Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 September 2020 doi:10.20944/preprints202009.0508.v1

Using Machine Learning to Perform Proximity Detection - Classifying Bluetooth Beacon RSSI Values

Karen Song
karen22@mit.edu

***Abstract*— This project focuses on using machine learning classification algorithms to determine whether two people are 6 feet apart or not. Two Raspberry Pis were used simulate smart phones. RSSI values of the Bluetooth beacons transmitted between the Raspberry Pis were collected and recorded to train the classifier. The Gaussian Support Vector Machine Classifer yielded the highest testing accuracy of 79.670 and the Decision Tree Classifier yielded the highest AUC of 0.80.**

Keywords—Bluetooth, RSSI, Classification, Machine Learning

I. Introduction

A. *Project Description*

This project addresses the feasibility of using machine learning classification algorithms to detect whether two Raspberry Pis (simulating phones held by humans) are ≤ 6 feet apart or not. The Center for Disease Control (CDC) has advised people to social distance when near other people at a distance of at least 6 feet apart, so contact tracing platforms must be able to perform proximity detection between people [4]. This project was executed by designating one Raspberry Pi to be a Bluetooth beacon advertiser, the other to scan for beacons, and collecting measured RSSI values of the beacons.

B. *Background Information*

A Bluetooth low energy beacon transmits a universally unique identifier (UUID) which is picked up by compatible devices [4]. The beacon also broadcasts "a Major number identifying a subset of beacons within a large group, a Minor number identifying a specific beacon, and a TX power level indicating the signal strength one meter from the device." [4]. RSSI, or "Received Signal Strength Indicator," is a measurement of how well a device can hear a signal from an access point [3].

II. Hypothesis

A machine learning classifier algorithm can be trained to accurately predict whether two Raspberry Pis are 6 feet apart or not based on RSSI values.

- Bluetooth-based contact tracing must be able to detect whether users are 6 feet apart or closer in order to report possible COVID-19 exposure.
- Large amounts of RSSI and distance data will need to be collected from the Raspberry Pi. Raspberry Pis will need to be placed at varying distances apart and RSSI values measured at those distances.

III. Experiment and Data Collections

A. *Plan and Execution*

The experiment was conducted by placing the Raspberry Pis at a measured distance apart and then recording the RSSI values measured by the scanner Pi. Figure 1 shows the setup of the 2 Pis, separated by a set distance and connected to power sources. The Pis were separated at a distance of 3 feet, 6 feet, and other arbitrary values both inside and outside the "6 feet apart rule". Multiple trials were taken to collect data. All RSSI data was compiled into a CSV file. 10 trials of each distance were taken.

- During each trial, 60 seconds of scanning and advertising of Bluetooth beacons took place.
- One Raspberry Pi was set up as a Bluetooth beacon advertiser that transmitted Bluetooth beacons once every 1 second. The other Raspberry Pi was set up as a Bluetooth scanner that scanned for beacons every 1 second.
- The experiment was conducted in a home setting due to the Raspberry Pis requiring power sources.

B. *Data Relevance*

The distance between Raspberry Pis must be measured in order to determine what RSSI values are associated with distances greater than or within 6 feet.

 doi:10.20944/preprints202009.0508.v1

C. Examples

Fig. 1 Experiment Setup

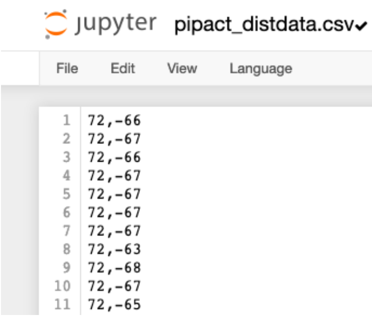

Fig. 2 "pipact_distdata.csv" contains RSSI values in the y-column and the distance between the Raspberry Pis (in inches) in the x-column.

