A CONTEXT-AWARE AND GEO-BASED MOBILE APPLICATION TO AUTOMATE THE NOTIFICATION OF PUBLIC HEALTH ISSUES USING BIG DATA ANALYSIS

Angela Xiang[1] and Yu Sun[2]

[1]Portola High School, Irvine, CA 92602
angelaxiang29@gmail.com
[2]California State Polytechnic University, Pomona, CA, 91768
yusun@cpp.edu

ABSTRACT

Coronavirus disease 2019 (COVID-19) is causing an ongoing pandemic. Social distancing and quarantine are the few effective methods to reduce the spreading risk of the coronavirus among people. As business starts to open up and quarantine policies become looser, the risk of COVID-19 spreading becomes greater [1]. This paper describes the development of a computer application to track the user's surroundings and calculate the exposure risk at a specific geographic location. Our application uses other user's data and online databases with information on COVID-19 cases to calculate a percentage revealing the user's possible risk at that location.

KEYWORDS

Flutter, python Flask, Firebase, iOS, Android

1. INTRODUCTION

Coronaviruses are a group of RNA viruses that usually cause infections in the respiratory tracts [2]. Most coronaviruses lead to mild illnesses, such as the common cold. However, some coronaviruses can be fatal. The SARS-CoV-2 virus that caused an outbreak in China quickly spread around the world in early 2020. The virus spreads primarily through person-to-person contact. This virus causes the COVID-19 disease, which leads to a respiratory tract infection and can affect the sinuses, nose, throat, windpipe, and lungs [3]. Within the span of a few months, the SARS-CoV-2 virus spread to 215 countries, causing many of these countries to enforce strict isolation policies [4]. As these policies loosen up and daily life resumes, the risk of SARS-CoV-2 spreading becomes greater. This application will assist users in enforcing social distancing measures by calculating a user's risk of contracting the virus based on their location and surroundings. Users will be notified when they are in high-risk areas, and using this application, they can seek areas with a lower risk level. Providing information on the risk level associated with a certain location may slow the spread of SARS-CoV-2 since users can find out the risk level based on their location at all times.

Some of the existing systems that have been proposed to slow the spread of SARS-CoV-2 are social distancing and isolation policies. Such policies have been enforced around the world to slow the spread of SARS-CoV-2. Since this virus spreads primarily through person-to-person contact, isolating people in their own homes and maintaining a 6-feet distance while in public helped to slow the spread of the virus [5]. However, as businesses begin to open up and isolation policies begin to loosen, the risk for SARS-CoV-2 exposure may increase again.

Another system proposed to slow the spread is facial masks, protective equipment, and frequent hand-washing. The idea is that these protective measures can prevent droplets from traveling and potentially spreading the virus into new hosts [6]. Facial masks that cover the nose and mouth would be particularly effective since they shield the nose and mouth from bacteria and viruses in droplets. Hand-washing is also very important to slow the spread of the virus. Washing hands for at least 20 seconds can help to prevent the spread of germs from the hands to other people [7]. Ideally, 100% of the population would wear masks and protective gear, but in reality, many people do not wear masks [6]. This exposes more people to the virus. The Institute of Health Metrics and Evaluation suggests that around 33,000 deaths can be avoided by October if 95% of people wear a mask [6]. Hospitals are also keeping medical records of patients infected with SARS-CoV-2. These medical records are used by public health workers to trace infections up the chain of command to learn how the disease spread [5]. This method, known as contact tracing, allows professionals to control the spread of disease. However, these medical records are not available to the general public, and cannot help people to identify when and where they may be at risk.

Our proposed method is a tracking application that provides a personalized real-time geo-based risk of potential COVID-19 exposure with context-aware features. The application uses location data to gather information about the user's surroundings, such as potential hot spots and large gatherings, as well as user data to find nearby users with diseases. When the user's position changes, the application recalculates a new risk percentage according to the new environment, providing the user with real-time updates calculated specifically for that user. Our application also provides the user with the risk at nearby locations, allowing users to scout for potentially safer locations if necessary. Many existing methods utilize data from medical records or confirmed COVID-19 cases, however some of this data is not available to the general public and is only updated every day or so. Our application uses geo-location and self-reported health data to predict and calculate a risk for the user. This allows our app to rapidly generate a real-time risk that is updated whenever and wherever the user moves to a new location. Our application is also widely available to the general public, and anyone can use it anywhere at any time. Therefore, we believe that this application could help users to potentially decrease their exposure to COVID-19 by warning users of the potential risk at their location.

In the case where geo-location data is limited or not available, we demonstrate how the combination of user data and a machine-learning algorithm predicts the user's risk. We experimented with different models, parameters, and data sets to produce the most accurate model. By conducting these experiments, we have found the machine learning model algorithm that predicts the risk with the highest accuracy. We also created a machine learning algorithm to predict the risk based on users most frequently visited locations. For this algorithm, we also experimented with different models, parameters, and data sets to produce the model with the highest accuracy. These experiments produced more accurate machine learning algorithms so that in the case where the application does not have sufficient data, it can predict a risk. Since our application primarily uses geo-location and self-reported user health data, these machine learning models fix the issue of not having sufficient data. Especially for locations where there is limited geo-data and when there are limited users, these experiments increase the accuracy of the risk algorithm. Increasing the accuracy of our risk calculation means that users will better understand their surroundings and be more aware of their risks. By providing more accurate warnings and risks, users can take more social distancing precautions or avoid high risk areas by finding and staying in locations with lower risk. This can assist users in potentially decreasing their exposure to COVID-19.

