A way to address safe navigation when traveling, that ensures personal safety to help make informed decisions.

Kritika Verma

Table of Contents

Abstract

This research looks at a new navigation system that can help women and students traverse new regions securely while also improving their user experience. According to the study, personalized route suggestions and crowdsourced data can lead to safer and more educated navigation selections. This system prioritizes well-lit regions, high-traffic routes, and real-time safety information to solve the issues encountered by these vulnerable groups. The user-centered strategy, simple design, and community participation efforts all contribute to a stress-free navigating experience. The study's findings are predicted to have a beneficial influence on safe navigation practices, resulting in more inclusive and safer travel conditions for women and students. This navigation system could transform navigation technology by providing consumers with the confidence and peace of mind they require when traveling. Future research could investigate additional safety features, evaluate user satisfaction, and conduct long-term safety assessments to improve navigation systems for enhanced user safety and experience.

Keywords: safe navigation, personalized suggestions, crowd-sourced data, women and students, user interaction, long-term safety assessments

1. Introduction

Navigation systems have become a popular and widespread user application over the past two decades. They play an essential role in today's navigation system. In 2005, about 10% of the drivers in Europe were using a navigation system[1]. For the Netherlands, this number was higher. Between 17 and 22% of the drivers in the group of men 30-55 years old owned a navigation system. In 2007, 23% of all cars in the Netherlands were equipped with a navigation system[2]. Offering real-time, user-friendly directions and access to up-to-date traffic information, navigation apps like Google Maps, Waze, and Apple Maps have become immensely popular. The increasing availability

of navigation-related data changes the perception of travel and safety. As technology advances and integrates with other services, navigation applications will influence and improve the way we traverse the world in the next few years[3]. With all this information in mind, an important question arises: do these apps really ensure the safety of users, especially women and students in new cities? Incorporating safety-focused elements into navigation applications has the potential to have a noteworthy impact on urban planning and public safety measures. Such systems enable users to make well-informed choices about their travels, especially in unfamiliar or high-risk areas, resulting in lower crime rates and improved overall sense of personal security, and confidence[4].

Previous research concerning the safety of users is rather heterogeneous. There are lots of applications available in this digital world, but these applications have some limitations. While several attempts have been made to reduce accidental risk, little has been done to detect criminal threats on the roadways. A few researchers collect crime information from government offices and map areas with high crime rates. A safety measure for a certain route in these works mostly shows the degree of criminal activity in the region surrounding the route. While such a metric is beneficial, it is insufficient. These measurements have the following limitations: (1) It is tough to re-obtain data from city offices and retrain the models on a regular basis, and (2) the past alone may not accurately predict the future. There are very few applications that comprehensively help the user traverse a safe path. This research aims to solve precisely that problem. Its objective is to propose a model where the user will get route suggestions based on a safety score assigned to any area/road and also enable the user to send out an emergency SOS alert to their immediate contacts in case of any danger or harm. In this research, we look at several aspects affecting road safety from diverse angles and offer a new and more robust safety model. Here is the visual representation of the data provided by FBI, what shows how important this project is,

