**A**

**PROJECT REPORT**

**ON**

**“ANALYSIS OF STUDENT MOVE-IN ISSUES”**

**USING CLOUD SERVICES AWS**

In Partial Fulfillment of

**Final Project Cloud Analytics (ITC-6460)**

A picture containing black, indoor

Description automatically generated

**Session Winter-2020**

**Northeastern University CPS, Boston**

**Group 4: Epsilon**

**Ashwin Nair | Rishab Taneja | Akash Singh | Sindhu Siddalingamurthy**

**Instructor Name: Sergiy Shevchenko**

**Table of Contents**

Abstract

Requirement Specifications

Introduction

Technologies and Libraries Used

AWS Services

Architecture Design

Data Analysis - QuickSight Visualization

Roadmap / Future Improvements

Code Snippet

References

**ABSTRACT**

The Design of architecture was based on the Data collected from <https://311.boston.gov> wherein the customer complaints were recorded for the issues in and around the area. The data scraped were images and used serverless services and other services based on Amazon Cloud. We gathered the dataset using python scripts via CLI of Amazon and performed Data Analysis over the gathered Datasets and built a solution of the whole flow of the project.

**REQUIREMENTS**

AWS free tier account | Libraries Used: Beautiful Soup, boto3, JSON.

Python Scripting knowledge | AWS Services Knowledge

Install boto3 | AWS CLI basic knowledge

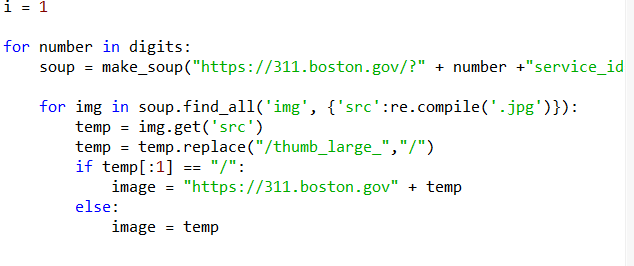
**INTRODUCTION**

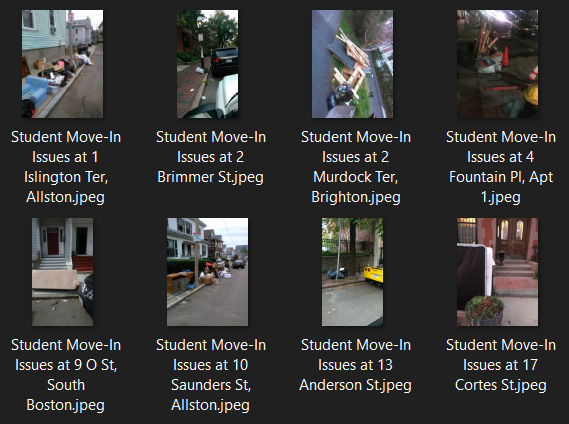
**Scope of the Project**

To visualize and gather insights about materials left on the streets by students - Scraping the 311 reports and performing image detection of all the materials, furniture and hardware in the “Student Move-in” section of Bos 311 database. In this project, we worked on an analysis of the data using the Amazon Cloud Services and API calls for the Command-Line.

**Gathering image data via Python Script:**

The image that showed up on the website was not of good resolution and the output image was small and had to be rescaled and image resolution was supposed to be taken care because we would need an input of good resolution image for performing analysis and gathering properties of the image. Finally, we created the same title as the name of the image file as seen on the 311Boston.gov by running the read and write function. We managed to scrape and store 70 images ready to be analyzed further.





**TECHNOLOGIES AND LIBRARIES USED**

**Beautiful Soup**

[Beautiful Soup](http://www.crummy.com/software/BeautifulSoup/) is a Python library for pulling data out of HTML and XML files. It works with your favourite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

Beautiful Soup is a Python library designed for quick turnaround projects like screen-scraping. Three features make it powerful:

Beautiful Soup provides a few simple methods and Pythonic idioms for navigating, searching, and modifying a parse tree: a toolkit for dissecting a document and extracting what you need. It doesn't take much code to write an application.

Beautiful Soup automatically converts incoming documents to Unicode and outgoing documents to UTF-8.

You don't have to think about encodings unless the document doesn't specify an encoding and Beautiful Soup can't detect one. Then you just have to specify the original encoding.

