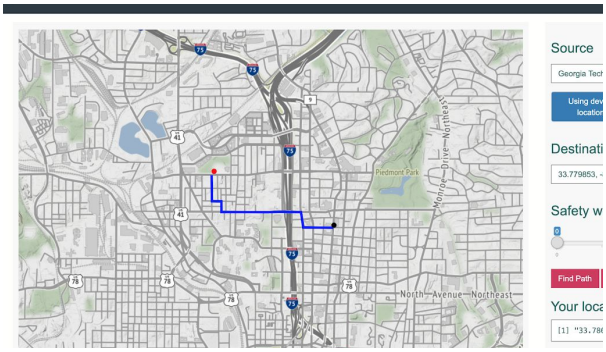
SPOT

Safe Path Optimization Tool

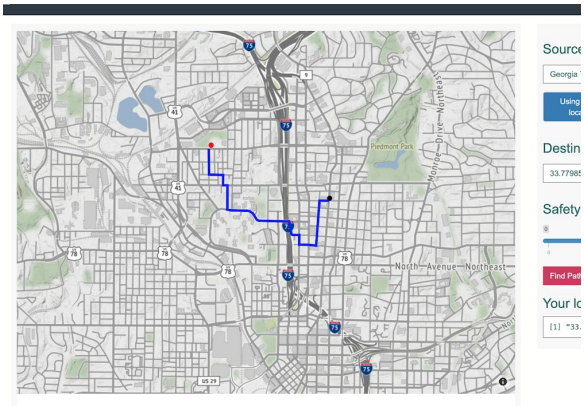
Introduction-Motivation:

Although being risky and dangerous, driving is an essential part of day-to-day life for most people. According to the report released by police, there are approximately 6.4 million accidents per year. In this project, we created a tool to help analyze and visualize the accident data. This enables end-users to take proactive measures to prevent accidents. The end-user can be anyone who commutes or the traffic authority itself. This helps a commuter by giving a real-time alert when he is close to an accident-prone region. Traffic authorities can use SPOT to analyse which are the areas with frequent accidents, when do accidents occur the most (hour/day/week/month), which states have the most accidents and their possible reasons. They can then change their patrolling accordingly, place signposts on the roads and change speed limits.

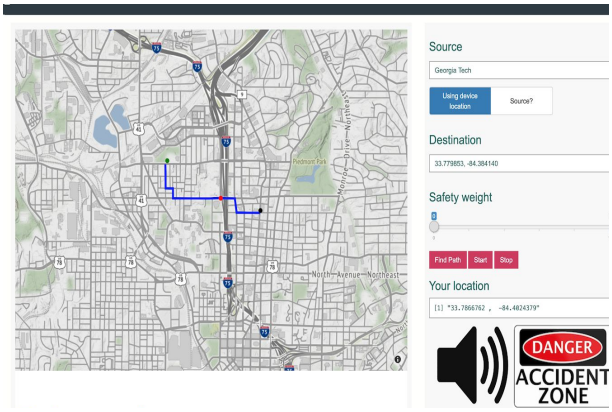
a) Map showing the accidents



b) Shortest path with safety weight set as 0



c) Safest path with full weight on safety. This path avoided the accident zones (circled in figure a) which was on shortest path



d) Once the driver reached the accident-prone area (close to highway intersection), alerts was generated

Figure 1) App screenshots showing different functionalities

Description of SPOT: Algorithm, user interface etc.

SPOT utilizes US accidents (2.25 Million rows) data from kaggle.com (Appendix 1) to cover three main areas: (1) the visualization and summary of historical accidents provides insights for proper preventive measures (Figure 1.a), (2) system generates the safest and shortest path between a source-destination pair (Figure 1.b) by using the Dijkstra's algorithm after updating the weights of edges taking number of accidents and severity into account. (3) While commuting, the user will receive an alert for the potential risk of an accident based on proximity to the accident-prone areas (Figure 1.c).

The closeness to an accident-prone area for generating alerts is calculated by estimating the probability of accident for the given coordinates using Kernel Density Estimation. Figure 2 shows an example of results generated by Kernel Density Estimation. Other external factors like rainfall, temperature, visibility, etc. can be used in the future to further improve the alert system.

