AI For Search and Rescue Public Transit Search Widget

CSC 491 Senior Project I
Completed June 9, 2024

Jadon Lai
Dept. of Computer Science and Software Engineering
California Polytechnic State University SLO
San Luis Obispo, USA
jadonlai.314@gmail.com

Franz Kurfess
Dept. of Computer Science and Software Engineering
California Polytechnic State University SLO
San Luis Obispo, USA
fkurfess@calpoly.edu

***Abstract*—Missing person cases can come in a variety of forms and first responders must find a person quickly to have the greatest likelihood of finding them. One particular place that people can easily go missing is urban environments. The person may go missing for a variety of reasons, but it can be particularly hard to find them if they use public transit, such as the Bay Area Rapid Transit System (BART). My goal is to provide first responders with a way to quickly find location(s) to look, if they know or think that the missing person used public transit last. This assumes that the person will have a "follow the crowd" mentality and go to/from stations that other people are going to/from. I've developed an AI for search and rescue (AI4SaR) widget that uses artificial intelligence (AI) methods to predict the probabilities that a person will be at a station. This also includes a walking radius around the most likely station for how far they could've traveled and the option to input details surrounding the time they went missing (e.g. the BART station from which they left, hour, weather conditions, etc.).**

The first responders will use the widget on a mobile device (such as their phone) and put in the features of the missing person case that are relevant to the model. The widget will then produce a map that has the aforementioned attributes. The widget will then be measured by the models' accuracy and the probability of a missing person going to the highest probability predicted station.

***Index Terms*—artificial intelligence, classification, supervised learning, data cleaning, features, label**

I. Introduction

The overarching goal of this widget is to help emergency responders find a place to search for a missing person, if they used public transit, in a reasonable amount of time. With over 42,000 adults and 60,000 children going missing in 2023 in California alone, it's crucial that emergency responders have a reliable, fast way of finding the missing person [1]. This includes examining popular entry/exit stops of the transit system and the common types of transit used. This paper will focus on the BART and where a missing person might end up in the Bay Area. The aim is to end up with an interactive map that displays the most likely spot or region where the missing person might be. There should be, but isn't required to be, a ranking system for the locations/stops that the missing person may go, in the form of probabilities or ordinal numbers.

I'll use an assumption that a missing person who last used public transit will most likely "follow the crowd." This means that whatever station they get on from, they'll exit on another station with the most people exiting. For instance, a child who doesn't know the subway system or an elderly person with a mental illness may try to just follow where everybody else is going and "move with the flow."

The BART consists of 50 different stations, spanning the Bay Area [3]. It also crosses between 5 different counties: Alameda, San Francisco, Contra Costa, Santa Clara, and San Mateo. With 131 miles of track, it covers a significant amount of the Bay Area, although by no means all of it.

The main uses will be emergency responders who're trying to actively find a missing person, but the widget can be made available to the public. This could be helpful if there's a public announcement about a missing person and it would be helpful if the public knows where to look. Similarly, people could be on the lookout if they so happen to be in the area of where the missing person is predicted to be.

However, this widget does come with some limitations. It's controlled by the accuracy of the AI model behind it, so if there aren't good results, then the widget may be functional, but it may not be very trustworthy. Likewise, if there are simply too many people using public transit to make accurate predictions, then it could also have an impact on performance. This could become a problem, as 48 million people used the BART in 2023 alone, with ridership on an increasing trend [2]. The main concern concerning the widget however is ethical. With the main purpose of the widget being to predict where a missing person might go, it could be misused to track not a missing person, but anybody of interest. One easy way to address this issue is to not release the widget to the public, but instead keep it private, only for emergency responders and authorized personnel.

II. Background

The core of the widget will be one or multiple AI models that will predict which station a missing person might end up at. There are a variety of ways to do this, but I will focus

on 3 main ones. These different methods are all ways of accomplishing the same task, just by different ways. At its core, the model will be a classification algorithm to predict a destination station among the 50 different stations. There are a multitude of different types of classification algorithms that fall under the umbrella of supervised learning, including decision trees [4], random forest classifiers, artificial neural network (ANN) classifiers, and recurrent neural networks (RNN), to name a few.

A. Decision Trees

Decision trees are a basic way to "break down a complex decision-making process into a collection of simpler decisions" [5]. The decision tree is an acyclic tree that consists of nodes and edges. At each node, a "decision" is made and depending on the value of an input, you would traverse down the tree, coming to a final output at a leaf node. Each leaf node is a classification of what I'm predicting, in this case the Destination Station. I use Shannon entropy as the "tree node splitting criterion," which is "equivalent to minimizing the log loss" [6]. Shannon's entropy is calculated as shown in Equation 1.

$$H = -\sum_{i} p_i \log p_i \tag{1}$$

The goal of the decision tree is to try to reduce the entropy as much as possible at each height of the tree. The log loss of the entire tree can then be computed using Equation 2, where D is a training set consisting of (x_i, y_i) pairs, T is a tree model.

$$LL(D,T) = \sum_{m \in T} \frac{n_m}{n} H(Q_m) \tag{2}$$

This is the same as the summation of each leaf's Shannon entropy, weighted by the number of training data that made it to each leaf.

In the end, this produces a tree such as Figure 1.

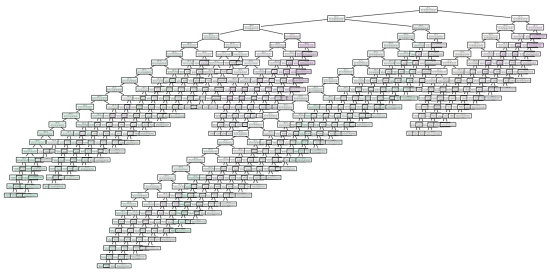

Fig. 1. Decision tree predicting a destination station from an origin station, month, day, day of the week, and county

The decision tree appears convoluted because of the number of classifications possible. Since there are 50 different stations (of the BART), there are 50 different possible classifications that the model could make, resulting in there being 50 leaf nodes. This, in addition to uncorrelated features, can lead to a hard to read decision tree, although it's still more user friendly to read the decision tree than trying to deduce outcomes from looking at just the data.

B. Random Forest Classifiers

Random forest classifiers are composed of decision trees classifiers, hence making a "forest" of the trees [7]. Each tree is created from a bootstrap sample of the training data, leading to every tree being slightly different. After the forest is created, predictions can be made by feeding in features to the forest to get a classification, just like in a decision tree. The features are then fed into every decision tree, producing a prediction. Then, a majority vote is held, and the highest voted classification is the prediction given by the forest.

Although tending to produce slightly better results than decision trees (due to the implementation of many decision trees), random forest classifiers tend to be slower than decision trees when making predictions. This intuitively makes sense, but it can be a major negative factor if the forest doesn't produce significantly better results than a single tree. In that instance, it would be worth it to use the tree, which would have similar results to the forest, but a much better response time. More information about random forests, in detail, can be found online [8], [9].