The rest of the paper is organized as follows: Section 2 discusses the challenges that people face when exercising social distancing. Section 3 describes our solution to the aforementioned challenges in Section 2. Section 4 presents the experiments we did, and Section 5 compares our

application to other SARS-CoV-2 tracking applications. Finally, Section 6 gives the conclusion remarks, as well as pointing out the future work of this project.

2. CHALLENGES

There are many challenges with minimizing the spread of COVID-19 and other dangerous diseases. Most of the current methods to reduce the spread rely on social distancing precautions and CDC data, however these methods do not provide people with real-time analysis of their surroundings. This application offers a personalized geo-based real-time context-aware risk analysis of the potential diseases that the user is exposed to.

2.1. Challenge 1: How to enable a personalized geo-based real-time analysis?

One challenge in developing the application is enabling a personalized geo-based real-time analysis. Since the goal of the application is to help users to minimize their potential exposure to fatal diseases such as COVID-19, providing a real-time analysis based on the user's position would achieve this goal. Using this application, users are able to see their risk of potential exposure to COVID-19 and other dangerous diseases. This risk is personalized for the user, taking into account their location, surroundings, and health status of nearby people. When one of these factors change, the risk is re-calculated, providing a real-time analysis that is up-to-date. Users can see their risk at any location and at any time, and if their surroundings have a high risk, then they can search for a safer area to stay.

2.2. Challenge 2: How to integrate context-aware types of risk features?

The second challenge is that the application uses context-aware features to calculate the risk. This context-awareness may help to increase the precision of the risk by taking the user's surroundings into account. Using the user's position, the application searches the surroundings for possible events or hot spots, such as restaurants, tourist attractions, and malls. If there are events or hot spots nearby, then the risk algorithm will factor that into the risk percentage. This allows the risk to calculate a more precise risk analysis by better understanding the surroundings. With a more precise risk analysis, the user can better understand the surroundings and find a relatively safe area to stay in.

2.3. Challenge 3: How to predict and provide the information for the areas that do not have sufficient data available?

The third challenge is providing a risk analysis for areas that do not have sufficient data available. For some areas, there could be little data on the COVID-19 spread rate, possible hot spots, and health status of nearby users. For areas with little data, the application uses a machine learning algorithm to predict and provide the information necessary to calculating a risk. The algorithm uses input data on other locations, including the latitude, longitude, number of hot spots nearby, and number of events occurring nearby. The algorithm also uses the output data for these locations: the calculated risk. These input and output data of other locations allows the algorithm to build a model for the relationship between the input (position, number of hot spots, number of events) and output (risk). Using the model, the machine learning algorithm then predicts a risk for the location that does not have sufficient data.

3. SOLUTION

This application provides a personalized geo-based real-time risk analysis of exposure to dangerous diseases such as SARS-CoV-2 based on the user's location and surroundings. It functions with three main components: the frontend UI, backend server, and database.

The user first interacts with the UI of the application. The frontend of the application consists of the visualization, format, pictures, textboxes, etc. The user first signs in or if they already created an account, they can log in. Once the user is logged in, the user can pick locations on a Google Map and see the COVID-19 risk at those locations.

The Google Maps on the frontend of the application is connected to a backend server. This component of the application contains algorithms and HTTP APIs to find the user's frequently visited locations, calculate a risk analysis, and search for the risk at other locations.

The backend server is also connected to a database that stores users' data. Storing latitude, longitude, name, age, email, etc., the database provides the necessary data for the server to calculate a real-time geo-based risk analysis. The database is also connected to the frontend of the application. In storing the login data for each user, it assists in the login and signup process in the UI.

This application provides a personalized geo-based real-time risk analysis of exposure to dangerous diseases such as SARS-CoV-2 based on the user's location and surroundings.