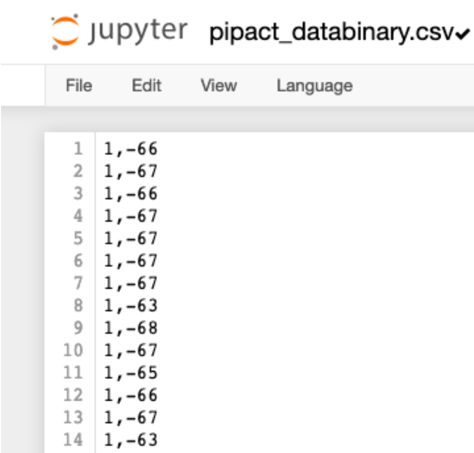

Fig. 3 "pipact_databinary.csv" contains RSSI values in the y column and a binary number (0 or 1) in the x column. 0 indicates that the Pis were separated by a distance greater than 6 feet, 1 indicates that the Pis were separated by 6 feet or less.

IV. Analysis and Algorithms

A. Description

I trained four different classifiers to classify the data collected into two classes: within 6 feet or outside 6 feet. Initially, I trained the models using the "pipact_distdata.csv" to try and predict the distance between beacons given RSSI values, but this yielded very low accuracies so I instead tried to predict whether beacons were within 6 feet apart or not using "pipact_databinary.csv" (Fig. 3). All models were programmed in Python. Python libraries used include Numpy, Pandas, Matplotlib, scikit-learn, and Seaborn. Sklearn's train_test_split to create an 80-20 train-test split of the dataset.

B. Results and Examples

Support Vector Machine

I implemented SVM using 4 different kernels: Gaussian (RBF), Linear, Polynomial (degree=4), and Sigmoid.

Kernel	Training Accuracy	Validation Accuracy	Testing Accuracy
Gaussian (rbf)	76.117	72.603	79.670
Linear	67.010	62.329	67.582
Poly (deg = 4)	67.010	63.699	67.582
Sigmoid	55.670	57.534	58.791

Fig. 4 Training, Validation, and Testing Accuracies of SVM model by kernels.

