**NightBot - Final Project Report**

**Project GitHub Code URL:** [**https://github.com/jeff-umkc/PG6\_FinalProject**](https://github.com/jeff-umkc/PG6_FinalProject)

1. **Motivation**

NightBot is an intelligent personal security assistant robot for the elderly. It answers the calling of a need among the elderly population for a 24/7 assistant to handle their safety needs while providing low operational costs. NightBot accomplishes this task by following a path throughout the home, monitoring for potential safety hazards such as open doors, interacting with the Cloud to process real-time crime data and send push notifications to the user, as well as responding to requests, such as finding and identifying a walker or cane for the elderly.

Our initial proposal was to do a stock prediction robot, but after our presentation and discussion with the Teaching Assistants, we realized that we could not utilize the robot’s features to the full extent, so we changed our robot idea to a security robot for the elderly.

1. **Related Work**

There are existing security robots in the market today, but they are either very expensive, or too generalized. The Knightscope K5 [1] is an amazing security and patrol robot, but it is tens of thousands of dollars, large and heavy, and is designed as a multi-purpose patrol and security robot. Another example is the iRobot for military defense and security [2], but it is also very expensive. After researching the Web, it became clear that there exists no specialized security robot for assisting the elderly on the market.

Our idea is novel and different from other solutions because it is specialized for the elderly, low cost, and made for home use. We utilized the Romotive Romo [3] robot as our base robot, since it is versatile, low cost, durable, made for home use, and is controlled by an iOS device brain.

1. **Architecture**

Our system consists of using iOS [4] for the Romo brain utilizing an iPhone or iPod touch device. We used Objective C [5] as the programming language. The Romo utilizes various sensors on the iOS device, such as the accelerometer for pitch detection and movement, the gyroscope for roll, yaw, and pitch detection, the camera for object detection, and location services (e.g. a mix of WiFi, Cellular network, and GPS). We utilize Android [6] as a client for controlling the Romo device via a peer-to-peer TCP socket connection. Apache Spark [7] is utilized for real-time big data analysis for machine learning using Object Detection, as well as our recommendation system, and Twitter Streaming [8] system. Our Romo communicates with Spark through a TCP socket connection and acts as a client sending requests (e.g. a camera taken image for object detection) and receives replies through MongoLab DB [9], which is our Mongo database. Below is a diagram of our architecture:

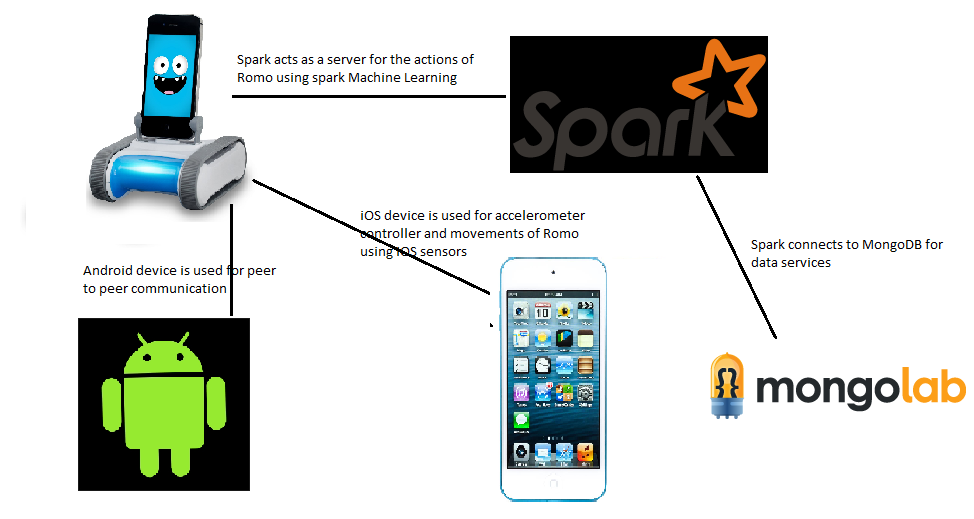


Fig 3.1 System Architecture

1. **System Features**

**Spark** **Object Detection/Notifications** - Romo takes a photo of the object to detect, sends it to our Spark Machine learning server for processing, detects the object, sends the information back to the Romo which notifies the user of the object via an SMS text notification.

