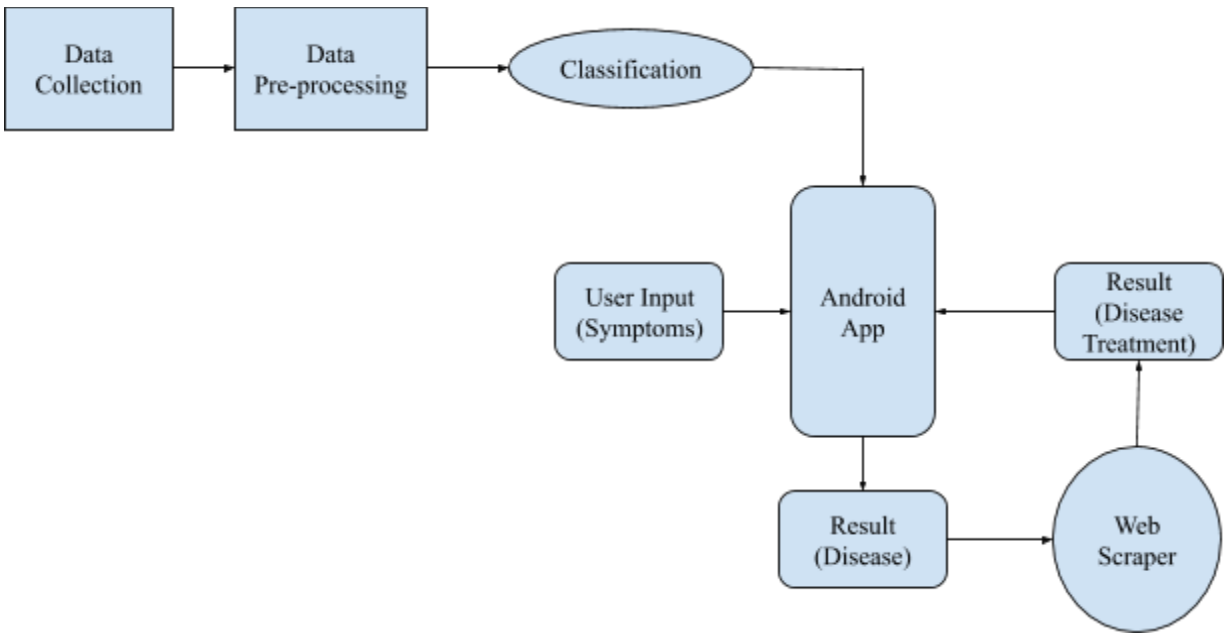


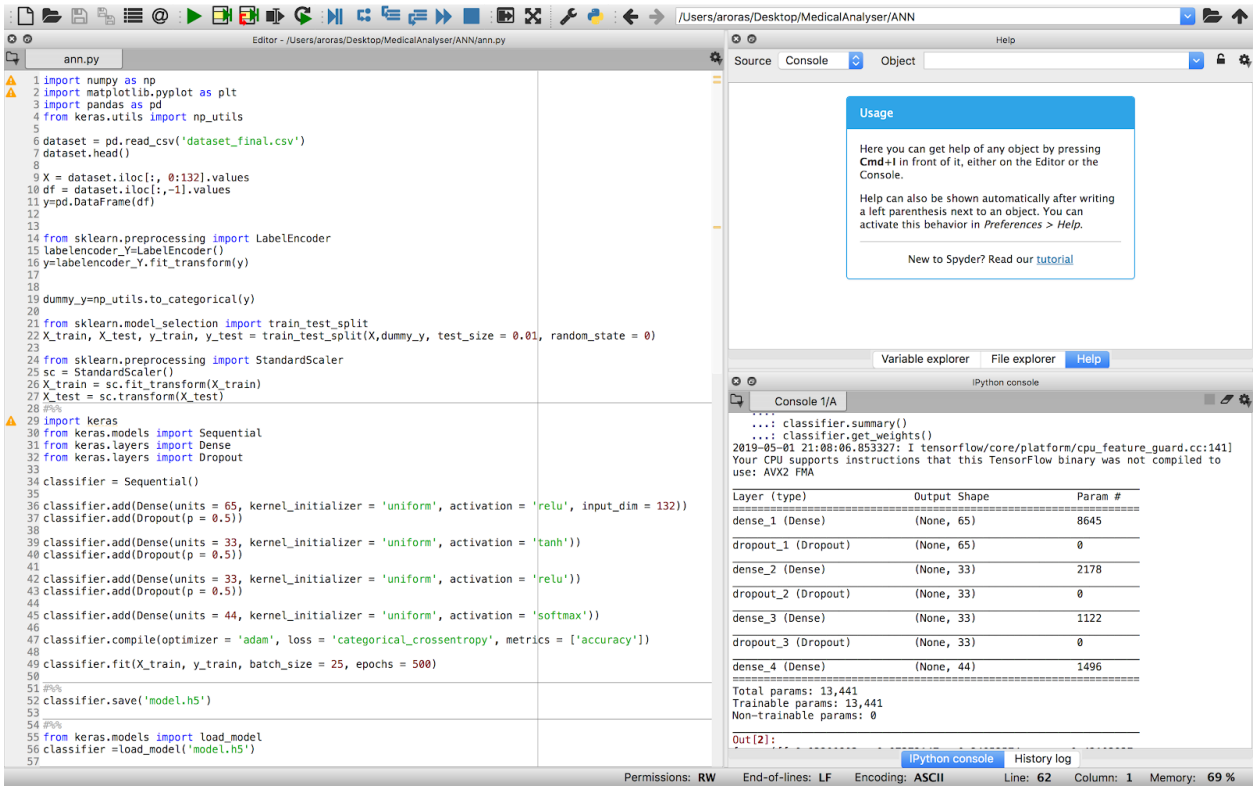
This is the screen displayed if the user clicked on Get Diagnosis button.It shows a selection of symptoms shown by a diabetic person.



This screen displays the correctly classified disease and its treatments.

For Model -2 :