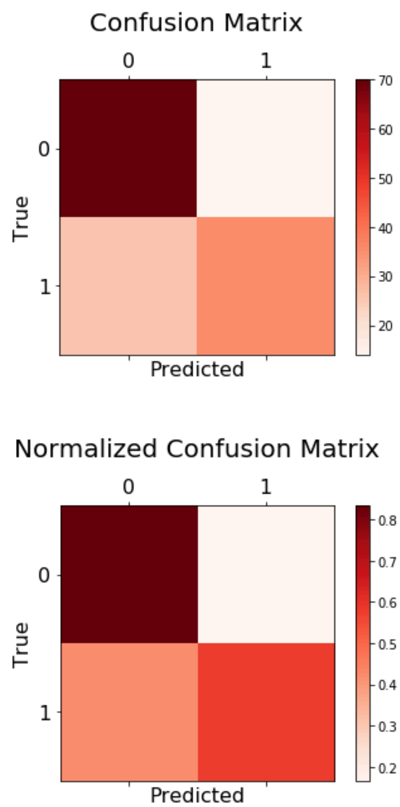

Fig. 5 Gaussian (RBF) SVM Confusion Matrix

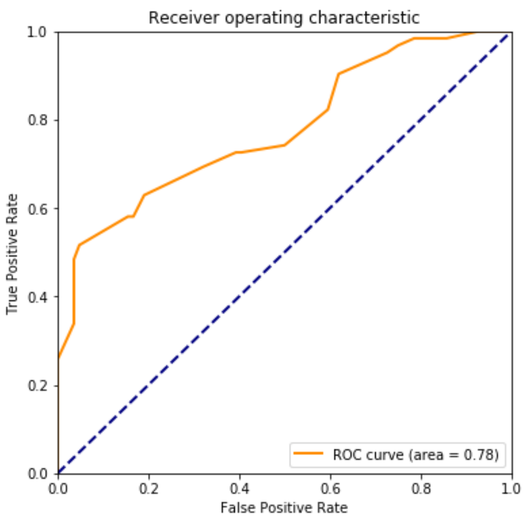

Fig. 6 AUROC Curve for Gaussian SVM

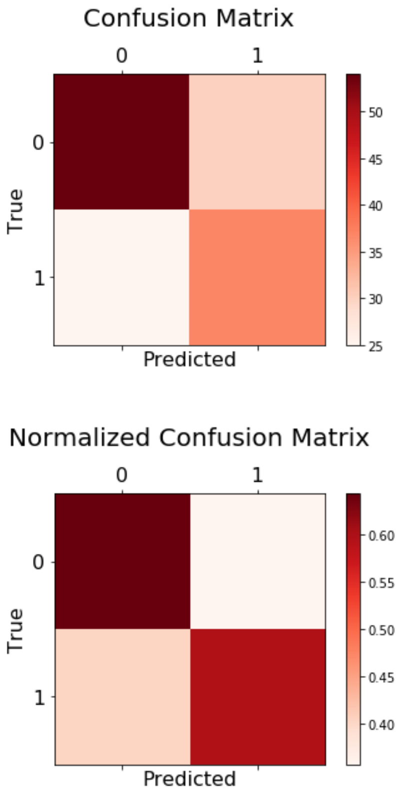

Fig. 7 Linear SVM Confusion Matrix

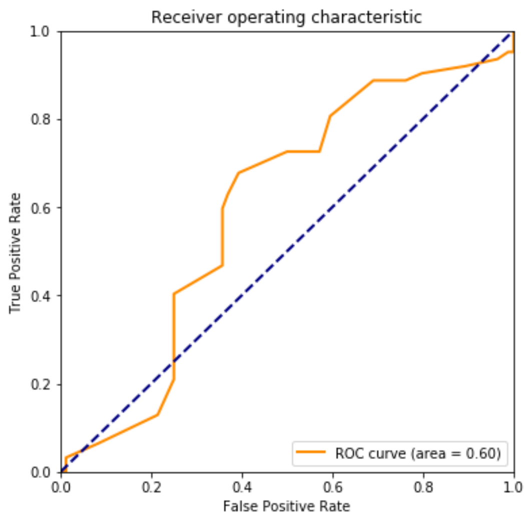

Fig. 8 AUROC Curve for Linear SVM

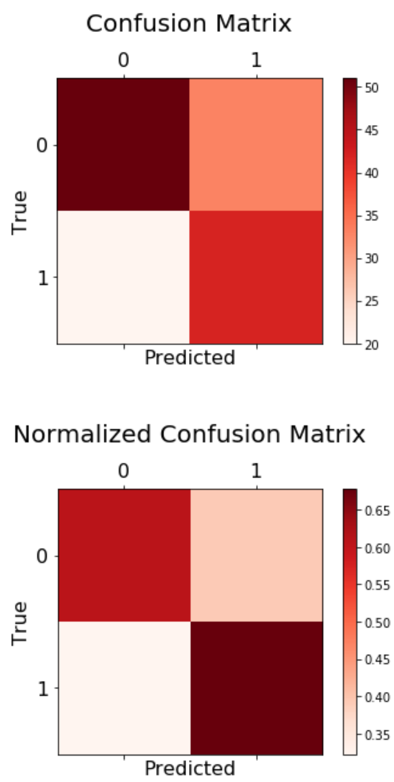

Fig. 9 Polynomial SVM Confusion Matrix

doi:10.20944/preprints202009.0508.v1

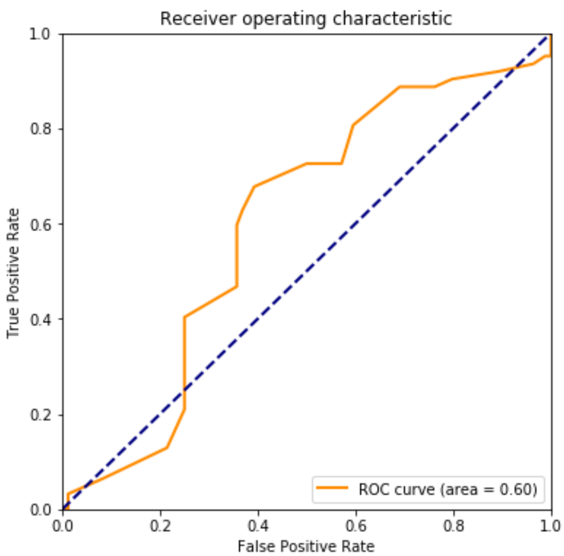

Fig. 10 AUROC Curve for Polynomial SVM

