Evaluating the Feasibility of Using Bluetooth Low Energy Signaling and Machine Learning for Proximity Detection and COVID-19 Contact Tracing

Samuel Wang
swang2023@gmail.com

***Abstract* —** During a global pandemic of COVID-19, contact tracing is key to curb the disease spread. The use of readily available Bluetooth Low Energy (BLE) signaling for automated exposure detection may augment the current labor-intensive and error-ridden manual contact tracing. To assess its feasibility, systematic studies were conducted to evaluate the impact of various factors on the Bluetooth received signal strength. In addition, various data analysis approaches were evaluated. A traditional method of linear regression-based analysis can be used to predict distances based on the Received Signal Strength Indication (RSSI) value of BLE in ideal settings. However, external factors such as orientation of the device can have detrimental effect on its accuracy and reliability. On the other hand, classification approaches based on machine learning is more suited to tackle this problem because of the binary nature in the contact tracing protocols. This paper presents the systematic studies of the relationship between proximity and RSSI in different settings under controlled environment. The relative orientation of the receiver is found to substantially impact the received signal strength. Three machine learning models were trained on the MIT-Matrix iOS Dataset and further tested using several datasets collected using Raspberry Pi BLE. Experimental findings indicate that machine learning models outperform traditional linear regression-based method in accuracy and reliability. A model of this fashion could prove instrumental to the success of an automated privacy-preserving contact tracing system.

***Keywords* —**

- ***Bluetooth Low Energy***
- ***Received Signal Strength Indication***
- ***Machine Learning***
- ***Contact Tracing***
- ***Private Automated Contact Tracing***
- ***Coronavirus***
- ***COVID-19***

I. Introduction

A. Project Description

Private Automated Contact Tracing (PACT) is a collaborative project led by MIT. PACT's mission is to enhance contact tracing in pandemic response by designing exposure detection functions in personal digital communication devices that have maximal public health utility while preserving privacy [1]. A key challenge with the PACT project is the proximity detection, which is needed to identify two individuals being closer than 6 feet for too long and thus leads to increased risk of COVID-19 infection [1].

This project use prototype device built using two Raspberry Pis to assess various factors that can influence the accuracy of proximity measurement using the Bluetooth received signal strength. Data are further analyzed using various models to develop algorithm for automated detection of closer than 6 feet proximity. By incorporating additional sensors and machine learning models, piPACT's objective of creating an automated contact tracing algorithm can potentially be satisfied. To demonstrate the device interoperability and model robustness, the machine learning models were first trained using the MIT-Matrix dataset collected using iOS devices with MIT-PACT testing protocol, and then subsequently tested with the data collected with the BLE sensors on Raspberry Pis. Additional information on device orientation was collected using MPU-9250 chip and added to the Raspberry Pi data set to correct its influences on received signal strength.

B. Background Information

As of July 26, 2020, there have been a total of 16.1 million infections and 0.6 million deaths as a result of the COVID-19 pandemic [2]. In US alone, more than 4.2 million people have been infected by COVID-19 with 146k reported death [2]. At the time of this paper, neither a cure nor a vaccine have been developed for this deadly illness [3, 4].

In the absence of a cure or vaccine, contact tracing is imperative to reducing the amount of cases and fostering a safe return to normalcy [5]. A successful implementation of effective contact tracing will enable not only more timely notification for the people exposed or at-risk, but also prompt quarantine, and ultimately slow the spread of the infection [6].

There are various methods of contact tracing. The most basic form commonly used is manual contact tracing. In this method, patients who test positive report the people they had in contact in the past 14 days to a human contact tracer. Manual contact tracing is not only time-consuming and costly, but also less reliable because its accuracy relies solely on the patient's ability of remembering the identity of their contacts, which is not always dependable, sometimes can be impossible in case of the encounter of strangers.

Automated contact tracing seeks to fix this problem by using technology to augment the tracing of the exposed [7]. There are three main types of automated contact tracing: accurate location-based, centralized proximity-based, and decentralized proximity-based.

Location based contact tracing has been implemented in countries such as China, India, and Iran using GPS [8]. The location of each user is tracked, and proximity of users can be calculated accurately using their GPS data. However, this form of contact tracing is not feasible to roll out in the US due to the myriad of privacy and security concerns associated with real-time individual location data being tracked.

Proximity-based contact tracing methods use relative rather than absolute positioning and are feasible to implement in the United States [9]. Typically, these types of contact tracing operate by collecting the Bluetooth chirps Proximity-based contact tracing methods use relative rather than absolute positioning and are feasible to implement in the United States. Typically, these forms of contact tracing operate by collect the Bluetooth chirps from all nearby devices, storing them, and then checking if the identifier becomes infected later. Centralized and decentralized proximity differ in that the former checks on the exposure in the database in a centralized location, such as government health agencies, whereas the latter allows individual to download the infected identifiers from the database and check his/her exposure with complete privacy. Decentralized tracking is much more secure and privacy-preserving because it requires less data to be stored in a central database that could be targeted by malicious actors. Due to this benefit decentralized tracking is favored by the US and has been implemented in countries such as Britain and Canada [7].