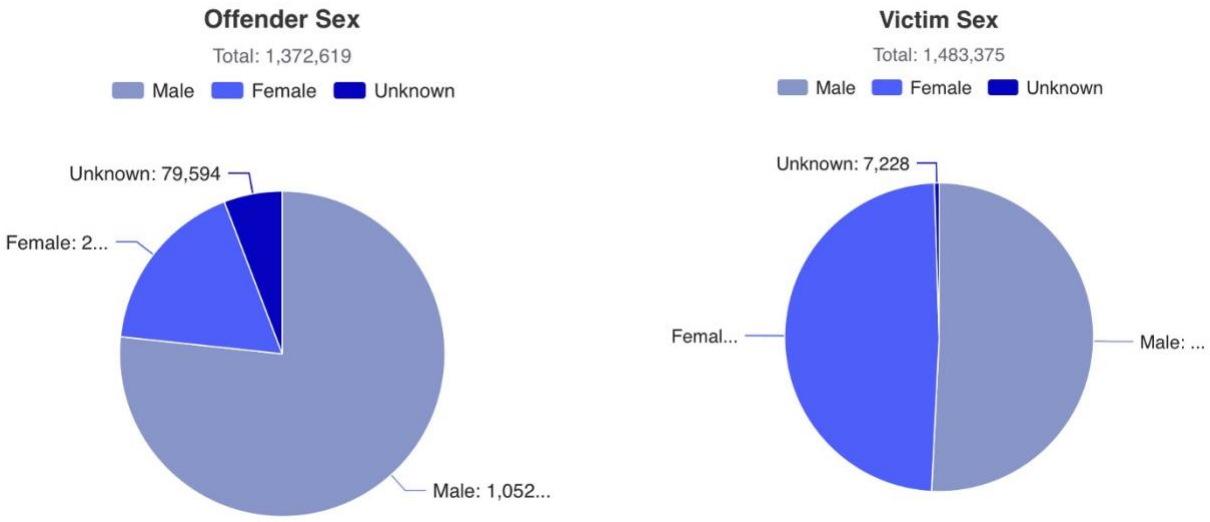

Figure 1: Visuals of all violent crime offenders vs. victim demographics

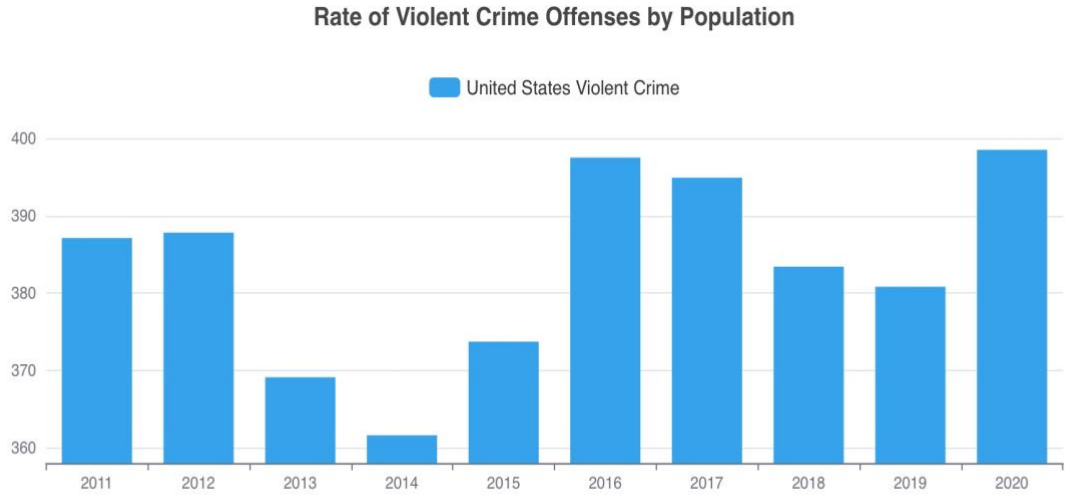

Figure 2: Trend of violent crime from 2011 to 2020 (Rate per 100,000 people, by Year)

2. Materials and Methods

2.1 Data Collection

The data collection process was the first step in this project. The goal of data collection was to gather a large and representative dataset of crime locations in a city. For this project, the data was limited to the crime dataset of Boston, Massachusetts, USA. This dataset was collected from “Crime Incident Report (August 2015 - Present)” which was being published and updated regularly by The Boston Police Department(BPD) [5]. This dataset was then filtered out to the year 2023 and hence was extracted around 35000 data points.

After Crime Dataset was collected from The Boston Police Department(BPD), the dataset looks like the figure given below,

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	INCIDENT_N	OFFENSE_CC	OFFENSE_CC	OFFENSE_DE	DISTRICT	REPORTING_	SHOOTING	OCCURRED_	YEAR	MONTH	DAY_OF_WE	HOUR	UCR_PART	STREET	Lat	Long	Location			
2	232000076	3115		INVESTIGATI	C6		0	2023-01-01 (	2023	1	Sunday	6		S POINT DR						
3	232052535	1102		FRAUD - FAL	E18	520	0	2023-01-01 :	2023	1	Sunday	12		HYDE PARK A	42.2562155	-71.12402	(42.256215542554976, -71.12402029533455)			
4	232000219	2647		THREATS TO	B2		0	2023-01-01 :	2023	1	Sunday	19		WARREN ST	42.3169661	-71.082541	(42.316966055161714, -71.08254116899847)			
5	232004918	613		LARCENY SH	C11	355	0	2023-01-02 (	2023	1	Monday	0		GIBSON ST	42.2975549	-71.05971	(42.297554856340426, -71.05970968687002)			
6	232000012	3201		PROPERTY -	C6		0	2023-01-01 (	2023	1	Sunday	1		MARINA PAF	42.353002	-71.045023	(42.35300196839949, -71.04502313135512)			
7	232000318	617		LARCENY TH	C6		0	2023-01-02 :	2023	1	Monday	11		LUCY ST						
8	232000532	3114		INVESTIGATI	B3		0	2023-01-03 (	2023	1	Tuesday	4		SCHOOL ST	42.297027	-71.075766	(42.29702697022541, -71.07576600299171)			
9	232003171	617		LARCENY TH	D4	156	0	2023-01-03 :	2023	1	Tuesday	12		MASSACHUSETTS AVENUE						
10	232003117	616		LARCENY TH	A1	77	0	2023-01-12 (	2023	1	Thursday	7		SUDBURY ST						
11	232000653	3115		INVESTIGATI	E18		0	2023-01-03 :	2023	1	Tuesday	15		SEFTON ST	42.2702416	-71.106871	(42.270241632190654, -71.10687078516219)			
12	232053829	1106		FRAUD - CRE	E18	520	0	2023-01-04 (	2023	1	Wednesday	0		HYDE PARK A	42.2562155	-71.12402	(42.256215542554976, -71.12402029533455)			
13	232002127	3801		M/V ACCIDEI	E18		0	2023-01-09 :	2023	1	Monday	10		HYDE PARK A	42.2633774	-71.121491	(42.263377376140255, -71.1214907733233)			
14	232001012	3801		M/V ACCIDEI	C11	355	0	2023-01-04 (	2023	1	Wednesday	0		GIBSON STREET						
15	232000496	1831		SICK ASSIST	B2	294	0	2023-01-02 (	2023	1	Monday	0		AKRON STREET						
16	232000511	724		AUTO THEFT	A1	77	0	2023-01-03 (	2023	1	Tuesday	0		SUDBURY ST						
17	232000773	3802		M/V ACCIDEI	D14		0	2023-01-04 (	2023	1	Wednesday	6		MARKET ST	42.3516407	-71.151928	(42.35164065997342, -71.1519276885547 8)			
18	232000781	1831		SICK ASSIST	D4	274	0	2023-01-04 (	2023	1	Wednesday	7		CAMDEN STREET						
19	232001042	301		ROBBERY	C6		0	2023-01-05 (	2023	1	Thursday	1		PREBLE ST	42.3292165	-71.054718	(42.329216475558326, -71.05471750473438)			
20	232000160	3831		M/V - LEAVI	C11	355	0	2023-01-01 (	2023	1	Sunday	1		GIBSON ST	42.2975549	-71.05971	(42.297554856340426, -71.05970968687002)			
21	232000876	1810		DRUGS - POS	C6		0	2023-01-04 :	2023	1	Wednesday	12		SOUTHAMPTON STREET						
22	232000561	3201		PROPERTY -	D14	787	0	2023-01-02 (	2023	1	Monday	3		SUTHERLANI	42.3402337	-71.148007	(42.34023372260431, -71.1480068289134)			
23	232001562	3115		INVESTIGATI	D4	167	0	2023-01-06 :	2023	1	Friday	20		HARRISON AVENUE						
24	232000035	2101		OPERATING	B3		0	2023-01-01 (	2023	1	Sunday	1		EPPING ST	42.2897076	-71.072574	(42.28970757497091, -71.07257357838927)			
25	232006522	3831		M/V - LEAVI	D4	167	0	2023-01-04 (	2023	1	Wednesday	5		HARRISON A	42.3395415	-71.069409	(42.33954152609665, -71.06940928222296)			
26	232000934	3301		VERBAL DISF	C6	925	0	2023-01-04 :	2023	1	Wednesday	16		MITCHELL STREET						
27	232002861	3805		M/V ACCIDEI	C11		0	2023-01-11 (	2023	1	Wednesday	8		GRANITE AV	42.2822116	-71.055562	(42.28221159263271, -71.05556214581358)			
28	232000040	801		ASSAULT - S	A1		0	2023-01-01 (	2023	1	Sunday	1		TREMONT	42.3510796	-71.06489	(42.35107955816716, -71.06489045879461)			
29	232000526	3802		M/V ACCIDEI	E18		0	2023-01-03 (	2023	1	Tuesday	4		HYDE PARK	42.2616116	-71.122176	(42.261611616819344, -71.12217586327527)			
30	232001296	3114		INVESTIGATI	C11		0	2023-01-05 (	2023	1	Thursday	0		GIBSON ST	42.2975549	-71.05971	(42.297554856340426, -71.05970968687002)			

