A Mini- Project Report

on

“Covid-19 infection Prediction based on questionnaire”

Submitted to the

Pune Institute of Computer Technology, Pune In partial fulfillment for the award of the Degree of Bachelor of Engineering

in

Information Technology

by

Shubham Darak	43211
Omkar Deshpande	43212
Piyush Agrawal	43202

Under the guidance of

Prof. R. R. Chhajed

Department Of Information Technology

Pune Institute of Computer Technology College of Engineering

Sr. No 27, Pune-Satara Road, Dhankawadi, Pune - 411 043.

2020-2021

CERTIFICATE

This is to certify that the project report entitled

Covid-19 infection Prediction based on questionnaire

Submitted by

Shubham Darak	43211
Omkar Deshpande	43212
Piyush Agrawal	43202

is a bonafide work carried out by them under the supervision of Prof. R. R.Chhajed and it is approved for the partial fulfillment of the requirement of **Computer Laboratory -X** for the award of the Degree of Bachelor of Engineering (Information Technology)

Prof. R. R. Chhajed
Lab Teacher
Department of Information Technology

Dr. A. M. Bagade
Head of Department
Department of Information Technology

Place : Pune
Date : 28/05/2021

ACKNOWLEDGEMENT

We are very grateful to our guide, Prof. R. R. Chhajed, Head of Department Dr. A. M. Bagade and our principal Dr. R. Sreemathy. They have been very supportive and have ensured that all facilities remained available for smooth progress of the project.

We would like to thank our professor and Prof. R. R. Chhajed for providing very valuable and timely suggestions and help.

We thank everyone who has helped and provided valuable suggestions for successfully creating a wonderful project.

Shubham Darak

Omkar Deshpande

Piyush Agrawal

ABSTRACT

The recent outbreak of the respiratory ailment COVID-19 caused by novel coronavirus SARS-Cov2 is a severe and urgent global concern. In the absence of effective treatments, the main containment strategy is to reduce the contagion by the isolation of infected individuals; however, isolation of unaffected individuals is highly undesirable. To help make rapid decisions on treatment and isolation needs, it would be useful to determine which features presented by suspected infection cases are the best predictors of a positive diagnosis. This can be done by analyzing patient characteristics, case trajectory, comorbidities, symptoms, diagnosis, and outcomes.

We have developed a model that employs supervised machine learning algorithms to identify the presentation features predicting COVID-19 disease diagnoses with high accuracy. We have implemented a machine learning model and integrated it with an android app which will ask questions to users and based on that it will predict the probability of infection. This prediction takes different features like age, gender, history of travel, etc under consideration. Along with that using a GPS sensor we are recognizing the city and taking it as an input parameter.

Keywords: LGBM model, GPS, Android app, questionnaire, Flask API, REST controller..

LIST OF FIGURES

LIST OF TOPIC

1..Introduction

There has recently been a rapid spread of the novel SARS-CoV2 coronavirus which gives rise to a respiratory disease COVID-19. The development and spread of the novel coronavirus causing COVID-19 has vastly outpaced the rate of vaccine and therapeutic development. Nevertheless, within weeks of the first observations of COVID-19 disease, the virus was isolated and characterized. One of the most significant SARS-CoV2 protein targets is a 3C-like protease for which the structure is already known. Much effort has been centered around re-purposing known clinically-tested drugs and virtual screening for possible targets using protein structure data.

In February 2020, the noted case fatality rate for COVID-19 in Wuhan, China, was 1.4%. However, accurate global estimates are far more challenging due to the vastly different response country to country. For example, in Italy during March 2020, it showed a case fatality rate of 7.2% . This may partly reflect the demographic differences between nations, with 23% of the Italian population being over 65. However, even when stratified by age, infection rates remain higher in Italians over 70 years of age compared to China. This highlights the critical need to have improved screening and prediction methods to stratify those at higher risk of infection in discrete populations in different Track changes on 39 nations. To this end, machine learning algorithms are ideally suited for improving patient stratification and can be widely and rapidly applied as needed during a pandemic.