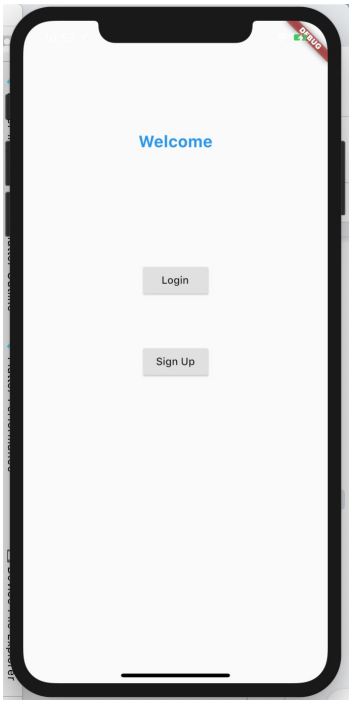


Figure 1: Welcome Page

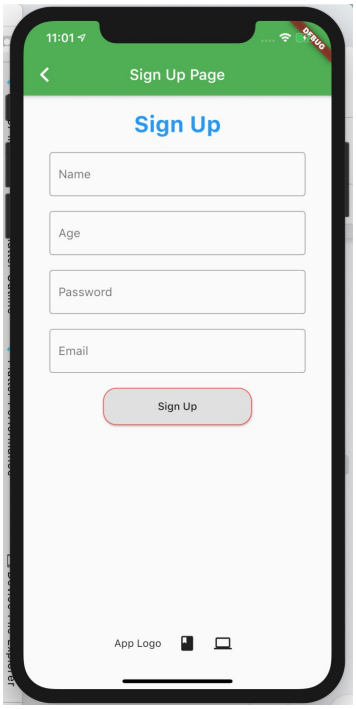


Figure 2: Sign Up Page

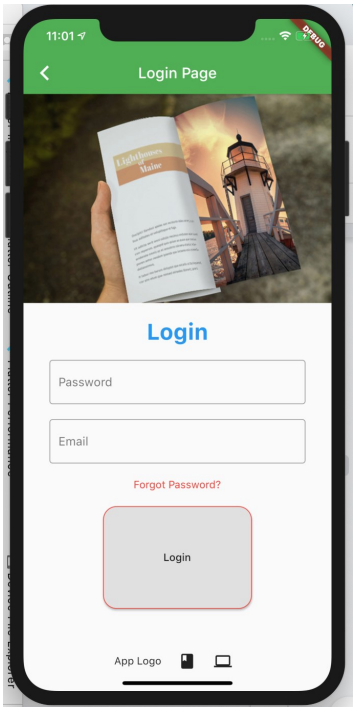


Figure 3: Login Page

The frontend of the application was developed using Flutter, an open-source UI development kit used to create both the Android and iOS versions of the application. The frontend consists of the visualization, format, pictures, textboxes, etc. When the application is initially opened, the user is directed to a welcome page that gives the option to either Sign Up or Login (Figure 1). If the user has not created an account yet, the Sign-Up button allows users to create an account and get started with the application. Clicking the Sign-Up button directs users to another screen that allows them to input their name, age, email, and password, creating a new user in the system (Figure 2). For returning users, the Login button allows access to the existing account, where the user's data is logged. Clicking the Login button takes users to a Login Page (Figure 3), where the user can input their email and password to access their account. After signing up or

logging into the account, the user can access three pages through the bottom navigation bar. The first page is the “My Locations” Page, where users can see a Google Map with pins that displays the user’s commonly visited locations (Figure 4). Clicking on each pin will reveal a textbox that shows the location’s name, description, as well as risk factor calculated by the app. The second page is the “Nearby” Page, which shows users a Google Map centered around their current location with pins displaying the surrounding locations with a high potential risk factor, such as restaurants, schools, shopping malls, tourist attractions, etc (Figure 5). Similar to the “My Locations” Page, clicking on the pins will open a textbox revealing the name, type of hot spot, and risk factor of the location. The third page is the “Profile” Page, where users can see their profile information, such as the name, age, and diseases they may have (Figure 6). On this page, users also have the option of editing their disease information by turning on a switch and entering the diseases that they have.

The backend server of the application was created using a Python Flask server to create 4 main HTTP APIs. Python Flask is a micro web framework, which routes an HTTP request to the specified controller. This calls a function in the server, and an HTTP response is then returned. The “My Locations” and “Nearby” Pages use this Python Flask server to find nearby users, find nearby hot spots, retrieve a user’s most visited locations, and calculate a risk percentage for the user. Within the Python Flask server, there are four main HTTP APIs that carry out these tasks. The first HTTP API is called getNearbyUsers, and it does the task of getting nearby users. This API uses three parameters, which are the user’s current latitude position, longitude position, and a given radius to search for nearby users. The function then loads all users by retrieving the data from the Firebase database. Using the Haversine formula, which calculates the distance between two points on a sphere, the function then returns the users that are located within the specified radius of the user.