Figure 3: The format of Boston Crime Dataset with all the columns

After extraction of crime data, we need to get a Boston graph. All the roads in the Boston city in the graph are represented by edges and all the intersections of the roads represented by the nodes. To extract all this information and graphs, we used a python module named OSMnx which is a python package to download, model, analyze, and visualize street networks and other geospatial features from OpenStreetMap[6]. The OSMnx Package extracted data in the form of nodes.shp and edges.shp (.shp is an extension which is

referred to as Shapefile). The shapefile format is a digital vector storage format for storing geographic location and associated attribute information [7].

This Shapefile format is then read and accessed by an online tool known as ArcGIS. ArcGIS Online is a cloud-based mapping and analysis solution. Use it to make maps, to analyze data, and to share and collaborate[8].

Both the shapefiles and the data inside looks like the figure provided below

(a)

	osmid	y	x	street_cou	lon	lat	highway	ref	geometry
0	30730954	4.692573e+06	333521.887579	3	-71.021817	42.367608	NaN	NaN	POINT (333521.888 4692572.502)
1	61441677	4.692605e+06	333528.489931	3	-71.021746	42.367901	NaN	NaN	POINT (333528.490 4692604.832)
2	1102741801	4.692474e+06	333304.232697	3	-71.024430	42.366676	traffic_signals	NaN	POINT (333304.233 4692474.133)
3	61178875	4.694305e+06	328698.115519	4	-71.080878	42.382149	NaN	NaN	POINT (328698.116 4694304.576)
4	61356567	4.694249e+06	328800.687430	3	-71.079616	42.381675	NaN	NaN	POINT (328800.687 4694249.343)
...	...	...	...	...	...	...	...	...	...
10988	10736649604	4.692414e+06	325406.262781	1	-71.120266	42.364403	NaN	NaN	POINT (325406.263 4692414.120)
10989	10764898444	4.686941e+06	327102.347253	3	-71.098044	42.315526	NaN	NaN	POINT (327102.347 4686941.304)
10990	10974090513	4.683504e+06	328732.145506	3	-71.077262	42.284950	NaN	NaN	POINT (328732.146 4683504.053)
10991	10974090519	4.683473e+06	328731.874580	3	-71.077256	42.284671	NaN	NaN	POINT (328731.875 4683473.005)
10992	11001004233	4.689897e+06	328434.853066	1	-71.082762	42.342424	NaN	NaN	POINT (328434.853 4689897.171)

(b)

osmid	oneway	lanes	highway	maxspeed	reversed	length	...	to	name	width	ref	bridge	access	junction	tunnel	service	geometry
197230699	1	2	residential	25 mph	False	33.072	...	61441677	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	LINESTRING (333521.888 4692572.502, 333524.915...
197230701	0	NaN	residential	25 mph	True	278.885	...	30730954	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	LINESTRING (333304.233 4692474.133, 333316.069...
29989698	1	NaN	residential	25 mph	False	34.493	...	30730954	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	LINESTRING (333516.798 4692606.543, 333517.948...
721665439	0	2	tertiary	NaN	False	11.786	...	61441677	Airport Way	NaN	NaN	NaN	NaN	NaN	NaN	NaN	LINESTRING (333516.798 4692606.543, 333528.490...
1096734572	0	2	tertiary	NaN	True	60.446	...	61441677	Airport Way	NaN	NaN	NaN	NaN	NaN	NaN	NaN	LINESTRING (333588.123 4692594.070, 333567.428...