**Line Following -** Romo uses color detection to follow a line - blue moves forward, yellow stops and moves backwards. Green to turn left and red to turn right. This is for patrolling the home.

**Voice commands -** Voice commands are received from the Android device and sent directly to the Romo to control it.

**Spark Twitter Streaming Analytics -** Uses location services to analyze the Twitter for crime data in the user’s location (such as Kansas city). Crime data is reported to the user via SMS text messages.

**Severe Weather Notifications/Reporting -** Reports severe weather conditions to the user, such as thunderstorms, floods, tornadoes, etc.

**Collaborative Filtering based Health Care Provider Recommendations -** Uses Spark for Collaborative Filtering for providing health care provider recommendations.

**Sign Recognition -** We utilized OpenCV to detect various signs, this is useful for navigation without line recognition or an Android controller.

**Accelerometer based movement -** Uses the Android device accelerometer to control the Romo remotely.

**Gyroscope based movement -** Utilizes the gyroscope in the iOS device to vary movement speed based off pitch, roll, and yaw.

**Twitter speech out -** Ability to communicate to the user via Twitter

1. **Implementation Details**

We utilized the camera on the iOS device to send images to Apache Spark for Object Detection. So far, our training set can detect Canes/Walkers, Open, and Closed doors. We utilized several machine learning algorithms in SparkML [10]. For Object Detection, we accepted a Base64 image from our TCP socket from the Romo device, extracted the feature descriptors using SURF, generated a K-Means cluster to find the centroid points, created a histogram, and used Naive Bayes to train the model based off our training image set (5 images per each object class) and predict what the input image from the Romo is.