C. Artificial Neural Networks

ANN's are meant to simulate the neurons in a brain. This means that they have the capability of learning and can detect patterns not visible to the naked eye. At its highest level, the neural network is composed of an input, output, and hidden layers [10]. The number of neurons in the input layer is equal to the number of input features. Each neuron in every layer is connected to every neuron in the next layer, as shown in Figure 2.

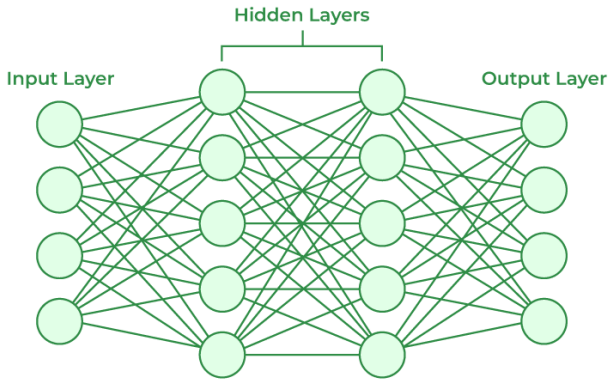

Fig. 2. Neural network showing the connections between neurons

Figure 3 shows that each neuron in the network has inputs, some summation function, an activation function, and an output.

The inputs to the neuron each have their own adjustable weight. There's also a bias input that's a constant value, still with a weight. This helps shift the activation function to give

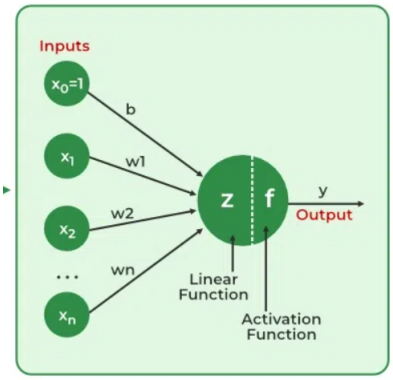

Fig. 3. Simplified structure of a single neuron in a neural network

certain outputs on certain inputs. The inputs are typically summed together, then fed into the activation function. A simple summation and popular activation function, Rectified Linear Unit (ReLU) [11] can be seen by Equations 3 and 4.

$$S = x_0 b + \sum_{i=1}^{n} w_i x_i \tag{3}$$

$$f(x) = \max(0, x) \tag{4}$$

The output produced will then be fed into another hidden layer or the output layer. For multiclass classifications, the Softmax function [12] is typically used in the output layer, as it provides the probabilities of each classification as shown in Equation 5. Then, the classification with the highest probability can be chosen as the ANN's prediction.

$$p_j = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{5}$$

The ANN learns by some optimization function, such as the Adaptive Moment Estimation (Adam) optimizer [13] or backpropagation [14]. These are used to update the weights on each step of the training cycle.

These optimization functions use loss functions, that determines the amount of error in the results. One such loss function that uses Softmax is categorical cross entropy (CCE), defined by Equation 6.

$$CCE = -\log\left(\frac{e^{s_p}}{\sum_j^c e^{s_j}}\right) \tag{6}$$

But when to stop is also an important detail about ANN's. I want to stop early enough so that the ANN won't overfit to the data, but also late enough to ensure that the ANN actually learns something. This can be done by a method called Early Stopping. Early Stopping is an extra parameter added to an ANN, with a metric that it monitors and a patience level p. Typically being one of four metrics, the training/test accuracy and training/test loss, after each epoch in the training phase, the metric is recorded. Then, if the metric doesn't improve by p epochs (increase for accuracy, decrease for loss), then the training will stop.

To further help the model prevent overfitting, there's an additional layer called a dropout layer that can be added to the ANN [15]. Dropout will "randomly drop units (along with their connections) from the neural network during training." This can be done by either fully dropping the node or setting the weight to 0.

D. Recurrent Neural Networks and Long Short-Term Memory

1) Recurrent Neural Networks: RNN's are a type of ANN that work particularly well with sequential data. This works using cycles within the network. The previous input(s) to the RNN are fed back into itself, allowing the RNN to learn based off of the current and previous input(s), not just the current input [16]. This can be seen by Figure 4.

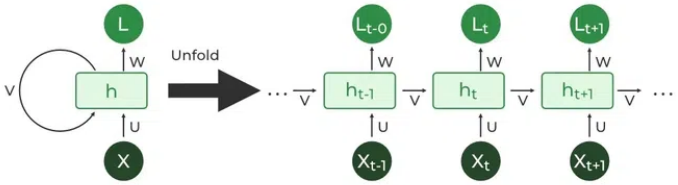

Fig. 4. RNN layer unfolded

If there exists a hidden layer H_t at time t, input X_t, weight matrix W, bias parameter b_h of the hidden layer, and activation function ϕ_h, then the hidden layer can be defined using Equation 7.

$$H_t = \phi_h(X_t W_{xh} + H_{t-1} W_{hh} + b_h) \tag{7}$$

Then, the network can be trained through methods similar to ANN methods, such as backpropagation through time. This is achieved by "unfolding" the RNN to a feedforward neural network. With the feedforward neural network, backpropagation can then be applied.

A major downside of RNN's though is that they suffer from the vanishing/exploding gradient problem. This occurs when matrix multiplications are perfoemd with weights < 1 or > 1, causing the gradient to approach 0 or ∞.

However, because RNN's and the variant LSTM's can learn based off of inputs, they work very well with time-series data.

2) Long Short-Term Memory: LSTM's are a variant of RNN's that attempt to solve the vanishing/exploding gradient problem [17]. Introducing an input, output, and forget gate defined by Equations 8, 9, and 10, these gates allow the LSTM to retain long-term dependencies.

$$I_t = \sigma(X_t W_{xi} + H_{t-1} W_{hi} + b_i) \tag{8}$$

$$O_t = \sigma(X_t W_{xo} + H_{t-1} W_{ho} + b_o) \tag{9}$$

$$F_t = \sigma(X_t W_{xf} + H_{t-1} W_{hf} + b_f) \quad (10)$$

The input and output gates control the information that enters/leaves the memory cell respectively. The forget gate controls the information discarded from the memory cell, storing or discarding information from previous time steps. This can be visualized by Figure 5.

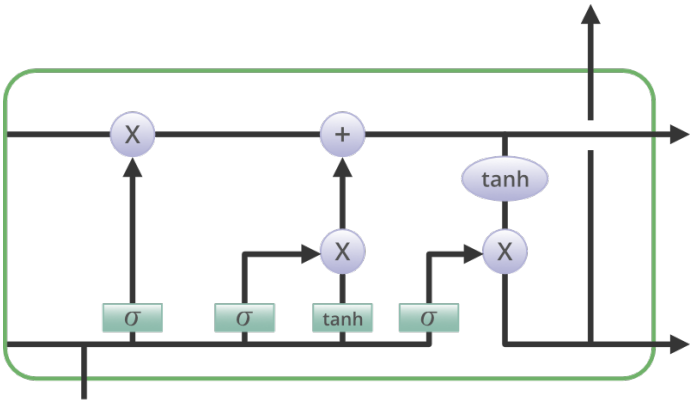

Fig. 5. LSTM memory cell

3) Bidirectional: There's also the option to change the RNN and LSTM to be bidirectional, instead of unidirectional. This will allow the model to receive information from the past, present, and future memory cells, relative to the current cell [18]. This can be seen in Figure 6.