We have developed a machine learning model which will predict the probability of a person getting infected from covid-19.We are taking a dataset having features such as age, sex, Cough, Shortness of breath, Fever, and others to make a prediction. We are using GPS as a sensor to get track of the city of the person using the app. In our app, the user will be asked several questions and based on the answers given by the user, a trained model will make predictions.

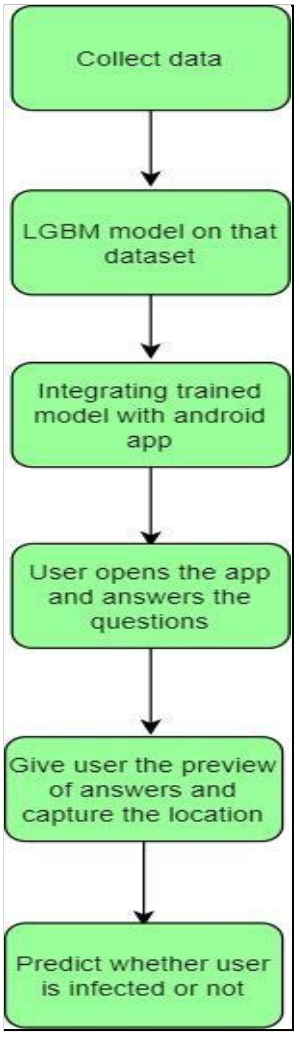
2. Scope and Objective:

- This app will help to make better and early predictions so that patients can be treated and get recovered.
- One can get a rough idea about his/her health without going out for a medical check up.
- Objective of this app is to make predictions based on questions and its answers.
- The App will collect data from questionnaires and send its responses to the ML model, which will in turn tell you Covid19 infection status.
- The scope of this project is currently limited to prediction based on answers of the user to the questions.

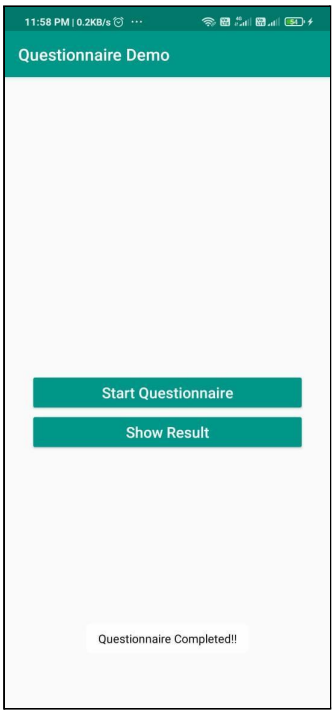
3. System architecture:

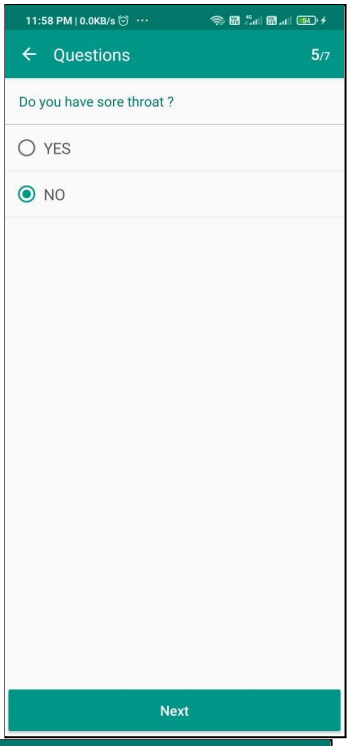
- The system is based on three tier architecture where we can assume three components as follows:
  - Model - the LGBM ML model for covid19 prediction.
  - View - The Android Application on which user interaction occurs, is the view layer in our architecture.
  - Controller - The Flask API on which the ML the model is hosted and will act as a Controller.