The adoption and success of a contact tracing solution are largely dependent on public and industry approval. Public approval of a contact tracing app is tied closely to its assurance of security and privacy. Additionally, for industry approval, the creators of Android and iOS (Google and Apple, respectively) have committed to user privacy and decentralized contact tracing apps [10].

Another key requirement for successful contact tracing is that the tracking technology used is energy efficient. This ties back into the importance of public and industry approval, as users expect their devices to maintain the same functionality as before installing the contact tracing app. Therefore, BLE is a popular option because it is already in many devices and operates using minimum energy [11].

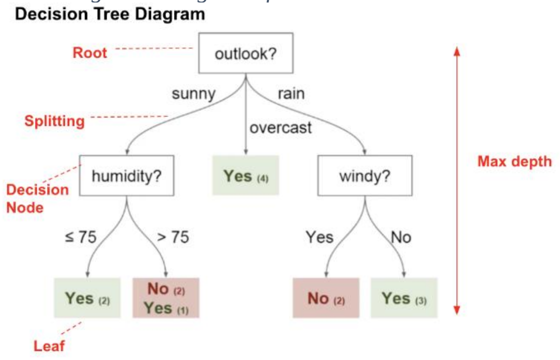
BLE is a protocol of Bluetooth. It relies on the same technology but is fundamentally different in that it transfers far less data than typical Bluetooth streaming. All modern phones are equipped with this technology, making it a highly effective sensor for contact tracing apps.

RSSI can be the metric used to measure the proximity for contact exposure. RSSI is always negative and ranges from -26 dBm (a few inches) to -100 dBm (40-50 m) [12]. Another piece of data that can be received from a BLE measurement is the universally unique identifier (UUID). The UUID is a unique identifier for a device that changes every 15 minutes and cannot be feasibly traced back to a certain device, satisfying privacy preservation requirements.

While we can treat contact tracing as a linear regression problem, it is optimal to treat it as a classification problem. This is because the CDC guidelines are binary [13]: if two people are within six feet for an extended period of time, they are at-risk; otherwise they are not. This means the exact distance measurement is not necessary. Therefore, machine learning, which is commonly used to solve these types of classification problems, is a logical approach for algorithm development. Supervised models were chosen for use in this experiment because the desired output is deterministic: a contact is either determined to at-risk or not.

Total three machine learning models were tested. The first one is Decision Tree classifiers, which are simple classifiers based on branching logic chains [14]. An example of a decision tree can be seen in Figure I [15]. The branches of the "tree" are questions about the data, guiding the model to conclusions about the class of the input, which are represented by the leaves of the tree. A consideration for decision trees is their tendency to over fit to the training set, in which a model creates correlations between the dataset and the result that do not exist.

Figure I. Diagram of a Decision Tree



To help solve the problem of overfitting, random forests were used as a second model. Random forests rely on an ensemble of decision trees to create predictions [16]. The output of the classifier is formed by the mode of the output of the trees. Each of these decision trees look at a subset of the features, rather than all of them. Random forests' usage of the collective conclusion of multiple decision trees reduces the overfitting that commonly affects decision tree classifiers. Random forests can be robust, an essential characteristic of models that have solved the problem of overfitting.

Naïve Bayes classifiers were used as a third model in this experiment. Naïve Bayes classifiers differs from Decision Trees and Random Forests because they are non-parametric, meaning they do not create rules [17]. This means that they do not assume anything about the structure of the data. Naïve Bayes is a probabilistic classifier which applies Bayes' theorem. A key condition of Naïve Bayes is the features are conditionally independent or Naïve, which allows it to be closed form. Due to the training being closed form, Naïve Bayes models can be trained in linear time, making them much less time-expensive than their peers.

Scikit is one of the primary modules in this experiment and was used heavily for the modeling portion. Scikit-learn is a powerful Python module for machine learning which contains regression, classification, model selection, and many other functions. The module had some portions written in Cython (an extension to Python that uses the C programming language) because it improves performance.

It should be noted that these current experiments do not address the influences from in indoor vs outdoor locations, humidity, temperature, surroundings, or other forms of interference. Such factors were kept as constant as possible throughout the experiment.

II. Hypotheses

Hypothesis 1: BLE RSSI measurement can be used for distance determination in ideal situations.

Hypothesis 2: Other factors, such as orientation of the advertiser and scanner will impact the accuracy of BLE Measurement.

Hypothesis 3: Traditional linear regression between the distance and RSSI Trendline can only work in ideal scenarios, but are not fit for more complex real-life situations.

Hypothesis 4: Adding additional sensors to assist the BLE improves the accuracy

Hypothesis 5: Machine Learning Classifiers will improve predictive accuracy compared to traditional regressive models.

