



CIS8395 –Final Paper  
Data Source, ETL, Cleaning, Storage, Analytics and Visualization

<b>Submitted By - 3TECH</b>
<b>Shrutika Potdar</b>
<b>Venkat Ramana</b>
<b>Souman Basha</b>

**Professor: VijayKumar Gandapodi**

## Table of Contents

<b>Introduction.....</b>	<b>3</b>
<b>Data Sources .....</b>	<b>4</b>
<b>Data extraction .....</b>	<b>5</b>
<b>Web Scraping .....</b>	<b>6</b>
<b>Data Cleaning and Transformation: .....</b>	<b>7</b>
<b>Data Loading/Storage .....</b>	<b>8</b>
<b>Data Analysis .....</b>	<b>9</b>
<b>Architecture and Implementation .....</b>	<b>13</b>
<b>Cloud Services .....</b>	<b>14</b>
<b>Deployment in Cloud .....</b>	<b>17</b>
<b>Data Visualization .....</b>	<b>18</b>
<b>Website creation .....</b>	<b>25</b>
<b>Future Implementation .....</b>	<b>26</b>
<b>Challenges Faced : .....</b>	<b>27</b>
<b>Experience &amp; Key Learnings .....</b>	<b>28</b>

## Introduction

### About Bridge Community

Bridge Community is a unique entrepreneurial community model, backed by an exclusive set of large corporate, focused on growing communities through corporate-startup collaboration. The Members of The BridgeCommunity Atlanta include an exclusive set of large corporations, focused on growing communities through corporate-startup collaboration. The Members seek to gain first mover advantage through the development of strong relationships with innovative software technology startups, supporting and facilitating their path to commercialization.

### Why Poochtree?

Poochtree is a location-based app for dog owners is to obtain reliable information on where to eat, drink, play, stay, and shop with their pooch. Poochtree's principal goal is to make dog ownership a seamless experience by delivering personalized content based on an owner's location and preferences and their dog's attributes. So, the Poochtree app will recommend these locations to its users based on accurate information about the dog-friendly establishments across the country.

Currently Poochtree is relying on its team and its users to provide the list of establishments and then have to call or email them to confirm the details.

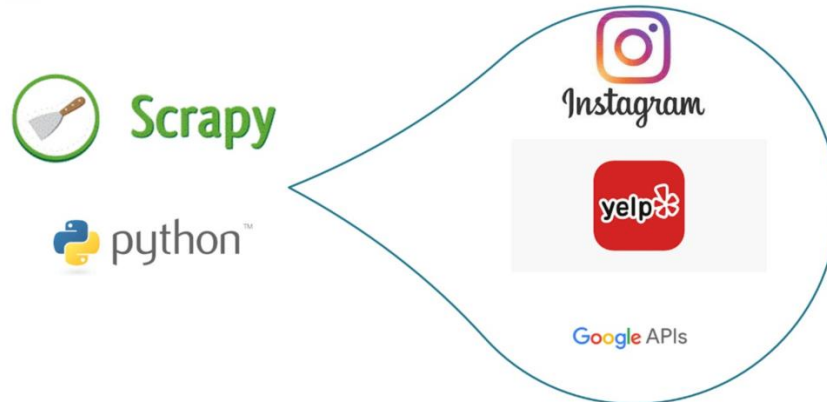
As par to the requirement we need to gather the list of dog friendly establishments by scraping social media sites and other sources. If an establishment that we eat at is dog friendly, they are also interested in identifying if they have the following attributes *if possible*; indoor/outdoor seating, type of service, and amenities like televisions in the dog friendly area, water bowls, treats, dog menu, heated patio, A/C or fans on the patio, covered patio, live music.

The other sources of information are Yelp and Google API. If Yelp has the flag and a comment stating a restaurant is dog friendly or if Yelp has a flag and Instagram has a tagged picture or id Yelp has a comment and Instagram has a picture, it is very likely it is dog friendly. In the similar manner need to get the dog friendly restaurants information from Google as well.

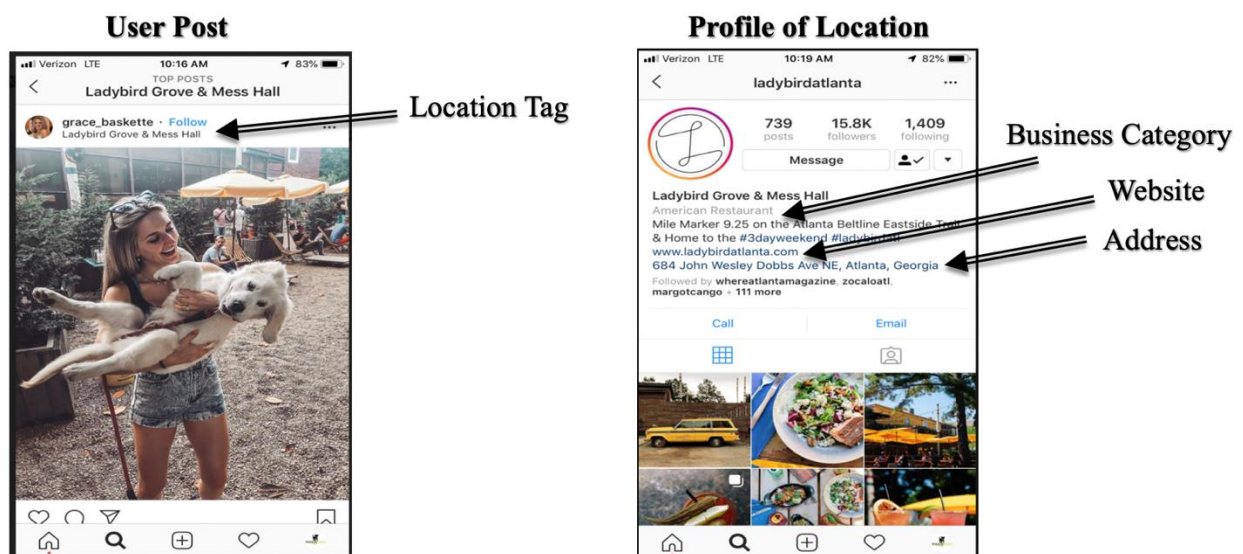
Should be able to filter and view the listings using the categories like (Name | Business Category |Address| City, State| Website| ) and attributes like "Brewery", "Hotel", "Shopping", "Retail", "Store etc .

## Data Sources

Along with the business requirements we were provided with around 800 Instagram images of either dogs or dogs in restaurants. Some of these images were redundant due to duplicity or pictures of same dog taken from different angles. Thus, we had around 550 images. These images were not enough to train our machine learning model. So, we have performed web scraping to extract a greater number of images from Instagram and also considered Kaggle dataset. Till submission of this project we were successfully able to scrap the data from Instagram and use it for further processing.



**Instagram:** First source is Instagram, where we are looking for pictures of dogs at locations or tagged with a location. If there are multiple images of dogs at a location or if there is an image of a dog and food at a location (for restaurant listings), then we would flag it for review. So it's basically like collecting data from various Instagram profiles which has the dog information.