Figure 4: The shapefile dataset including (a) nodes.shp, and (b) edges.shp

2.2 Feature Definition and Visualisation

Although we have extracted the Boston Crime dataset, we cannot use all the columns. We can use only those features/columns which can be used to make a model. For Boston Crime Dataset, we have used some columns which are listed below,

- OFFENCE_DESCRIPTION: The type of crime that was committed

- OCCURRED_ON_DATE: This columntells us the date and time that offense was

  conducted.

- STREET: The street address of the crime location.
- LONGITUDE: The longitude coordinate of the crime location.
- LATITUDE : The latitude coordinate of the crime location.
- LOCATION: The Geo-Coordinates of the crime location.

After defining the features going to be used in this model, we need to visualize the data. For that we have used GeoPandas python Library. GeoPandas is an open source project to make working with geospatial data in python easier. GeoPandas extends the datatypes used by pandas to allow spatial operations on geometric types. Geometric operations are performed by shapely.

Geopandas further depends on fiona for file access and matplotlib for plotting [9].

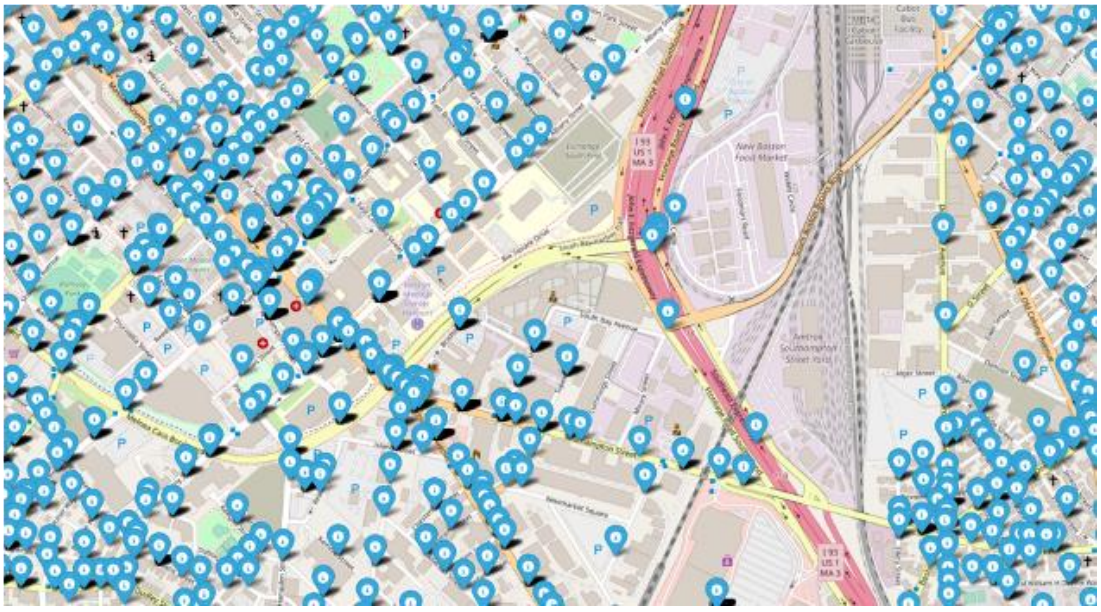

Figure 5: Plotting all the Crime Location on the map

But because this plotting is complex, we used folium to make clusters for a region as shown in the figure below,