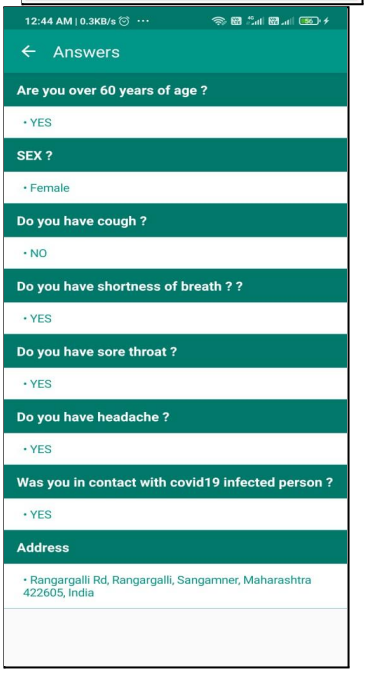
- The flow of the data will as shown in the adjacent figure.

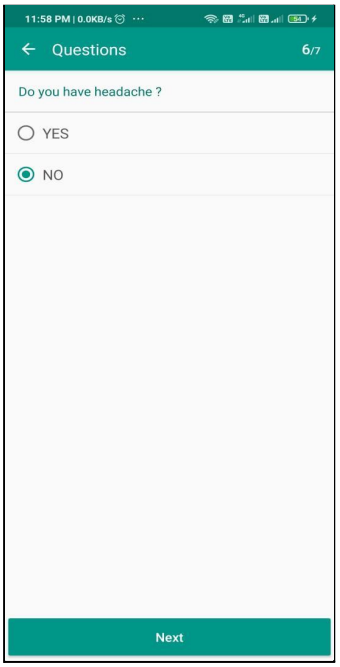


4.Snapshots:









*NOTE : THE LAST PAGE SHOWS THE LOCATION OF THE USER AT THE END OF THE LIST. THIS DATA IS COLLECTED AUTOMATICALLY BY THE GPS SENSOR. (the actual data format is {latitude, longitude}).

CODE :

MainActivity.java

```
package com.spk.questionnaire;

import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.location.LocationManager;
import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.provider.Settings;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import com.spk.questionnaire.questions.AnswersActivity;
import com.spk.questionnaire.questions.QuestionActivity;

import java.io.IOException;
import java.io.InputStream;

public class MainActivity extends AppCompatActivity
{
   private static final int QUESTIONNAIRE_REQUEST = 2018;
   Button resultButton;

   @Override
   protected void onCreate(Bundle savedInstanceState)
   {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);

       setUpToolbar();
       LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
       if (!locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {
           OnGPS();
       }
       Button questionnaireButton = findViewById(R.id.questionnaireButton);
       resultButton = findViewById(R.id.resultButton);

       questionnaireButton.setOnClickListener(v -> {
           resultButton.setVisibility(View.GONE);

           Intent questions = new Intent(MainActivity.this, QuestionActivity.class);
           //you have to pass as an extra the json string.
           questions.putExtra("json_questions",
loadQuestionnaireJson("questions_example.json"));
           startActivityForResult(questions, QUESTIONNAIRE_REQUEST);
       });
       resultButton.setOnClickListener(v -> {
           Intent questions = new Intent(MainActivity.this, AnswersActivity.class);
```

```
            startActivity(questions);
        });
    }
    private void OnGPS() {
        final AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage("Enable
GPS").setCancelable(false).setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                startActivity(new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));
            }
        }).setNegativeButton("No", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        });
        final AlertDialog alertDialog = builder.create();
        alertDialog.show();
    }
    void setUpToolbar()
    {
        Toolbar mainPageToolbar = findViewById(R.id.mainPageToolbar);
        setSupportActionBar(mainPageToolbar);
        getSupportActionBar().setTitle("Questionnaire Demo");
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data)
    {
        if (requestCode == QUESTIONNAIRE_REQUEST)
        {
            if (resultCode == RESULT_OK)
            {
                resultButton.setVisibility(View.VISIBLE);
                Toast.makeText(this, "Questionnaire Completed!!",
Toast.LENGTH_LONG).show();
            }
        }
    }

    //json stored in the assets folder. but you can get it from wherever you like.
    private String loadQuestionnaireJson(String filename)
    {
        try
        {
            InputStream is = getAssets().open(filename);
            int size = is.available();
            byte[] buffer = new byte[size];
            is.read(buffer);
            is.close();
            return new String(buffer, "UTF-8");
        } catch (IOException ex)
        {
            ex.printStackTrace();
            return null;
        }
    }
}
```