### Secondary sources (Kaggle):

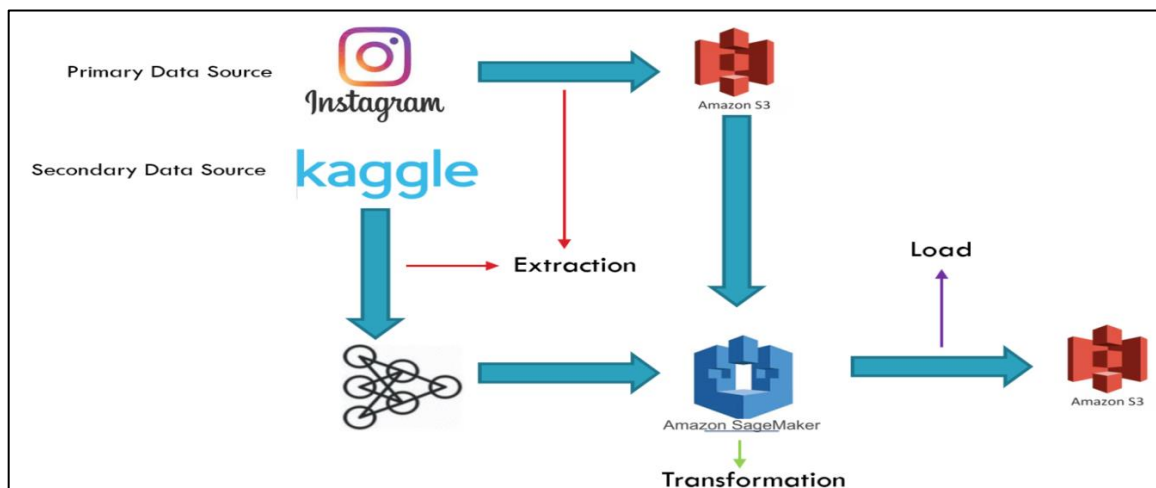
We are using a combination of the image dataset provided by business and also the below dataset to train our image classification machine learning model.

- <https://www.kaggle.com/c/dogs-vs-cats/data>

The data from the secondary source was used to train the image classification model and the data from Instagram (primary data source) was used to test the model. It actually was the live data we used to present a demonstration of the working model. We used 2000 images for training dataset, 500 for validation dataset and 500 images for test dataset.

## Data extraction

ETL provides the underlying infrastructure for integration by performing three important functions. ETL refers to Extract, Transform and Load process.



### Extract:

ETL is a very important component for devouring data warehouses, business intelligence, and big data management. During this process, data is taken (extracted) from heterogeneous source systems, converted (transformed) into a format that can be easily analysed, and then stored (loaded) into a data warehouse (DWs) or operational data stores (ODSs). For our project, extraction begins by scraping the social media websites like Instagram, Yelp and Google for listing dog-friendly establishments. We are performing web scraping to extract images of dogs along with the location information. For example, if an establishment where you eat at is dog friendly, we are also identifying if they have the attributes such as; indoor/outdoor seating, type of service, and amenities like televisions in the dog friendly area, water bowls, treats, dog menu, heated patio, A/C or fans on the patio, covered patio, live music. All these factors will be taken into consideration while extracting data from the websites. When we extracted data from Instagram, say for example Business Category, searched for posts that include words like “Restaurant”, “Brewery”, “Hotel”, “Shopping”, “Retail”, “Store” (i.e. could be “Book Store” or “Clothing Store”), “Park”, “Business”, “Shopping District”, “Salon”, Winery/Vineyard, “Service”, “Local Service”, “Sports”, “Recreation”, “Clothing”, “Salon”, “Food”, “Beverage”,

“Grocery”, “Garden”, “Hot Dog Joint”, “Just for Fun”, “Club”, “Company”, “Vet”, “Service”, “Lodging”, “Lounge”, “Theatre”, “Pizza Place”, “Transportation”, “Travel”, “Studio”, “Lumber Yard”, “Groomer”, “Bar”, “Wine” etc.

### **Transformation:**

Convert the format of the extracted data so that it conforms to the requirements of the target data. Transformation is done by using rules or merging data with other data. Once the data is extracted by web scraping, the resultant csv files and the images will pass through pre-processing steps. The data-set contains more than 13,000 images of dogs collected from the websites. In our case, we are dealing with large image dataset. These images always vary a lot in their background, image quality, lighting etc. We have taken into account all these factors and try to create as realistic dataset as possible. Some of the factors are as below:

## **Web Scraping**

For any business to be successful, it needs data. Data required can vary with market performance to the competitor’s data. Web scraping allows the business to get data from various sources. There are several ways to extract information from the web. Like the use of APIs. Large websites like Twitter, Facebook, Google provide APIs to access their data in a more structured manner. So web scraping technique focuses on transformation of unstructured data (HTML format) on web into structured data (database/spreadsheet).

### **How to achieve Web Scraping?**

First way to achieve web scraping is by using APIs, since Instagram depreciated the API platform in December 2028 and early January 2019 mainly for user’s privacy and security reasons, we decided to use Python programming language with its libraries BeautifulSoup and urllib2.

**BeautifulSoup:** It is a python library designed to parse data, i.e., to extract data from HTML or XML documents. It can be used to extract tables, lists, paragraphs and we can also put filters to extract information. Extracting images from Instagram based on hashtags and downloading those images can also be done using this library.

**Urllib2:** The urllib2 is a Python 2 module (called *urllib.request* and *urllib.error* in Python 3) that defines functions and classes that help in opening URLs in a complex world (i.e. basic and digest authentication, redirections, cookies etc). urllib2’s biggest advantage is that it’s considered Python standard library module, which means as long as you have Python installed, you are good to go.

## Data Cleaning and Transformation:

### Problems faced:

- **Data Inconsistency:** The data we download was incoherent and inconsistent. There were no images and we had the video files downloaded. Need to delete the videos and other non-image format data.
- **Duplicate values:** There were multiple images with the same name, so we have deleted the duplicate images as part of the cleaning process.
- **Data errors:** The labels for the images were sometimes alphanumeric, to be consistent we have deleted or renamed the name of the images.
- **Unnecessary formats:** Even non-image formats were downloaded while scraping. Videos, text and .csv file were downloaded apart from images. We have removed all unnecessary formats.

### Solutions implemented:

<b>MISSING DATA</b>	The content that have been downloaded with Scraping code contains missing information like Label Names.
<b>INCONSISTENT DATA</b>	The data that is downloaded from the Instagram Scarper is inconsistent. Apart from the Images, Videos and text files are also downloaded. Need to remove the inconsistent data.
<b>DUPLICATE DATA</b>	Few images are downloaded multiple times. Since the duplication of data will impact the analysis, the data need to be removed.
<b>DATA ERRORS</b>	While downloading the images and URL's, we got some bad URLs. In this case the connection link has been removed.

Here is a snippet of how the same file looks now:

## Key takeaways:

There are several types of data quality problems and before doing any kind of analysis it's important that we take into account all the quality concerns. With our data, we faced various data quality issues and we tried to solve them step by step, following were the key takeaways for us-

- **Missing attributes:** We have downloaded few missing which has missing or inconsistent values. Those values need to be eliminated.
- **Spot checks:** While the data might appear clean it is important to make sure that we spot check the data. While doing so in this data set, we could find spelling errors that might have caused problems in later querying and transformations.
- **Noise Vs phenomena:** The incoherent fields might not be noise all the time, in our case, we found that tile field had some encrypted data and that led us to realizing that we might want to focus on the movies belonging to one particular region.

## Data Loading/Storage

For data storage we have gone through multiple options and after analysing the pros and cons of multiple storage, data retrieval and compute options we have decided to move ahead with AWS S3 (Simple storage Service) and LAMBDA. Mentioned below are the details of the challenges we have faced in each approach and how have we decided on our primary storage.

### Approach 1 - AWS-S3

After scrapping the raw data from Instagram (or other websites), which would be the picture and restaurant data in csv format, the data would be stored in one S3 bucket (Bucket-

1. And that raw data would be in turn fed to the machine learning model for image classification.
2. As per the machine learning model output, if the image is classified to have a dog in it, the content will be transferred to another S3 bucket (Bucket-2) for final storage and that data will be



exposed to the user. If the image is classified as not featuring a dog, then the content will be transferred to another bucket (Bucket-3) and marked for archival.