1. **Path Finding Algorithm**
   a. I constructed the road network of a region from scratch using the OpenStreetMap API (OSM). A road is identified by list of coordinate points. The graph was constructed with nodes as the intersections of roads in the dataset. The edges in the network are the roads. I assigned weights to the edges based on the severity and number of accidents close to the nodes present on the edge. I used the following formula to get the path which minimizes the following cost:

$$D(s,d) = \lambda * d_{safety} + (1-\lambda) * d_{act}$$

Where, $D(s,d)$ is the final distance metric between source s and destination d, d_{safety} is the distance calculated using the number of accidents and d_{act} is the actual distance. λ is the weight assigned to d_{safety}

```
<node id="69370177">
  <data key="d4">33.7751189</data>
  <data key="d5">-84.3895323</data>
  <data key="d6">69370177</data>
</node>
```

Figure 2 a) Node '69370177' with latitude 33.7751189 and longitude of -84.3895323 stored as a graphml file

```
<edge source="69372783" target="69256234" id="0">
  <data key="d9">175629473</data>
  <data key="d12">True</data>
  <data key="d10">West Peachtree Street Northwest</data>
  <data key="d11">primary</data>
  <data key="d18">30 mph</data>
  <data key="d13">140.112</data>
  <data key="d15">LINESTRING (-84.387435 33.770011, -84.3874296 33.7704935, -84.38742240000001
  33.7711421, -84.387421 33.771271)</data>
  <data key="d14">44</data>
</edge>
```

Figure 2b) Edge connecting nodes '69372783' and '69256234' stored as a graphml file. This edge has number of accidents stored in key 'd14' and length in key 'd13'. 'D15' key stores the intermediate coordinates required to plot curves of the path

2. **Alert Generation:**
   a. While the driver is commuting, I used his location through his device's current location
   b. I then measured the risk of potential accidents by calculating the probability of accident at that location using fitted KDE. The bandwidth for KDE was selected as 0.001 using cross-validation.
   c. If the probability of accident is higher than a threshold, an audio and visual alert was generated, so that the user will be cautious near this area

3. **User Interface:**
   https://drive.google.com/file/d/16zXKFrugJQB2sC1hacSBrwUfmcx2CaX1/view?usp=sharing

   The SPOT app works with multiple tabs where each tab is designed for different aims. After the user closes the welcome screen, he moves to the dashboard where he can do exploratory analysis on the accident data, and find the major accident prone zones. This tab gives the flexibility to filter for particular years through a dropdown or select a particular state by clicking on the US choropleth map.

Figure 3) Welcome screen of SPOT

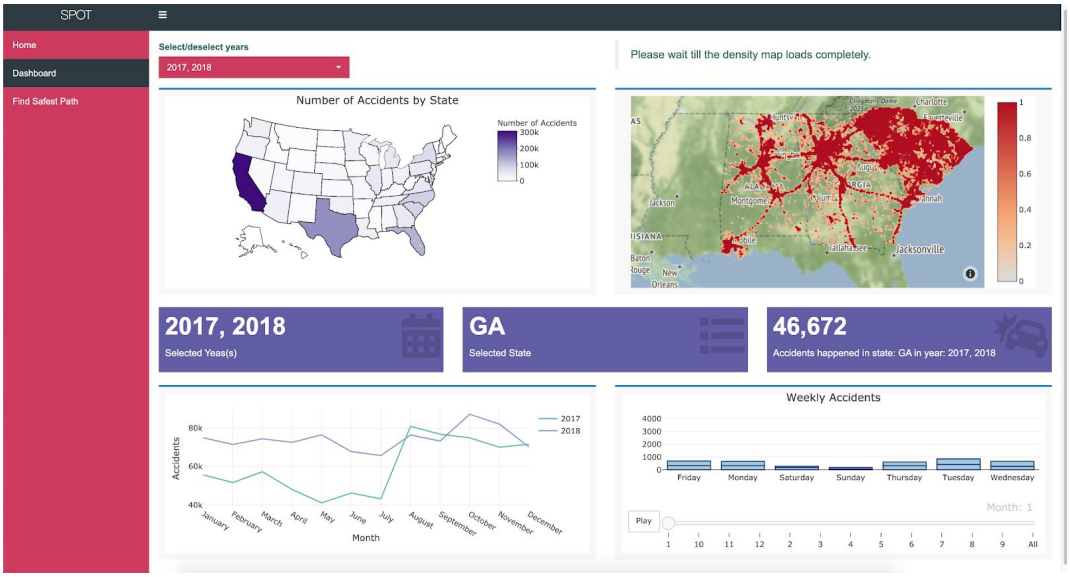


Figure 4) Dashboard tab showing different plots for the filtered state of Georgia and years 2017 and 2018

Next tab is dedicated for commuting purposes where the user can find his preferred path and will get alerted when he is close to the accident prone area. User can enter his source, destination and the weight he wants to give to safety while finding the path. He can also choose to use the device location by clicking on the ‘Source’ switch. User can either give the coordinates or type the name of places as source and destinations. By changing the weight to the safety user can get either the shortest, safest or something in between (Fig 5). If the plot is not getting updated after changing the weights, then that means that the path is the same for all the weights. This happens when very close locations are selected as source and destination and there are very less accidents between them.