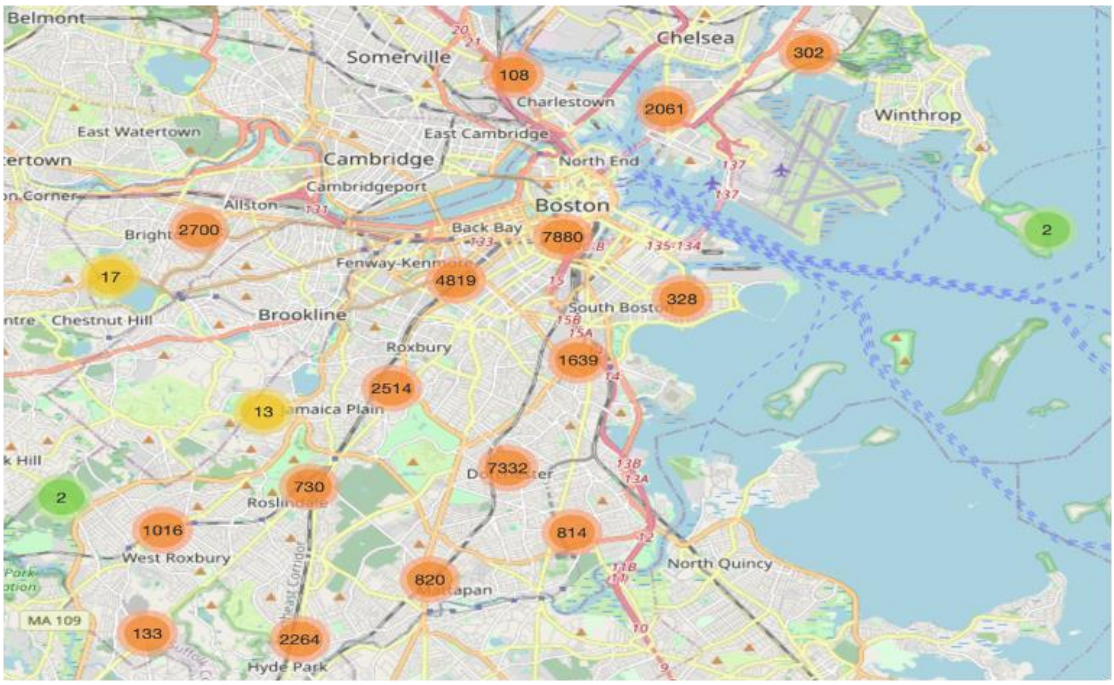

Figure 6: Plotting all the crime location by adding cluster features

Folium makes it easy to visualize data that's been manipulated in Python on an interactive leaflet map [10].

For visualizing nodes and edges we extracted, we used a tool ArcGIS. The Fig is shown below,

Figure 7: Plotting nodes and edges using ArcGIS

To combine those two plots, we again used ArcGIS which shows all the nodes(orange dots) and edges(purple line) with the crime location (black points).

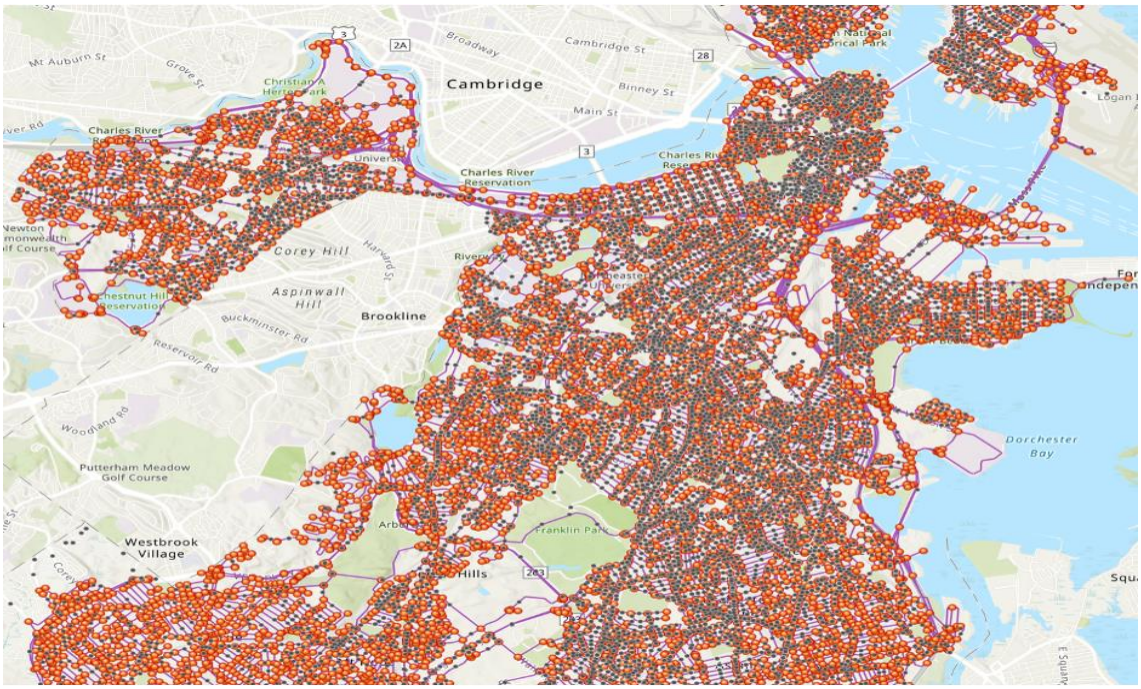

Figure 8: Showing overall combined dataset that shows nodes as intersection(orange points), edges as roads(purple line) and Crime Location (Black points)

2.3 Applying suitable Algorithm

Many projects require data modeling, especially those employing graph-based data structures. Dijkstra's method and A* (A-star) algorithms are two frequently employed pathfinding and graph traversal algorithms.

2.3.1 Dijkstra’s Algorithm

Single-source shortest path algorithms are what Dijkstra's algorithm is. In a weighted network with a non-negative weight assigned to each edge, it determines the shortest route between a given beginning node and all other nodes. In order to keep track of the subsequent node to visit based on the known distances at any given time, the algorithm maintains a set of visited nodes and a priority queue (often implemented with a min-heap).

Dijkstra's algorithm is an algorithm for finding the shortest path between two nodes in a graph.

The algorithm works by iteratively adding nodes to a set of nodes that have already been visited. The nodes are added to the set in order of their distance from the start node. The Dijkstra algorithm is a greedy algorithm, which means that it makes the best choice at each step based on the information that it has at that time. This makes the algorithm efficient, but it does not guarantee that it will find the shortest path.