3. AWS Glue is a cloud service for data for analysis through automated extract, transform and load (ETL) processes. Glue supports multiple databases like MySQL, Oracle, Microsoft SQLServer and PostgreSQL databases that run on Amazon Elastic Compute Cloud (EC2). It can be used for batch processing. We are planning to use AWS Glue for our ETL process. It is a fully managed extract, transform, and load (ETL) service that makes it easy for to prepare and load their data. Amazon DynamoDB is a fully managed proprietary NoSQL database service that supports key value and document data structures. We might use the DynamoDB as one of the options for saving the image details. Considering the image size limit might be using the DynamoDB.

### **Approach 2 - AWS-DynamoDB**

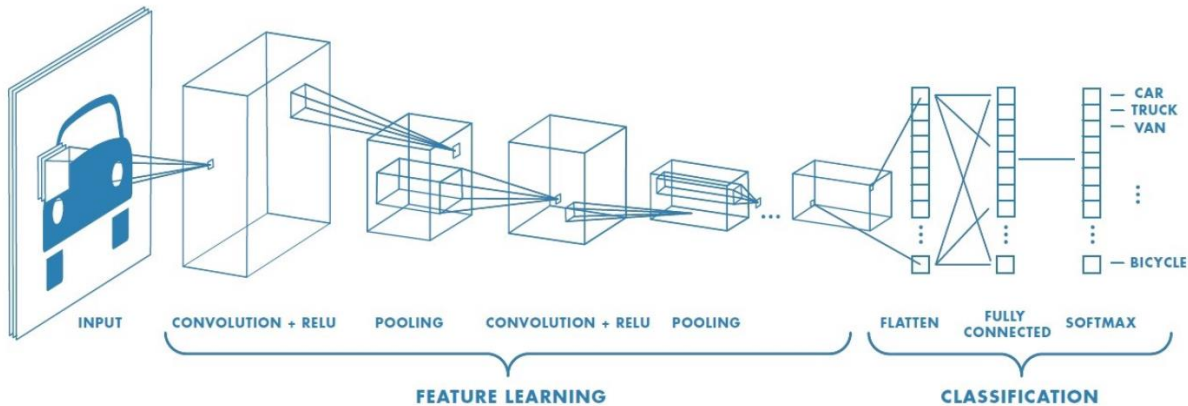
Dynamo DB is a key-value and document database, which delivers millisecond performance at any scale. It is a fully managed AWS service which has in memory caching capabilities and suitable for internet scale applications

#### **Reason why we planned to consider Dynamo DB:**

1. Dynamo DB is a no-SQL database which is suitable for the storage of unstructured data and when combined with Dynamo DB accelerator (In -Memory Caching) gives a higher throughput for every query made
2. Dynamo-DB is a fully scalable and managed database solution provided by AWS, which would automatically address, any scalability and availability related concern which might raise in the future
3. Dynamo-DB allows the use of Local Secondary Indexes (LSI) and Global Secondary Indexes (GSI), with the help of LSI and GSI, we would have the capabilities to set up indexes based on our requirement and that would efficient and customizable way of querying the data

### **Data Analysis**

The methodology used for our analysis and modelling is Convolutional Neural Network (CNN). CNN image classification takes an input image, process it and classify it under certain categories (E.g., Dog, Cat, Tiger, Lion). Computers sees an input image as array of pixels, and it depends on the image resolution. Based on the image resolution, it will see  $h \times w \times d$  ( $h$  = Height,  $w$  = Width,  $d$  = Dimension). E.g., An image of  $6 \times 6 \times 3$  array of matrix of RGB (3 refers to RGB values) and an image of  $4 \times 4 \times 1$  array of matrix of grayscale image. A CNN is a neural network that typically contains several types of layers, mainly convolutional layer, pooling layer and activation layers.



The convolutional layer is the core building block of a Convolutional Network that does most of the computational work. Convolution preserves the relationship between pixels by learning image features using small squares of input data. The Convolutional Network (ConvNet) layer is coupled with ReLU. ReLU's purpose is to introduce non-linearity in our ConvNet. Since, the real-world data would want our ConvNet to learn would be non-negative linear values. Pooling layers section would reduce the number of parameters when the images are too large. A typical ConvNet contains 3 iterations of the above layers based of the quality and also the quantity of the data. However, since we are dealing with bigger images and a more complex problem, we will make our network accordingly larger: it will have one more Conv2D + MaxPooling2D stage. This serves both to augment the capacity of the network, and to further reduce the size of the feature maps, so that they aren't overly large when we reach the Flatten layer. Here, since we start from inputs of size 150x150 (a somewhat arbitrary choice), we end up with feature maps of size 7x7 right before the Flatten layer.

Below is a snippet of the layer creation we did: -

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
                        input_shape=(150, 150, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

While creating the feature maps, the depth has been progressively increased in the network (from 32 to 128), while the size of the feature maps has been decreased (from 148x148 to 7x7). This is a pattern that is followed generally while defining the convnets.

Below is the summary of the model that we defined: -

```
model.summary()
```

Layer (type)	Output Shape	Param #
=====		
conv2d_1 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_1 (MaxPooling2)	(None, 74, 74, 32)	0
conv2d_2 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_2 (MaxPooling2)	(None, 36, 36, 64)	0
conv2d_3 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_3 (MaxPooling2)	(None, 17, 17, 128)	0
conv2d_4 (Conv2D)	(None, 15, 15, 128)	147584
max_pooling2d_4 (MaxPooling2)	(None, 7, 7, 128)	0
flatten_1 (Flatten)	(None, 6272)	0
dense_1 (Dense)	(None, 512)	3211776
dense_2 (Dense)	(None, 1)	513
=====		
Total params: 3,453,121		
Trainable params: 3,453,121		
Non-trainable params: 0		

The `ImageDataGenerator` function of Keras was used for train and test data generation. We configured the `ImageDataGenerator` to yield batches of 150x150 RGB images (shape (20, 150, 150, 3)) and binary labels (shape (20,)). 20 is the number of samples in each batch (the batch size). We fit the model using the `fit_generator` method, the equivalent of `fit`. It expects as first argument a Python generator that will yield batches of inputs and targets indefinitely. Because the data is being generated endlessly, the generator needs to know example how many samples to draw from the generator before declaring an epoch over. This is the role of the `steps_per_epoch` argument: after having drawn `steps_per_epoch` batches from the generator, i.e. after having run for `steps_per_epoch` gradient descent steps, the fitting process will go to the next epoch.

After training the model, upon testing we did not get very good results. The results have been shown in the visualization part. Hence, we used the process of Image Augmentation which takes the approach of generating more training data from existing training samples, by "augmenting" the samples via a number of random transformations that yield believable-looking images. The

goal is that at training time, our model would never see the exact same picture twice. This helps the model get exposed to more aspects of the data and generalize better.