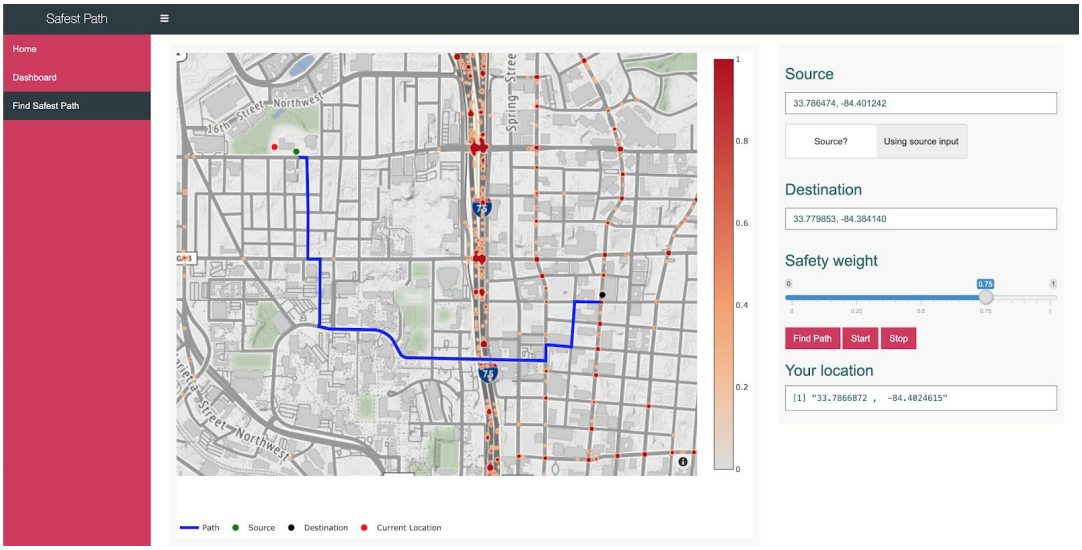


Figure 5) Next tab showing the path interface where user can select his source/destination and weights he want to give for safety

To give a demo of alert generation, simulation mechanism has been added in the app which can be tested by clicking on start after selecting the path. This will simulate the driving process on the selected path. Audio and visual alerts will be generated on close proximity to accident prone zone. This can be easily converted for use in real driving by changing two lines in the source code.

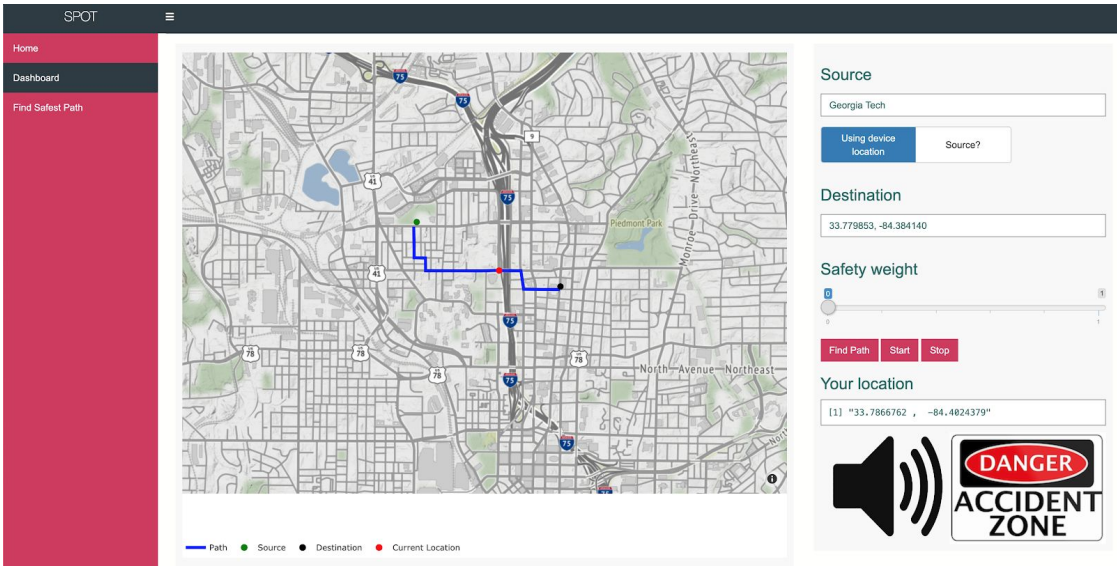


Figure 6) Audio and visual alerts were generated when driver (marked with a red dot) reached near the highway (dense accident zone)