Beautiful Soup sits on top of popular Python parsers like lxml and html5lib, allowing you to try out different parsing strategies or trade speed for flexibility.

**boto3**

Boto is the Amazon Web Services (AWS) SDK for Python. It enables Python developers to create, configure, and manage AWS services, such as EC2 and S3. Boto provides an easy to use, object-oriented API, as well as low-level access to AWS services.Boto3 has two distinct levels of APIs. Client (or "low-level") APIs provide one-to-one mappings to the underlying HTTP API operations. Resource APIs hide explicit network calls but instead provide resource objects and collections to access attributes and perform actions.

**AWS Cloud Services**

**AWS S3**

Simple Storage Service (S3) in which we have stored the images by scraping the 311boston webpage using Python and Beautiful Soup library as explained in the earlier paragraph. The number of objects on S3 which can be stored is 2000 put objects out of which we scraped 100+ images to pass on to the recognition. It is object storage which provides scalability, security and durability of the data along with the performance. It provides easy to manage access and we could store any kind of object on s3.

**AWS Lambda:**

We used AWS lambda as we could run our code without worrying about the servers. It provides a virtual environment to run our code of any programming language with no administration required.

**Lambda Specs**

The technical specifications of AWS Lambda. Lambda supports the following runtimes.

* Node.js: v10.15 and v8.10
* Java 8
* Python: 3.7, 3.6, and 2.7
* .NET Core: 1.0.1 and 2.1
* Go 1.x
* Ruby 2.5
* Rust

Each function runs inside a container with a 64-bit Amazon Linux AMI. And the execution environment has:

* Memory: 128MB - 3008MB, in 64 MB increments
* Ephemeral disk space: 512MB
* Max execution duration: 900 seconds
* Compressed package size: 50MB
* Uncompressed package size: 250MB

You might notice that CPU is not mentioned as a part of the container specification. This is because you cannot control the CPU directly. As you increase the memory, the CPU is increased as well.

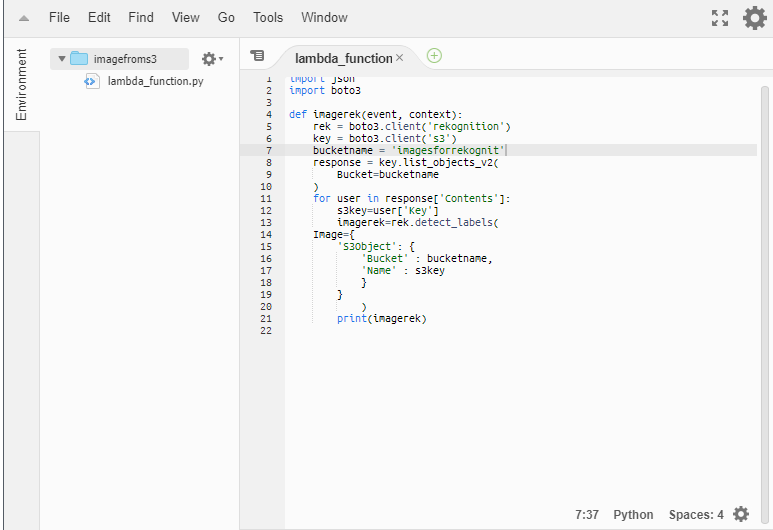
The ephemeral disk space is available in the form of the /temp directory. You can only use this space for temporary storage since subsequent invocations will not have access to this. We’ll talk a bit more on the stateless nature of the Lambda functions below.

The execution duration means that your Lambda function can run for a maximum of 900 seconds or 15 minutes. This means that Lambda isn’t meant for long-running processes.

The package size refers to all your code necessary to run your function. This includes any dependencies (node\_modules/ directory in case of Node.js) that your function might import. There is a limit of 250MB on the uncompressed package and a 50MB limit once it has been compressed. We’ll take a look at the packaging process below.

* Lambda Function

We did upload the code on the lambda function created imagerekeg() is our lambda function:



Our Lambda function runs for two minutes as the size of the images on S3 is 200MB. Lambda function getobjects() the data from Amazon’s S3 and then places it on to the AWS Rekognition using the detect\_label() API call which has been made and accepts only .jpeg and .png format files. Few of the images which were scraped were not in the right format and hence we would face errors in parsing the images which are not of the right format.