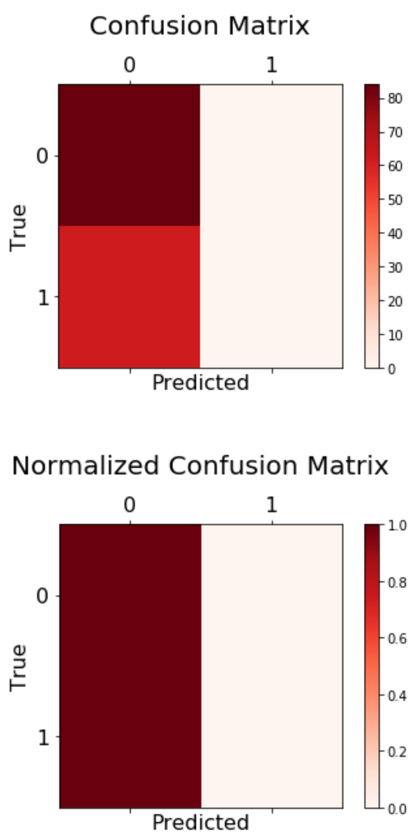

Fig. 11 Sigmoid SVM Confusion Matrix

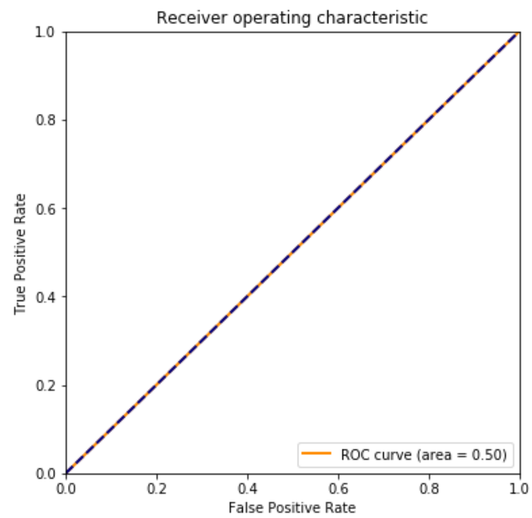

Fig. 12 AUROC Curve for Sigmoid SVM

Logistic Regression

A logistic regression model was also implemented to classify the dataset.

- Training Accuracy = 59.107
- Training Accuracy = 53.846
- Validation Accuracy = 52.055

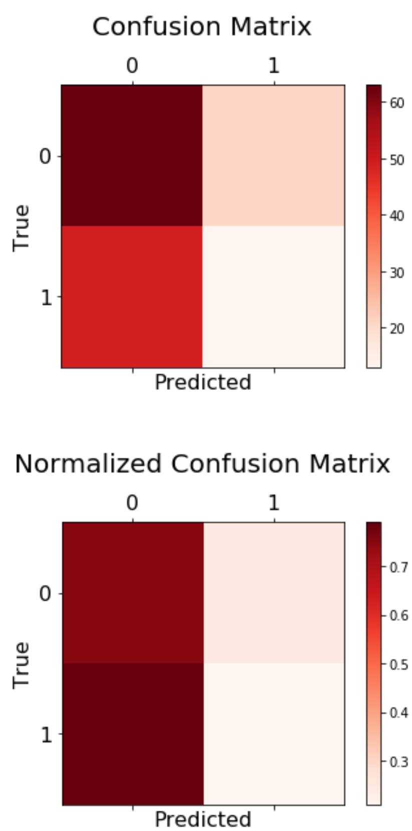

Fig. 13 Logistic Regression Confusion Matrix

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 September 2020 doi:10.20944/preprints202009.0508.v1