Observations

To give the best suited route to user two factors were combined- distance and safety. Change in path was easily seen as the weight assigned to each factor changed (Figure 9-12)

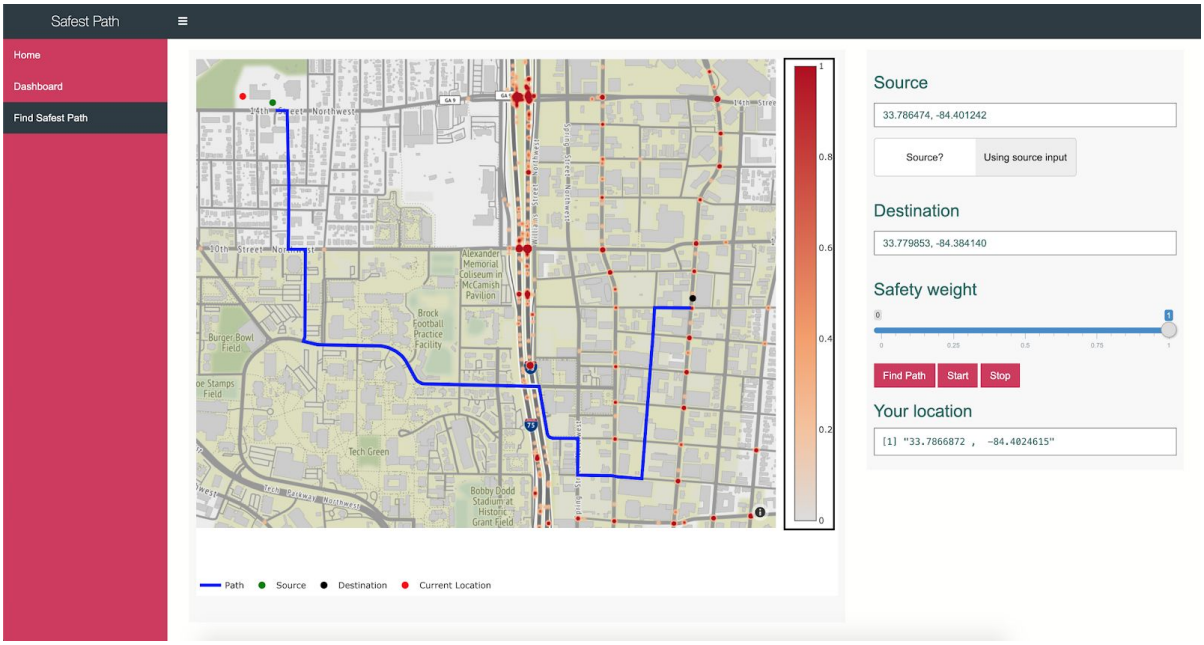


Figure 9) Map showing the safest route with safety weight = 1

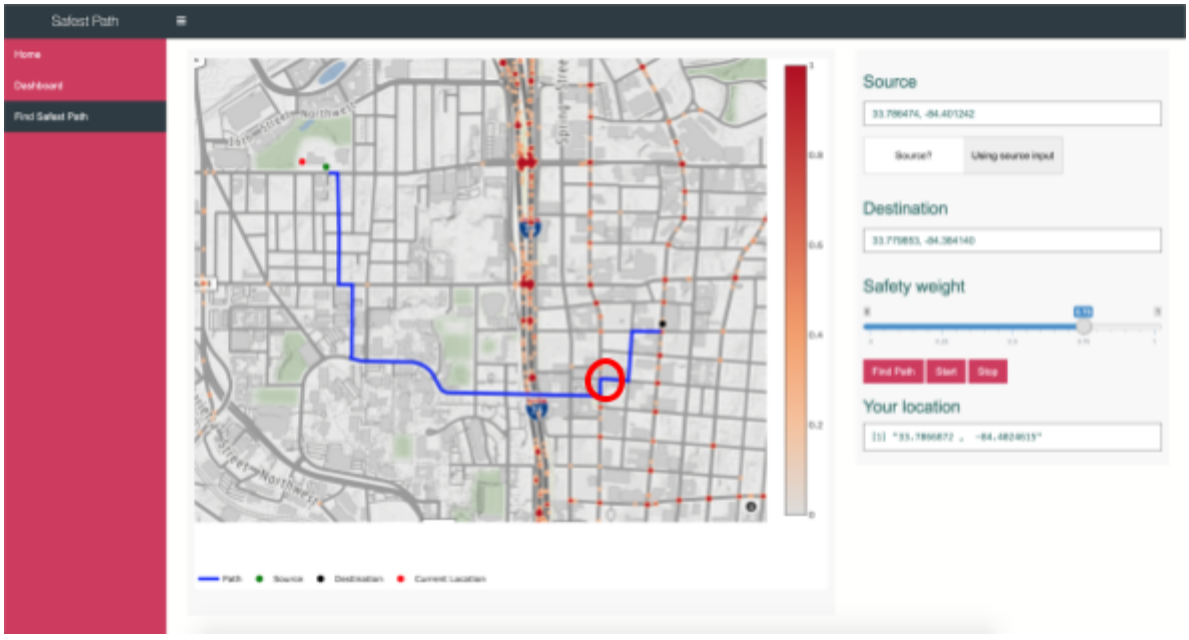