AnswerActivity.java

```
package com.spk.questionnaire.questions;

import android.Manifest;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Typeface;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.provider.Settings;
import android.util.TypedValue;
import android.view.View;
import android.view.ViewGroup;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonArrayRequest;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.google.gson.JsonArray;
import com.spk.questionnaire.R;
import com.spk.questionnaire.ResultActivity;
import com.spk.questionnaire.questions.database.AppDatabase;
import com.spk.questionnaire.questions.qdb.QuestionEntity;
import com.spk.questionnaire.questions.qdb.QuestionWithChoicesEntity;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Locale;
import java.util.Map;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import io.reactivex.Completable;
import io.reactivex.CompletableObserver;
```

```
import io.reactivex.android.schedulers.AndroidSchedulers;
import io.reactivex.disposables.Disposable;
import io.reactivex.schedulers.Schedulers;

public class AnswersActivity extends AppCompatActivity
{
   Context context;
   private static final int REQUEST_LOCATION = 1;
   LinearLayout resultLinearLayout;
   List<QuestionEntity> questionsList = new ArrayList<>();
   List<QuestionWithChoicesEntity> questionsWithAllChoicesList = new
ArrayList<>();
   private AppDatabase appDatabase;
   private String MyLat, MyLong;
   double[] d = new double[2];
   LocationManager locationManager;
   @Override
   protected void onCreate(Bundle savedInstanceState)
   {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_answers);

       context = this;
       appDatabase = AppDatabase.getAppDatabase(this);

       resultLinearLayout = findViewById(R.id.resultLinearLayout);
       toolBarInit();

       getResultFromDatabase();
   }

   private void toolBarInit()
   {
       Toolbar answerToolBar = findViewById(R.id.answerToolbar);
       answerToolBar.setNavigationIcon(R.drawable.ic_arrow_back);
       answerToolBar.setNavigationOnClickListener(v -> onBackPressed());
   }
   private void OnGPS() {
       final AlertDialog.Builder builder = new AlertDialog.Builder(this);
       builder.setMessage("Enable
GPS").setCancelable(false).setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
           @Override
           public void onClick(DialogInterface dialog, int which) {
               startActivity(new
Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));
           }
       }).setNegativeButton("No", new DialogInterface.OnClickListener() {
           @Override
           public void onClick(DialogInterface dialog, int which) {
               dialog.cancel();
           }
       });
       final AlertDialog alertDialog = builder.create();
       alertDialog.show();
   }
   private double[] getLocation() {
       if (ActivityCompat.checkSelfPermission(
               AnswersActivity.this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(
```

```
                AnswersActivity.this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, REQUEST_LOCATION);
        } else {

            Location locationGPS =
locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
            if (locationGPS != null) {
                double lat = locationGPS.getLatitude();
                double longi = locationGPS.getLongitude();
                return new double[]{lat , longi};
            } else {
                System.out.println("Unable to find location.");
                return new double[]{};
            }
        }
        return new double[]{};
    }
    /*After, getting all result you can/must delete the saved results
    although we are clearing the Tables as soon we start the
QuestionActivity.*/
    private void getResultFromDatabase()
    {
        Completable.fromAction(() -> {
            questionsList = appDatabase.getQuestionDao().getAllQuestions();
            questionsWithAllChoicesList =
appDatabase.getQuestionChoicesDao().getAllQuestionsWithChoices("1");
        }).subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(new CompletableObserver()
                {
                    @Override
                    public void onSubscribe(Disposable d)
                    {

                    }

                    @Override
                    public void onComplete()
                    {
                        makeJsonDataToMakeResultView();
                    }

                    @Override
                    public void onError(Throwable e)
                    {

                    }
                });
    }

    /*Here, JSON got created and send to make Result View as per Project
requirement.
    * Alternatively, in your case, you make Network-call to send the result to
back-end.*/
    private void makeJsonDataToMakeResultView()
    {

        try
```

```
       {
            JSONArray questionAndAnswerArray = new JSONArray();
            int questionsSize = questionsList.size();
            locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
            d = getLocation();
            System.out.println(d[0] + " " + d[1]);
            Geocoder geocoder = new Geocoder(this, Locale.getDefault());

            List<Address> addresses = geocoder.getFromLocation(d[0], d[1], 1);
            String cityName = addresses.get(0).getAddressLine(0);
            Toast.makeText(this, cityName, Toast.LENGTH_SHORT).show();
            if (questionsSize > 0)
            {
                for (int i = 0; i < questionsSize; i++)
                {
                    JSONObject questionName = new JSONObject();
                    questionName.put("question",