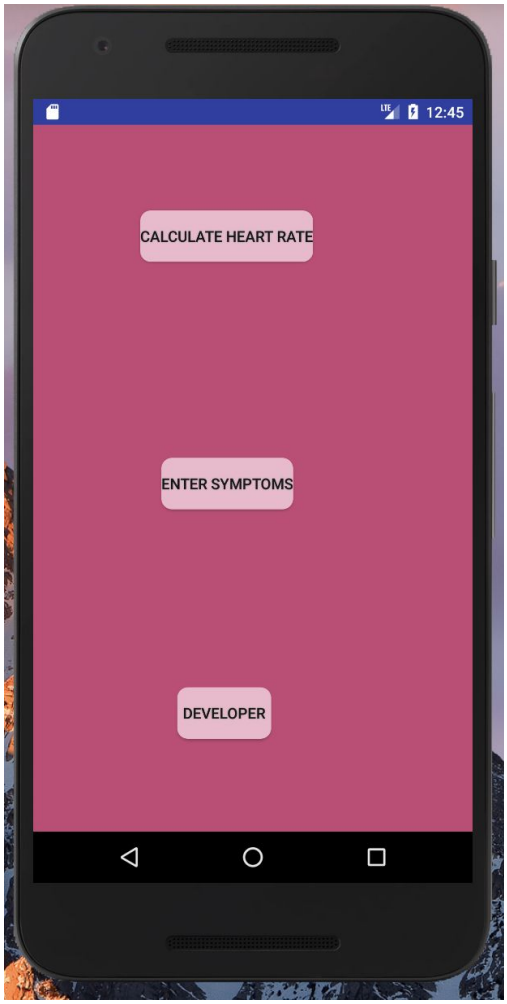


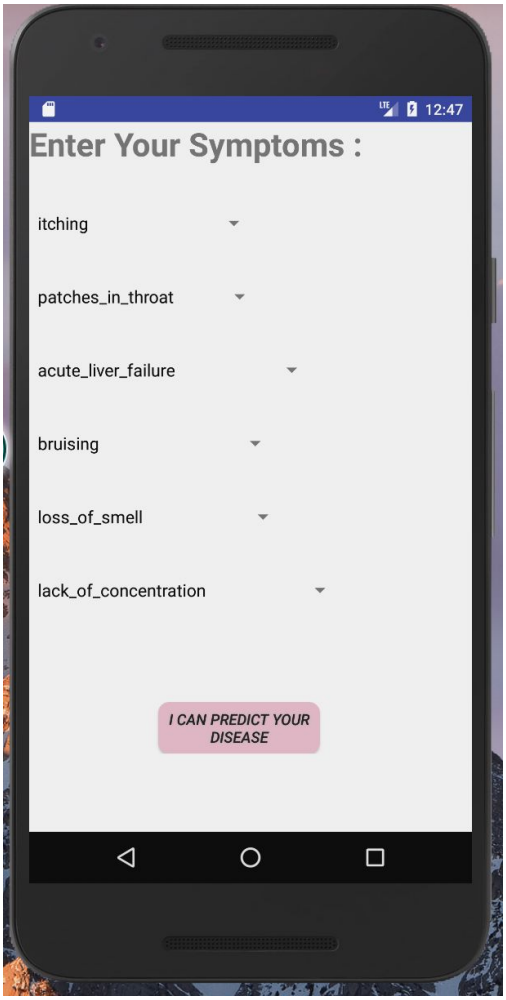


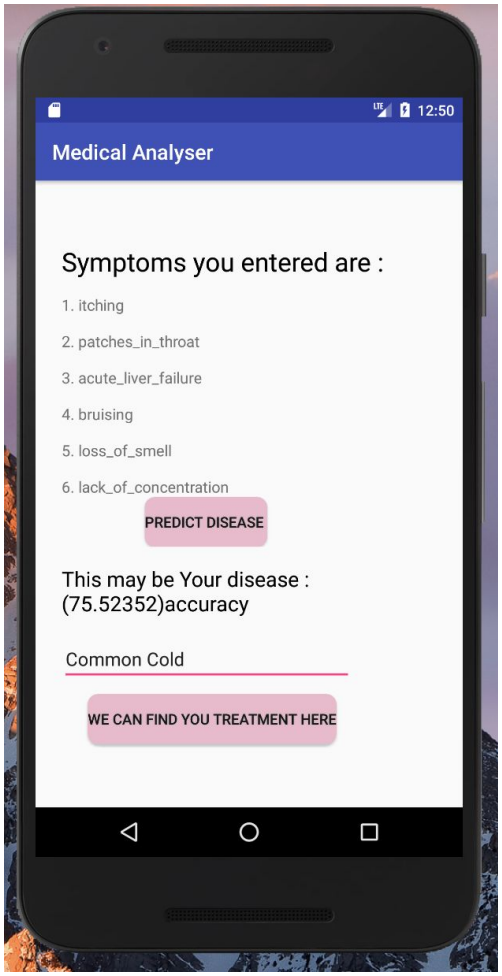
The above screen displays the code for creation and summary of the ANN model after training

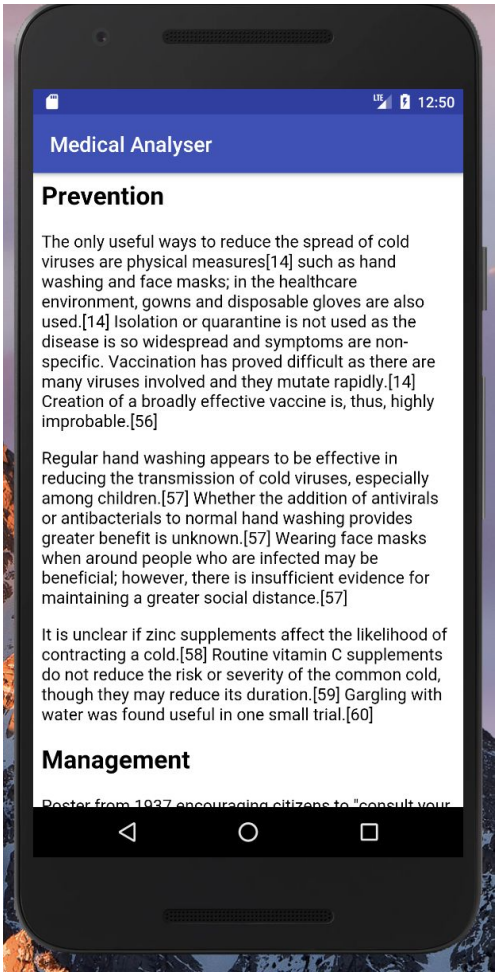
```
In [4]: y_pred = classifier.predict(X_test)
   ...:
   ...: y_pred = (y_pred > 0.5)
   ...:
   ...: from sklearn.metrics import confusion_matrix
   ...: from sklearn.metrics import accuracy_score
   ...:
   ...: cm=confusion_matrix(y_test.argmax(axis=1), y_pred.argmax(axis=1))
   ...: accuracy_score(y_test, y_pred)
Out[4]: 0.96

In [5]:
```