**AWS Rekognition**

We had a thought to use Rekognition because we had numerous images and the accuracy provided by Rekognition is good enough to build a data set for analysis. We put our images on S3 and used the API detect\_label() and boto3 library which provides the AWS services functionality. We did run a loop via lambda function using Python to get the images from S3 and provide it as an input to Rekognition which helps in Data Analysis.

**AWS CloudWatch:**

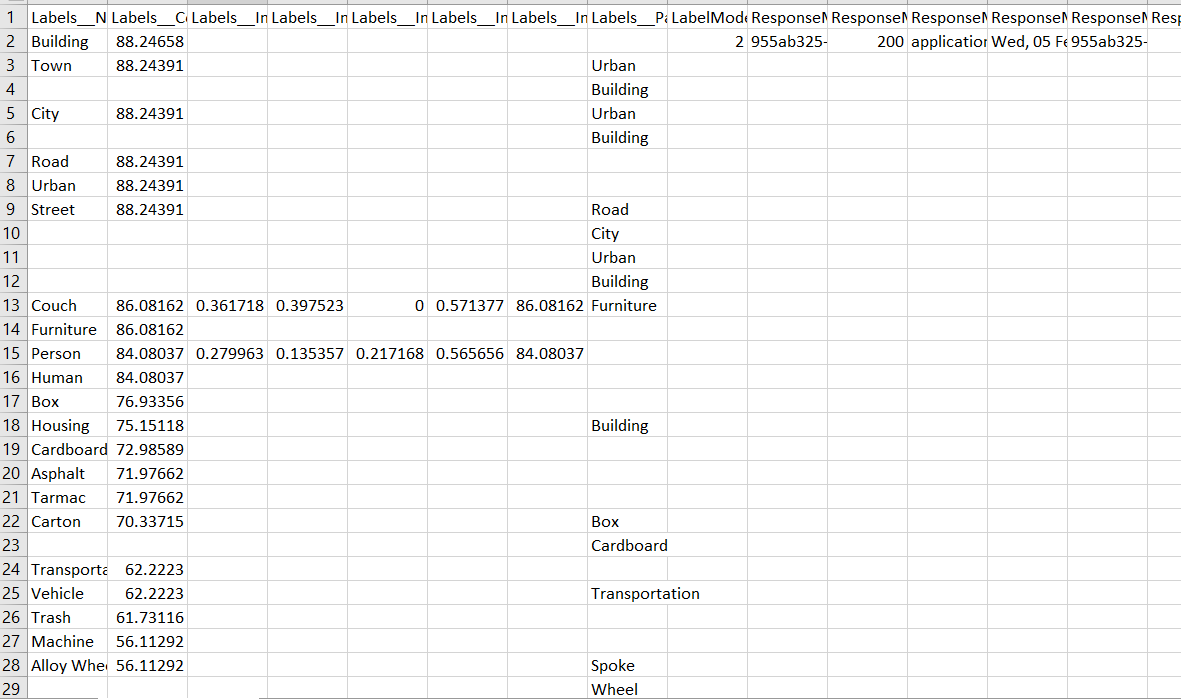
Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real-time. We have used CloudWatch to collect and track metrics through which our images get recognized. Cloud Watch helped us to collect the log data through which we export the data on to the S3 using the Grouplogs on which the .json file is present.

**AWS Quicksight:**

Amazon QuickSight is a fast, cloud-powered business intelligence service that makes it easy to deliver insights to everyone in your organization. As a fully managed service, QuickSight lets you easily create and publish interactive dashboards that include ML Insights. Dashboards can then be accessed from any device and embedded into your applications, portals, and websites. We have used Quicksight for our project to Visualize the Datasets which were scraped from the website.

**Data Cleaning:**

In this step, we cleaned the data which is considered as one of the most important tasks as a data expert. Below we have an image showing the CSV file which is what we got from the AWS Rekognition.



As we can see that above dataset is having almost 465 missing rows which need to be removed and then we removed a few labels as well which were insignificant for our analysis like - human, planks etc. Besides that certain columns were also needed to be removed such as:

Labels\_\_Instances\_\_BoundingBox\_\_Width, Labels\_\_Instances\_\_BoundingBox\_\_Height, Labels\_\_Instances\_\_BoundingBox\_\_Left etc.