questionsList.get(i).getQuestion());
                    //questionName.put("question_id",
String.valueOf(questionsList.get(i).getQuestionId()));
                    String questionId =
String.valueOf(questionsList.get(i).getQuestionId());

                    JSONArray answerChoicesList = new JSONArray();
                    int selectedChoicesSize =
questionsWithAllChoicesList.size();
                    for (int k = 0; k < selectedChoicesSize; k++)
                    {
                        String questionIdOfChoice =
questionsWithAllChoicesList.get(k).getQuestionId();
                        if (questionId.equals(questionIdOfChoice))
                        {
                            JSONObject selectedChoice = new JSONObject();
                            selectedChoice.put("answer_choice",
questionsWithAllChoicesList.get(k).getAnswerChoice());

                            //selectedChoice.put("answer_id",
questionsWithAllChoicesList.get(k).getAnswerChoiceId());
                            answerChoicesList.put(selectedChoice);
                        }
                    }
                    questionName.put("selected_answer", answerChoicesList);

                    questionAndAnswerArray.put(questionName);
                }
                JSONObject questionName = new JSONObject();

                questionName.put("question", "Address");
                JSONArray answerChoicesList = new JSONArray();
                JSONObject selectedChoice = new JSONObject();
                selectedChoice.put("answer_choice", cityName);

                //selectedChoice.put("answer_id",
questionsWithAllChoicesList.get(k).getAnswerChoiceId());
                answerChoicesList.put(selectedChoice);
                questionName.put("selected_answer", answerChoicesList);
                questionAndAnswerArray.put(questionName);
            }

            questionsAnswerView(questionAndAnswerArray);
```

```
        } catch (JSONException | IOException e)
        {
            e.printStackTrace();
        }
    }



    private void questionsAnswerView(JSONArray questionsWithAnswerArray)
    {
        if (questionsWithAnswerArray.length() > 0)
        {
            try
            {
                for (int i = 0; i < questionsWithAnswerArray.length(); i++)
                {
                    String question =
questionsWithAnswerArray.getJSONObject(i).getString("question");

                    TextView questionTextView = new TextView(context);
                    questionTextView.setTextSize(TypedValue.COMPLEX_UNIT_SP,
16);

questionTextView.setTextColor(ContextCompat.getColor(context,
R.color.colorWhite));
                    questionTextView.setPadding(40, 30, 16, 30);

questionTextView.setBackgroundColor(ContextCompat.getColor(context,
R.color.colorPrimaryDark));
                    questionTextView.setLayoutParams(new
LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
                    questionTextView.setTypeface(null, Typeface.BOLD);
                    questionTextView.setText(question);

                    resultLinearLayout.addView(questionTextView);

                    JSONArray selectedAnswerJSONArray =
questionsWithAnswerArray.getJSONObject(i).getJSONArray("selected_answer");

                    for (int j = 0; j < selectedAnswerJSONArray.length(); j++)
                    {
                        String answer =
selectedAnswerJSONArray.getJSONObject(j).getString("answer_choice");
                        String formattedAnswer = "• " + answer; // alt + 7 -->
•

                        TextView answerTextView = new TextView(context);
                        answerTextView.setTextSize(TypedValue.COMPLEX_UNIT_SP,
14);

answerTextView.setTextColor(ContextCompat.getColor(context,
R.color.colorPrimary));
                        answerTextView.setPadding(60, 30, 16, 30);
                        answerTextView.setLayoutParams(new
LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.WRAP_CONTENT));

answerTextView.setBackgroundColor(ContextCompat.getColor(context,
R.color.colorWhite));
```

```
                        answerTextView.setText(formattedAnswer);

                        View view = new View(context);

view.setBackgroundColor(ContextCompat.getColor(context,
R.color.colorPrimary));
                        view.setLayoutParams(new
LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT, 1));

                        resultLinearLayout.addView(answerTextView);
                        resultLinearLayout.addView(view);
                    }
                }

            } catch (JSONException e)
            {
                e.printStackTrace();
            }

            RequestQueue queue = Volley.newRequestQueue(this);
            String url ="https://www.google.com";

// Request a string response from the provided URL.
            JsonArrayRequest stringRequest = new
JsonArrayRequest(Request.Method.POST, url, questionsWithAnswerArray,
                    new Response.Listener<JSONArray>() {
                        @Override
                        public void onResponse(JSONArray response) {
                            System.out.println(response.toString());

                            Intent intent = new Intent(AnswersActivity.this,
ResultActivity.class);
                            intent.putExtra("result", response.toString());
                            startActivity(intent);
                        }
                    }, new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {
                    System.out.println(error);
                }
            }){

            };

// Add the request to the RequestQueue.
            queue.add(stringRequest);
        }
    }
}
```