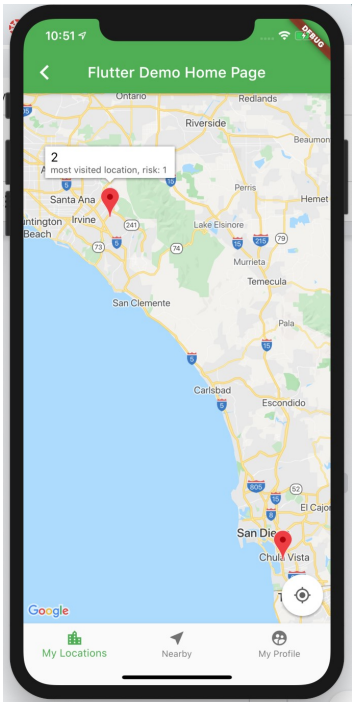


Figure 4: My Locations Page

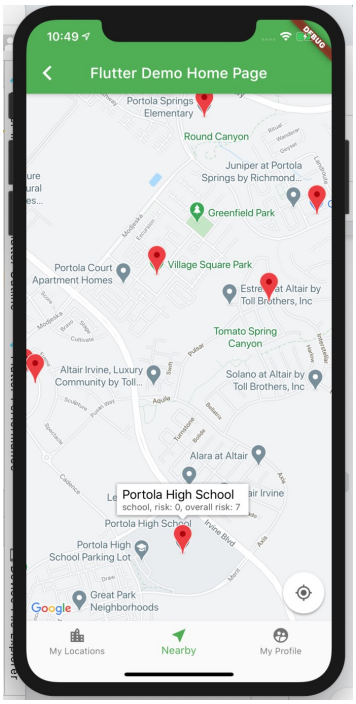


Figure 5: Nearby Page

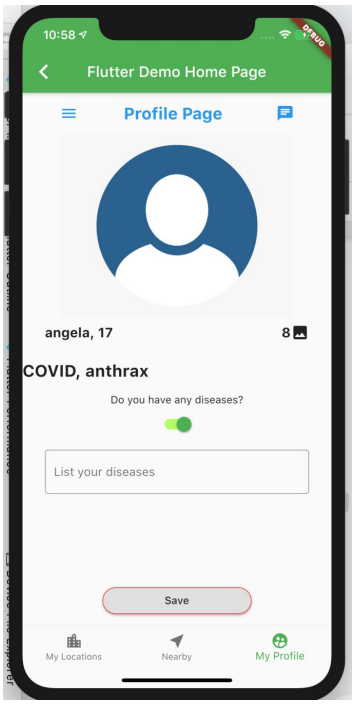


Figure 6: Profile Page

The second HTTP API is called getNearbyHotPlacesWithAllTypes, and it gets the nearby hot spots (Figure 8). This function uses three parameters: the user’s current latitude, current

longitude, and a specified radius. The function first specifies the types of hot spots, such as bars, restaurants, tourist attractions, malls, etc. The function then loads the hot spots that are located within the specified radius around the user's location using the Google Maps API. Finally, the function returns the hot spots, as well as the risk at that hot spot.

The third HTTP API is called getMostVisitedLocations, and it gets the user's most frequently visited places (Figure 9). This function loads the user's data from the Firebase database, and it counts the number of times the user visited each location. Counting the number of visits, the function returns the locations with the highest number of visits.

The last API, get_location_risk, calculates the risk for each location (Figure 10). It works by taking the current location from a user, and it checks if another user with a disease is within a specified radius of the user. For every user nearby that has diseases, the risk increases by one. The function then returns the risk for the user's current position.

This application uses a Firebase database to store users' login, profile, and location information. The Firebase Realtime Database stores and syncs user data from the application. Firebase stores this user data in JSON format, which consists of attribute-value pairs and array data types. Within the database, there are two main branches in which the data is sorted: users and logs. The users branch is further categorized into smaller branches for each user, which is labeled with a unique uid. Within these individual users' branches, the name, age, email, and diseases are stored. The other main branch, logs, tracks the user's location. Within the logs branch, there are also branches for each user labeled with the user's unique uid. These branches contain the latitude and longitude of the user at different time stamps, creating a log of all locations that the user has visited.

```
34  @app.route('/getNearbyUsers/<cur_lat>/<cur_lon>/<radius>')
35  def get_nearby_users(cur_lat, cur_lon, radius):
36      cur_lat = float(cur_lat)
37      cur_lon = float(cur_lon)
38      radius = float(radius)
39      print(cur_lat, cur_lon, radius)
40      # 1. get all the user data from the firebase database
41      # 2. return all the users near by based on a given radius
42      result = json.loads(
43          requests.get(
44              'https://coronavirus-app-1637b.firebaseio.com/users.json').text)
45      users_list = []
46
47      for key in result:
48          # print(key)            # key
49          # print(result[key])    # value
50          if 'longitude' in result[key] and 'latitude' in result[key]:
51              # print(result[key]['longitude'])
52              # print(result[key]['latitude'])
53
54              # filter out the users who's distance is out of the radius range
55              if distance(cur_lat, result[key]['latitude'], cur_lon,
56                          result[key]['longitude']) <= radius:
57                  users_list.append(result[key])
58
59      print(users_list)
60      return json.dumps(users_list)
```

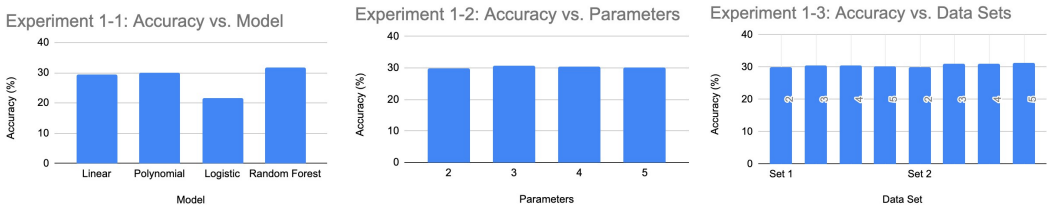
Figure 8: getNearbyHotPlacesWithAllTypes

```python
@app.route('/getNearbyHotPlacesWithAllTypes/<cur_lat>/<cur_lon>/<radius>')
def getNearbyHotPlacesWithAllTypes(cur_lat, cur_lon, radius):

    types = [
        'bar', 'restaurant', 'school', 'airport', 'university', 'hospital',
        'supermarket', 'store', 'shopping_mall', 'tourist_attraction'
    ]
    nearby_places = []

    # call the server to get all the list
    # and SAVE it here in the variable
    user_list = get_all_users_object()
    overall_risk = 0

    for type in types:
        res = requests.get(
            "https://maps.googleapis.com/maps/api/place/nearbysearch/json?location="
            + cur_lat + "," + cur_lon + "&radius=" + radius + "&type=" + type +
            "&key=AIzaSyCxFM696RNw-aqQmM67jA7-7LogJ9uBUt0")
        print(
            "https://maps.googleapis.com/maps/api/place/nearbysearch/json?location="
            + cur_lat + "," + cur_lon + "&radius=" + radius + "&type=" + type +
            "&key=AIzaSyCxFM696RNw-aqQmM67jA7-7LogJ9uBUt0")
        res_obj = json.loads(res.text)
        # print(res_obj['results'])
        list_places = res_obj['results']

        for place in list_places:
            # print(place['name'])
            # print(place['geometry']['location']['lat'])
            # print(place['geometry']['location']['lng'])
            loc_risk = get_location_risk(
                user_list, place['geometry']['location']['lat'],
                place['geometry']['location']['lng'], 1)
            new_place = {
                'name': place['name'],
                'type': type,
                'risk': loc_risk,
                'lat': place['geometry']['location']['lat'],
                'lng': place['geometry']['location']['lng']
            }
            nearby_places.append(new_place)
            overall_risk += loc_risk

    res = {'overall_risk': overall_risk, 'places': nearby_places}

    return json.dumps(res)
```