Figure 5.1 - Object Detection Training Sets - Cane/Walker

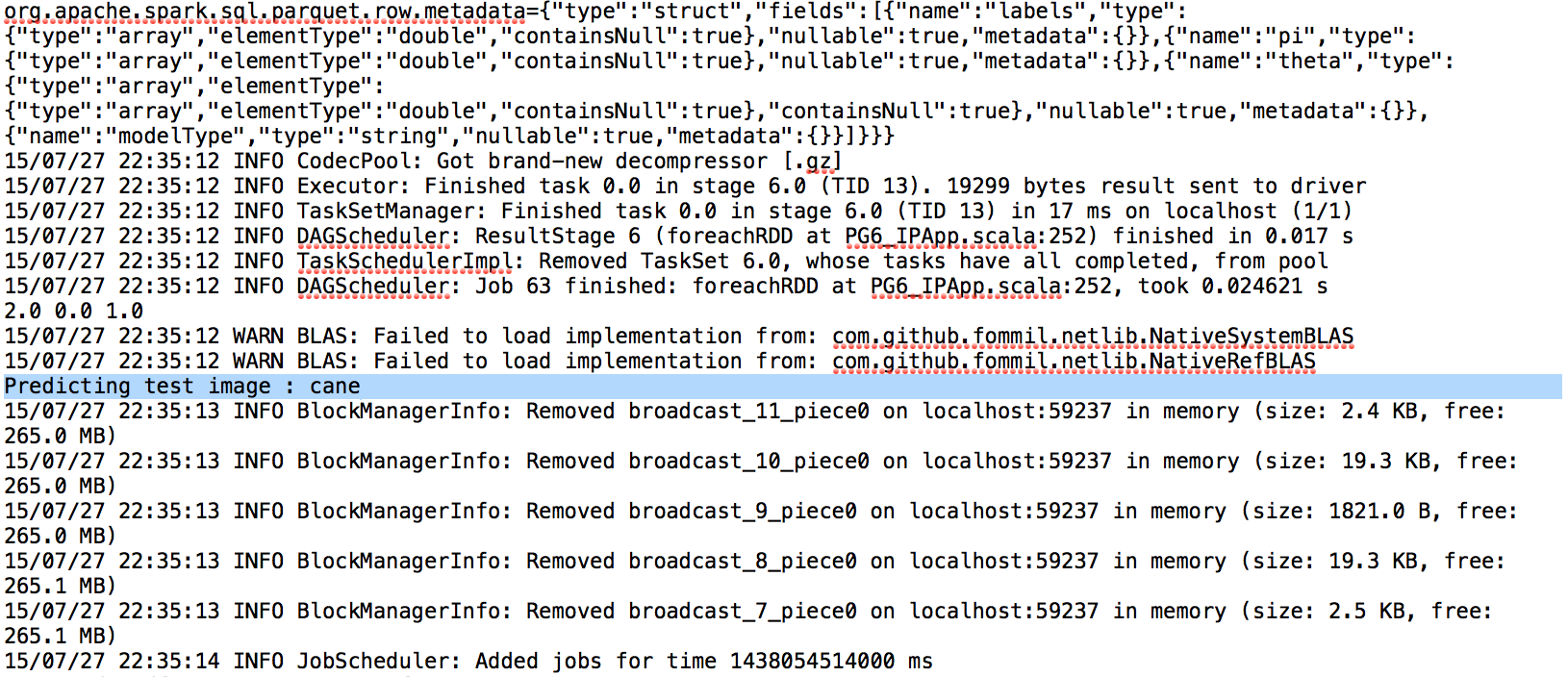


Figure 5.2 - Spark Output (Server) of Cane Object Recognition

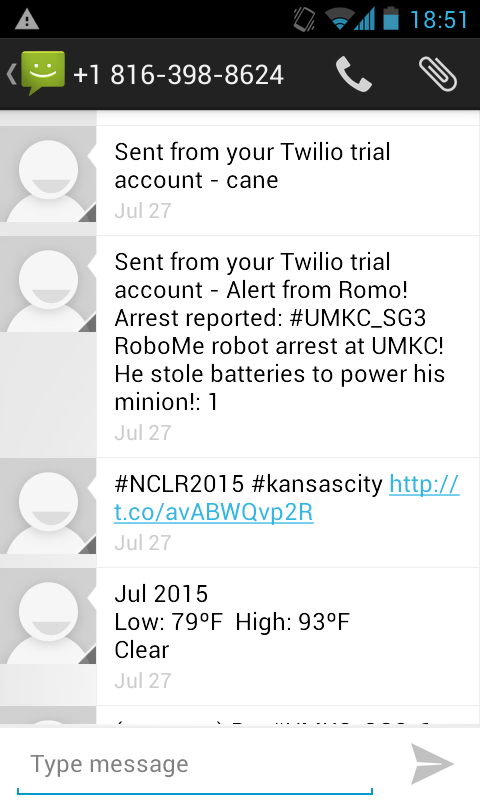


Figure 5.3 - SMS Notifications to user for Object Detection (Cane), Twitter Streaming of Crime data based off Location, and Weather notifications.

For Line Following, we utilized OpenCV’s [11] color recognition API. Essentially, we utilize the camera to take video images, convert them to a matrix of HSV values and recognize colors based off the range of HSV values. Blue is for moving forward, Yellow for stopping and moving backwards.

For Voice commands, we utilize an Android device Speech to Text API [12] using a RecognitionListener intent. Our Android device captures the speech to text output, and sends the command to our Romo robot client via a TCP socket.