Below is a code snippet of our augmentation part: -

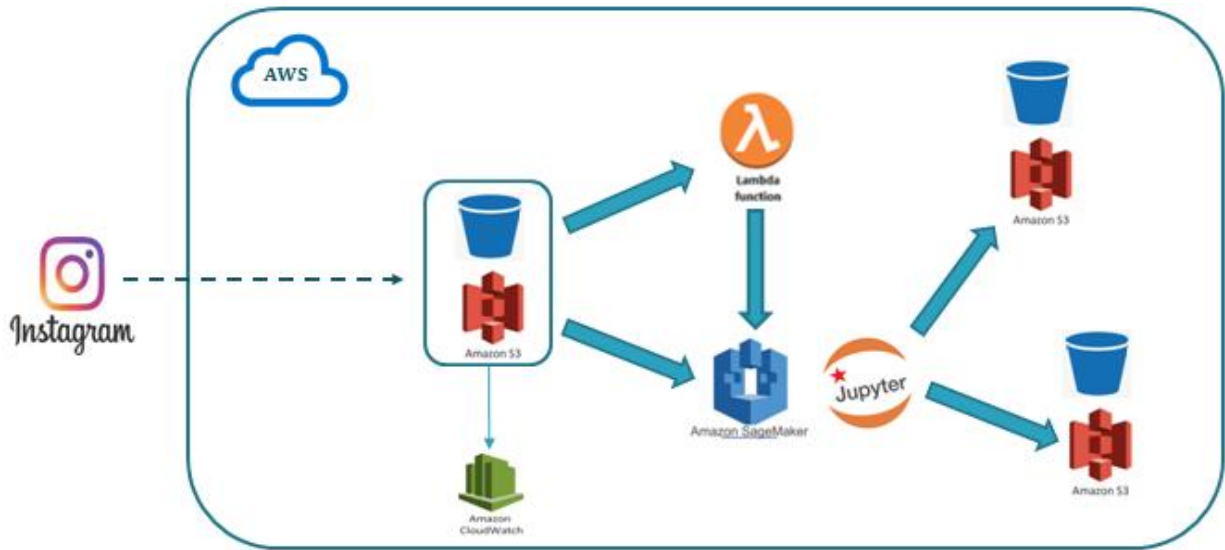
```
datagen = ImageDataGenerator(  
    rotation_range=40,  
    width_shift_range=0.2,  
    height_shift_range=0.2,  
    shear_range=0.2,  
    zoom_range=0.2,  
    horizontal_flip=True,  
    fill_mode='nearest')
```

These are just a few of the options available in Keras: -

- `rotation_range` is a value in degrees (0-180), a range within which to randomly rotate pictures.
- `width_shift` and `height_shift` are ranges (as a fraction of total width or height) within which to randomly translate pictures vertically or horizontally.
- `shear_range` is for randomly applying shearing transformations.
- `zoom_range` is for randomly zooming inside pictures.
- `horizontal_flip` is for randomly flipping half of the images horizontally -- relevant when there are no assumptions of horizontal asymmetry (e.g. real-world pictures).
- `fill_mode` is the strategy used for filling in newly created pixels, which can appear after a rotation or a width/height shift.

After using the augmentation, we saw a remarkable increase in the model accuracy, close to 98%.

## Architecture and Implementation



### Process Description:

- As per the requirement of the business the process developed should be able to fetch details from sources such as Instagram, Yelp and Google APIs. As of now, we have implemented the data scrapping of Instagram and fetching data from Yelp and Google APIs will be part of future implementation.
- The web scrapping process is a python code which invokes the API of Instagram and downloads the information of a specific Instagram account such as user details, location, tags and images.
- The images are uploaded to an AWS S3 bucket.
- The upload event of the S3 bucket triggers a Amazon Lambda function. The Lambda function in turn will trigger a Sagemaker Jupyter Notebook.
- The Sagemaker Jupyter Notebook features the CNN Image classification code and will classify the image as per the requirement. The code will give the result in terms of accuracy, whether the image has a dog or not.
- The accuracy parameter has been defined by us for now (this will be further verified with the business and changed accordingly if required). An image with an accuracy of more than 75% will be classified as having a dog in it and the images with accuracy less than 75% will be classified as without dog.
- Based on the accuracy output of the machine learning model, the content of the first bucket will be transferred to two different buckets. First, the images classified to featuring dog will be sent to a bucket which will be referred by business for further action. And the images not featuring a dog will be moved to another bucket and marked for archiving.

## Cloud Services

### **SageMaker:**

Amazon SageMaker is a fully managed service that enables data scientists and developers to quickly and easily build, train, and deploy machine learning models at any scale. Amazon SageMaker includes modules that can be used together or independently to build, train, and deploy your machine learning models.

Different phases that we typically have in the Machine Learning project Build, Train and Deploy phase. SageMaker makes it very easy to access to Data Scientist training job. One click training is the objective. SageMaker manages the configuring and deploying all the infrastructure needed to run your training jobs.

It's similar to an EC2 instance that provision on your behalf. EC2 can have a GPU power or not , it's about the compute power you use. EC2 instance has EBS volume attached to it and 5GB in size and put the Jupyter open source software installed and give an endpoint so that the developers can start working on the Data Analysis. On the Notebook instance we have bunch of prebuilt examples books like Recommendations Engine, Fraud Detection etc

SageMaker has bunch of prebuilt algorithms, we can use 3 ways to bring in Algorithms:

- Built In Algorithm developed based on the customer needs.
- We can bring on the Tensorflow or MXnet algorithms and AWS manages the Containers.
- We can bring in our own algorithm, here we need to build a Docker Image for the training job.

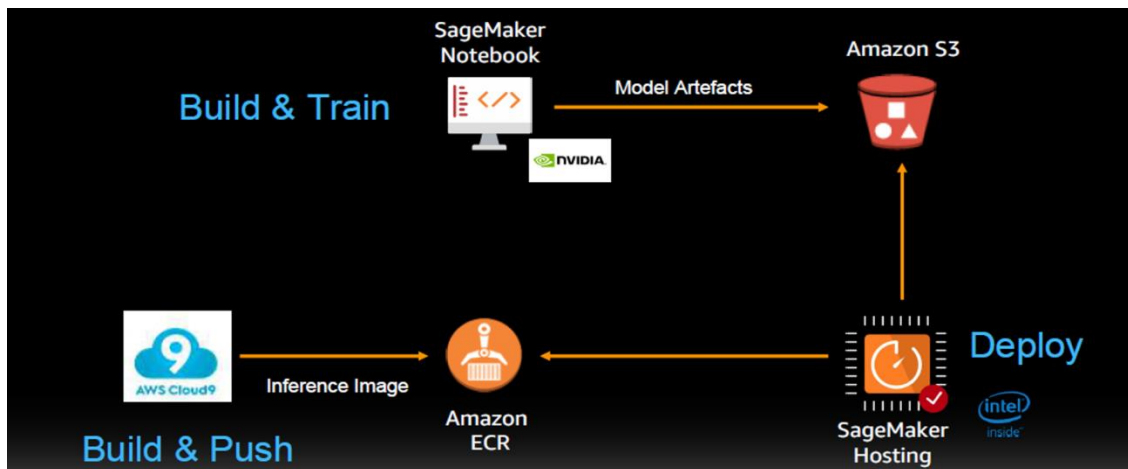
Spark Cluster does lot of data pre-processing and offloading it to the SageMaker

The foundation for the ML training Service is Docker. The way the Training flow works, it all starts with the Training data, needs to be hosted into the S3. SageMaker will pull the data the training component, depending on what algorithm you are using, built in or own algorithm it would determine the Docker Image that will pull into the service. It will spin up the infrastructure needed to train the model and at the end of it will generate a Model artefact.

E.g. if it's a NN, all of the weights, neurons in some sort of format typically a zip file. Then it will upload the model artefacts into S3

It builds on the foundation of Docker. The model artefacts are pulled into the hosting service. SageMaker will pull up the hosting infrastructure using the inference algorithm that we decide; it could be a managed one or your own custom algorithm. Then it will create an endpoint. Endpoint is an API that you give to the end-users.