Here's how Dijkstra's algorithm works:

***Step 1**: Initialize distances and visited status for all nodes. Set the distance of the starting node to 0 and all others to infinity.*
***Step 2**: While there are unvisited nodes, do the following:*
a. Select the unvisited node with the smallest distance from the starting node (from the priority queue).
b. For the selected node, update the distances of its neighbors if a shorter path is found.
c. Mark the selected node as visited.
***Step 3**: After visiting all nodes, the distances represent the shortest paths from the starting node to each node in the graph.*

In the following pseudocode algorithm, dist is an array that contains the current distances from the source to other vertices, i.e. dist[u]is the current distance from the source to the vertex u. The prev array contains pointers to previous-hop nodes on the shortest path from source to the given vertex (equivalently, it is the next-hop on the path from the given vertex to the source). The code u ← vertex in Q with min dist[u], searches for the vertex u in the vertex set Q that has the least dist[u] value. Graph.Edges(u, v) returns the length of the edge joining (i.e. the distance between) the two neighbor-nodes u and v. The variable alt on line 14 is the length of the path from the root node to the neighbor node v if it were to go through u. If this path is shorter than the current shortest path recorded for v, that current path is replaced with this alt path.

```
1  function Dijkstra(Graph, source):
2
3      for each vertex v in Graph.Vertices:
4          dist[v] ← INFINITY
5          prev[v] ← UNDEFINED
6          add v to Q
7      dist[source] ← 0
8
9      while Q is not empty:
10          u ← vertex in Q with min dist[u]
11          remove u from Q
12
13          for each neighbor v of u still in Q:
14              alt ← dist[u] + Graph.Edges(u, v)
15              if alt < dist[v]:
16                  dist[v] ← alt
17                  prev[v] ← u
18
19      return dist[], prev[]
```

2.3.2 A* Search Algorithm

Another pathfinding algorithm, the A* algorithm, outperforms Dijkstra's algorithm by employing heuristics to more effectively direct the search. Finding the shortest route in a weighted graph that contains a potential goal node is where it shines the most. A* combines a heuristic estimate of the cost from a node to the objective (h-value) with the actual cost to reach a node (g-value).The A* Algorithm is an improvement on the Dijkstra algorithm. The A* algorithm works by taking into account the estimated cost of the remaining path from the current node to the destination node. This makes the A* algorithm more likely to find the shortest path than the Dijkstra algorithm. The A* algorithm is also a greedy algorithm, but it uses a more sophisticated heuristic to estimate the cost of the remaining path. This makes the A* algorithm more efficient than the Dijkstra algorithm.

Here's how A* algorithm works:

Step 1*: Initialize distances, visited status, and f-values (f = g + h) for all nodes. Set the distance of the starting node to 0 and others to infinity. Set the f-value of the starting node using the heuristic estimate to the goal.*

Step 2*: While there are unvisited nodes, do the following:*

a. Select the unvisited node with the smallest f-value.

b. For the selected node, update the distances and f-values of its neighbors based on the heuristic estimate and the actual cost to reach that node.

c. Mark the selected node as visited.

***Step 3**: If the goal node is reached or there are no more unvisited nodes, the algorithm terminates, and the shortest path can be reconstructed.*

The goal node is denoted by node_goal and the source node is denoted by node_start We maintain two lists: OPEN and CLOSE:

- OPEN consists of nodes that have been visited but not expanded (meaning that successors have not been explored yet). This is the list of pending tasks.
- CLOSE consists of nodes that have been visited and expanded (successors have been explored already and included in the open list, if this was the case).

```
1 Put node_start in the OPEN list with f(node_start) = h(node_start)
2(initialization)
3                   while the OPEN list is not empty {
4                   Take from the open list the node node_current with the
                    lowest
5                   f(node_current) = g(node_current) + h(node_current)
6                   if node_current is node_goal we have found the solution;
                    break
7                   Generate each state node_successor that come after
                    node_current
8                   for each node_successor of node_current {
9                   Set successor_current_cost = g(node_current) +
                    w(node_current,
10                  node_successor)
11                  if node_successor is in the OPEN list {
12                  if g(node_successor) ≤ successor_current_cost continue
(to line 20)
13                  } else if node_successor is in the CLOSED list {
14                  if g(node_successor) ≤ successor_current_cost continue
                    (to line 20)
15                  Move node_successor from the CLOSED list to the OPEN
                    list
16                  } else {
17                  Add node_successor to the OPEN list
18                  Set h(node_successor) to be the heuristic distance to
                    node_goal
18            }
```

```
20            Set g(node_successor) = successor_current_cost
21            Set the parent of node_successor to node_current
22            }
23            Add node_current to the CLOSED list
24            }
25            if(node_current != node_goal) exit with error (the OPEN list
              is empty)
```

Here's is a comparison between the two algorithm, A* algorithm and Dijkstra's algorithm. The visualization shows how A* Algorithm outperforms Dijkstra's algorithm,