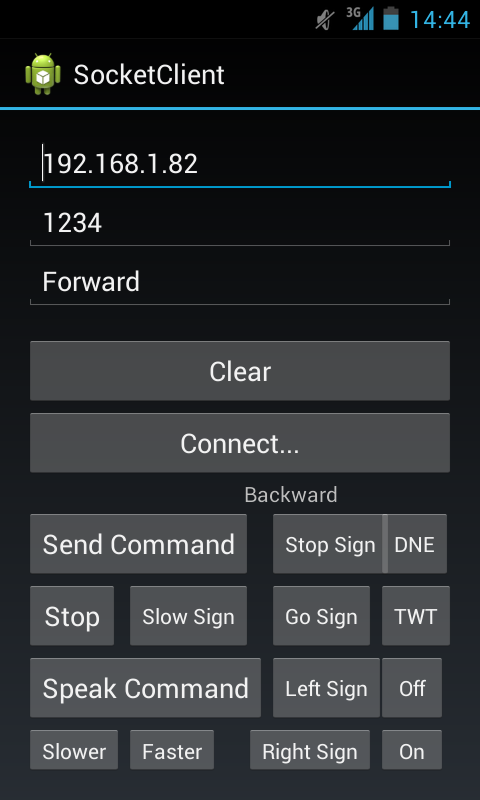


Figure 5.4 - Android Client

For our Spark Twitter Streaming, we used Spark’s real time Twitter Streaming API to capture twitter data using the user’s location (such as Kansas City). We captured the hash tags of crime related data in the area, and captured the Twitter Statuses, Places, and Location information. Relevant crime-related data is stored in MongoLab DB and accessed by our Romo robot, which it then uses the Twilio [13] API to send an SMS notification of crime data to the user. See Figure 5.3.

For severe weather notifications, we used a weather API YWeatherUtils, which uses the Yahoo Weather API [14]. See Figure 5.3.

For Provider Recommendations, we used the Medicare.gov [15] Big Data set of Health Care providers across the US. We used SparkML’s Collaborative Filtering using the Alternating Least Squares method and MSE to produce a recommendation. The recommendation of the providers meeting the criteria is then sent to the user via an SMS notification.