III. Experiments and Data Collections

The experiments for each hypothesis are summarized in Table I.

TABLE I. Experiment Overview

Exp #	Hypothesis	Objectives	Rep
1	None.	Establish the percentage variance of RSSI strength	2
2	BLE can be used for distance measurement in ideal settings	Assess whether BLE is viable for contact tracing	2
3	Other factors will impact the accuracy of BLE strength	Test the impact of vertical rotation	4
4	Other factors will impact the accuracy of BLE strength	Assess the impact of horizontal rotation	4
5	Other factors will impact the accuracy of BLE strength	Test the effect of occlusion by clothing	1

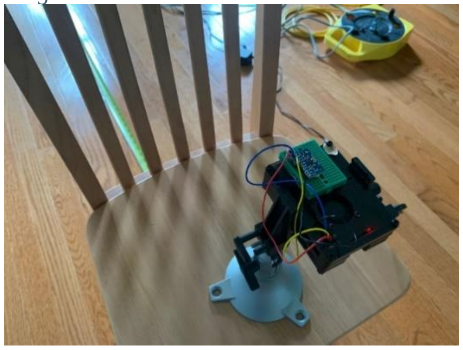
Note: Rep stands for the number of repetitions for each experiment

A. Plan and Execution

Two Raspberry Pis were used in the experiments: one was stationary serving as an advertiser which broadcasted a BLE signal while the other was designated as a scanner collecting packets and recording them as csv files.

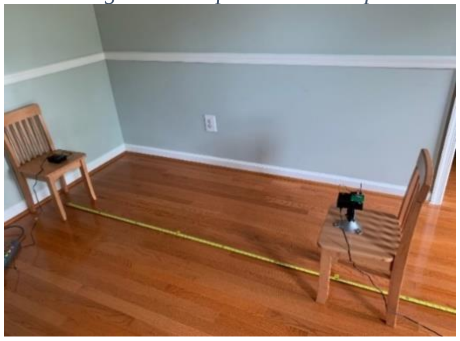
- To ensure controllability/reproducibility of the data, the scanner Raspberry Pi which has a MPU-9250 chip (A 9-axis Accelerometer-Gyroscope-Magnetometer inertial measurement sensor) was secured with a small vise that allows it to be set at different angles both horizontally and vertically (Figure II).

Figure II. Scanner with MPU-9250 on a vise



- The experiments were carried out in multiple days where temperature and humidity variation were minimum and therefore not taken into consideration.
- The reference code provided by MIT PiPact was modified to enable calibration and data collection from the MPU-9250 chip that was added to the Scanner Pi. The code was also configured to support the packetization of data and feed buckets into models Otherwise, the code remained unchanged.
- The experiments were carried out in a large room with all the furniture removed. The advertising Raspberry Pi is fixed on a small chair while the scanner Pi on an identical chair can be place at various distances from the broadcaster while the Bluetooth chips of both Pis are aligned in a straight line (tape measure on the floor) as shown in Figure III.

Figure III. Experimental setup



The experiment was designed to be as uniform as possible between different measurements. Two identical chairs of equal heights were used to elevate the two Raspberry Pis and a measuring tape was used to mark off different distance intervals for the scanner to be moved.

A-1. Distance vs RSSI

In Experiment 1, the variability of RSSI measurement was established by taken a series RSSI value with the exact same condition at distance of 3 feet.

In Experiment 2, a baseline was created by collecting measurements of the RSSI vs Distance in an ideal indoors scenario: everything was kept the same as much as possible; the two Raspberry Pi devices were elevated to the same height while the distance between the two Raspberry Pis was varied between six intervals: 1, 3, 6, 8, 10, and 12 feet. They were also placed flat with ports facing each other to ensure proper alignment of the BLE chips. This minimizes any effects of orientation effects in this experiment, which was in the hypothesized to have an impact on the RSSI of the devices (Hypothesis 2). The position of the scanner was varied between intervals, whereas the advertiser remained stationary. At each distance, the scanner collected the BLE data from the advertiser for 4 minutes, resulting in approximately 250 measurements. These points were sorted into around 50 "buckets" each consist of the average of accumulated data during 5-second intervals. The term "bucket" will be explained with more depth in the Analysis and Algorithms Subsection.

A-2. Effects of Oritations on RSSI

In Experiment 3, the impact of vertical rotation of the scanner on the RSSI measurements was evaluated. Through these series of experiments, the distance between the Scanner and Advertiser was set constant at 3 feet and 6 feet. A vise was used to secure the Scanner while it was carefully rotated through the four different angles of 0, 90, 180, and 270 degrees on the vertical axis, with the vector horizontally pointing the Advertiser defined as 0 degree.