Below is the model workflow for Sagemaker:



We used Amazon SageMaker to create the Jupyter notebooks which in turn were used to write our machine learning model code. Below is a screenshot of the working directory of the Jupyter notebook on Amazon SageMaker:

The screenshot shows the JupyterLab interface with the file explorer open. The files and folders in the directory are as follows:

Name	Last Modified	File size
export	11 days ago	
poochtree_v3.ipynb	Running 10 days ago	367 kB
test_poochtree1.ipynb	Running 10 days ago	5.3 kB
Untitled.ipynb	12 days ago	5.76 kB
Untitled1.ipynb	Running 10 days ago	20.8 kB
Untitled2.ipynb	Running 11 days ago	15.7 kB
46566546_511817795962668_1377583705355458276_n.jpg	10 days ago	109 kB
9747.jpg	13 days ago	47.8 kB
cats_and_dogs_small_1.h5	22 days ago	27.7 MB
dog.jpg	10 days ago	592 kB
model.tar.gz	11 days ago	49.2 MB
ntest2.jpg	10 days ago	151 kB
ntest6.jpg	10 days ago	39.9 kB
poochtree_demo_v2.h5	10 days ago	27.7 MB
test6.jpg	10 days ago	109 kB
test8.jpg	10 days ago	171 kB
train.py	11 days ago	0 B
WhatsApp1234.jpeg	10 days ago	592 kB

### CloudWatch:

Amazon Cloud Watch is a monitoring and management service that provides data and actionable insights for AWS, hybrid, and on-premises applications and infrastructure resources. With Cloud Watch, you can collect and access all your performance and operational data in form of logs and metrics from a single platform.

The main advantages of Cloud Watch are

- **Performance:** Monitoring each layer performance is a difficult task in a Multilayer architecture. Since we are not using any kind of monitoring tool here, it is difficult for us to know how our application is performing in the cloud. One solution is to employ a monitoring tool.
- **Cost:** If multiple high-performance servers are running all night. During daytime the servers are handling most of the traffic and during night they are ideal. CPU utilization is less. We are paying for the resources that we are not using. One this is to appoint a monitoring tool; the tool will send notification when the server is ideal and most used.

Amazon Cloud Watch is a monitoring tool that offers most reliable, scalable and a flexible way to monitor your resources or applications which are currently active on cloud. It offers with 2 levels of Monitoring

- **Basic level:** Free tier, will be monitored every 5 minutes and provided with limited metrics to choose from
- **Detailed Monitoring:** Will be monitored every 5 minutes with detailed metrics

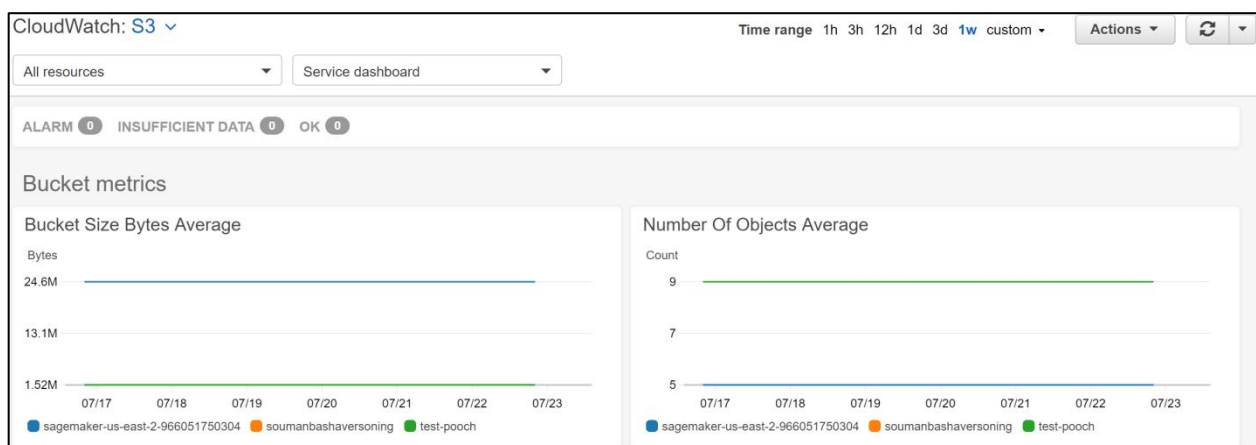
**Metrics:** A metric represents a set of time-ordered data points. Think of a metrics a variable to monitor and data points represent the values of that variable.

**Dimensions:** A dimension is a name/value pair that uniquely identifies a metrics. They can be considered as a category of characteristics that describe the metric. We can assign up to 10 dimensions to a metric

**Statistics:** Statistics are the metric data aggregations over a specific period of time. Aggregations are made using the namespace, metric name, dimensions within the time period you specify.

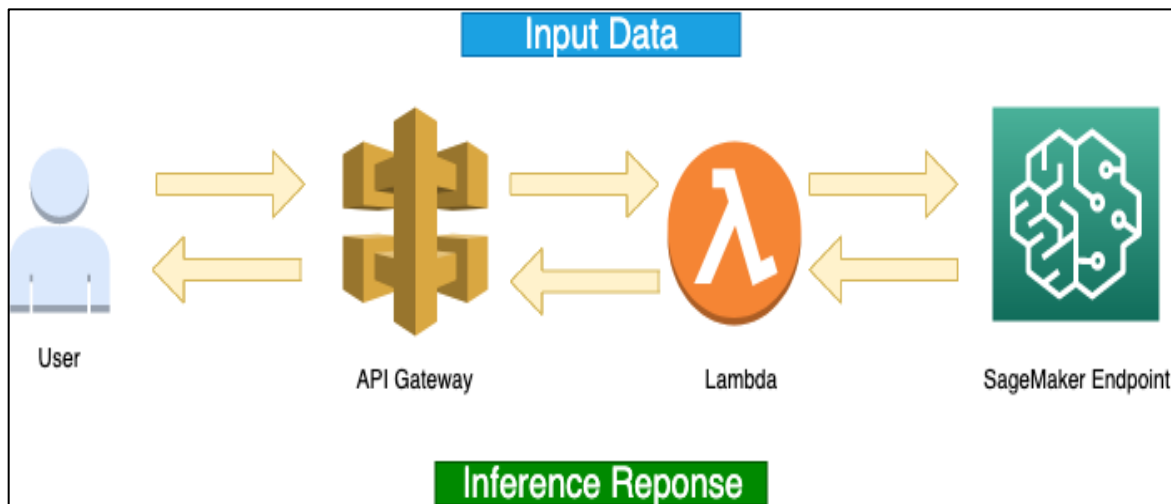
**Alarm:** An alarm can be used to automatically initiate actions on your behalf. It watches a single metric over a specified period of time and performs one or more specified actions.

In our project we used Amazon CloudWatch to monitor the S3 buckets we created, so that we can keep an eye on the usage of the space and also the utilization of the resources. Below is a screenshot of the dashboard:





## Deployment in Cloud



### AWS Lambda

Here, we have created a Lambda function which is an event-driven, serverless computing platform that calls the SageMaker Runtime `Invoke_Endpoint`. We have passed the endpoint name in the lambda function created. `Endpoint_name` is an environment variable that holds the name of the SageMaker model endpoint that is deployed.

### Why Lambda?

1. AWS Lambda supports languages like Python, Java, Node.js, Ruby, GO, C# and PowerShell.
2. You pay only for what you use.
3. There's no infrastructure to manage.
4. For the supported languages (JavaScript and Python), there's an online editor in the Lambda interface on the AWS web console. You can literally code and edit the function using the web browser.
5. It connects to API Gateways and other connection points. It acts as an interlink between other AWS services.