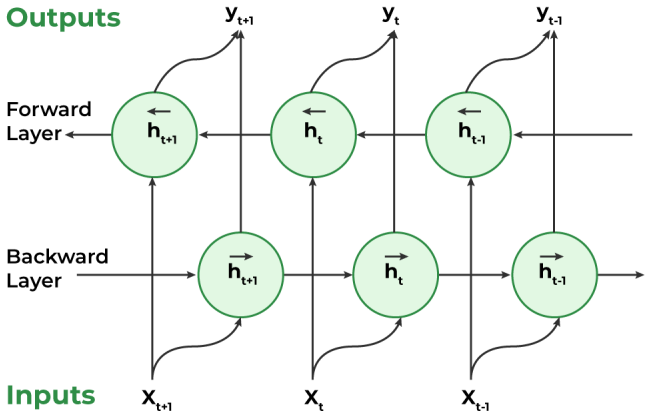

Fig. 6. BRNN layer unfolded

III. Related Work

While there are a lot of related papers and research, there isn't much information about predicting where a missing person might go, urban or otherwise. This is due to the numerous characteristics related with a missing person case. These characteristics are also very hard to find and measure, on top of seeing if they're even related at all. As such, the following works are similar, but not the same as this paper. Instead, I've found research done on separate aspects of searching for missing persons using public transit. This includes, but isn't limited to, public transit density estimators and predictors, missing persons probability of area techniques, feature selection, and map creation.

A. Ridership and Density Estimators

There has been quite a bit of research done on trying to find the ridership of public transportation, such as the BART. This is because of applications outside of search and rescue. One simple example is improving the efficiency of public transit and deciding where to put new transit lines. Studying these papers [19], [20], autoregressive integrated moving average (ARIMA) models, deep neural networks (DNN), bayesian structural time series, LSTMs, and bidirectional RNN's (BRNN) can be applied to predicting how crowded the BART is with decent success.

B. Probability of Area Techniques

Probability of area (POA) is the likelihood that a missing person would be in a certain region [21]. Given that a person can walk $1.42m/s$, then, in just an hour, the "search area could theoretical[ly] expand to $82km^2$." This is more ground than is feasible to cover in a reasonable amount of time. Thus, it's worthwhile to put probabilities to certain areas in order to find the "most probable location of the subject." This is especially important in Yosemite National Park, since there's hundreds of places a person can get lost. POA can easily be translated to the probability of station (POS), since a person could travel from any BART station to any of the other 49 BART stations within minutes.

With the creation of a ring and mobility model, the ring model simply calculates a circle of probability around the last known location. Each circle is based on the average walk speed of a person, and the farther away, the less likely it is that they'll have traveled there.

The mobility model instead uses the average effort that a person expends, as the geography of Yosemite consists of hills and mountains. This means that a person would be less likely to travel up a mountain, than stay on low ground.

C. Relevant Features

Deciding what is actually relevant to a missing persons case can be difficult. This is due to the nature of each case being distinct and unique from any others. However, it is still possible to find these features using certain methods. One such method is Shapley Additive Explanation (SHAP) values [22], [23]. This metric was used to find that "a longer time elapsed (in days) since the missingness report, a small municipality (i.e., with a relatively lower population), being male, and more advanced age of the missing person were all associated with a decreased probability of a missing older adult to be found later."

Another work used the person's gender, previous suicide attempts, somatic illnesses, traumatic events/experiences, and relationships with other people [24]. These were used in machine learning models to determine the "search level" of the missing persons case. The search level is the amount of

resources spent on the search, with levels I, II, and III. The lower the search level, the higher the urgency and importance of finding the missing person.

D. Geographic Information Systems

Geographic information systems (GIS) are meant to integrate data into a map. It's important to decide what features to add to the map, ensuring that the map isn't too cluttered or that it has enough information [25]. Programs "such as Maptech Terrain Navigator, National Geographic Topo and Delorme Topo North America" were often used by first responders, which don't have all the capabilities that GIS have. Instead of being able to manipulate and visualize data, these tools enabled responders to gather data in the field (such as GPS coordinates), display multiple layers (such as "topographic maps and aerial imagery"), and take measurements and 3D views. These are features that should be added to any search and rescue mapping program, while the addition of data and data manipulation may not be necessary.

IV. System Design

My system can be divided into 3 parts. The input data, AI model, and output map.

The input data will determine the capabilities of my model, the features that will be allowed to be fed into the model, and the output of the model.

The model will be the center of the widget and will make the predictions based off of the input data. Different models can yield different results and producing a model with high accuracy and probabilities for each BART station is important. The reason for the probabilities for each station is that, ideally, the widget doesn't tell the first responder where the missing person is, but instead gives a suggestion as to where they might be. The model will then be able to evaluate the predicted station with the highest probability with an observed station. This can be done with evaluation metrics such as accuracy, precision, recall, and F1-score [26]. Given that $TP = true\ positive$, $FP = false\ positive$, $TN = true\ negative$, and $FN = false\ negative$, the aforementioned evaluation metrics can be defined using Equations 11, 12, 13, and 14.

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

$$P = \frac{TP}{TP + FP} \tag{12}$$

$$R = \frac{TP}{TP + FN} \tag{13}$$

$$F = 2\frac{PR}{P + R} \tag{14}$$

In my case, it's important that recall is higher than precision because it's better to have false negatives than false positives. If there's a false positive, then responders will spend time and resources on a location that isn't where the missing person is. On the other hand, a false negative will indicate that the person might not be there, even if they are. This is better because the widget is meant to be used as a suggestion, not as fact. With this in mind, I would rather the model produce more uncertain results with a low precision than give a high probability to a random, unrelated location with a low recall.

Finally, I'll output the results of the model onto a map. The map will have an outline of the BART lines and stations, with the probability of each station being the destination included. Additionally, the map will include a measuring tool, zooming capabilities, and options to change the map type. These are enough features to ensure that the map is useful to search and rescue responders, but not too many to clutter the screen with unnecessary information.

A. Features and Labels

Using BART ridership data from 2023 [27], I'm able to clean the data so that each record is a BART trip by a person, as shown by the sample in Table I. Consisting of roughly 15 million records in 2023, there's more than enough data to train and test on.

TABLE I
BART Ridership Data Sample

Hour	OS	DS	DoW	Month	Day	County
0	12TH	12TH	Sunday	1	1	Alameda
0	12TH	16TH	Sunday	1	1	Alameda
0	12TH	19TH	Sunday	1	1	Alameda
0	12TH	19TH	Sunday	1	1	Alameda
⋮	⋮	⋮	⋮	⋮	⋮	⋮
23	WOAK	WCRK	Tuesday	5	2	Alameda

OS = Origin Station, DS = Destination Station, DoW = Day of Week

The DS will be the label that we're trying to predict and the other 6 attributes are the features. These simple features are easy to input and should be known in a missing person case. However, we can see that it's not easy to predict the DS just from inference, as shown by how convoluted Figure 7 is.