This snippet shows the Validation criteria of the model i.e. accuracy_score , which means 96% predictions from the model are correct

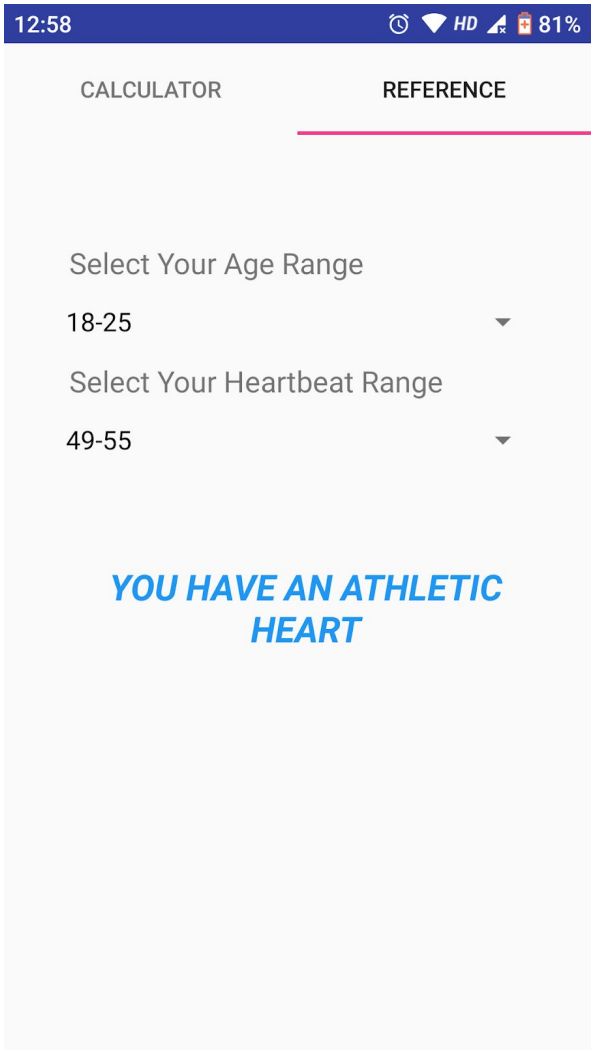

12:45
CALCULATE HEART RATE
ENTER SYMPTOMS
DEVELOPER


12:47
Enter Your Symptoms :
itching
patches_in_throat
acute_liver_failure
bruising
loss_of_smell
lack_of_concentration
I CAN PREDICT YOUR DISEASE