### AWS API Gateway

Next step is to create API which is a fully managed service that makes it easy for developers to publish, maintain, monitor, secure, and operate APIs at any scale and setup the integration request, where the integration type is lambda. The event that invokes the lambda function is triggered by API Gateway. API simply passes the test data through event. After API setup is done, all we need to do is to deploy this API created which will invoke the URL.

### Why API Gateway?

1. Elastic, Self-Service and Pay-by-Use API Facade in the Cloud.
2. It easily integrates with AWS Lambda, IAM and AWS services.
3. API Gateway supports staging and versioning that can be aligned with the rest of the application.

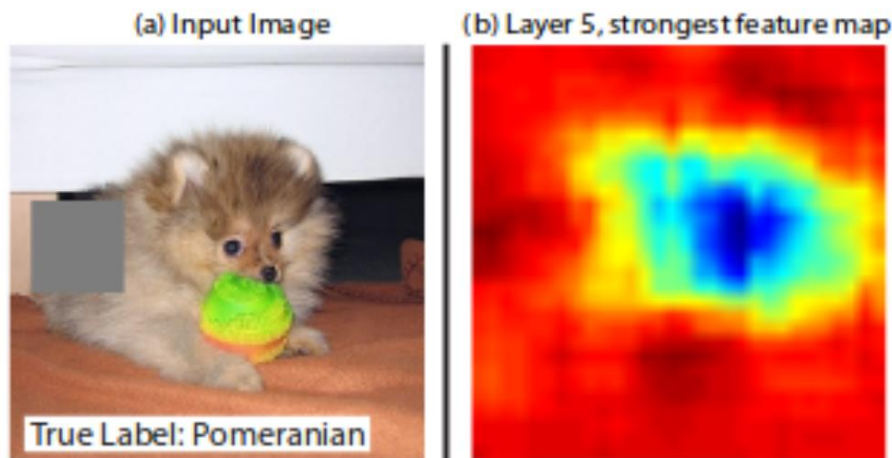
## Postman

Once API is created and URL is invoked, we can now test it with Postman which is an API development environment where you can design, debug, test, monitor or publish your API's all in one place. Here, you can paste the URL invoked and test data. We can thus see the returned predicted results.

## Data Visualization

Data Visualization is an essential phase of a deep learning process, as it helps to analyse the various parameters of the input data and which in turn facilitates the configuration of a model to provide best results. Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, pictures and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

In an image classification problem, a natural question is if the model is truly identifying the location of the object in the image, or just using the surrounding context. We took a brief look at this in gradient based methods. There is also something called Occlusion based method which attempts to answer this question by systematically occluding different portions of the input image with a grey square and monitoring the output of the classifier. The examples clearly show the model is localizing the objects within the scene, as the probability of the correct class drops significantly when the object is occluded.



In our model the below steps are essential: -

- Read the picture files.
- Decode the JPEG content to RGB grids of pixels.
- Convert these into floating point tensors.
- Rescale the pixel values (between 0 and 255) to the  $[0, 1]$  interval

But the above steps were implemented by us using Keras, which has utilities to take care of these steps automatically. Keras has a module with image processing helper tools, located at `keras.preprocessing.image`. In particular, it contains the class `ImageDataGenerator` which allows to quickly set up Python generators that can automatically turn image files on disk into batches of pre-processed tensors.

```
# We preprocess the image into a 4D tensor
from keras.preprocessing import image
import numpy as np

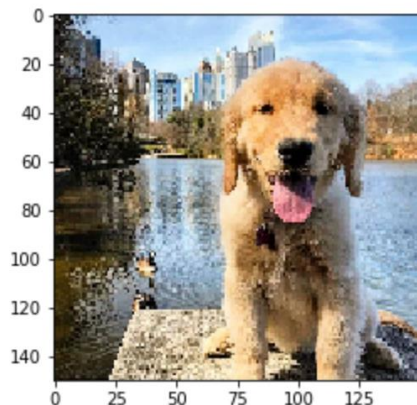
img = image.load_img(img_path, target_size=(150, 150))
img_tensor = image.img_to_array(img)
img_tensor = np.expand_dims(img_tensor, axis=0)
# The model was trained on inputs that were preprocessed in the following way:
img_tensor /= 255.

# Its shape is (1, 150, 150, 3)
print(img_tensor.shape)
```

```
(1, 150, 150, 3)
```

```
import matplotlib.pyplot as plt

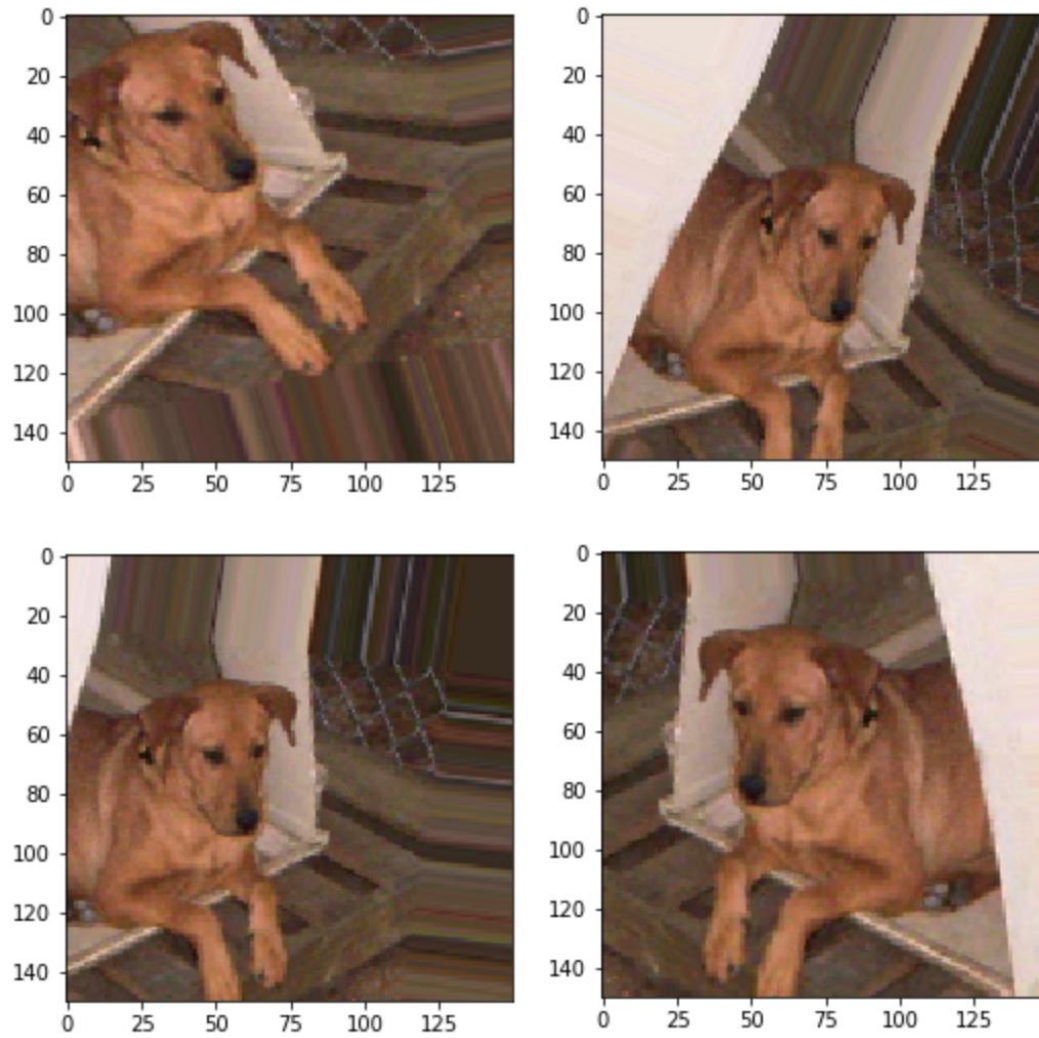
plt.imshow(img_tensor[0])
plt.show()
```