Figure 10) As the safety weight decreased to 0.75, the path became a little shorter but it crossed an area where accident took place (marked with red circle) which was avoided by previous path

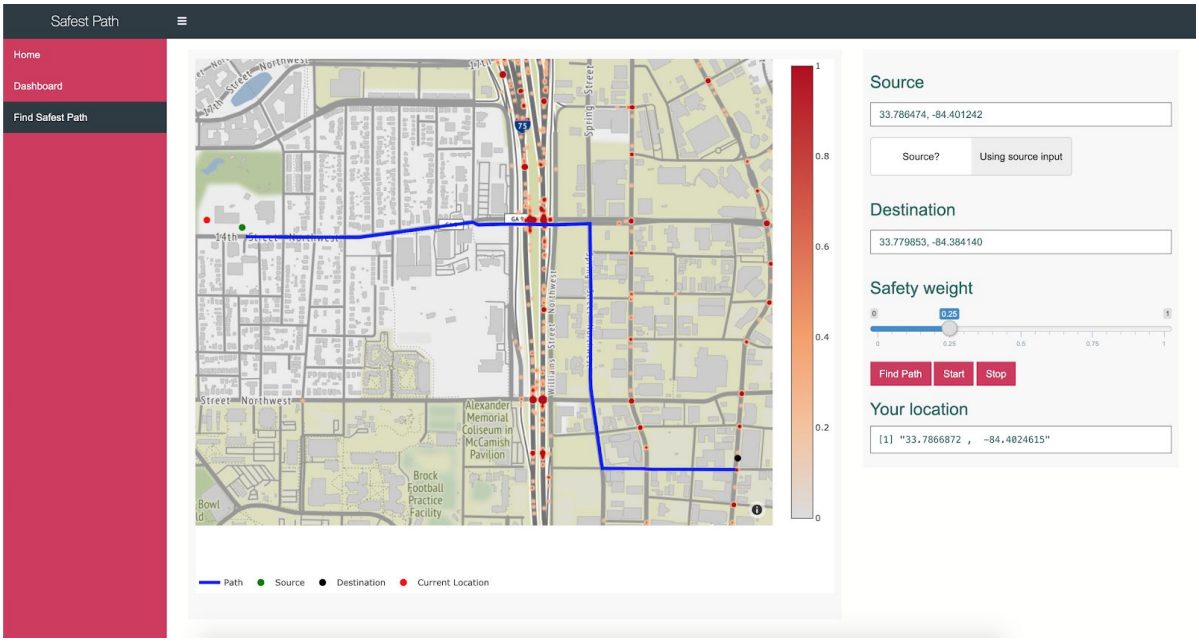


Figure 11) For safety weight = 0.25, the path became more short but increased the exposure to more accident areas.