In Experiment 4, the impact of horizontal rotation of the scanner on the RSSI was evaluated in similar fashion as Experiment 3, only the Scanner was rotated through the four different angles of 0, 90, 180, and 270 degrees on the horizontal axis, with the vector horizontally pointing the Advertiser defined as 0 degree.

A-3. Effects of Occlusion on RSSI

In Experiment 5, the impact of occlusion on RSSI measurement was assessed by covering the Scanner with different materials, including none, jeans, cotton shirt, winter jacket, fleece, sweater and windbreaker (a very light jacket). The distance and orientation of the two Raspberry Pis were kept constant.

B. *Data Relevance*

Experiment 1 established the variability of measurement of RSSI in current setting.

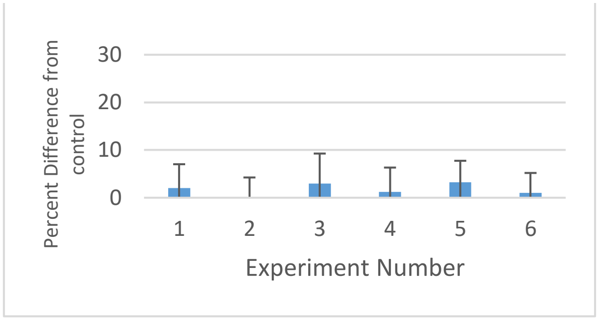
Experiment 2 was conducted to test Hypothesis 1, which states "BLE RSSI measurement can be used for distance determination in ideal situations".

Experiment 3, 4 and 5 were carried out to test Hypothesis 2, which states "Other factors, such as orientation of the advertiser and scanner will impact the accuracy of BLE Measurement". The combined datasets will be used in the second round of traditional linear regression between distance and the measured RSSI value to test Hypothesis 3, which states "Traditional linear regression between the distance and RSSI Trendline can only work in ideal scenarios, but not fit for more complex real-life situation.". And the results also will provide evidence that will test Hypothesis 4, which states "Adding additional sensors to assist the BLE improves the accuracy".

C. *Examples*

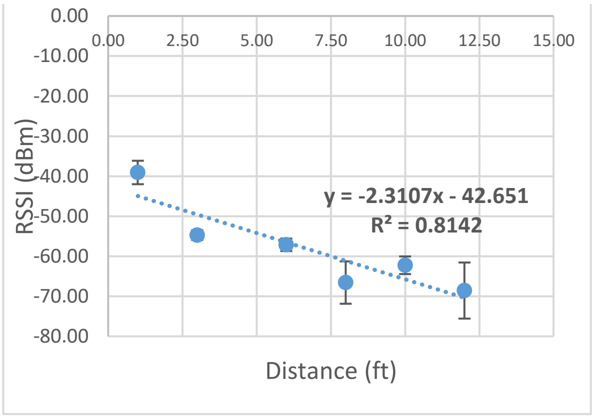
The variability of RSSI measurement was determined by data from Experiment 1. As shown in Figure IV, the RSSI can have a percent difference around roughly 3.2%, even in exact conditions. This established the baseline intrinsic variability for all other RSSI measurements. As such the threshold for a variable satisfying Hypothesis 2 was set as having a percent difference higher than 5%.

Figure IVIV. Percentage Difference of RSSI under Identical Conditions



Hypothesis 1 was confirmed by data from Experiment 2 as shown in Figure V. A traditional linear regression of the measured RSSI vs the distance between the two Raspberry Pis yielded a correlation with $R^2 = 0.81$ in the ideal and controlled indoor scenario. There is a steady inverse relationship between distance and RSSI measurement. Additionally, as the distance between the two devices increased, so did the amount of deviation.

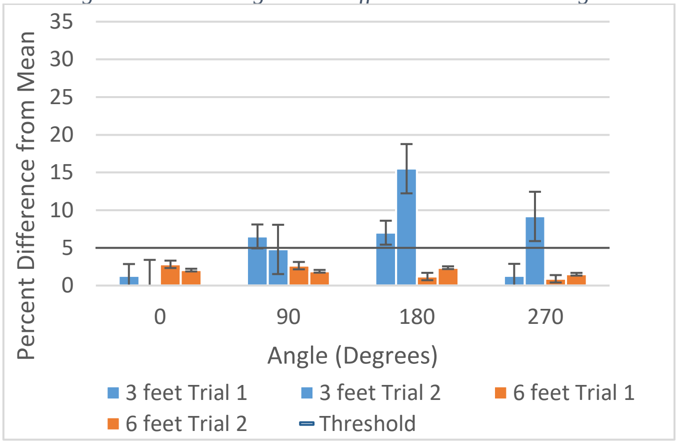
Figure V. RSSI VS Distance in Ideal Indoor Situation



Due to the effects of rotation, as presented by Experiment 3 and 4, Hypothesis 2 was also confirmed.