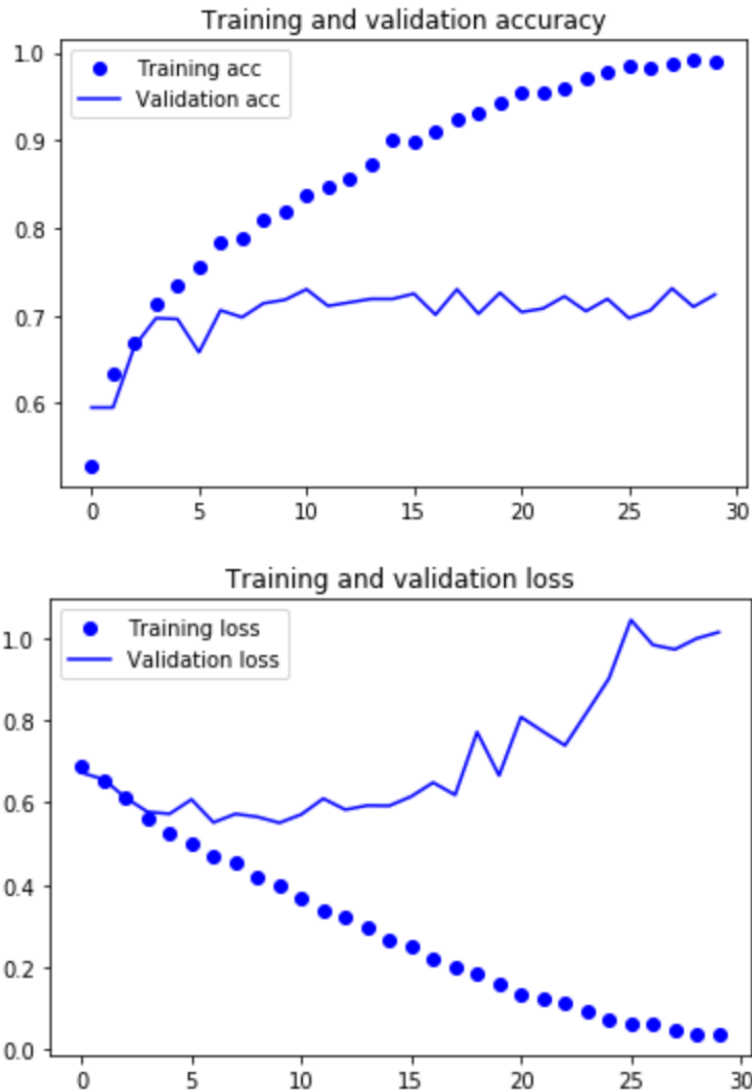
## Using Data Augmentation

Data augmentation takes the approach of generating more training data from existing training samples, by "augmenting" the samples via a number of random transformations that yield believable-looking images. The goal is that at training time, our model would never see the exact same picture twice. This helps the model get exposed to more aspects of the data and generalize better.

We used some of the options available in the `ImageDataGenerator` method of Keras module to augment our image and also visualized them to see the image transformations. Below is an example of one of the plots: -

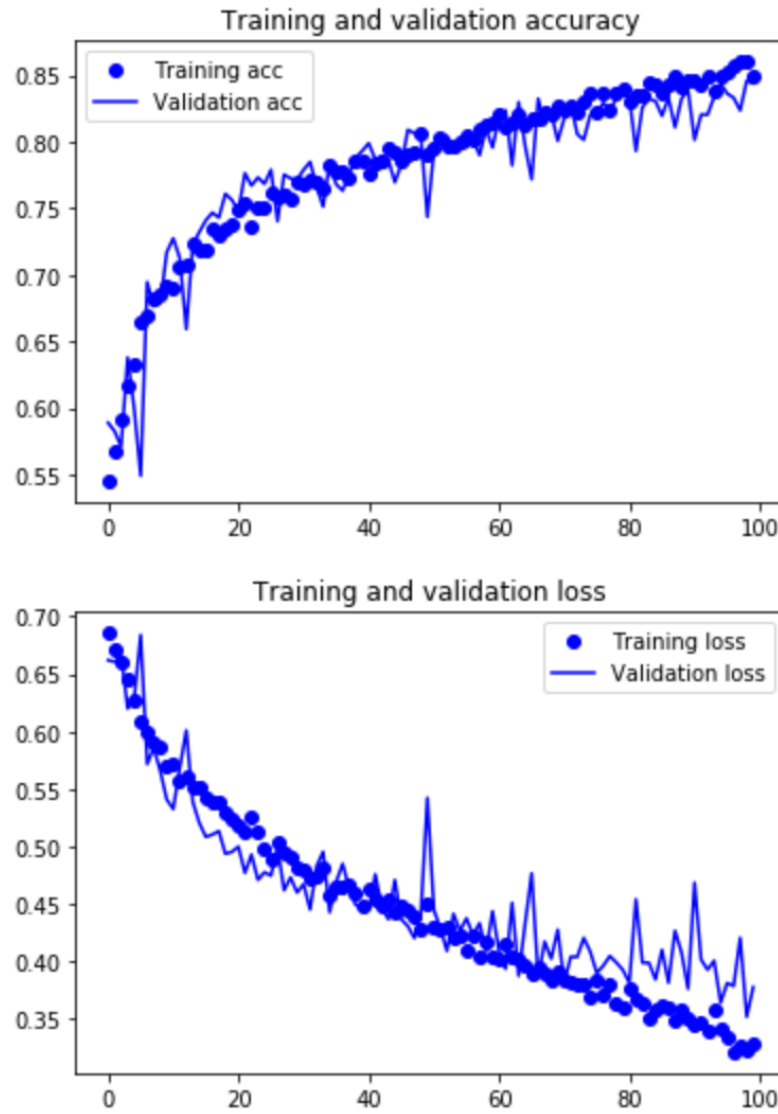


We also measured our model performance by plotting the loss and accuracy of the model over the training and validation data during training. Below is the plot we got below doing the image augmentation: -



These plots are usually the characteristics of overfitting. They show our training accuracy increases linearly over time and it reaches nearly 100%, while our validation accuracy remains stagnant at 70-72%. Also, the validation loss reaches the minimum after five epochs then stalls, whereas the training loss keeps decreasing until it reaches nearly 0.

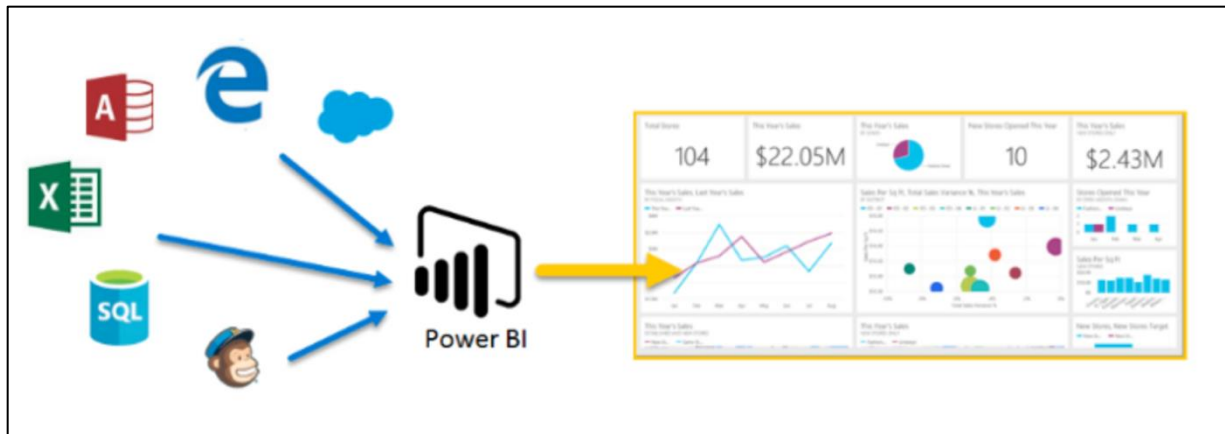
Because we only used 2000 training samples, we faced the issue of. Hence, we used image augmentation to mitigate it, and below is the result we got after image augmentation: -



## Power BI for Data Visualization

We have used Power BI to get insights of our data in such a way that everybody who sees them is able to understand their implications and acts on them accordingly. Power BI is a cloud-based business analytics service from Microsoft that enables anyone to visualize and analyze data, with better speed and efficiency. It is a powerful as well as a flexible tool for connecting with and analyzing a wide variety of data. It gives the ability to analyze and explore data on-premise as well as in the cloud.