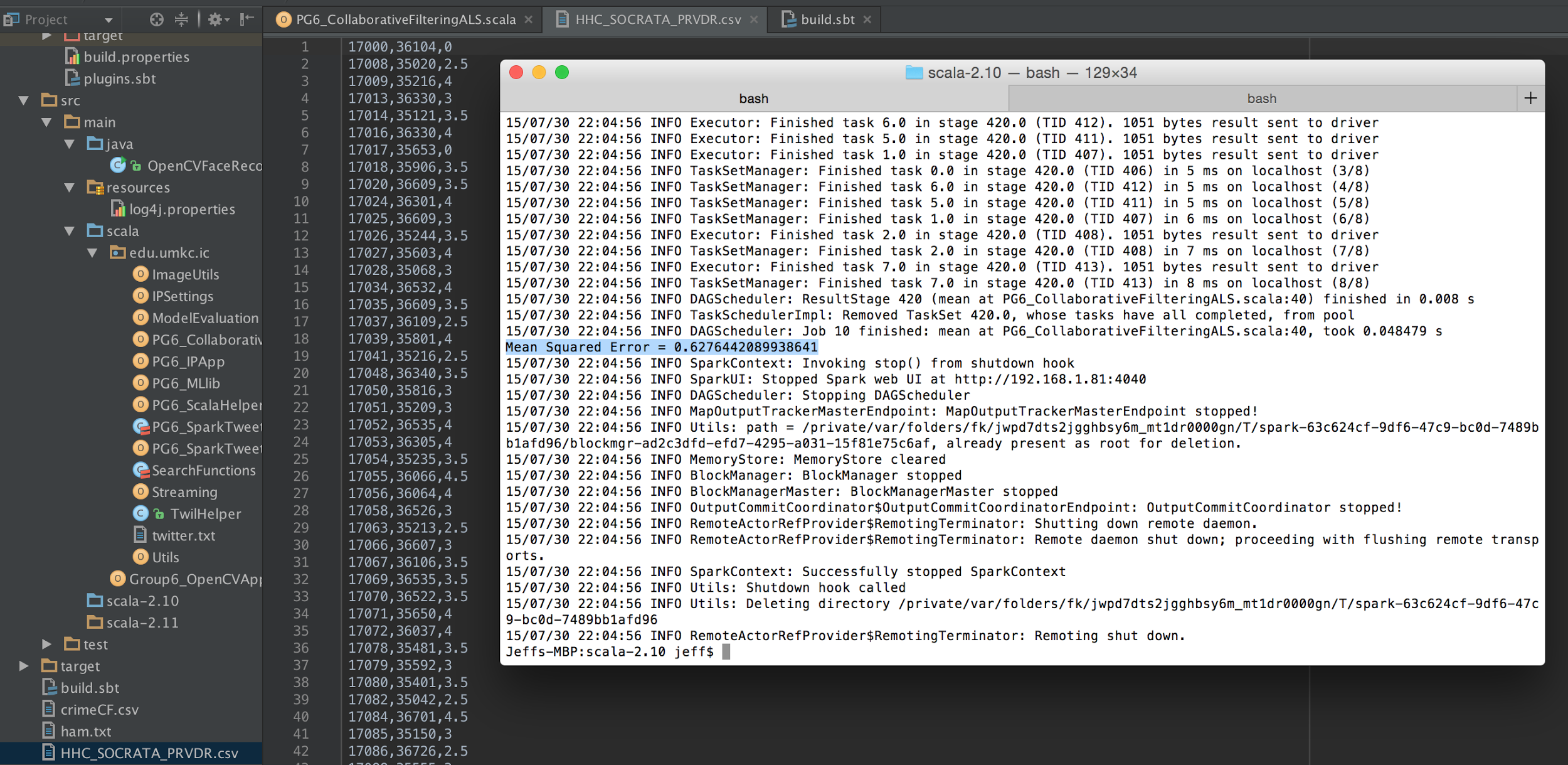


Figure 5.5 - Healthcare Provider Collaborative Filtering Recommendation System (Server)

For Sign Detection, we utilized the iOS device camera to take video images, converted the images to a grayscale matrix, and applied feature detection based off key feature descriptors to see how closely they matched our given set of signs (including: Stop, Go, Left, Right, Do Not Enter).

For our accelerometer based movement, we used the Android’s accelerometer data and utilized a TCP socket to send movement commands to our Romo. For Gyroscope data, we utilized the Romo’s iOS device gyroscope to determine pitch, roll, and yaw and varied the Romo’s speed based off this data, so that the Romo could move regardless of if it was at an incline or decline, which should help it navigate various home terrains.

Finally, we used MongoLab DB to store our NoSQL document based Twitter crime data and retrieved it using a REST API [16] from our Romo iOS device.