In Experiment 3, the relationship between vertical rotation and RSSI values at distance of both 3 feet and 6 feet were each measured twice. Even though the devices remained at a constant distance, the change in vertical rotation affected the RSSI value, as shown by the bars chat of percentage RSSI difference vs vertical angles in Figure VI. There was a maximum range of 12.7 dBm between the four different angles and a maximum percentage RSSI difference of 15.5%, statistically significant over the 5%, normal variability of RSSI established in Experiment 1.

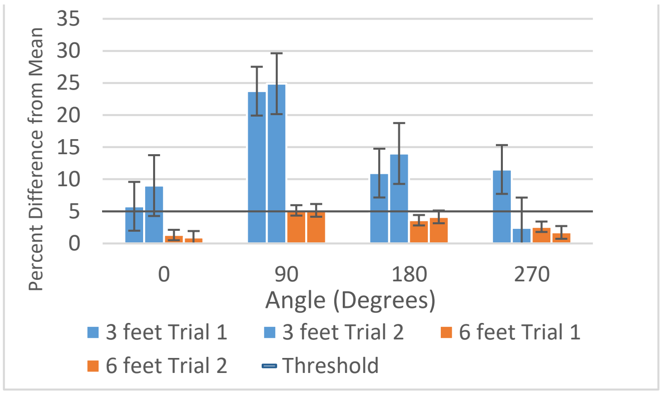
Figure VI. Percentage RSSI Difference vs Vertical Angles



Similarly, Experiment 4 provided the relationship between horizontal rotation and RSSI at both 3 feet and 6 feet distances. Again, even though the distance between the two devices remained constant, the changes of horizontal orientation affected the RSSI measurement. In Figure VII, The RSSI value had a range of 20 dBm through the experiment and a maximum percent difference of 24.9%. The point at 90 degrees and 3 feet was believed to be an outlier, but this was retested six times, resulting in a maximum percent difference of 4.14% and a range of 5.9 dBm.

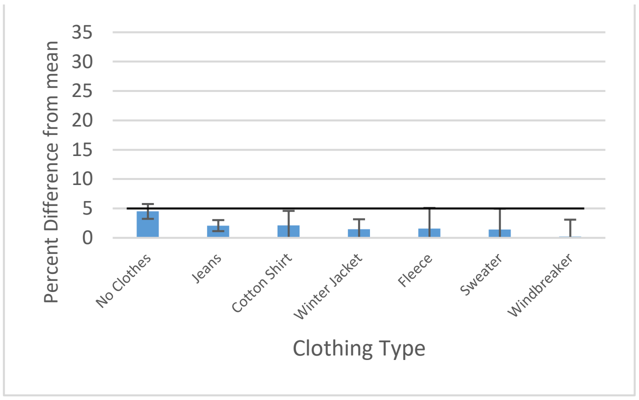
Through Experiments 3 and 4 on angles, it was found that there is a larger variance at 3 feet than 6 feet. In experiment 3, vertical rotation, the Average percent difference was 5.6% at 3 feet, whereas it was 1.9% at 6 feet. In experiment 4, vertical rotation, the Average percent difference was 9.8% at 3 feet, whereas it was 3.1% at 6 feet. Given the threshold for significant percent difference established through experiment 1 is 5%, rotation at 3 feet on average was a significant factor, whereas for 6 feet it was not found to be one.

Figure VII. Percentage RSSI Difference vs Horizontal Angles



Experiment 5 shows that there is some effect of different clothing on the RSSI measurement. However, due to the difficulty in determining the clothing of the user, this effect was deemed negligible and was ignored by the rest of the experiment.

Figure VIII. Percentage RSSI Difference vs Different Occlusion



IV. Analysis and Algorithms

A. Description

The MIT-Matrix Dataset is a dataset collected by PACT members. The MIT Testing protocol was used to collect it. Data from the Range, Angle, Activity, Bluetooth, Heading, Gyroscope, Attitude, Gravity, Magnetic-field, Pedometer, and Altitude sensors were all recorded as features in the collection. Data was collected at 10 distance intervals: 3, 4, 5, 6, 8, 10, 12, 15 feet.

The MIT-Matrix Dataset organizes its data in log files. Each entry that is recorded is of a specific sensor at a given timestamp. To efficiently feed the data into machine learning models, pre-processing of the log files was done by converting the log files into separate “buckets”. A bucket contains all the sensor readings that occurred during a certain time interval and each sensor data point is averaged. This means that once every bucket is processed, the size is the uniform, regardless if there were more readings in a specific bucket. This also helped remove the dimension of time, as the model can approach each bucket independently and make a prediction as to whether the devices were within six feet during the time interval covered by the bucket.

In the MIT-Matrix Dataset, each sensor reading was collected at different intervals and some sensors were not used in certain experiments. In the parser we wrote, any sensors that were not used throughout all the experiments in this dataset were removed because of the inconsistency it would cause between different buckets in the final processed dataset. For sensors missing in a bucket, but not an experiment, previous values from the same sensor were filled in instead by the parser. This is an inherently flawed assumption. However, due to the lack of data to begin with, the parser was designed to maximize the amount of the processed data the models could be trained on.