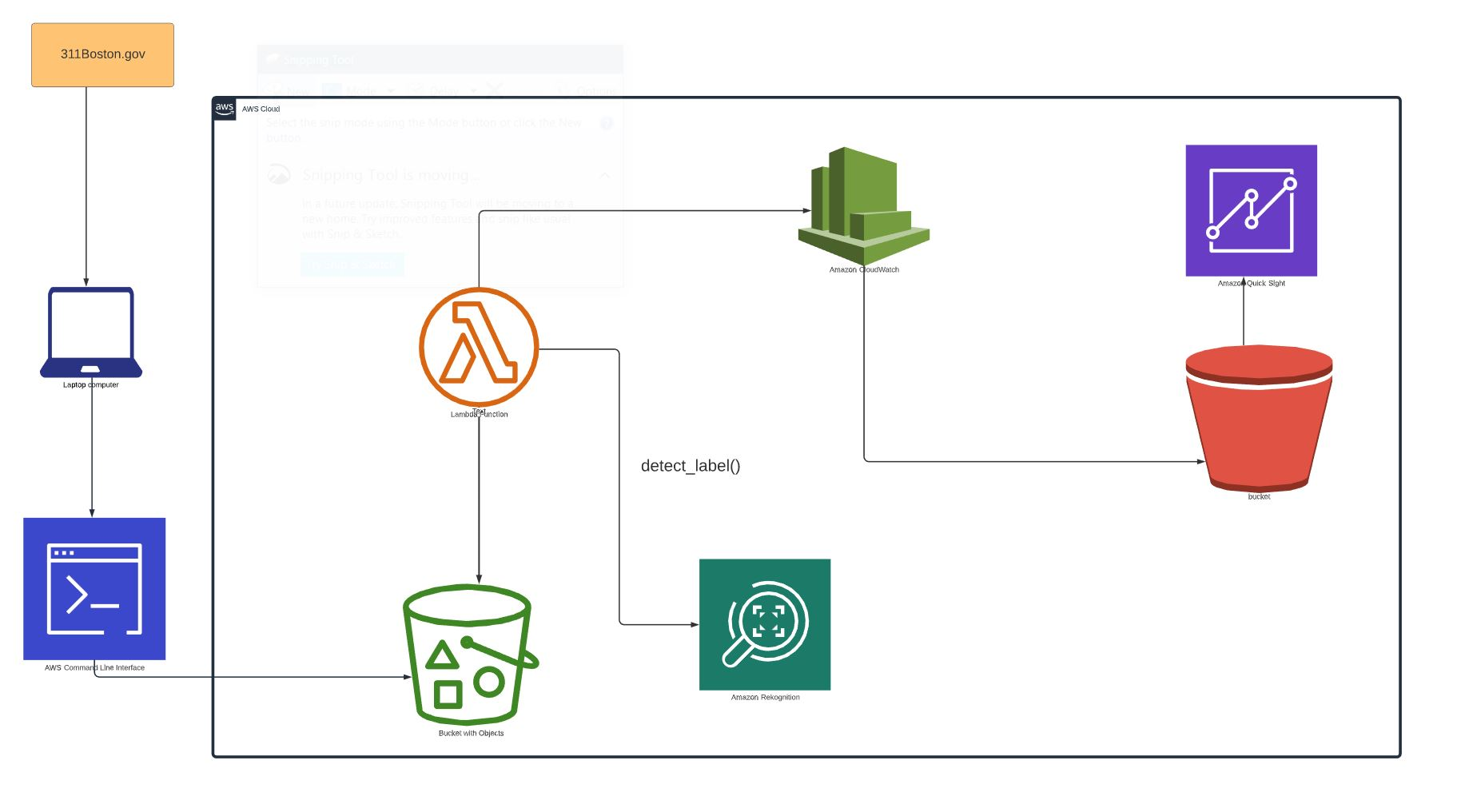
Since we weren't able to get a sufficient amount of data from them to get meaningful insight from it. All of these things were done using the quick sight since the tools which we are going to use as visualization tools to get insights.