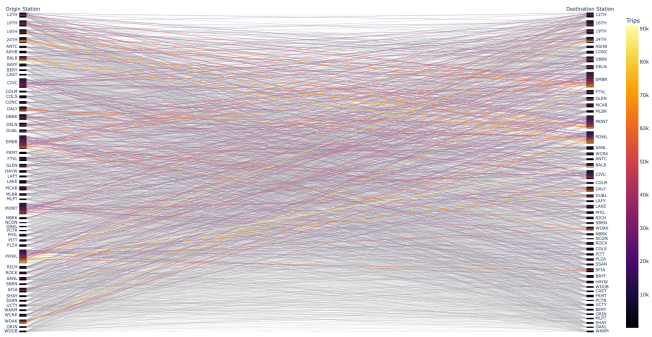

Fig. 7. Network of BART stations. The figure shows the OS on the left hand side, mapping trips to DSs on the right hand side. A color gradient also shows the number of trips made in 2023 from certain OSs to certain DSs, with yellow indicating the highest number of trips and black/grey indicating the lowest number of trips.

It's also worthwhile to note, from Figure 8, that some of the busiest stations are Embarcadero, Powell St., and Montgomery

St, all with over 1 million trips made to/from the stations in 2023. On the other hand, the Pittsburgh Center (PCTR) is the least busiest station, so there will most likely be a lower probability of a person going there. However, despite being the least traveled to/from station, PCTR still had 52.09 thousand trips made from and 50.157 thousand trips made to it in 2023.

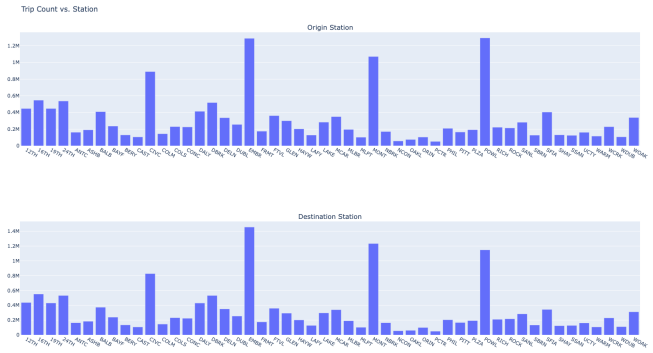

Fig. 8. Trip counts to/from BART stations in 2023

After cleaning and concatenating weather data as well [29], I was able to obtain the correlation matrix shown in Figure 9. It's evident that most of the features aren't correlated with the DSs, which could possibly results in a model with poor accuracy.

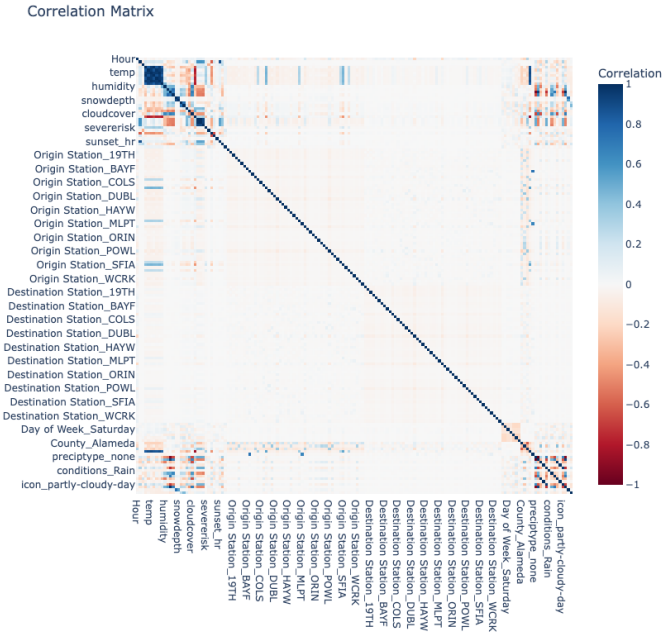

Fig. 9. Correlation matrix of all the features, including the initial BART dataset and concatenated weather attributes

The idea of poor correlation is further reinforced when performing a Chi-Squared Test [28] on all of the possible features. The Chi-Squared Test is given by Equation 15, where χ^2 is the statistic, O_i is the observed value, and E_i is the expected value of each record.

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i} \tag{15}$$

The χ^2 for the data can be seen in Figure 10. While hard to tell, we can see from the scale of the y-axis (in the millions) that the statistics are very poor. The lowest statistic, precipitation probability (precipprob) has $\chi^2 = 3,533.399$.

Additionally, the main feature that we would expect to predict the DS from, the OS, has $\chi^2 = 7.462$ million. As such, I chose to use the features from the original BART dataset to predict DSs, since none of the features from either the BART or weather datasets have proven to be correlated or associated with DSs.

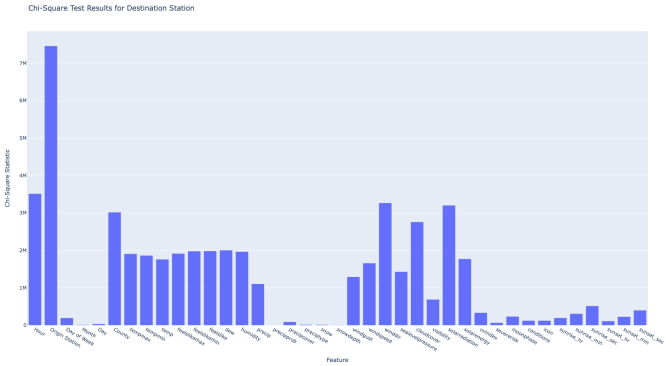

Fig. 10. χ^2 for each feature

The data will then either consist of the full dataset or be a random sample of 1 million records of the dataset based on the model. Due to time constraints, the neural networks will be trained using the random sample and the decision tree and random forest classifiers will be trained using the whole dataset.

Then, the data will be split into training and testing data, with 80% being training data and 20% being testing data. The testing data will be used during each epoch to see how good our model will perform on data the model hasn't seen before and can help ensure that the model doesn't overfit.

B. Models

When making a model, it's important to choose good hyperparameters, as they can affect the accuracy of the model severely. I've tuned my hyperparameters to work as well as possible, but there is still room for optimization.

1) Decision Tree Classifier: The decision tree is simple and doesn't need many parameters. Mine was created using the Shannon entropy from Section II-A and allowing the tree to grow infinitely until there are 50 leaves (for 50 classifications). The tree can be seen in Figure 1.

2) Random Forest Classifier: The random forest is also as simple as the decision tree. The forest consists of 100 trees and each tree also uses Shannon entropy.

3) ANN: The ANN consists of an input layer, 4 hidden layers, dropout layers, and an output layer. The input layer has a size of 6 for the 6 features and the output layer has 50 neurons with a softmax activation, one for each station. The 4 hidden layers have 64, 32, 16, and 8 neurons with a ReLU activation. In between each of the hidden layers and before the output layer, there are dropout layers that drop 20% of the weights, to prevent overfitting.