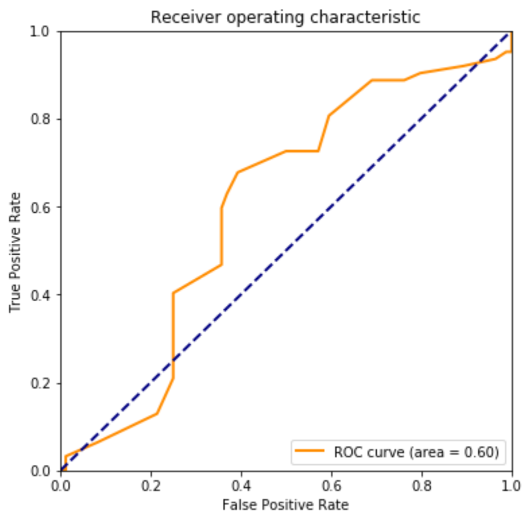

Fig. 14 AUROC Curve for Logistic Regression

k-Nearest Neighbors

The next model tested was the k-Nearest Neighbors classifier. I experimented with different n values, and accuracies did not improve beyond n=3.

- Training Accuracy = 78.179
- Testing Accuracy = 76.923
- Validation Accuracy = 78.082

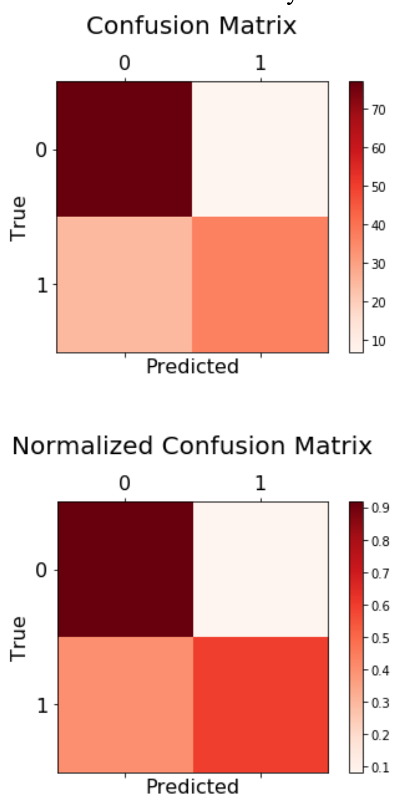

Fig. 15 kNN Confusion Matrix

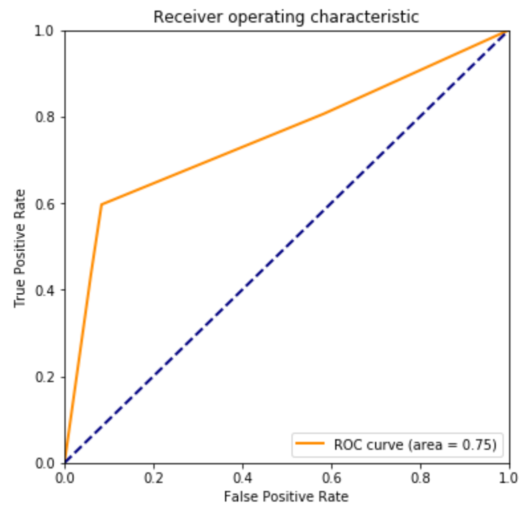

Fig. 16 AUROC Curve for kNN Model

Decision Tree

The final model tested was the Decision Tree Classifier. A max depth of 5 and a minimum number of leaves of 5 was chosen as accuracies did not improve beyond these values.