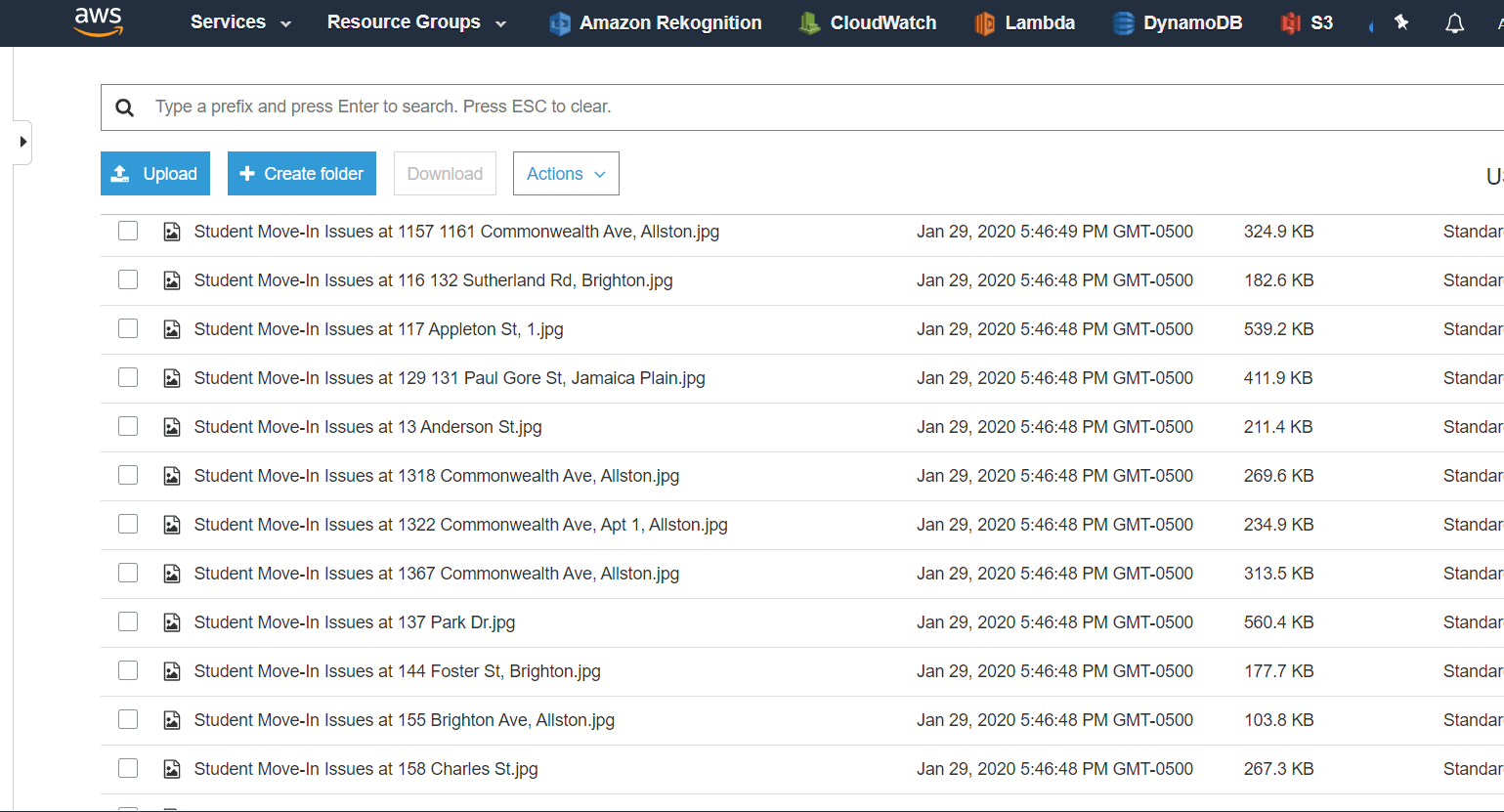
**ARCHITECTURE DESIGN**

Initially, we faced challenges in gathering the dataset. As the website could not be scraped easily as we expected. Once we did scrape the images of the website using scripts we would need an image with better quality and resolution. An image with better resolution provides the best datasets. Using the Beautiful Soup Library we did get the images with better resolution and stored the images on to the Amazon’s Simple Storage Service.