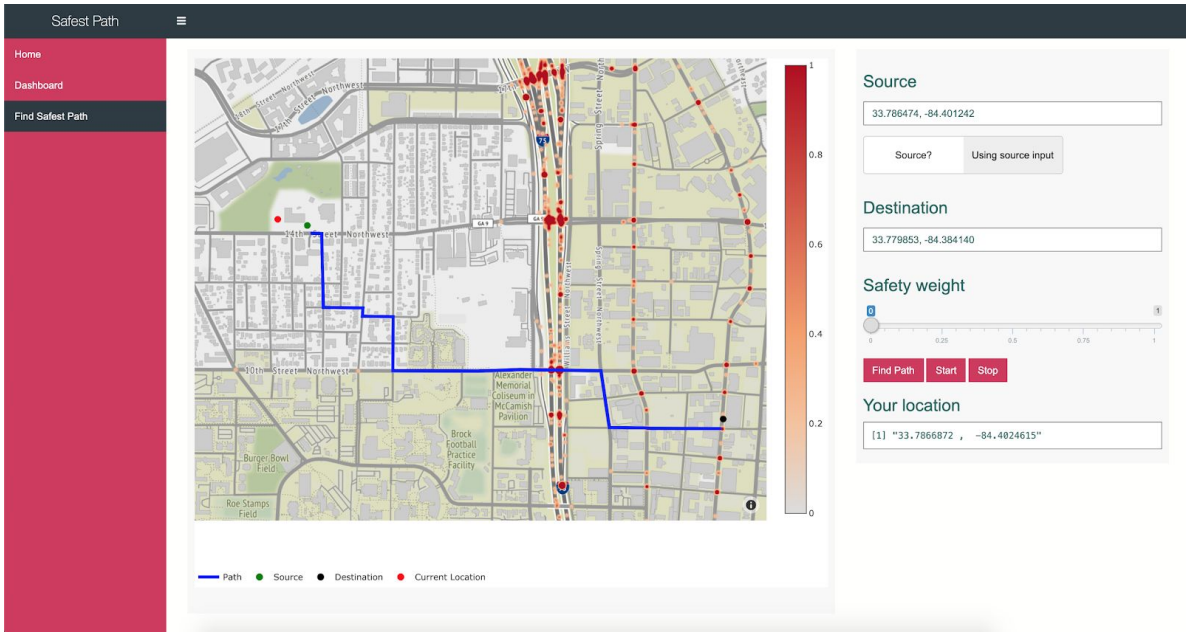


Figure 12) For safety weight = 0, SPOT gave the shortest possible path

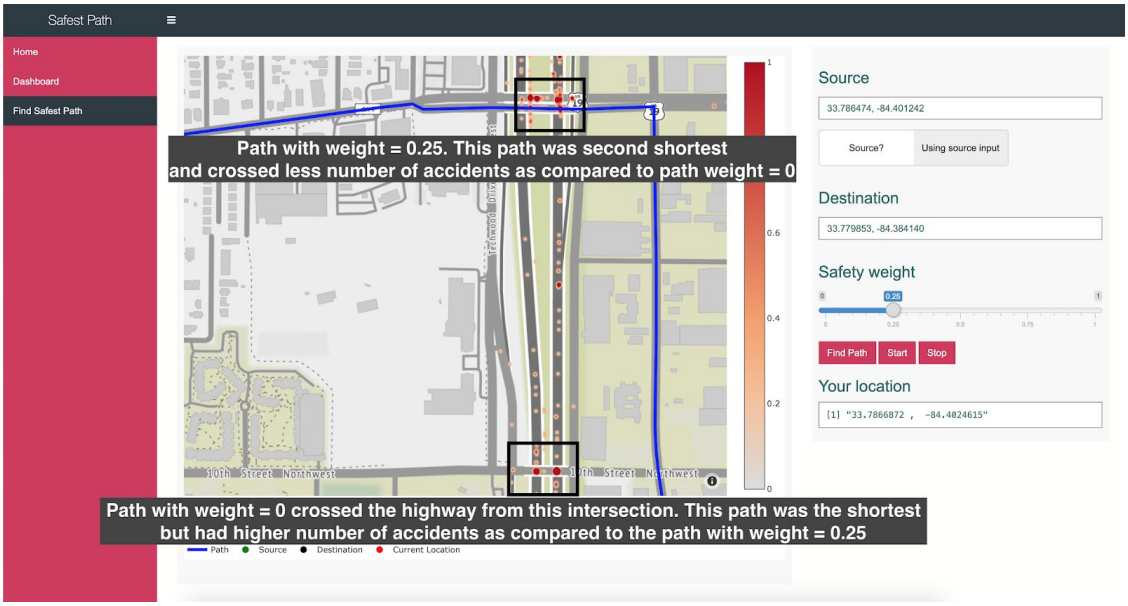


Figure 13) Path given by SPOT used that side of the road which was legally allowed to drive. Since this side of the road has less number of accidents it was included in the path

Although at first glance it appears as if path in figure 11 with weight = 0.25 used the intersection with higher accidents, but on closer inspection it was observed that more accidents have happened on the opposite side of the road and not on the road which was used in the path. Comparison of paths with weight = 0 and 0.25 are marked with black boxes in Figure 13. This showed the accuracy of SPOT while giving the safest path.

To generate the alert, Kernel Density Estimation (KDE) was used to assign the probability of accident to the current location of the driver. A KDE model was fit on the complete data to identify the dense accident regions (Figure 2). This gave intuitive results as we can see the probability of accident near high accident prone regions (like intersections on highways) is higher as compared to when the driver is inside the ‘Georgia Tech’ campus (Figure 3). Many locations were successfully tried. We have used ‘epanechnikov’ kernel as it gives faster probability calculation than gaussian and we believe that the probability of a place being an accident zone (based only on historical accidents) should go to 0 as we move very far away from the historic accidents.