Then, the ANN is compiled using the CCE and uses the Adam optimizer. Then, I train the model for a maximum of 1,000 epochs and a batch size of 64. Additionally, early stopping will be used to monitor the validation loss at each epoch and will stop the training if the loss doesn't improve after 5 epochs in a row.

4) RNN, LSTM, and Bidirectional: Due to LSTMs being variants of RNNs, they will have similar architectures. From experimentation, I've found that feeding the inputs in as embeddings with an input and output size of 64 works best. From there, the hidden layers consist of an RNN or LSTM with 64 units and dropout rate of 20%. Finally, the same output layer as the ANN, with 50 neurons and Softmax activation is used. These models are also compiled and trained the same way as the ANN. BRNNs and Bidirectional LSTMs (BLSTMs) can easily be created by wrapping the RNN and LSTM in a Bidirectional layer.

C. Map

The map focuses on the Bay Area, outlining the BART tracks, stations, and counties. The BART tracks and stations are added to the map as .shp files. The counties are created by subtracting the county water .shp file from the faces .shp file.

There will be 4 tile layer options, a street map, terrain, black and white (for printing), and CartoDB Positron. Each station will have a tooltip that contains the station's name, acronym, and probability of that station being the DS. The OS will be red and the predicted DS will be green, with a "walking circle" of around it. This is a circle that the person could've traveled, if they were walking, within a certain amount of time. The walking radius is given by Equation 16, where $r_{walking}$ is in meters and t is in minutes. The walking circle will be set to 60 minutes by default. This assumes that the normal walking speed of a person is 1.42m/s.

$$r_{walking} = 1.42m/s \times t \times 60 \tag{16}$$

To further signify their importance, the OS and DS will have a larger marker than the other stations. The rest of the stations will be on a color gradient, with blue being the stations with the least probability of being the DS and green stations having a higher probability. There will also be a measuring tool and the ability to find any latitude/longitude by selecting any empty space on the map.

V. Implementation

The entire implementation was done in Python, due to the large number of AI libraries that are well maintained and documented. The inputs to program are the type of model to use, a number that'll indicate how many features to use, and an optional input sample .csv file. If the input .csv file isn't given, then the 5 millionth record will be used to make a prediction.

A. Data

Data was obtained from online sources and saved as .csv files. Then, I clean and concatenate the data using Pandas and Numpy.

Data analysis is achieved using the aforementioned libraries, as well as Matplotlib and Plotly Express for visualizations. For instance, we're able to view the trips from the Civic Center Station (CIVC) on June 15 at noon in Figure 11. We can see that most people go from CIVC to 12th St. (12TH), however there's many people going to all the other stations as well.

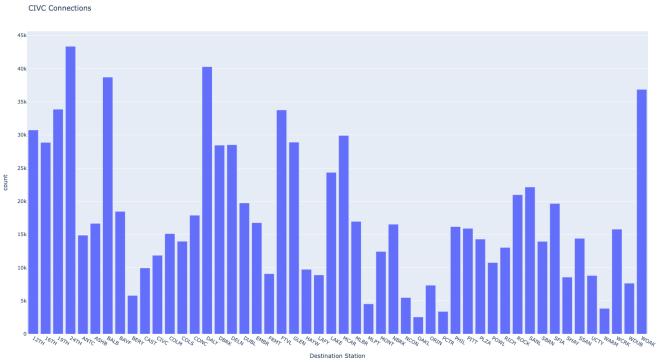

Fig. 11. Trip counts to each station from the CIVC station on June 15 at noon

Figures such as these will be useful in analyzing the models' output probabilities. For instance, we can directly compare the two and the closer they match, potentially the more accurate the model is.

B. Model Libraries

There are many ways of creating AI models, but the two I used are Scikit-Learn (SKL) and Tensorflow. SKL is also useful for data transformation, manipulation, and scoring metrics. After the models are trained, they're tested on the input sample and the resulting prediction, station probabilities, and map are produced.

1) Scikit-Learn: SKL was used to one hot encode the data for both the SKL and Tensorflow implementations. I created the decision tree and random forest using this library.

After the models are created, they're saved as .joblib files, allowing for easy retrieval and usage.

2) Tensorflow: Tensorflow was used to create the ANN, RNN, LSTM, and the BLSTM. A history is recorded of the models' validation accuracy/loss and test accuracy/loss. The models are also saved as .keras files.

C. Map

The map is created using Folium, using xyzservices and stadia for the tile layers. All of the shapefiles are read in using geopandas and I use the branca library to make the color gradient for the station probabilities. This is then saved as a .html file so that it can be viewed on any device.

VI. Results: Testing and Validation

The main evaluation metrics I'll be using are the accuracy, recall, precision, and F1-score of each model. These can determine how good the models are, but we'll also be able to see some patterns in the probabilities that each model produces. They'll be calculated using the test data from the BART dataset.

A. Example of Model Probability Patterns

After training the model, the probabilities of each station being the DS can easily be retrieved, then plotted. For example, we can see the trips to every station from South Hayward (SHAY) on February 13, 8am (the 5 millionth record) in Figure 12. The observed DS for this record is Hayward (HAYW).

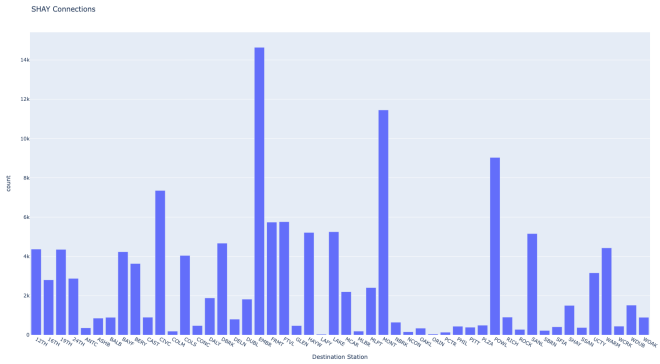

Fig. 12. Trip counts to each station from the SHAY station on February 13 at 8am

Then, from the decision tree, we get the following probabilities in Figure 13. With the blue and red bars indicating the base probabilities from the dataset and the probabilities from the model respectively, we can see that the decision tree probabilities very closely match the base probabilities. The random forest classifier has very similar probabilities to the decision tree. For every bar chart shown in this example, the format will consist of blue bars being the base probabilities and red bars being the model probabilities.

From the ANN, RNN, LSTM, and BLSTM, we see slightly different patterns. We can see from the ANN in Figure 14 that the probabilities are more uniformly distributed. This means that stations that have a lower base probability are given a slightly higher probability from the model and vice versa. The RNN also has similar probabilities, although less uniform than the ANN.