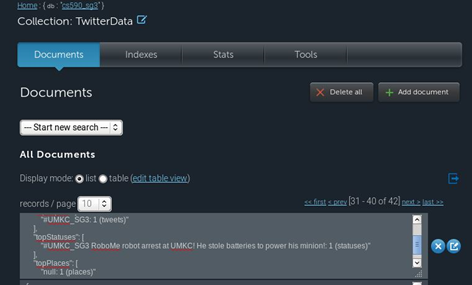


Figure 5.6 - MongoLab DB - Twitter Crime Data

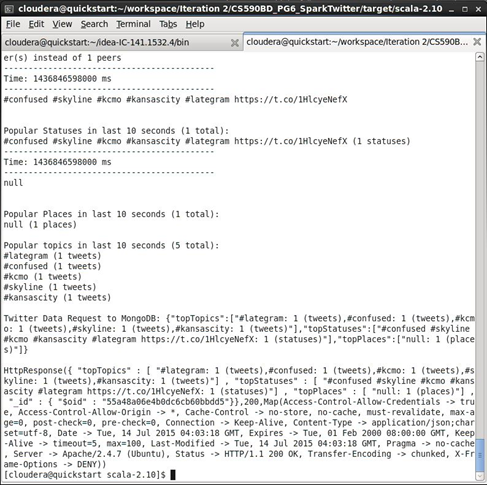


Figure 5.7 - Spark Streaming Twitter Crime Data

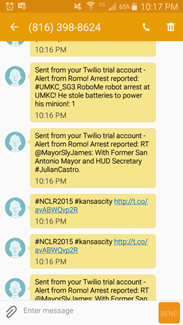


Figure 5.8 - Spark Streaming output

1. **Results/Evaluation**

We calculated a Confusion Matrix to determine the Accuracy of our Naive Bayesian based Object Recognition model. From our test sample, we received an 86% accuracy rating, which is quite acceptable for our use case. One example of our use case is for our robot to detect Canes, and notify the user when it’s detected the cane, along with the Cane’s location. This is to help the elderly user find their walking durable medical equipment.

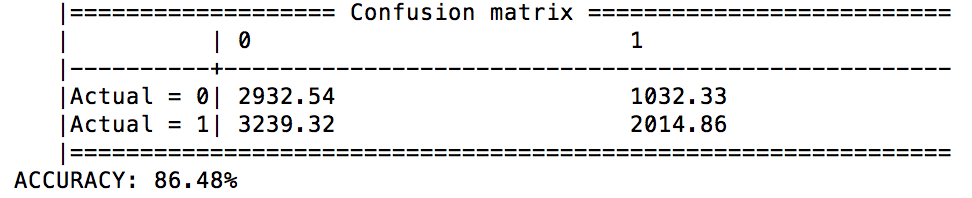


Figure 6.1

Line following proved to be a useful feature, since now our robot can patrol a set path along a user’s home. The only downside is that we needed to use colored painters tape to utilize line following, and setting this up requires someone to have tape down in their house. In addition, this will not work effectively at night because it utilizes color recognition. We hope to improve this in the future by not using color recognition, but implementing path finding based off computer vision along with a room coordinate system (so the robot can differentiate between the kitchen and bedroom, for example).

1. **Conclusion**

From the beginning, this was a very complex project. We began by familiarizing ourselves with the Romo SDK and controlling its basic movements with the Android client. Once we defined our robot’s purpose to provide safety and security for its owner, we began implementing useful APIs. Twitter was used to provide the user with information on local crime. The robot would communicate with the user by sending the Twitter information via SMS text message using the Twilio API. Weather information was also sent using Twilio. OpenCV was used for color detection and object detection.

The primary goal however, was to implement Big Data features relevant to NightBot’s purpose. Using Spark and the machine learning library, we were able to implement a HealthCare recommendation system using collaborative filtering, parse through Twitter data with natural language processing, and image processing. Through a socket connection, this information was either sent or received from Spark and the iOS application.

1. **Future Work**

Future research is in making the NightBot autonomous and not need to follow a pre-set line by using advanced Image Recognition to sense the environment and move/interact on it’s own without the need for a controller.We hope to expand the NightBots capabilities by adding additional machine learning techniques for recommending hospitals based off location and ratings, in addition to our provider recommendation. In addition, we hope to test the various features of the Romo by letting an elderly person use it and observing areas of improvement and listening to feedback for new features.

1. **References**

[1] - Knightscope K5: <http://knightscope.com/about.html>

[2] - iRobot: <http://www.irobot.com/>

[3] - Romotive Romo: <http://www.romotive.com/>

[4] - iOS: <https://developer.apple.com/ios/>

[5] - Objective C: <https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html>

[6] - Android: <http://developer.android.com/index.html>

[7] - Apache Spark: <http://spark.apache.org/>

[8] - Spark Twitter Streaming: <https://spark.apache.org/docs/latest/streaming-programming-guide.html>

[9] - MongoLab DB: <https://mongolab.com/>

[10] - Spark ML: <http://spark.apache.org/docs/latest/ml-guide.html>

[11] - OpenCV: <http://opencv.org/>

[12] - Android Speech to Text: <http://developer.android.com/reference/android/speech/package-summary.html>

[13] - Twilio API: <https://www.twilio.com/api>

[14] - Yahoo Weather API: <https://developer.yahoo.com/weather/>

[15] - Medicare Data Sets: <https://data.medicare.gov/>