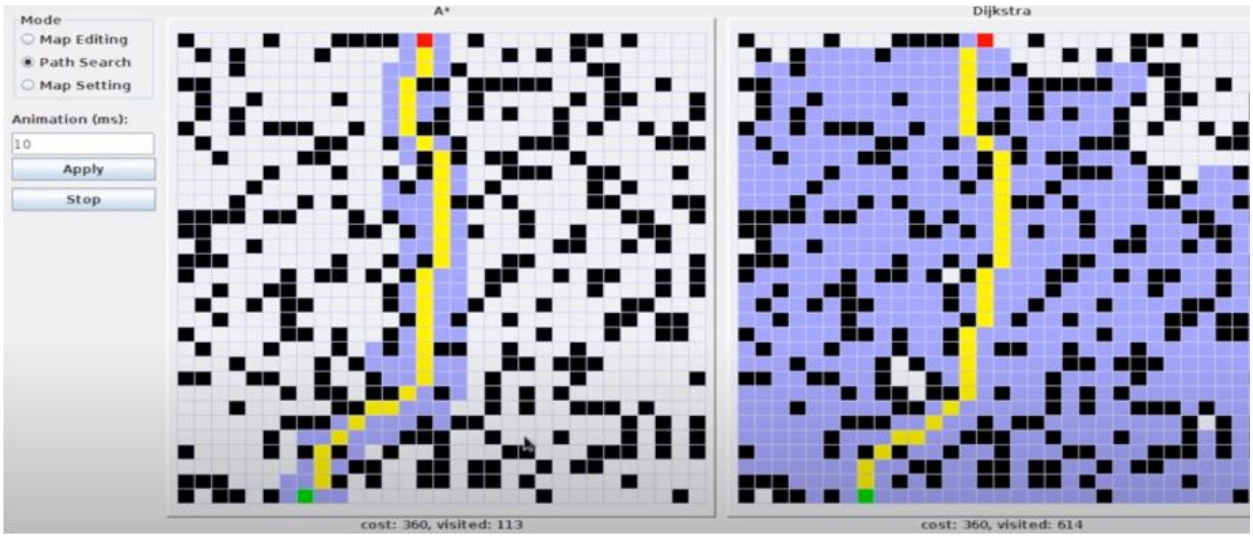

Figure 9: Visual Comparison between A Search Algorithm and Dijkstra's algorithm*

We can see that the Red point denotes the destination point and GREEN point denotes Starting point. The Dijkstra's Algorithm works in a circular way where you can see the PURPLE region denotes traversed nodes. While A* Search Algorithm straight to the point with the better cost calculation algorithm.

3. Hypothesized Results

3.1 Efficiency evaluation

To evaluate the efficiency of our prototype we have considered various parameters. We are using huge crowd-sourced data that has been gathered by us. Therefore, the data is based on a precise location (due to the accuracy of GPS) and is up-to-date. With a meticulous comparison between the Algorithm, we chose the A* (A-star) algorithm to suggest the users with the optimized safe route that is generated based on factors of crime location, street address and the date of the crime. This data and algorithm are visually observed in the map of the prototype.

3.2. Evaluating the app performance

These methods and models have been integrated into our experimental prototype of a simulated safe navigation app. Visual design elements, such as color schemes, fonts, and iconography, were thoughtfully chosen to promote clarity and visual hierarchy.

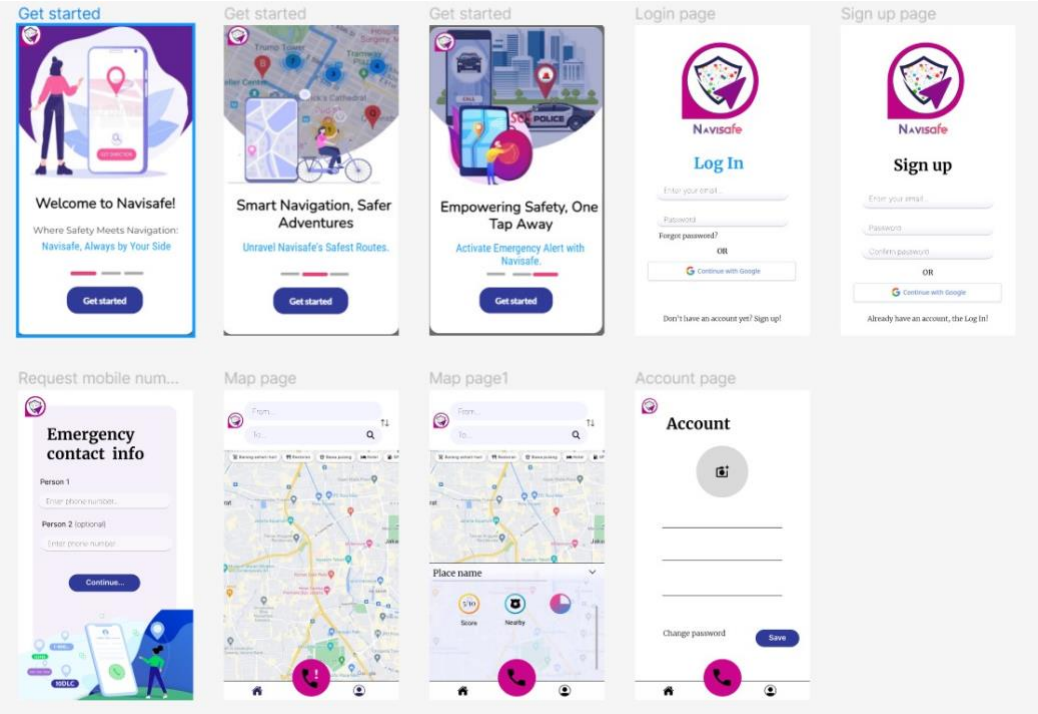

Fig. 10: The screens of prototype [11]