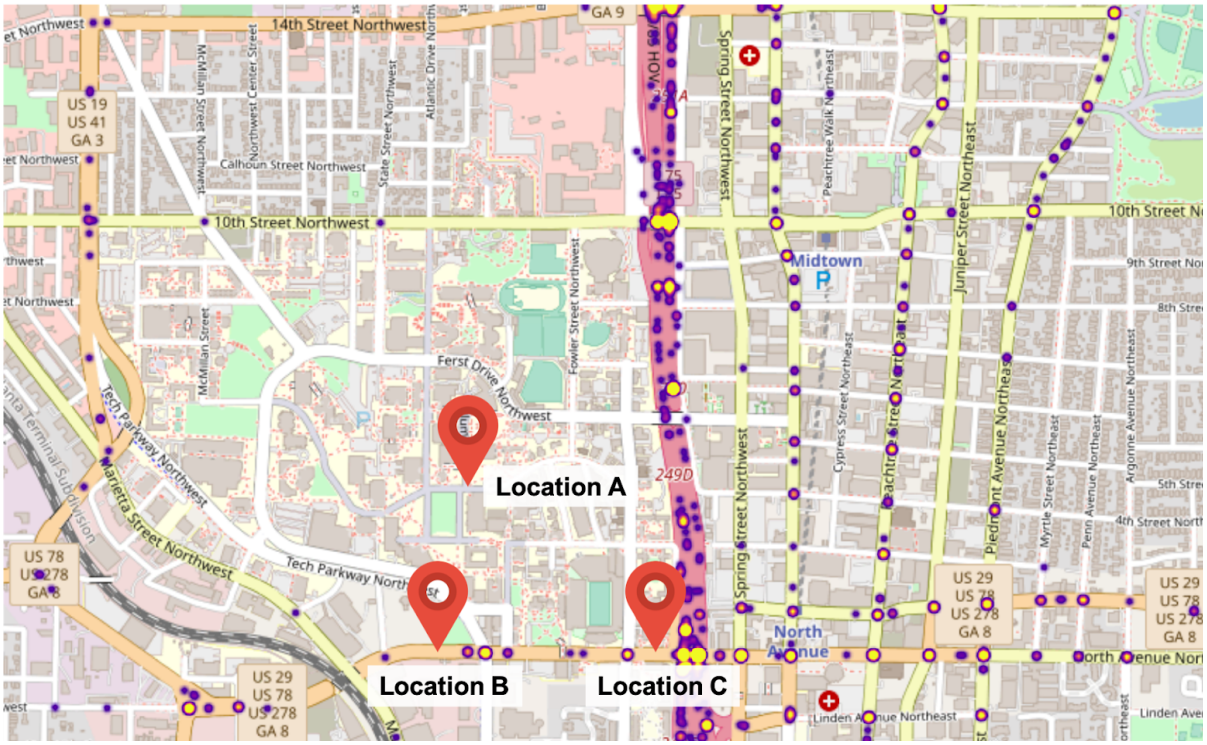
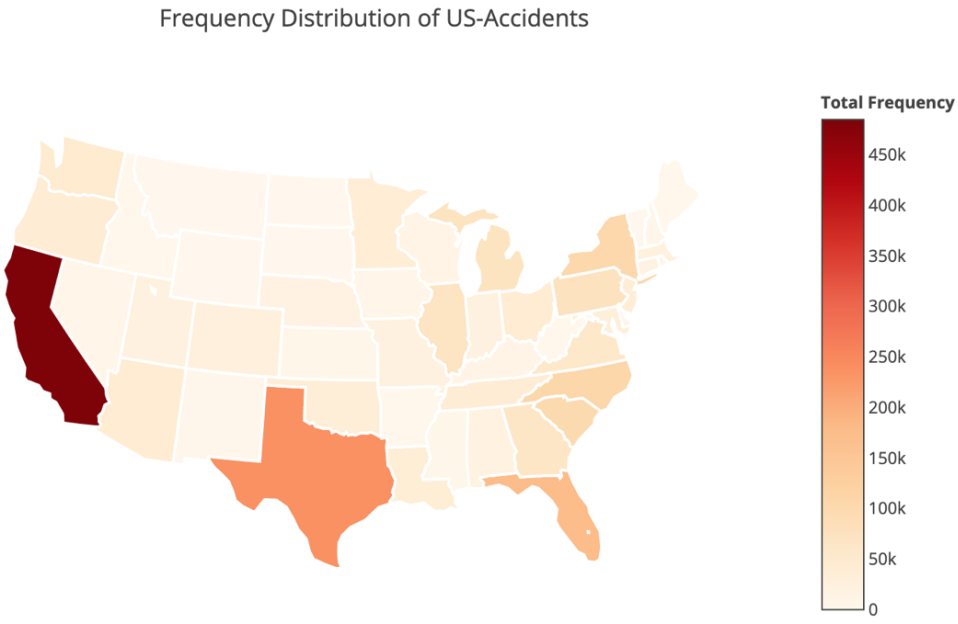