Figure 9: getMostVisitedLocations

4. EXPERIMENT

To evaluate the accuracy of the risk algorithm, we experimented with the different models, parameters, and data set features to find the most accurate machine learning model. The first experiment was conducted to find a machine learning model that predicts the risks for a new geo-location without available data. This experiment has three parts: experiment 1-1 tests different regression models, experiment 1-2 tests different polynomial parameters, and experiment 1-3 tests different data set features. The machine learning model was created with 5000 data sets for each experiment. The accuracy of each experiment was calculated using the machine learning algorithm and comparing the prediction with the actual risk. For experiment 1-1, different machine learning models were applied to the same data set to find out which model would produce the most accurate algorithm. The models used are linear, polynomial with a power of 2, logistic, and random forest regressions. Experiment 1-2 tested the polynomial model parameters 2, 3, 4, and 5, and tested which model would have the highest accuracy. Experiment 1-3 tested two data sets: one set with four inputs (latitude, longitude, number of hotspots, and number of events) and the other set with two inputs (number of locations and number of events).

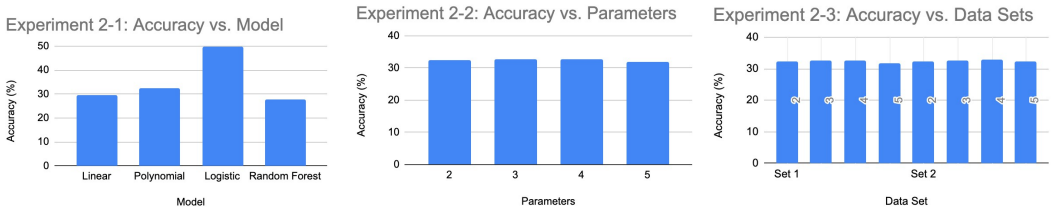


(a) (b) (c)

Figure 10: results of experiment 1

The results of experiment 1-1 can be seen in Figure 10. The Random Forest model has the highest accuracy at 31.772%. In comparison, the Logistic model has the lowest accuracy of the four models, with an accuracy of 21.6%. This proves that the geo-based risk does not follow a Logistic model. The results of experiment 1-2 reveals that 3 parameters in the polynomial model produces a higher accuracy at 30.568%, but not by much. The models with 2, 4, and 5 parameters have similar accuracies to the model with three parameters, with the least accurate model (2 parameters) being less than 1% away from the most accurate (3 parameters). As mentioned previously, experiment 1-3 compares two data sets with different inputs and different parameters. The most accurate model was produced using Data Set 2 and 5 parameters, and on average, Data Set 2 produced more accurate models than Data Set 1.

The second experiment was conducted to find a machine learning model that predicts the risks for a user based on the most-visited locations. This experiment has three parts: experiment 2-1 tests different regression models, experiment 2-2 tests different polynomial parameters, and experiment 2-3 tests different data set features. The machine learning model was created with 5000 data sets for each experiment. The accuracy of each experiment was calculated using the machine learning algorithm and comparing the prediction with the actual risk. For experiment 2-1, different machine learning models were applied to the same data set to find out which model would produce the most accurate algorithm. The models used are linear, polynomial with a power of 2, logistic, and random forest regressions. Experiment 2-2 tested the polynomial model parameters 2, 3, 4, and 5, and tested which model would have the highest accuracy. Experiment 2-3 tested two data sets: one set with four inputs (type of location, number of nearby hot spots, number of visits, and the duration of each visit) and the other set with three inputs (type of location, number of hotspots, and duration of each visit).



(a) (b) (c)

Figure 11: results of experiment 2

The results of experiment 2-1 can be seen in Figure 14. The Logistic model has the highest accuracy at 49.8%. In comparison, the Random Forest model has the lowest accuracy of the four models, with an accuracy of 27.721%. This proves that the geo-based risk does not follow a Random Forest model. The results of experiment 2-2 (Figure 11) reveals that 4 parameters in

the polynomial model produces a higher accuracy at 32.649%, but not by much. The models with 2, 3, and 5 parameters have similar accuracies to the model with four parameters, with the least accurate model (5 parameters) being less than 1% away from the most accurate (4 parameters). As mentioned previously, experiment 2-3 (Figure 16) compares two data sets with different inputs and different parameters. The most accurate model was produced using Data Set 2 and 4 parameters, and on average, Data Set 2 produced more accurate models than Data Set 1.