The LSTM on the other hand, in Figure 15 has much different probabilities, without a distinct pattern compared to the base probabilities. This may be due to some underlying

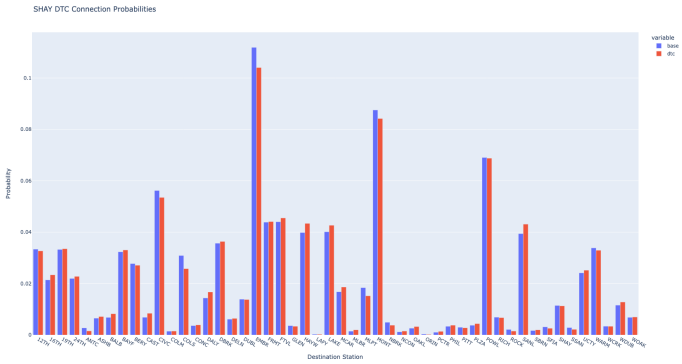

Fig. 13. Station probabilities for each station being the DS from the SHAY station on February 13 at 8am. The station probabilities in blue are the probabilities generated from the BART dataset for 2023 and the probabilities in red were generated from the decision tree.

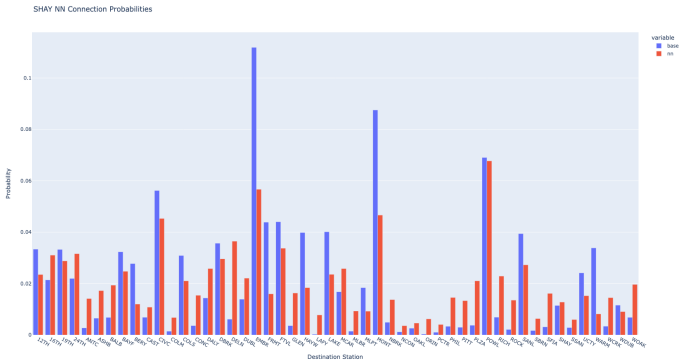

Fig. 14. Station probabilities for each station being the DS from the SHAY station on February 13 at 8am. The station probabilities in blue are the probabilities generated from the BART dataset for 2023 and the probabilities in red were generated from the ANN.

trend in the data that isn't obvious to the naked eye. The BLSTM is also similar to the LSTM, without any obvious trends.

While these observations are interesting, it can be hard to understand what they mean and how they relate to the accuracy of the models. Because the decision tree and random forest closely match the record, they could possibly be overfitting to the training data and may not be super useful for generalized data. On the other hand, the ANN and RNN may be able to more accurately predict test records, due to their generalization. Similarly, the LSTM and BLSTM are also generalized and might be capturing some of the temporal patterns (since they don't match the base probabilities that well), although this isn't guaranteed.

The models predict the DS as the station with the highest probability. Because of the low probabilities of every station, since there are so many, we can already tell that the models may not be very accurate. None of the models are able to correctly predict this example's DS, as you can see by Table II.

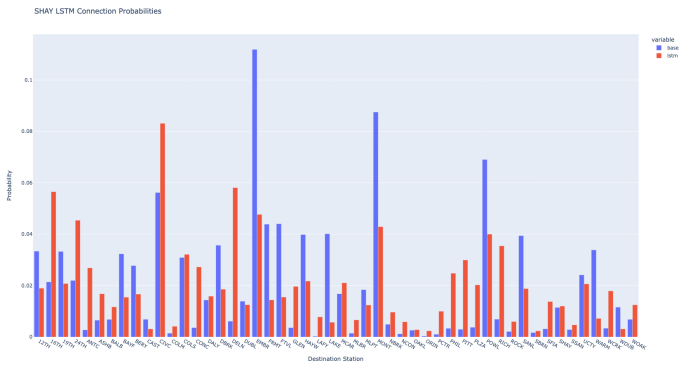

Fig. 15. Station probabilities for each station being the DS from the SHAY station on February 13 at 8am. The station probabilities in blue are the probabilities generated from the BART dataset for 2023 and the probabilities in red were generated from the LSTM.

TABLE II
EXAMPLE PREDICTIONS AND PROBABILITIES FROM SHAY

Model	Predicted DS	Probability
Observed	HAYW	N/A
DT	EMBR	0.104
RF	EMBR	0.104
ANN	POWL	0.068
RNN	EMBR	0.071
LSTM	CIVC	0.083
BILSTM	POWL	0.055

DT = Decision Tree, RF = Random Forest
EMBR = Embarcadero, POWL = Powell St.
CIVC = Civic Center

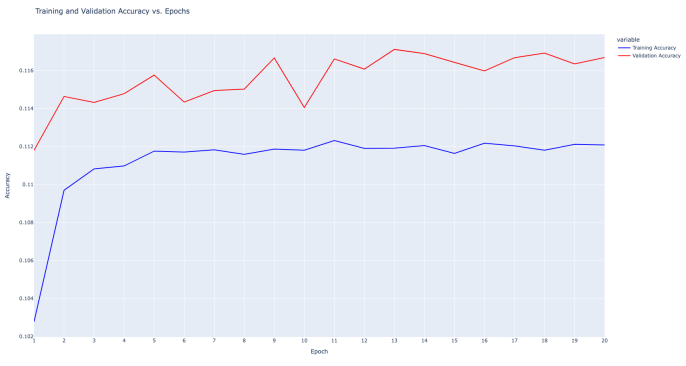

Fig. 16. Accuracy of the ANN model for each epoch during training

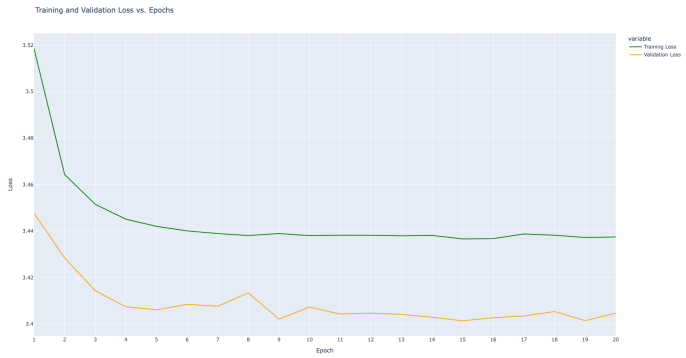

Fig. 17. Loss of the ANN model for each epoch during training

B. Model Accuracy

The accuracy of the models can be crucial, however it may prove difficult to get a good accuracy due to the lack of correlation described in IV-A. However, it's important to note that, with 50 possible DSs, there's a 2% chance of a random guess being right. So, if the model performs better than 2%, then it may be useful.

1) Training: Each model performed slightly differently during training. From Figures 16 and 17, we can see the ANN during training. The model converged very quickly, in only 20 epochs. The validation accuracy isn't very good, reaching around 11.2% accuracy and 3.4 loss.

The RNN followed a similar training pattern, converging in 22 epochs. However, it ended up having a slightly better accuracy and loss than the ANN.

One interesting characteristic about the training history for these two models is that the validation accuracy/loss was better than the training accuracy/loss. This could be due to not enough training time, since early stopping was used. Another possibility is that the ANN and RNN could be very generalized so they perform better on data it hasn't seen, as demonstrated in Section VI-A.

The LSTM and BLSTM performed better in accuracy and loss than both the ANN and RNN, finishing training in 94 and 79 epochs respectively. However, there's the possibility that the BLSTM overfits the data as seen by Figures 18 and 19. We can see that the validation accuracies for each model are about equal. However, the LSTM's training accuracy approaches a limit, while the BLSTM's training accuracy continues to increase as the validation accuracy stays constant.