The parser algorithm was written with Python, and periodically employed the use of various Python Modules. To best explain this algorithm, we will split it into two sections: Bucket Collection and Bucket Processing.

During the bucket collection period, data from the log files were read in and placed into a 3-level dictionary. This dictionary had three levels of keys: first the distance between the devices, then the bucket id, and the sensor name last. The log files were iterated through with the Python module OS. To find the bucket id, the difference between the starting time of the experiment (defined as the timestamp of the first entry) and the current entry's timestamp was floor divided by the set size of the buckets.

In the bucket processing section, the data in the multilevel dictionary created in the first section was compacted into one row of a CSV file. To do this, the dictionary was iterated through and each bucket was processed with a helper function. In this function, the sensors of each bucket had all their values processed. If the sensor was missing in each given bucket, then a helper function backfilled with the last found value for the sensor. The resulting processed data was then dumped into a CSV file.

The models from the Sklearn modules were then trained on the processed data set. These models were Linear Regression, Decision Tree Classifier, Random Forest Classifier, and Naïve Bayes Classifier.

Each of these models was trained on the processed data from the MIT-Matrix Dataset, which had 1161 data points. The training was repeated 5 times for each model architecture with randomized subsets of the data and the resultant models were pickled using the joblib module so that they could be loaded later.

To select the best model, cross-validation experiments were executed, using the distance data collected on the in Experiment 1. The training and testing data were intentionally collected on different device types to test the robustness of the models. In addition, the metrics of Area Under the ROC Curve (AUC or AUROC), recall, and precision were recorded.

Area Under Curve is calculated by taking the definite integral under the Receiver Operating Characteristic Curve (ROC Curve). The ROC Curve plots the True Positive Rate against the False Positive Rate at different classification thresholds. The lower the classification threshold, the more items marked positive, increasing the number of True and False positives. An example ROC from this experiment can be found in 0. The higher the AUC, the better the model performs. AUC was maximized in this experiment.

Recall is the ratio of correctly predicted positive examples and actual positives. A model that only produces false negatives has a recall of zero, whereas a model that produces no false negatives has a recall of one. Recall is valued more when False Negatives are worse than False Positives in a problem.

Precision is the ratio of correctly predicted positive examples and the total number of points predicted to be positive. A model that produces only false positives has a precision of zero, whereas a model producing no false positives has a precision of one. Precision is valued more when False Positives are worse than False Negatives.

Recall is typically inverse to Precision. In this problem, the cost of a false positive is stress on a person for two weeks, who believes that they falsely were exposed. The cost of a False Negative is keeping someone who was exposed on the street and as a potential carrier. Thus, we believe the cost of a False Negatives is greater than that of a False Positive in this problem. Therefore, although both metrics were considered, we valued recall over precision in this experiment.

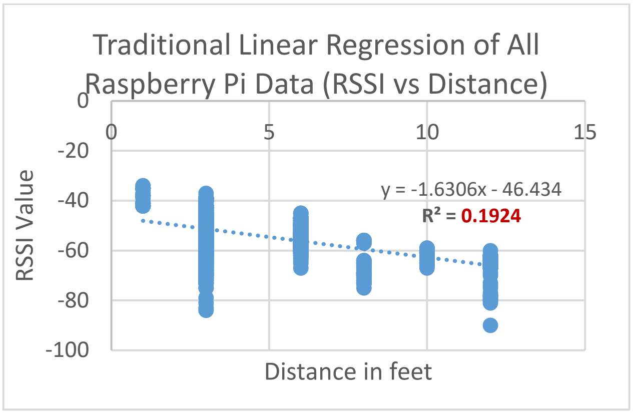
To test Hypothesis 3 a linear regression model was trained on the MIT-Matrix Dataset and was compared to the classifiers in this experiment. To test Hypothesis 4, a linear regression model was trained only on Bluetooth Measurements in the MIT-Matrix Dataset and compared to the AUC from the other models.

B. Results and Examples

The data collected in Experiment 2 was in the format of a N x 2 matrix. It was analyzed by a traditional linear regression model to evaluated whether or a linear relationship existed between RSSI and distance in an ideal situation. Evidence of a linear relationship would imply that Hypothesis 1, i.e. Bluetooth can be used for distance determination would be true. An R^2 indicates the fit of the model to the data. Given an R^2 of 0.81, with a maximum of 1, we can reasonably conclude that there is a linear relationship between distance and RSSI, confirming Hypothesis 1.