Experiment 1 tried to find the most accurate model for predicting risks for new geo-locations without prior data. The most accurate algorithm was one with a Random Forest model, 3 parameters, and using Data Set 2 (a data set with 2 inputs, specifically the number of hot spots and number of locations). Experiment 1-1 proved that using a Random Forest model resulted in a higher accuracy, and a Logistic model had the lowest accuracy, revealing that the geo-location vs. risk does not follow a Logistic pattern. Experiment 1-2 found that a 3-parameter polynomial model had the highest accuracy, but overall, there was not a significant difference between a 2, 3, 4, or 5-parameter polynomial model. Experiment 1-3 found that the model using Data Set 2 and a 5-parameter polynomial model was the most accurate. This is interesting since in experiment 1-2, the 3-parameter model was the most accurate, not the 5-parameter model. This is perhaps due to the fact that in experiment 1-2, the models had very similar accuracies that only deviated by less than 1%. Experiment 1-3 also reveals that on average Data Set 2 had a higher accuracy, suggesting that the input of latitude and longitude in Data Set 1 made the risk algorithm less accurate. Since the risk algorithm did not use the latitude and longitude, it makes sense that Data Set 2 was more accurate. The latitude and longitude position does not directly relate to the risk at the geo-location, and the number of nearby hot spots and events have a greater impact on the risk.

Experiment 2 tried to find the most accurate model for predicting risks for users based on their most visited locations. The most accurate algorithm was one with a Logistic model, 4 parameters, and using Data Set 2 (a data set with 3 inputs, specifically the type of location, number of hotspots, and duration of each visit). Experiment 2-1 proved that using a Logistic model resulted in a higher accuracy, and a Random Forest model had the lowest accuracy, revealing that the geo-location vs. risk does not follow a Random Forest pattern. Experiment 2-2 found that a 4-parameter polynomial model had the highest accuracy, but overall, there was not a significant difference between a 2, 3, 4, or 5-parameter polynomial model. Experiment 2-3 found that the model using Data Set 2 and a 4-parameter polynomial model was the most accurate, although there is less than a 1% difference in accuracy between models trained with Data Set 1 and Data Set 2.

5. RELATED WORK

Coveley, M. et al [8] developed a tracking system for infectious diseases. This system consists of transmitter circuits that record the infected people within the confined space. The system works by identifying the people within the location, establishing a record for each person with a time and date, and storing these records [8]. These records can be retrieved to generate an audit record that can help in tracking the disease [8]. Since this tracking system relies on transmitter circuits spaced out over a confined location, it is designed for and best suited to be used in a hospital with medical professionals and potential carriers of the disease inside. Our application also tracks users with diseases; however, it utilizes location data of nearby users to warn someone of potential exposure. This difference allows our application to be used anywhere and anytime as long as the user has a mobile device with them.

Segal, E. et al built an international consortium to track COVID-19 spread [9]. This system relies on participants to self-report COVID-19 symptoms. Participants will also report their geospatial location, time, age, demographic information, and pre-existing medical conditions

[9]. This data is used to help track COVID-19 around the world. This is very similar to my application, since both rely on self-reported data and a large group of users/participants. However, my application takes this one step further by using the self-reported data to calculate a personalized risk analysis for users. Not only does the risk analysis utilize other user's self-reported data, but it also uses geo-location data to find nearby hot spots and large events that could increase the risk of COVID-19.

Boulos, M. et al describes a range of online and mobile GIS mapping systems that track COVID-19 [10]. Some of these systems include Johns Hopkins' CSSE dashboard (Figure 12), World Health Organization's dashboard (Figure 13), and HealthMap (Figure 14). All of these systems track COVID-19 cases on a map, as well as figures, tables, and graphs showing the trends. The CSSE dashboard uses diagnosed cases based on symptom array and chest imagine, and the WHO dashboard uses laboratory-confirmed cases [10]. HealthMap uses machine learning and language processing to sift through news reports to track COVID-19 [10]. HealthMap also offers a feature to find "outbreaks near me" [10]. This is very similar to my application, that uses user and location data to find the risk and nearby users with diseases.