The UI of [Fig .10], the app incorporates intuitive navigation, responsive layouts, and accessibility to all the safety-focused features such as the emergency SOS alert. The user interface evaluation results drove iterative refinements, resulting in a user-centric and visually engaging interface that enables seamless access to crime-related insights, empowering users to make informed decisions

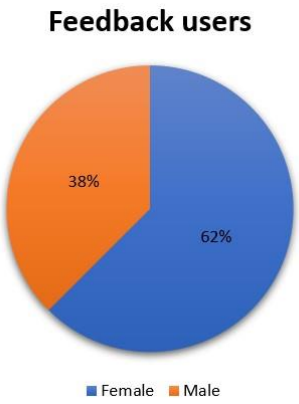

Fig. 11: 16 Users feedback data

effortlessly. To assess the overall performance of our safe navigation app, we conducted user feedback collection [Fig .11]. The feedback highlighted that the app has a positive impact on user awareness of potential risks in different destinations, helping them to navigate confidently.

4. Discussion

According to the hypothesized results above, we can get insights that might prove valuable in further research and development in the safety navigation field. Developing this app is feasible, but constant updating and refining, taking user experience into account will be an essential constant in our research.

4.1 Inferences

Based on this trained safety score model, a lot can be inferred -

4.1.1 Importance of features

All features have some kind of impact on the final safety score, but some factors carry more weight when it comes to the impact on the score. Understanding the importance of this, we only consider crime data as the main factor in this case. In future research, a complex algorithm can be constructed which includes other factors too to help in app UI design.

4.1.2 Up-to-date predictions

Real-world implementation of constant real-time data collection for this algorithm entails a higher level of difficulty, but certainly not an unreachable level. When the app can ensure this, it in turn ensures the user of a smooth experience.

4.1.3 Algorithm type

Through the illustrations above, we observe what a major effect using the A* algorithm in place of Dijkstra's algorithm made in the effectiveness of the model's chosen route. This means that, for our research in particular, the A* algorithm is better suited.

5. Conclusions & Future Outlook

For now, this algorithm maps out a safe route based only on crime data. Lots of other data also influence how safe a road is - like the type of street, lighting, etc. Moving forward, we aim to include code that takes into account above mentioned factors too to ensure that the level of accuracy is always up to the mark.

Futuristically, this app can be integrated with commonly used navigation tools like Google Maps/Apple Maps where users can select the safest and most efficient navigation option within the app itself, giving a wider reach to our application. Thus, this project idea holds immense scope for development and improvement.

Acknowledgments

The research in this report was initiated and supported by Incognito Blueprints. We thank our TA/advisor Ms. Kritika for guidance throughout the research.

Footnotes

Author contributions

S.A. drafted the design for all respective pages of the app; Y.M., and S.K. constructed the algorithm; S.K., P.K., H.C., Y.M., and S.A. constructed the UI design of the app; S.K., P.K., H.C., Y.M., and S.A. wrote the report. All authors have given approval to the final version of the manuscript.

Competing financial interests

The authors declare no competing financial interests.

References

1. Nagel, K., & Krüger, H. (2008). The use of navigation systems in Europe: A comparison of 2005 and 2008.Transportation Research Part F: Traffic Psychology and Behaviour, 11(3), 237-247.
2. Van der Horst, A. W., De Waard, D., & Brookhuis, K. A. (2007). The impact of navigation systems on traffic safety. Transportation Research Part F: Traffic Psychology and Behaviour, 10(4), 259-273.

3. Chen, J., & Liu, Y. (2017). The future of navigation: Challenges and opportunities. IEEE Intelligent Transportation Systems Magazine, 9(3), 26-35.
4. Zhang, J., & Axhausen, K. W. (2016). The potential of safety-focused navigation applications for urban planning and public safety measures. Transportation Research Part C: Emerging Technologies, 68, 1-14.
5. "Crime Incident Reports (August 2015 - To Date) (Source: New System) - Crime Incident Reports - 2023 to Present - Analyze Boston." Available: https://data.boston.gov/dataset/crime-incident-reports-august-2015-to-date-source-new-system/resource/b973d8cb-eeb2-4e7e-99da-c92938efc9c0
6. Boeing, G. 2017. OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks. *Computers, Environment and Urban Systems* 65, 126-139.
7. Wikipedia contributors, "Shapefile," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=Shapefile&oldid=1161215530 (accessed July 25, 2023)
8. "An introduction to ArcGIS Online—ArcGIS Online Help | Documentation." Available: https://doc.arcgis.com/en/arcgis-online/get-started/what-is-agol.htm
9. "GeoPandas 0.13.2 — GeoPandas 0.13.2+0.gd5add48.dirty documentation." Available: https://geopandas.org/en/stable/
10. "Folium — Folium 0.14.0 documentation." Available: https://python-visualization.github.io/folium/
11. "Figma," *Figma*, 2023.

https://www.figma.com/file/HYPWXhwZsCfFTdLScuEt2W/NavIsafe?type=design&node-id=0%3A1&mode=design&t=36W3sHQ2xU60JaFu-1 (accessed Jul. 15, 2023).