After combining all the data collected using Raspberry Pis including the ones from Experiment 3-5 where orientation and occlusion changes were included, another traditional linear regression analysis was performed on the whole Raspberry dataset. As shown in Figure IX. Traditional Linear Regression on Combined Raspberry Pi RSSI Data vs Distance, now the correlation only has $R^2 < 0.2$. This along with the result from Experiment 2, provided direct evidence to support Hypothesis 3, which states "Traditional linear regression between the distance and RSSI Trendline can only work in ideal scenarios, but are not fit for more complex real-life situations".

Figure IX. Traditional Linear Regression on Combined Raspberry Pi RSSI Data vs Distance



The data collected from the Pi in Experiments 1 - 5 was used to test all the models. Tables II-IV through show the results from these tests. To compare the models, the AUC and recall were used as primary metrics. Given this, the best model was the was Decision Tree 2, which had an AUC of 0.7, a recall of 0.46, and a precision of 0.89.

This data also confirmed Hypothesis 5 "Machine Learning Classifiers will improve predictive accuracy compared to traditional regressive models". The classifiers all vastly outperformed the regressor, which had an R^2 value of less than 0.2. On the other hand, all the Machine learning classifiers had an AUC greater than 0.49.

FIGURE X. AN EXAMPLE ROC CURVE

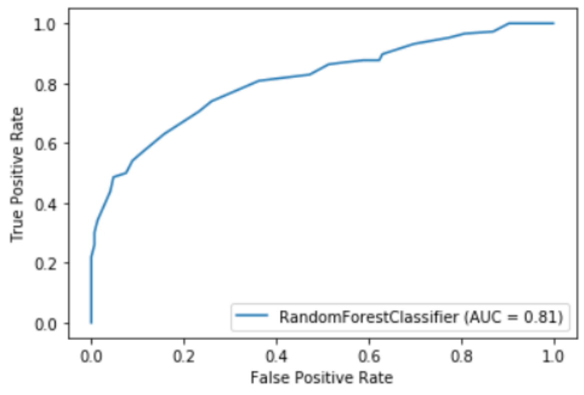


TABLE II. DECISION TREE CLASSIFIER

Model Number	AUC	Recall	Precision
Decision Tree 1	0.54	0.53	0.54
Decision Tree 2	0.70	0.46	0.89
Decision Tree 3	0.50	1.00	0.50
Decision Tree 4	0.58	0.26	0.73
Decision Tree 5	0.55	0.56	0.55

TABLE III. RANDOM FOREST CLASSIFIER

Model Number	AUC	Recall	Precision
Random Forest 6	0.69	0.38	1.00
Random Forest 7	0.60	0.37	0.69
Random Forest 8	0.62	0.25	1.00
Random Forest 9	0.69	0.38	1.00
Random Forest 10	0.60	0.19	1.00

TABLE IV. NAÏVE BAYES

Model Number	AUC	Recall	Precision
Naïve Bayes 11	0.50	0.00	0.00
Naïve Bayes 12	0.49	0.54	0.49
Naïve Bayes 13	0.50	0.45	0.50
Naïve Bayes 14	0.52	0.10	0.61
Naïve Bayes 15	0.51	0.01	1.00

V. CONCLUSIONS

A. *Hypothesis Evaluation*

Hypothesis 1, "BLE can be used for distance determination in ideal situations" was confirmed through Experiment 2. By proving this hypothesis, we reaffirmed the feasibility of BLE as an appropriate signal for contact tracing. Without this foundation to build on, there would be no reason to evaluate the remaining hypotheses. As shown in Section III-C, there is a clear negative linear relationship (Figure V) between RSSI and distance in between the two devices.

Hypothesis 2, "Other factors, such as orientation of the advertiser and scanner will impact the accuracy of BLE Measurement" was confirmed by results from Experiments 3-5. Specifically, our experimentation showed that horizontal and vertical orientation have a significant impact on RSSI. As shown in Section III-C, vertical and horizontal orientation of the scanner device have a significant impact on the RSSI signals. The impact of orientation is very relevant and significant to real-life contact tracing because people naturally will have their phones in various orientations. This justify the notion that we need extra sensor to gauge the phone orientation, which is exactly Hypothesis 4 calls for.

Hypothesis 3, "Traditional linear regression between the distance and RSSI Trendline can only work in ideal scenarios, but not fit for more complex real-life situation" was confirmed by both Hypothesis 2 and the combined Raspberry Pi data as described in Section IV-B.

Hypothesis 4, "Adding additional sensors to assist the Bluetooth sensor improves the accuracy" was also confirmed. As shown in Section IV, machine learning models that incorporate accelerometer and gyroscope readings in addition to RSSI signals have performed better, as evidenced by the greater AUC values.

Hypothesis 5, “Creating Machine Learning Classifier will improve predictive accuracy compared to traditional regressive models” was also improved. As shown in Section IV, a total of 15 machine learning classification models of three major types were trained and validated using MIT-Matrix data. This led to a model whose AUC is 0.70 on the Raspberry pi data. The AUC of all three classification models were higher than that of the linear regression model, indicating superior performance of the classification method.