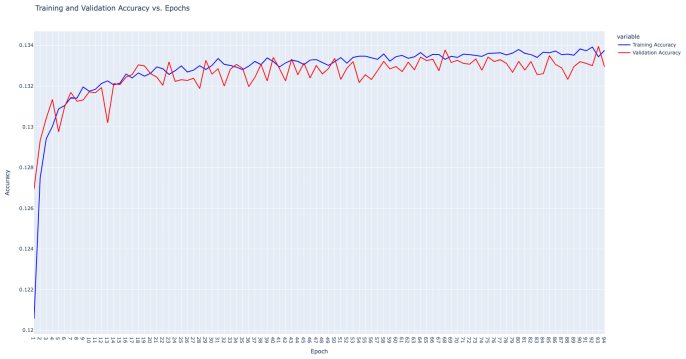

Fig. 18. Accuracy of the LSTM model for each epoch during training

2) Accuracy, Precision, Recall, and F1-Score: Getting the final results of every model, we can see how they perform on the test data with Table III.

We can see that the LSTM is the most accurate model, with an accuracy 6.6x better than randomly guessing. The LSTM also has the highest precision and recall as well, although the BLSTM has a slightly higher F1-Score. The lowest accuracy

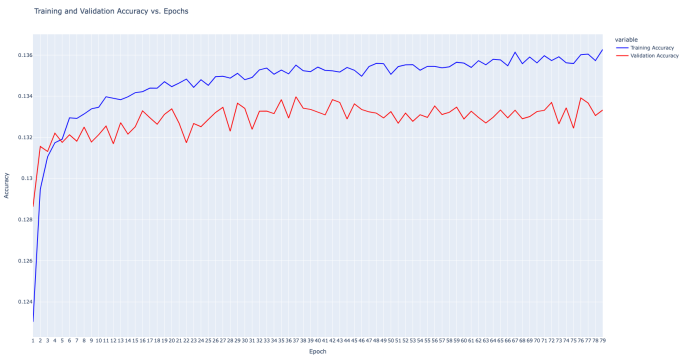

Fig. 19. Accuracy of the BLSTM model for each epoch during training

TABLE III
MODEL EVALUATION METRICS

Model	Accuracy	Precision	Recall	F1-Score
DT	0.121	0.054	0.121	0.067
RF	0.121	0.054	0.121	0.067
ANN	0.118	0.042	0.118	0.055
RNN	0.126	0.084	0.126	0.083
LSTM	**0.132**	**0.100**	**0.132**	0.095
BILSTM	0.132	0.089	0.132	**0.096**

models, DT and RF, are also still an improvement. They're 6.05x better than randomly guessing. Even though the model accuracies aren't very high, because there are so many possible classifications and the data isn't correlated, they're still an improvement and have found some patterns in the data.

All of the models also have a higher recall than precision, which will mean that they'll have less false positives. This is beneficial, as described in Section IV.

C. Accuracy Based on Reasoning

Producing the map with the example from Section VI-A and the LSTM model gives Figure 20.

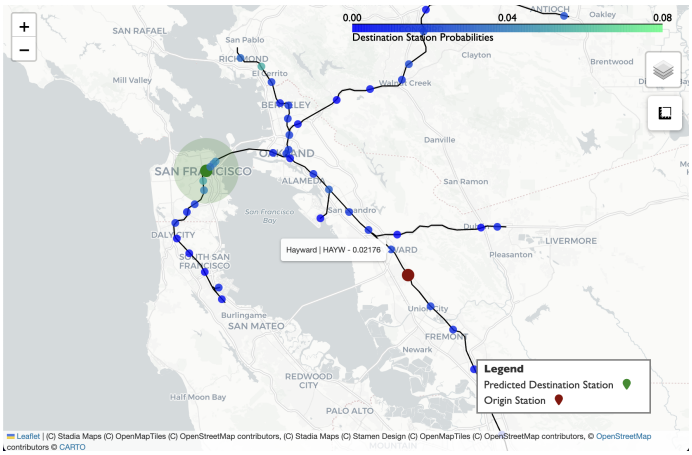

Fig. 20. Map of the station probabilities for each station being the DS from the SHAY station on February 13 at 8am using the LSTM model. The observed DS is HAYW and the predicted DS is CIVC.

Even though the model doesn't correctly predict the DS, the map is meant to be used as a suggestion, not as fact. Keeping this in mind, the predicted DS being CIVC is the fourth most traveled to station in this scenario, from Figure 12.

It also associates higher probabilities with stations that have high observed probabilities and are near CIVC, such as EMBR, Montgomery St. (MONT), and POWL. The stations 16th St. and 24th St. don't have high observed probabilities, but are near CIVC. There is an outlier, El Cerrito del Norte (DELN), that isn't near the other stations, yet still has a high predicted probability.

These stations make sense in relation to the predicted DS CIVC, since the person might go to nearby stations as the DS. However, due to the incorrect predicted DS, these aren't near the observed DS. The produced map may provide inaccurate results, but because it's a suggestion and better than blindly guessing a station with a 2% chance of achieving the right station, it's a useful tool.

VII. FUTURE WORK

Obviously there's a lot of room for improvement. Due to the lack of documentation, research, and data in the area, this is a hard problem to tackle. Yet, there are some definable improvements to be made on this work.

A. Data

At the forefront of the problems is a good dataset. Finding data with features similar to or containing those in Section III-C would be especially useful. This would allow the model to be generalized towards missing persons cases, so that it could learn what a missing person might do based on their history and details. As it currently exists, the model assumes that a missing person will "follow the crowd," which isn't always the case.

Another option is to include the missing person's travel habits. An example of this would be the person working a 9am-5pm job and using the BART every weekday to travel from station A to station B. Then, further training the model on these kinds of travel habits would yield results specific to the missing person. However, there is no obvious evaluation metric for this kind of data, unless label DSs were given.

B. Frontend

The program is still missing a UI. A streamlined frontend would allow first responders to easily input the facts of the missing person case and produce the map within seconds, without a computer science background. This would also enable more feedback to be given, as they could provide input on where the model succeeds and where it fails.

C. Model Improvements

With the LSTM model achieving an accuracy of 13.2%, it's a significant improvement over guessing, but could still be improved. Ideally the model produces 100% accuracy, otherwise it is only useful a small portion of the time. Tuning the hyperparameters further could produce some marginally better results.

Different models could also be used, such as some mentioned in Section III. Additionally, the idea of POA could be introduced to the map. It already exists in part with the 1 hour walking circle, but that could be improved by limiting the "circle" to only where a person can actually walk (roads, sidewalks, etc.). This would more accurately simulate where a person could walk to in a given time range, since it's based on the map features.

D. Increasing Coverage

Currently, the widget only covers the BART. Ideally, it spans across the entirety of the US and potentially other countries as well. This could be done by retrieving a standardized format for public transit and missing person information, then the model could produce results anywhere. This idea could also expand to other forms of public transit, such as buses, trains, etc. Even branching out from public transit, any transportation that consists of "stations," such as planes, could be analyzed.