- Training Accuracy = 78.694
- Testing Accuracy = 78.571
- Validation Accuracy = 76.027

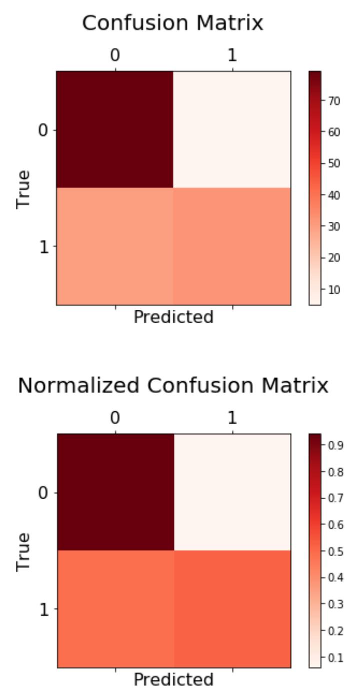

Fig. 17 Decision Tree Confusion Matrix

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 September 2020 doi:10.20944/preprints202009.0508.v1

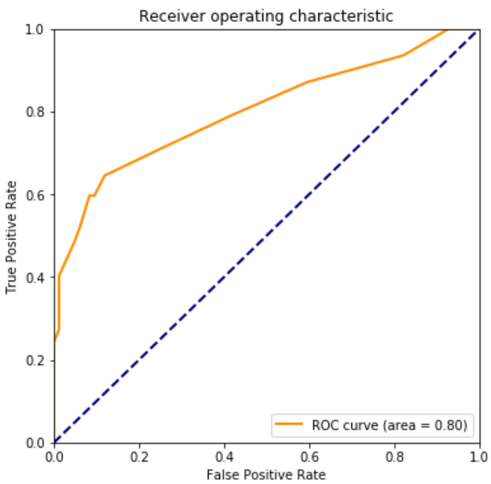

Fig. 18 AUROC Curve for Decision Tree Model

V. Conclusions

A. *Hypothesis Evaluation*

The most accurate machine learning classification algorithm was moderately accurate (~80%). This indicates that my hypothesis (a machine learning classification algorithm can be trained to accurately predict whether two Raspberry Pis are 6 feet apart or not based on RSSI values) is correct. However, this method of proximity detection may not be the most accurate option

B. *Noteworthy Conclusions*

Of the classifiers tested, the Gaussian Support Vector Machine Classifier yielded the highest test accuracy of 79.670. The Decision Tree Classifier yielded the Area Under Curve of 0.80.

C. *General Lessons Learned*

RSSI values are insufficient in accurate proximity detection between 2 Bluetooth devices, as even in ideal circumstances the data collected was classified with moderate accuracy (~80%). Factors such as humidity, obstacles, and other interferences can further lower the accuracy of a machine learning proximity detector.

VI. Next Steps

The experiment was conducted under near ideal circumstances (no weather fluctuations, no obstacles, etc). With the presence of interferences such as clothing fabric, humidity, and other Bluetooth devices, the machine learning model would most likely yield lower accuracies. In the future, I will collect data under different weather conditions, cover the Pis in different types of fabric, and also attempt to send phone notifications or pings from Raspberry Pis to a user's smart phone about their proximity to other people from RSSI values. I also plan on exploring metrics other than RSSI values to detect proximity, as well as other hardware additions that can add functionality to the Raspberry Pis.

References

[1] Arm MBED. (2015, March 24) *Understanding the different types of BLE Beacons.* Retrieved July 20, 2020, from https://os.mbed.com/blog/entry/BLE-Beacons-URIBeacon-AltBeacons-iBeacon/

[2] *Medlytics: Classification Metrics*, Beaver Works Summer Institute, July 2020, https://beaverworks.ll.mit.edu/CMS/bw/bwsi

[3] Metageek. *Understanding RSSI*. Retrieved July 20, 2020, from https://www.metageek.com/training/resources/understanding-rssi.html

[4] National Center for Immunization and Respiratory Diseases (NCIRD), Division of Viral Diseases, & Centers for Disease Control and Prevention (CDC). (2020, July 15). *Social distancing*. Centers for Disease Control and Prevention. Retrieved July 20, 2020, from https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/social-distancing.html

[5] piPact, Beaver Works Summer Institute, July 2020, https://beaverworks.ll.mit.edu/CMS/bw/pipact