B. Noteworthy Conclusions

A noteworthy conclusion of these experiments is that device orientation has a significant impact on the measured RSSI signal. Accelerometer and gyroscope on a MPU-9250 chip were successfully used on Raspberry Pi scanner to provide relative device orientation data when combined with machine learning models can improve the accuracy.

Another noteworthy conclusion is that machine leaning classifier is more effective than traditional linear regression method in estimating proximity.

C. General Lessons Learned

A general lesson learned is that external factors other than the distance affect RSSI measurement. These factors must be addressed so that BLE-based contact tracing correctly classify whether certain received BLE chirps belong to “too close” or not. Once that is correctly determined, the “too long” part of the contact tracing should be relatively easier to calculated. This is an important lesson for Bluetooth based contact tracing as it means that is imperative to incorporate other sensors to improve the accuracy.

VI. Next Steps

I plan to take a few next steps to further improve the model accuracy. First, I would like to further compare the machine learning models used in these experiments and incorporate additional models. For example, I would like to divide the data set into training, validation, and testing triplet sets, so that I can objectively compare these models.

After optimizing the machine learning model, I plan to port the model to the Raspberry Pi so it can generate “too close” classification real time which allows further calculation to address “too long” part of the contact tracing. This will also allow updating model real time on the device. My current efforts to move the model to the Raspberry Pi were unsuccessful due to the different 32-bit architecture used on the Pi, which I hope to solve soon.

Finally, I would like to collect additional data to improve my model, so that it can incorporate the impact from other factors, such as humidity, temperature, surroundings, the presence of conductors, and more.

References

[1] *PACT: Private Automated Contact Tracing*. Available: https://pact.mit.edu/

[2] *Johns Hopkins Coronavirus Resource Center*. Available: https://coronavirus.jhu.edu/us-map

[3] A. Abrams-Downey, J. Saabiye, and M. Vidaurrazaga, "Investigational Therapies for the Treatment of COVID-19: Updates from Ongoing Clinical Trials," (in eng), *Eur Urol Focus*, Jun 2020.

[4] P. Vijayvargiya, Z. Esquer Garrigos, N. E. Castillo Almeida, P. R. Gurram, R. W. Stevens, and R. R. Razonable, "Treatment Considerations for COVID-19: A Critical Review of the Evidence (or Lack Thereof)," (in eng), *Mayo Clin Proc*, vol. 95, no. 7, pp. 1454-1466, 07 2020.

[5] B. J. Ryan, D. Coppola, J. Williams, and R. Swienton, "COVID-19 Contact tracing solutions for mass gatherings," (in eng), *Disaster Med Public Health Prep*, pp. 1-16, Jul 2020.

[6] C. R. MacIntyre, "Case isolation, contact tracing, and physical distancing are pillars of COVID-19 pandemic control, not optional choices," (in eng), *Lancet Infect Dis*, Jun 2020.

[7] L. Ferretti *et al.*, "Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing," (in eng), *Science*, vol. 368, no. 6491, 05 2020.

[8] S. Wang, S. Ding, and L. Xiong, "A New System for Surveillance and Digital Contact Tracing for COVID-19: Spatiotemporal Reporting Over Network and GPS," (in eng), *JMIR Mhealth Uhealth*, vol. 8, no. 6, p. e19457, 06 2020.

[9] M. J. Parker, C. Fraser, L. Abeler-Dörner, and D. Bonsall, "Ethics of instantaneous contact tracing using mobile phone apps in the control of the COVID-19 pandemic," (in eng), *J Med Ethics*, vol. 46, no. 7, pp. 427-431, 07 2020.

[10] "Apple Newsroom."

[11] J. Abeler, M. Bäcker, U. Buermeyer, and H. Zillessen, "COVID-19 Contact Tracing and Data Protection Can Go Together," (in eng), *JMIR Mhealth Uhealth*, vol. 8, no. 4, p. e19359, 04 2020.

[12] *IOT and Electronics*. Available: https://iotandelectronics.wordpress.com/2016/10/07/how-to-calculate-distance-from-the-rssi-value-of-the-ble-beacon/#:~:text=The%20signal%20strength%20depends%20on,40%2D50%20m%20distance).

[13] *CDC Guideline on Contact Tracing for COVID-19*. Available: https://www.cdc.gov/coronavirus/2019-ncov/php/contact-tracing/contact-tracing-plan/contact-tracing.html

[14] J. Quinlan, "Induction of Decision Trees. Mach. Learn," 1986.

[15] *Meet Nandu in Good Audience*. Available: https://blog.goodaudience.com/machine-learning-

using-decision-trees-and-random-forests-in-python-with-code-e50f6e14e19f

[16] A. Liaw and M. Wiener, "Classification and regression by randomForest," *R news*, vol. 2, no. 3, pp. 18-22, 2002.

[17] T. J. Watson, "An empirical study of the naive Bayes classifier," 2001.