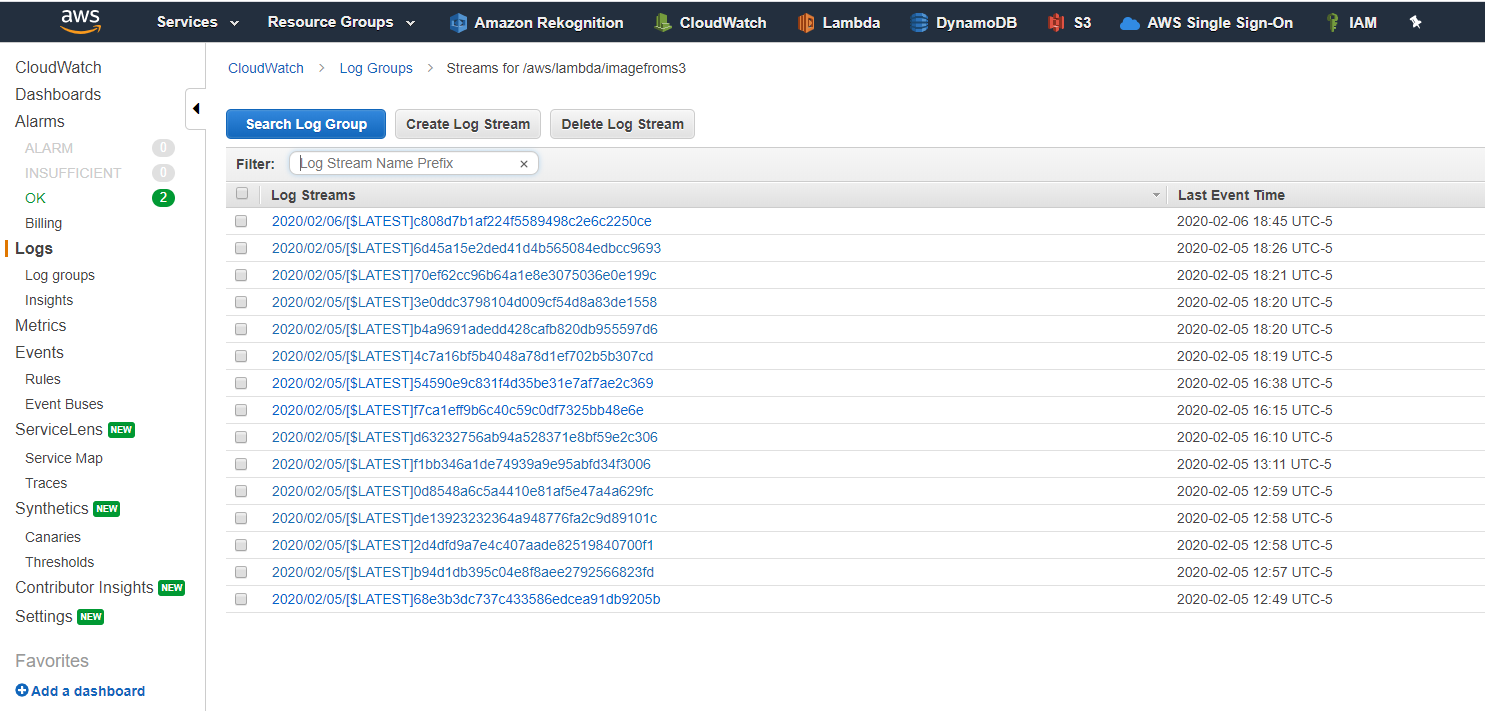
Images stored on S3 were run against Lambda, using a Python script which ran a loop for all the images on the S3. We did use the boto3 library to perform the tasks on getobject() from S3 and putobject() to Lambda and fed into Rekongition.

detect\_Label(), takes only .jpeg and .png format as input and few of the images were not of the right format. The images in the right format were run and the JSON property file which had an object Rekognition were exported from the cloud watch logs to Amazon’s S3. The log group in the Cloud Watch had the detailed object detection JSON which helped us to continue with the analysis. These .json files obtained from cloud watches were converted to CSV for the convenience of performing Analyzing the data. These data sets were cleaned based on certainly required columns against which we are performing Analysis.