VIII. Ethics

Despite the intent to create a tool to help first responders find missing persons, the widget could easily be used for malintent. It predicts where a person might go if they use public transit, no matter if the person is missing or not. In the future, it could even predict where the person might go given their travel history, background, details, etc. Additionally, there exists personal data that isn't meant to be shared outside of the scope of first responders.

As such, it's important to ensure that, should be widget continue to develop to more accurately predict where a missing person might go, it should remain in the hands of first responders and not be available to the public. The model isn't meant to be used to predict where any person may go, but only be used to help first responders find a missing person as quickly as possible.

It makes sense that the model produces inaccurate results, despite the improvements over random guesses. If it were to be extremely accurate without much effort, then it means that peoples' private information is probably easily accessible online and can be used for malice.

For these reasons, the code will not be open-source.

IX. Conclusions

Using AI models such as decision trees, random forests, ANNs, RNNs, LSTMs, and BLSTMs, I'm able to effectively create a Public Transit Search Widget that will help first responders find missing persons using the BART. With the assumption that a missing person will "follow the crowd," the most accurate model, LSTM, is 6.6x better than a random guess.

The models produce probabilities that are fed into an effective map to give first responders suggestions on where to search. This map is saved as a .html file, allowing it to be viewed on almost any device. So, there are no hardware requirements to use the model.

References

[1] "Missing Persons Statistics," State of California - Department of Justice - Office of the Attorney General, Feb. 28, 2011. https://oag.ca.gov/missing/stats

[2] "BART ridership numbers were up in 2023. Here's how much," KRON4, Jan. 12, 2024. https://www.kron4.com/news/bay-area/bart-ridership-numbers-were-up-in-2023-heres-how-much/ (accessed May 19, 2024).

[3] "About — Bay Area Rapid Transit," www.bart.gov. https://www.bart.gov/about#:~:text=BART\%20operates\%20in\%20five\%20counties

[4] Melanie, "Classification algorithms: Definition and main models," Data Science Courses — DataScientest, Nov. 30, 2023. https://datascientest.com/en/classification-algorithms-definition-and-main-models

[5] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," in IEEE Transactions on Systems, Man, and Cybernetics, vol. 21, no. 3, pp. 660-674, May-June 1991

[6] "1.10. Decision Trees," scikit-learn. https://scikit-learn.org/stable/modules/tree.html#tree-mathematical-formulation

[7] V. Kulkarni, "Random Forest Classifiers :A Survey and Future Research Directions," International Journal of Advanced Computing, vol. 36, no. 1, pp. 2051-0845, Apr. 2013, Available: https://adiwijaya.staff.telkomuniversity.ac.id/files/2014/02/Random-Forest-Classifiers_A-Survey-and-Future.pdf

[8] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001, doi: https://doi.org/10.1023/a:1010933404324.

[9] G. Biau and G. Fr, "Analysis of a Random Forests Model," Journal of Machine Learning Research, vol. 13, pp. 1063–1095, 2012, Available: https://jmlr.org/papers/volume13/biau12a/biau12a.pdf

[10] M. S. B. Maind and M. P. Wankar, "Research Paper on Basic of Artificial Neural Network," International Journal on Recent and Innovation Trends in Computing and Communication, vol. 2, no. 1, pp. 96–100, Jan. 2014, doi: https://doi.org/10.17762/ijritcc.v2i1.2920.

[11] F. Abien and Agarap, "Deep Learning using Rectified Linear Units (ReLU)." Accessed: May 13, 2022. [Online]. Available: https://arxiv.org/pdf/1803.08375

[12] D. Wei, H. Wu, M. Wu, P.-Y. Chen, C. Barrett, and E. Farchi, "Convex Bounds on the Softmax Function with Applications to Robustness Verification." Accessed: Jun. 06, 2024. [Online]. Available: https://arxiv.org/pdf/2303.01713

[13] D. Kingma and J. Lei Ba, "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION," 2015. Available: https://arxiv.org/pdf/1412.6980

[14] R. Hecht-Nielsen, "Theory of the backpropagation neural network," Neural Networks, vol. 1, p. 445, Jan. 1988, doi: https://doi.org/10.1016/0893-6080(88)90469-8.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," Journal of Machine Learning Research, vol. 15, pp. 1929–1958, 2014, Available: https://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

[16] R. Schmidt, "Recurrent Neural Networks (RNNs): A gentle Introduction and Overview." Available: https://arxiv.org/pdf/1912.05911

[17] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 1, pp. 1–42, Jan. 1997, doi: https://doi.org/10.1162/neco.1997.9.1.1.

[18] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," IEEE Transactions on Signal Processing, vol. 45, no. 11, pp. 2673–2681, 1997, doi: https://doi.org/10.1109/78.650093.

[19] R. Hu, Y. Chiu, and C. Hsieh, "Crowding prediction on mass rapid transit systems using a weighted bidirectional recurrent neural network," IET intelligent transport systems, vol. 14, no. 3, pp. 196–203, Feb. 2020, doi: https://doi.org/10.1049/iet-its.2018.5542.

[20] "SMU Data Science Review SMU Data Science Review." Available: https://scholar.smu.edu/cgi/viewcontent.cgi?article=1136&context=datasciencereview

[21] P. J. Doherty, Q. Guo, J. Doke, and D. Ferguson, "An analysis of probability of area techniques for missing persons in Yosemite National Park," Applied Geography, vol. 47, pp. 99–110, Feb. 2014, doi: https://doi.org/10.1016/j.apgeog.2013.11.001.

[22] A. L. Ruiz-Rizzo, M. Eduardo, and J. Fredy, "Predicting the probability of finding missing older adults based on machine learning," Journal of Computational Social Science, vol. 5, no. 2, pp. 1303–1321, Jun. 2022, doi: https://doi.org/10.1007/s42001-022-00171-x.

[23] S. Lundberg, P. Allen, and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions.” Available: https://arxiv.org/pdf/1705.07874
[24] Pierzchała, Dariusz and Gutowski, Tomasz and Czuba, Przemysław and Antkiewicz, Ryszard. (2021). Machine Learning-Based Method for Recommendation of Missing Person’s ”Search Level”.
[25] L. Pfau, “Wilderness Search and Rescue: Sources and Uses of Geospatial Information,” 2013. Accessed: Jun. 07, 2024. [Online]. Available: https://handbook.geospatial.psu.edu/sites/default/files/capstone/PfauCapstone_Final_20130502.pdf
[26] H. Dalianis, “Evaluation Metrics and Evaluation,” Clinical Text Mining, pp. 45–53, 2018, doi: https://doi.org/10.1007/978-3-319-78503-5_6.
[27] “BART Ridership,” www.kaggle.com. https://www.kaggle.com/datasets/mrgeislinger/bartridership (accessed Jun. 07, 2024).
[28] M. L. McHugh, “The Chi-square Test of Independence,” Biochemia Medica, vol. 23, no. 2, pp. 143–149, Jun. 2013, doi: https://doi.org/10.11613/bm.2013.018.
[29] Visual Crossing Corporation, “Historical Weather Data & Weather Forecast Data — Visual Crossing,” Visualcrossing.com, 2021. https://www.visualcrossing.com/weather-data