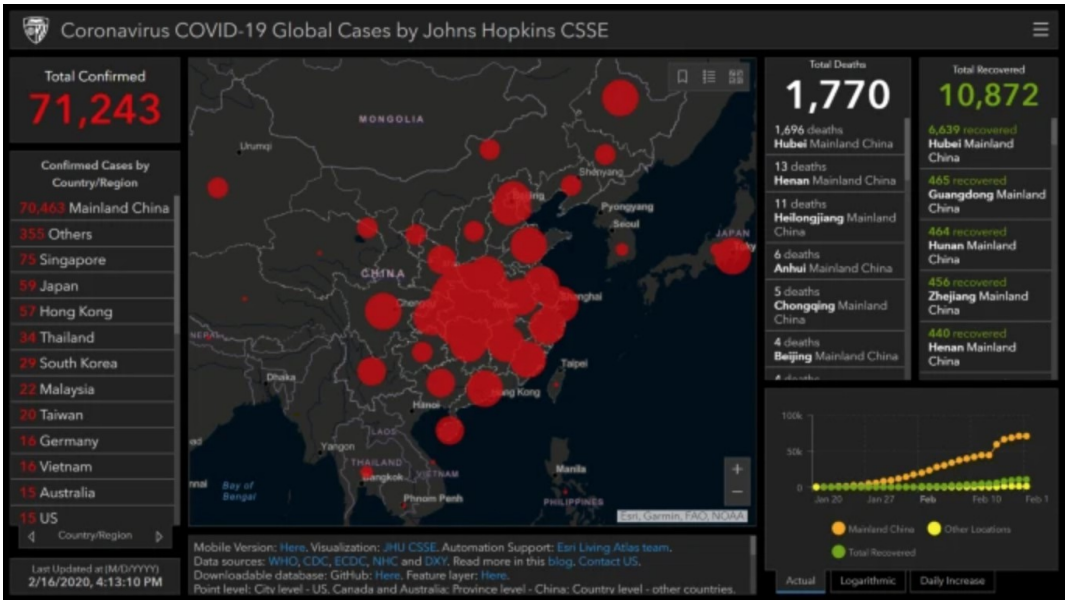


Figure 12: Johns Hopkins CSSE dashboard

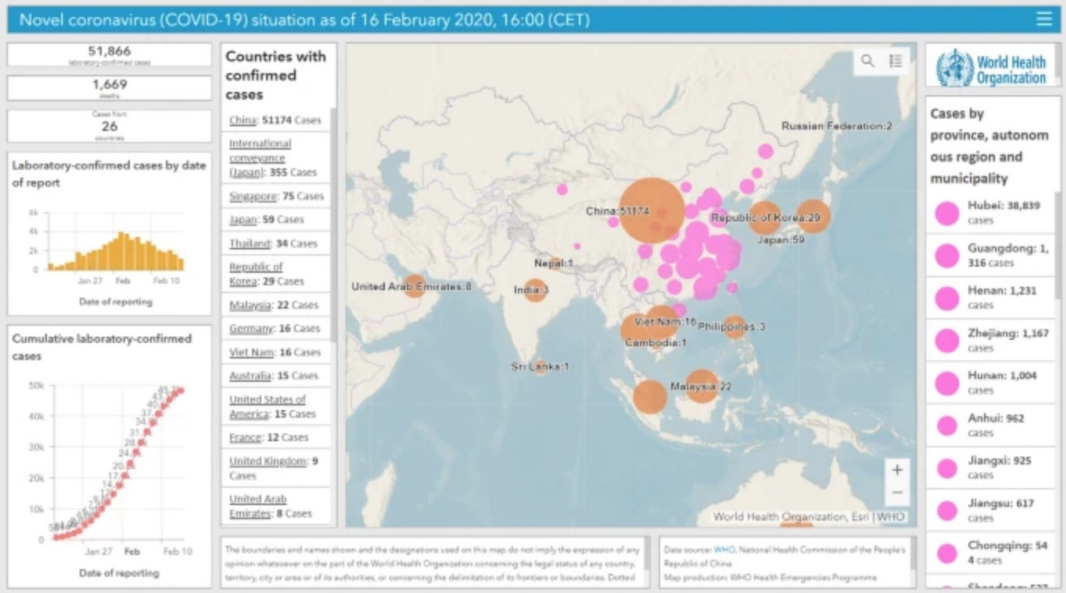


Figure 13: WHO dashboard

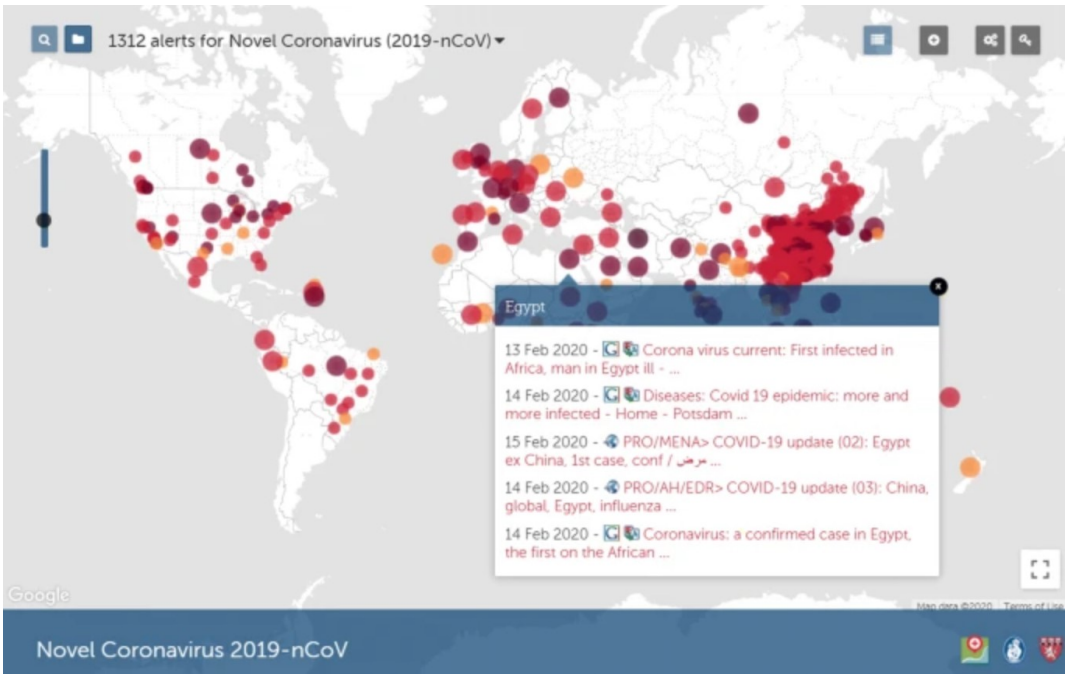


Figure 14: HealthMap

6. Conclusion and Future Work

In summary, my application tracks COVID-19 among its users and provides a personalized real-time geo-based risk analysis for its users. Using geo-location data and self-reported disease data from users, the application calculates a risk factor with the nearby users, hot spots, and events.