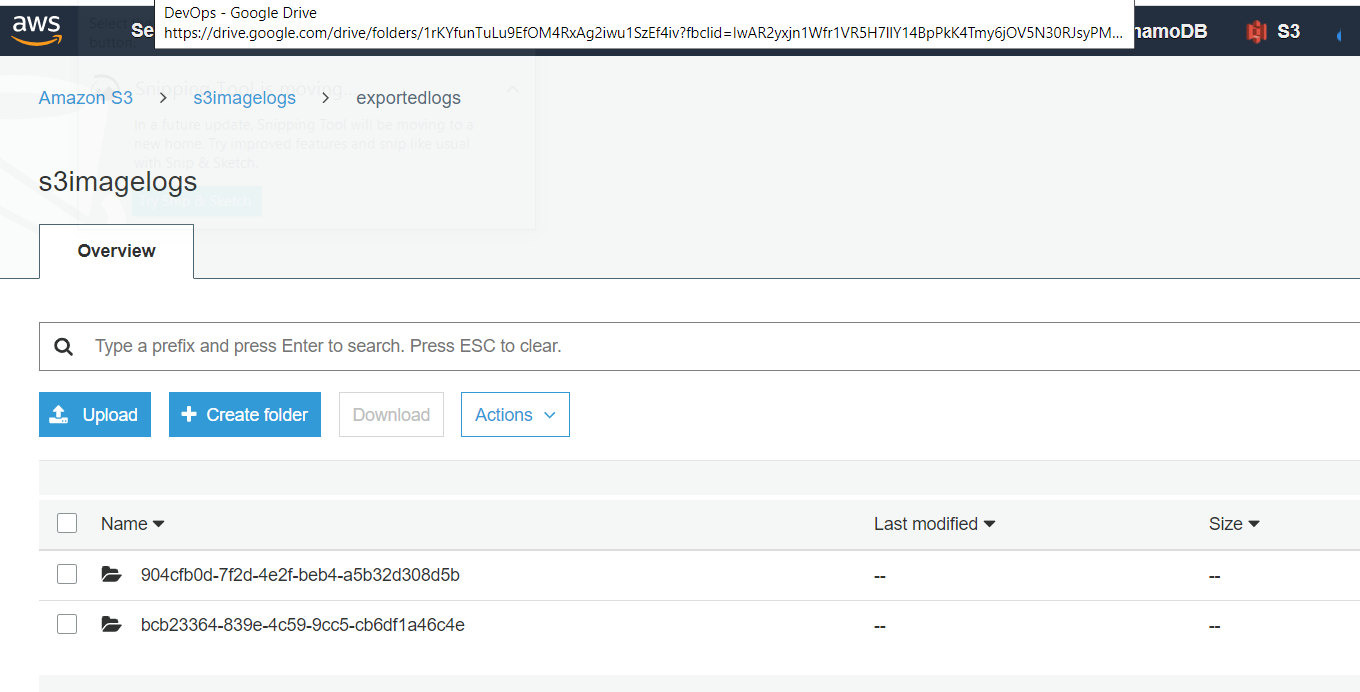
****

****

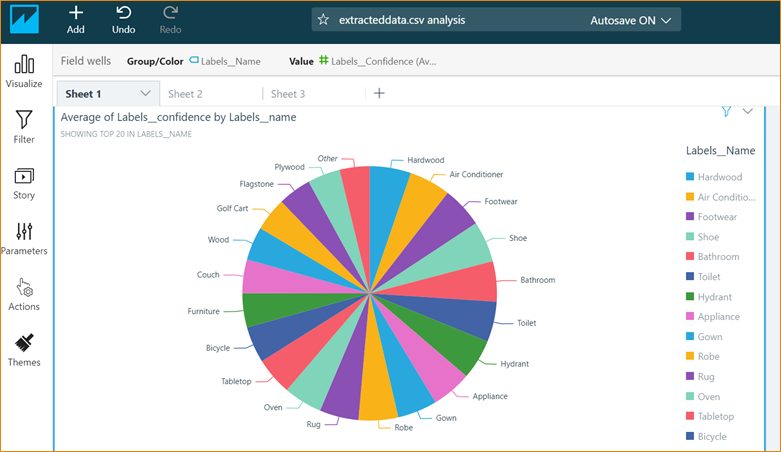
**CLOUDWATCH LOGS**

****

**IMAGELOGS**

****

**Data Analysis – Quicksight Visualizations**

****

Furniture & Wood are the main types of materials left out on the streets

Plywood and Cardboard are the most frequent items seen

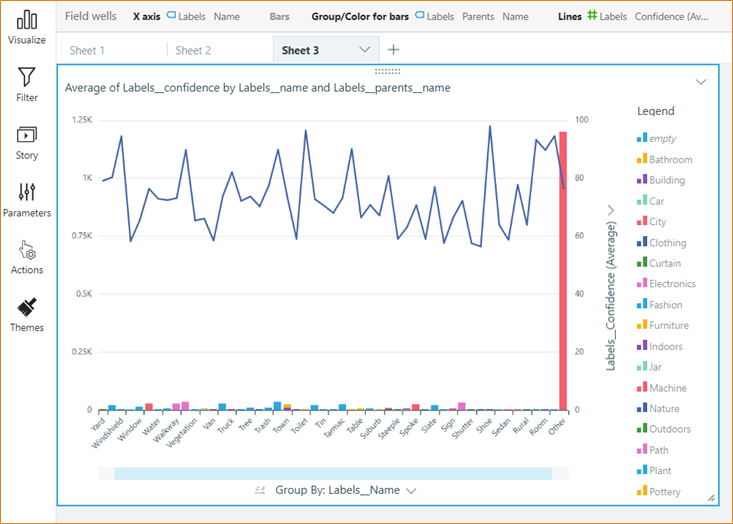
Carton, Bags, Chairs & Glass are also seen in high numbers

****

Categorizing the label parent name as x-axis, we show how each label name is varying in confidence level.

Hardwood shows the highest confidence with 98.8%

‘Shoes’ seem to show a confidence level of 97.9%

****

This visual gives a better idea of the label names, the count and the confidence level as detected. There are clothing material – leather, rugs and coupe glass that are common

**Roadmap / Future Improvements**

* Textiles, mattresses and bedding shouldn’t be thrown away – connect with local charities and MassDEP Beyond the Bin Recycling Directory can be used to find nearby drop-off locations
* Any wood or metal products are recycled by Wood Waste of Boston Inc for energy production. Understand the feasibility of them opening a facility near Northeastern
* Glass cannot be recycled with other products and its around 70% of glass is recycled back into shelves within 30 days. Earth911 provides a directory of redemption centres for glass around Boston.

**Code snippets**

***imagerek.py*,** this is the script which was written to getobject() from S3 and detect\_label() of the objects. Since the number of objects was more than 100+ a loop was created to run from 1st object on the S3 until the end.