Figure 14) Example showing the risk of accident at three locations: Location A (Georgia Tech campus), Location B (Away from intersection and many accidents) and Location C (near the intersection with many accidents). Our algorithm showed that Location C has >100 and 7 times higher chances of accidents than Location A and Location B respectively

Appendix 1

This is a countywide traffic accident dataset, which covers 49 states of the United States. We are using data for one of the larger states. The data is collected from February 2016 to March 2019, using several data providers, including two APIs which provide streaming traffic event data. These APIs broadcast traffic events captured by a variety of entities, such as the US and state departments of transportation, law enforcement agencies, traffic cameras, and traffic sensors within the road-networks. Currently, there are about 2.25 million accident records in this dataset. Below is the data distribution over all of the states:



Attribute	Description
ID	This is a unique identifier of the accident record.
Source	Indicates source of the accident report (i.e. the API which reported the accident.).
TMC	A traffic accident may have a Traffic Message Channel (TMC) code which provides more detailed description of the event.
Severity	Shows severity of the accident (a number between 1 and 4).
Start_Time	Shows start time of the accident in local time zone.
End_Time	Shows end time of the accident in local time zone.
Start_Lat	Shows latitude in GPS coordinate of the start point.
Start_Lng	Shows longitude in GPS coordinate of the start point.
End_Lat	Shows latitude in GPS coordinate of the end point.
End_Lng	Shows longitude in GPS coordinate of the end point.

Distance(mi)	The length of the road extent affected by the accident.
Description	Shows natural language description of the accident.
Number	Shows the street number in address field.
Street	Shows the street name in address field.
Side	Shows the relative side of the street (Right/Left) in address field.
City	Shows the city in address field.
County	Shows the county in address field.
State	Shows the state in address field.
Zipcode	Shows the zipcode in address field.
Country	Shows the country in address field.
Timezone	Shows timezone based on the location of the accident (eastern, central, etc.).
Airport_Code	Denotes an airport-based weather station which is the closest one to location of the accident.
Weather_Timestamp	Shows the time-stamp of weather observation record (in local time).
Temperature(F)	Shows the temperature (in Fahrenheit).
Wind_Chill(F)	Shows the wind chill (in Fahrenheit).
Humidity(%)	Shows the humidity (in percentage).
Pressure(in)	Shows the air pressure (in inches).
Visibility(mi)	Shows visibility (in miles).
Wind_Direction	Shows wind direction.
Wind_Speed(mph)	Shows wind speed (in miles per hour).
Precipitation(in)	Shows precipitation amount in inches, if there is any.
Weather_Condition	Shows the weather condition (rain, snow, thunderstorm, fog, etc.)
Amenity	A POI annotation which indicates presence of amenity in a nearby location.
Bump	A POI annotation which indicates presence of speed bump or hump in a nearby location.
Crossing	A POI annotation which indicates presence of crossing in a nearby location.
Give_Way	A POI annotation which indicates presence of give_way in a nearby location.

Junction	A POI annotation which indicates presence of junction in a nearby location.
No_Exit	A POI annotation which indicates presence of no_exit in a nearby location.
Railway	A POI annotation which indicates presence of railway in a nearby location.
Roundabout	A POI annotation which indicates presence of roundabout in a nearby location.
Station	A POI annotation which indicates presence of station in a nearby location.
Stop	A POI annotation which indicates presence of stop in a nearby location.
Traffic_Calming	A POI annotation which indicates presence of traffic_calming in a nearby location.
Traffic_Signal	A POI annotation which indicates presence of traffic_signal in a nearby location.
Turning_Loop	A POI annotation which indicates presence of turning_loop in a nearby location.
Sunrise_Sunset	Shows the period of day (i.e. day or night) based on sunrise/sunset.
Civil_Twilight	Shows the period of day (i.e. day or night) based on civil twilight.
Nautical_Twilight	Shows the period of day (i.e. day or night) based on nautical twilight.
Astronomical_Twilight	Shows the period of day (i.e. day or night) based on astronomical twilight.

Source: https://www.kaggle.com/sobhanmoosavi/us-accidents