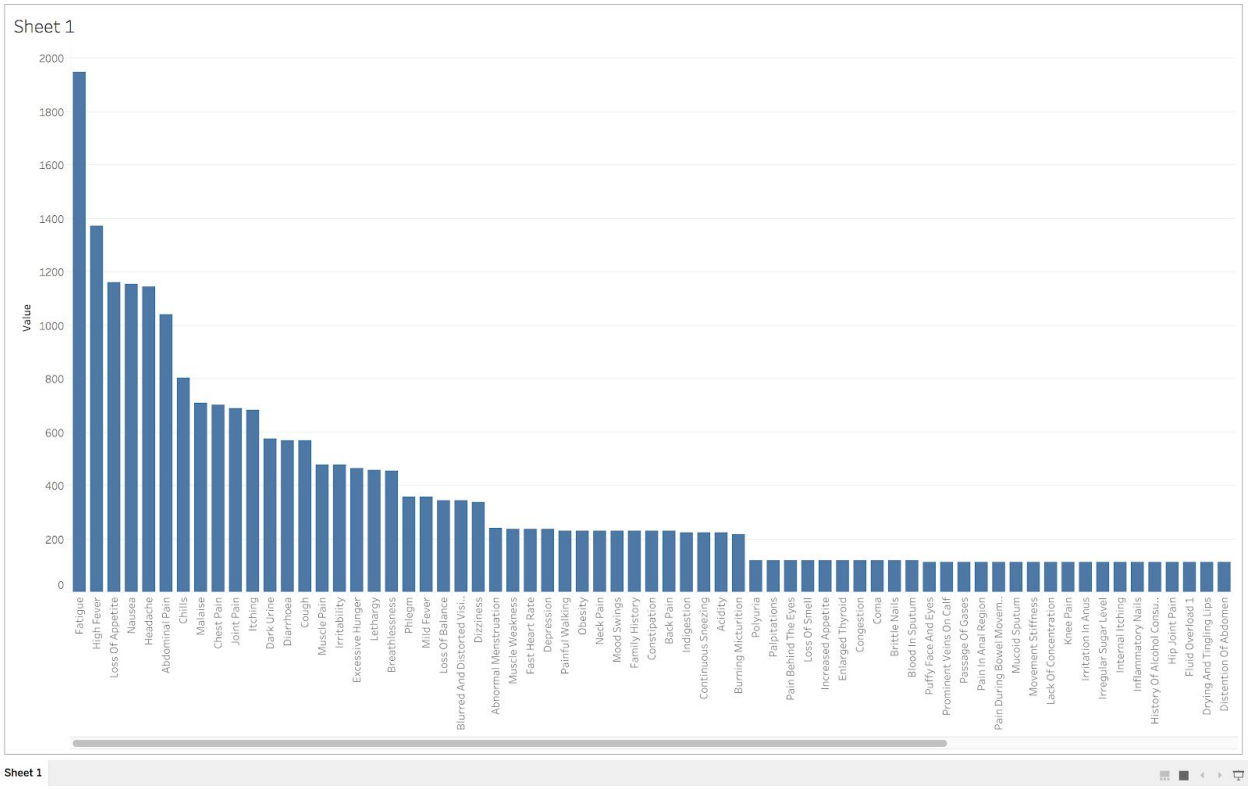




The above snippets shows the working of the app



The above images shows the working of the Heart Rate Calculator



This image shows the Visualization of the most prominent disease that is the cause of maximum number of diseases i.e. Fatigue in our dataset.

References:

[1]: Mani Shankar, Mayank Pahadia, Divyang Srivastava, Ashwin T S, G. Ram Mohana Reddy, A Novel Method for Disease Recognition and Cure Time Prediction Based On Symptoms

[2]: A. Piñango, and R. Dorado, A Bayesian Model for Disease Prediction Using Symptomatic Information

[3]:Imam M Shofi, Luh Kesuma Wardhani, and Ghina Anisa, Android Application for Diagnosing General Symptoms of Disease Using Forward Chaining Method

[4]Sellappan Palaniappan, Rafiah Awang ,Intelligent Heart Disease Prediction System Using Data Mining Techniques

[5]Dharavath Ramesh, Pranshu Suraj, and Lokendra Saini, Big data Analytics in Healthcare: A Survey Approach

[6]Min Chen, Yixue Hao, Kai Hwang, Lu Wang, and Lin Wang,Disease Prediction by Machine Learning Over Big Data From Healthcare Communities

[7] Prabakaran.N , and Kannadasan.R, Prediction of Cardiac Disease Based on Patient's Symptoms

[8]Aditi Gavhane, Gouthami Kokkula ,Isha Pandya ,Kailas Devadkar ,Prediction of Heart Disease Using Machine Learning

[9]http://people.dbmi.columbia.edu/~friedma/Projects/DiseaseSymptomKB/index.html

[10]http://pythonanywhere.com