**# -\*- coding: utf-8 -\*-**

**import boto3**

**rek = boto3.client('rekognition')**

**key = boto3.client('s3')**

**bucketname = '311bostonimages'**

**response = key.list\_objects\_v2(Bucket=bucketname)**

**for user in response['Contents']:**

**s3key=user['Key']**

**imagerek=rek.detect\_labels(**

**Image={**

**'S3Object': {**

**'Bucket' : bucketname,**

**'Name' : s3key**

**}**

**}**

**)**

**print(imagerek)**

***This is the script to scrape the website:***

**import urllib**

**import urllib.request**

**from bs4 import BeautifulSoup**

**import string**

**from warnings import warn**

**from string import digits**

**import re**

**def make\_soup(url):**

**thepage = urllib.request.urlopen(url)**

**soupdata = BeautifulSoup(thepage, "html.parser")**

**return soupdata**

**i = 1**

**for number in digits:**

**soup=make\_soup("https://311.boston.gov/?"+number “service\_id=54c65c0669632a1ac4f2acb1")**

**for img in soup.find\_all('img', {'src':re.compile('.jpg')}):**

**temp = img.get('src')**

**temp = temp.replace("/thumb\_large\_","/")**

**if temp[:1] == "/":**

**image = "https://311.boston.gov" + temp**

**else:**

**image = temp**

**nametemp = img.get('alt')**

**if len(nametemp)==0:**

**filename = str(i)**

**i = i + 1**

**else:**

**filename = nametemp**

**imagefile = open(filename + ".jpeg", "wb")**

**imagefile.write(urllib.request.urlopen(image).read())**

**imagefile.close()**

**References**

[**https://311.boston.gov/**](https://311.boston.gov/)

[**https://aws.amazon.com/qi**](https://aws.amazon.com/qi)

[**https://www.bostonbuildingresources.com/advice/recycling-where-does-it-all-go**](https://www.bostonbuildingresources.com/advice/recycling-where-does-it-all-go)

[**https://www.boston.gov/**](https://www.boston.gov/)

[**https://docs.aws.amazon.com/cli/latest/reference/s3/cp.html**](https://docs.aws.amazon.com/cli/latest/reference/s3/cp.html)

[**https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html#using-boto-3**](https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html)

[**https://www.crummy.com/software/BeautifulSoup/**](https://www.crummy.com/software/BeautifulSoup/)

[**https://www.python.org/**](https://www.python.org/)