This method could assist in slowing the spread of COVID-19 by warning users of the risk at their location, and thus users can take more precautions in high-risk areas. Users are also able to see nearby locations that may have a smaller risk, allowing them to move to a relatively safer location. In cases where the geo-location of a user has no available data, the application would use a machine learning algorithm to predict the risk. Our experiments tried to find the most accurate algorithm by adjusting the regression model, polynomial parameter, and features of the input set. The results show that using a Random Forest model with 3 parameters trained with an input of [number of hotspots, number of events] will result in a more accurate model. The accuracy of these models are between 20-30%, which indicates that separately, these features produce a fairly accurate prediction. Even without prior geo-location data, the machine learning algorithm can predict a fairly accurate risk, solving the challenge of inability to calculate a risk without sufficient data. With the addition of the machine learning algorithm, this application is able to predict a risk at every position, regardless of existing geo-location data. This allows the application to be used by users around the world at every location.

However, there are some limitations in this application. First, the risk algorithm partially relies on self-reported disease data. If there are not enough users, then the risk might be underestimated, losing the risk algorithm's accuracy. Another limitation is that the machine learning algorithm may not be optimized. The experiment revealed that individually, the best features led to a higher accuracy. But when the best features are combined, the resulting algorithm may not have the highest accuracy. A third limitation is that the risk algorithm could be better optimized using more factors. Currently, the risk algorithm uses self-reported user disease data, nearby hot spots, and nearby large events. This algorithm could be further developed to be a more accurate reflection of the area.

Further experimentation should be implemented to solve these limitations. The risk algorithm could be further experimented with and developed so that it does not heavily rely on self-reported user data. Using more location data, such as the number of nearby laboratory-identified COVID-19 cases, could lessen the reliance on many users self-reporting data. Furthermore, conducting experiments that combine the best features could optimize the accuracy of the machine learning algorithm. Testing different combinations of features would lead to the best overall model. Lastly, the risk algorithm could also be further experimented with to find the most accurate way to calculate risk.

REFERENCES

[1] Johns Hopkins Coronavirus Resource Center (2020). Impact of Opening and losing Decisions by State. https://coronavirus.jhu.edu/data/state-timeline/new-confirmed-cases/california/1

[2] Nat'l. Inst. Allergy & Infectious Diseases (2020). Coronaviruses Overview, https://www.niaid.nih.gov/diseases-conditions/coronaviruses

[3] WebMD (2020). Coronavirus and COVID-19: What You Should Know, https://www.webmd.com/lung/coronavirus

[4] Coronavirus Update (Live). Worldometer Reported Cases and Deaths by Country, Territory, or Conveyance, https://www.worldometers.info/coronavirus/?utm_campaign=homeAdUOA?Si

[5] Centers for Disease Control and Prevention (2020). Monitoring and tracking the disease, https://www.cdc.gov/coronavirus/2019-ncov/cases-updates/about-epidemiology/monitoring-and-tracking.html

[6] Bai, N. (2020). Still Confused About Masks? Here's the Science Behind How Face Masks Prevent Coronavirus , University of California San Francisco, https://www.ucsf.edu/news/2020/06/417906/still-confused-about-masks-heres-science-behind-how-face-masks-prevent

[7] Centers for Disease Control and Prevention (2020). Follow Five Steps to Wash Your Hands the Right Way, https://www.cdc.gov/handwashing/when-how-handwashing.html.

[8] Coveley, M. and Huang, Y. (2010). Method and apparatus for cataloging and poling movement in an environment for purposes of tracking and/or containment of infectious diseases. US Patent No. US 7,817.046 B2. https://patents.google.com/patent/US7817046B2/en

[9] Segal, E., et al. (2020). Building an International Consortium for Tracking Coronavirus Health Status. Nat. Med. Vol. 26, 1161–1165. https://doi.org/10.1038/s41591-020-0929-x.

[10] Kamel Boulos, M.N. and Geraghty, E.M. Geographical tracking and mapping of coronavirus disease COVID-19/severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) epidemic and associated events around the world: how 21st century GIS technologies are supporting the global fight against outbreaks and epidemics. Int. J. Health Geogr. Vol.19, 8. https://doi.org/10.1186/s12942-020-00202-8.

[11] Oliver, N., et al. (2020). Mobile phone data for informing public health actions across the COVID-19 pandemic life cycle. Science Advances Vol. 6, eabc0764. DOI: 10.1126/sciadv.abc0764

[12] Motley, J., et al. (2020). Ethical guidelines for COVID-19 tracing apps. Nature, Vol. 582, 29 - 31. https://www.nature.com/articles/d41586-020-01578-0

[13] Drew, D., et al. (2020). Rapid implementation of mobile technology for real-time epidemiology of COVID-19. Science, Vol. 368,1362-1367. DOI: 10.1126/science.abc0473.

[14] Couture, V. et al. (2020). Measuring Movement and Social Contact with Smartphone Data: A Real-Time Application to COVID-19. NBER Working Paper No. 27560, https://www.nber.org/papers/w27560.pdf

[15] Ahmed, N et al. (2020). A Survey of COVID-19 Contact Tracing Apps, IEEE Access, Vol. 8,134577 - 134601, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9144194.