ML MODEL :

```
# split the dataset into the training set and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
```

```
[45] import lightgbm as lgb
clf = lgb.LGBMClassifier(num_leaves= 20, min_data_in_leaf= 4, feature_fraction= 0.2, bagging_fraction= 0.8, bagging_freq= 5, learning_rate= 0.05,verbose=1)
clf.fit(X_train, y_train)
```

```
LGBMClassifier(bagging_fraction=0.8, bagging_freq=5, boosting_type='gbdt',
               class_weight=None, colsample_bytree=1.0, feature_fraction=0.2,
               importance_type='split', learning_rate=0.05, max_depth=-1,
               min_child_samples=20, min_child_weight=0.001, min_data_in_leaf=4,
               min_split_gain=0.0, n_estimators=100, n_jobs=-1, num_leaves=20,
               objective=None, random_state=None, reg_alpha=0.0, reg_lambda=0.0,
               silent=True, subsample=1.0, subsample_for_bin=200000,
               subsample_freq=0, verbose=1)
```

FLASK API :

```
# flask_ngrok_example.py
from flask import Flask
from flask_ngrok import run_with_ngrok
from flask import request, jsonify
from collections import defaultdict
import json

answer_dict = {'YES':1,'NO':0,'Male':1,'Female':0}
app = Flask(__name__)
run_with_ngrok(app)  # Start ngrok when app is run

@app.route("/",methods=['GET', 'POST'])
def hello():
    data = request.json
    print(data)
    features = []

    for obj in data:
      if obj.get('question') != 'Address':
        features.append(answer_dict[obj['selected_answer'][0]['answer_choice']])
    answer = clf.predict([features])
    print(answer)
    return json.dumps([{'answer':str(answer[0])}])

if __name__ == '__main__':
    app.run()
```

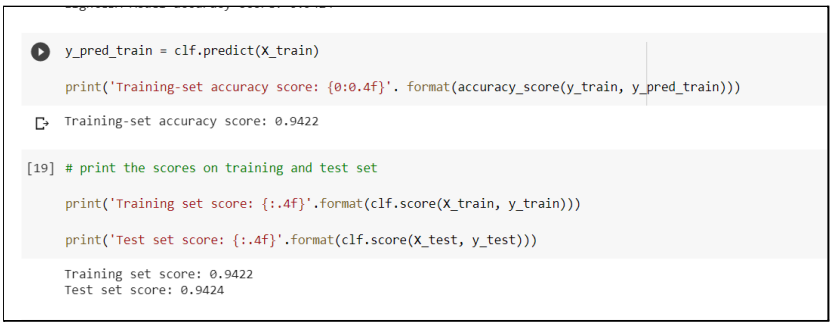
5. Result:

In this project, we have used the LGBM model which gave us an accuracy around

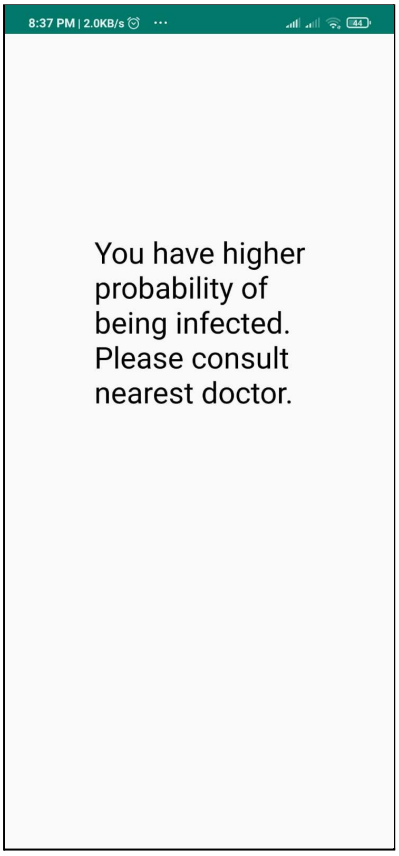
94%.

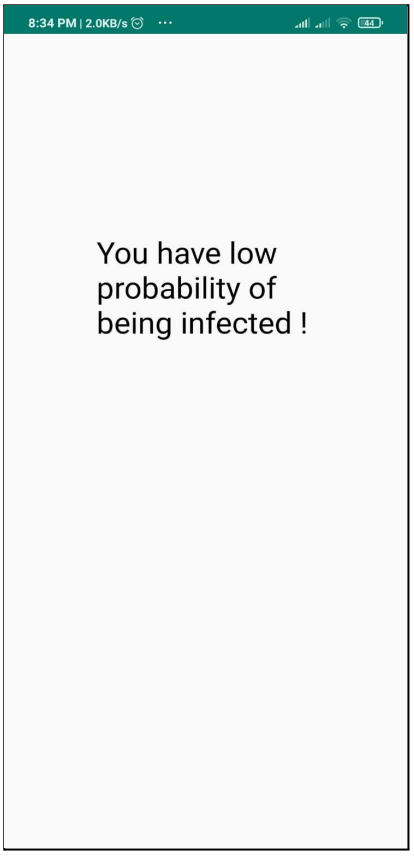
After asking questions to the user, a preview of answers is given to the user and then we capture the current location(city) of the user.

We were able to predict successfully whether a person is infected or not based on the answers received from the user.



```
y_pred_train = clf.predict(X_train)

print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pred_train)))

Training-set accuracy score: 0.9422

[19] # print the scores on training and test set

print('Training set score: {:.4f}'.format(clf.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(clf.score(X_test, y_test)))

Training set score: 0.9422
Test set score: 0.9424
```





6. Conclusion

Fast and timely detection of COVID +ve patients is necessary to avoid spreading the disease and keeping it in control. This work has been done to detect the COVID +ve patients from question answer patterns in a simple and inexpensive way. The proposed model has achieved a classification accuracy of 94 %.It is believed that this work along with the GUI interface will help the doctors to detect the affected patients with the help of computer-aided analysis, that too within a few seconds. We believe that this will significantly add value to the medical field.

7. Limitations

This app heavily relies on the honesty of the user. It is assumed that the user is trustworthy and will enter the answers genuinely. If not, the results may deviate significantly from the actual path.

8. Future Scope:

- Right now, questions have been asked to users and based on that predictions are made, but we can include X-ray, heartbeat sensor to make a better prediction.
- Model can be optimized to get better accuracy.

9. References:

1. Md. Martuza Ahamad, Sakifa Aktar, Md. Rashed-Al-Mahfuz, Shahadat Uddin,Pietro

Liò, Haoming Xu, Matthew A. Summers,Julian M.W. Quinn, and Mohammad Ali Moni, “ A machine learning model to identify early stage symptoms of SARS-Cov2 infected patinets”.

2.Ashish U Mandayam,Rakshith A.C, S Siddesha ,S K Niranjan, “Prediction of Covid-19 pandemic based on regression” Nov 2020.

3.Areej A.wahab Ahmed Musleh,Ashraf Yunis Maghari,” Covid-19 detection in X-ray images using CNN algorithm” Dec 2020