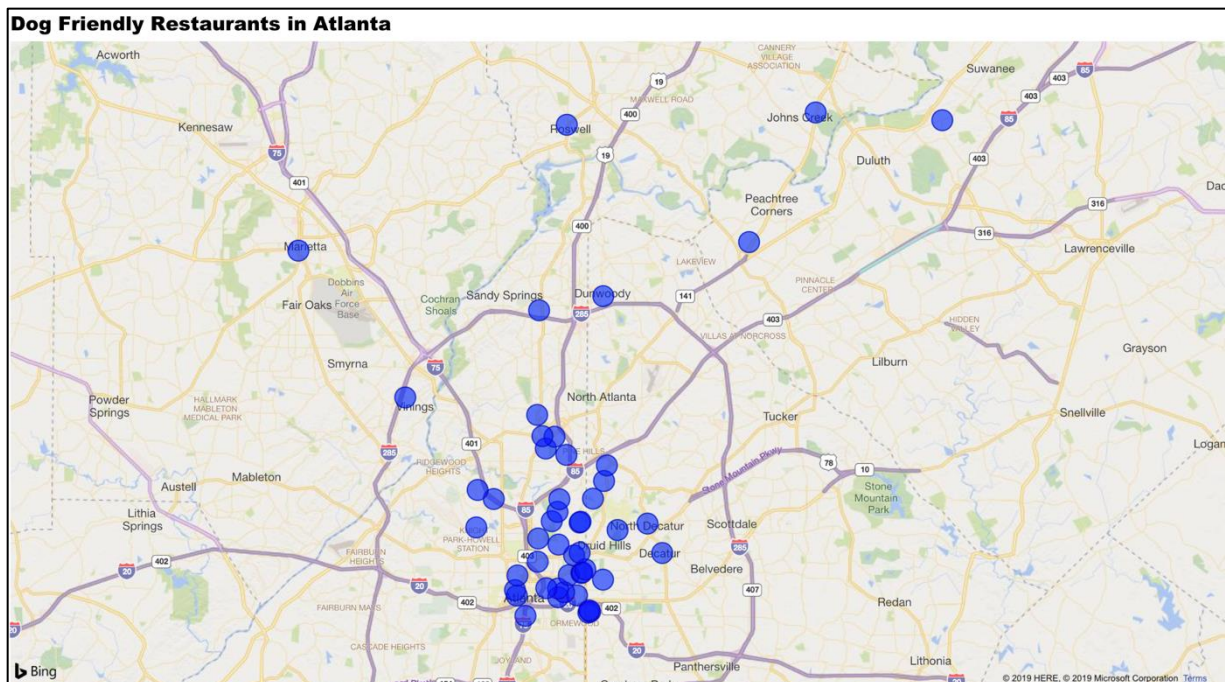


## Benefits of Power BI:

- Power BI provides certain benefits which make it superior to the existing analytical tools:
- Provides a cloud-based as well as a desktop interface.
- Provides capabilities like data warehousing, data discovery and interactive dashboards.
- Ability to load custom visualizations, and
- Easily scalable across the entire organization

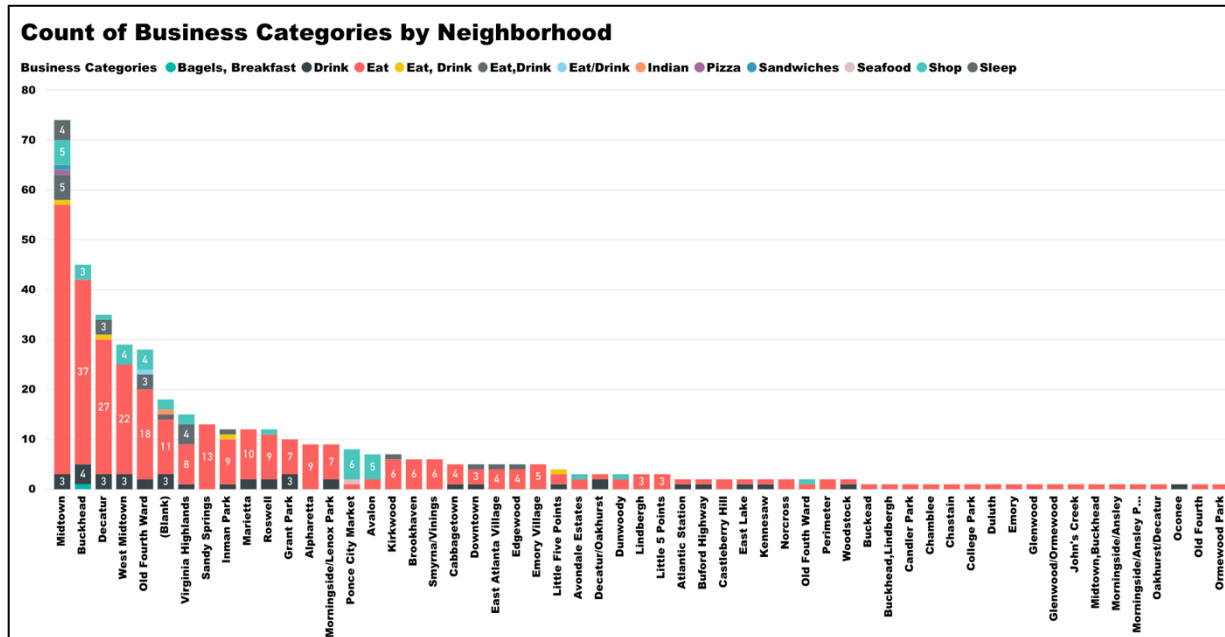
### 1. Total restaurants in Atlanta which are dog friendly

Here, we have plotted all the establishments with business category of “Restaurants” located in different areas of Atlanta. This will help the user to navigate through all the locations and pick one restaurant to visit with the pooch.



## 2. Total number of dog friendly establishments in Atlanta based on the different business categories and by neighborhood

The different business categories considered are “bagels, breakfast”, “drink”, “eat”, “eat, drink”, “eat/drink”, “Indian”, “Pizza”, “sandwiches”, “seafood”, “shop”, “sleep”. From this, we were able to understand that the highest number of dog-friendly establishments are in Midtown followed by Buckhead and Decatur. Such type of information can help the user to select the area which has many options to visit.



### Why not Tableau?

There is always a debate of whether Power BI or Tableau should be used for data visualization. Power BI aims to provide interactive visualizations and business intelligence capabilities with an interface simple enough for end users to create their own reports and dashboards. While Tableau is a business intelligence and data visualization tool that has a very intuitive user interface.

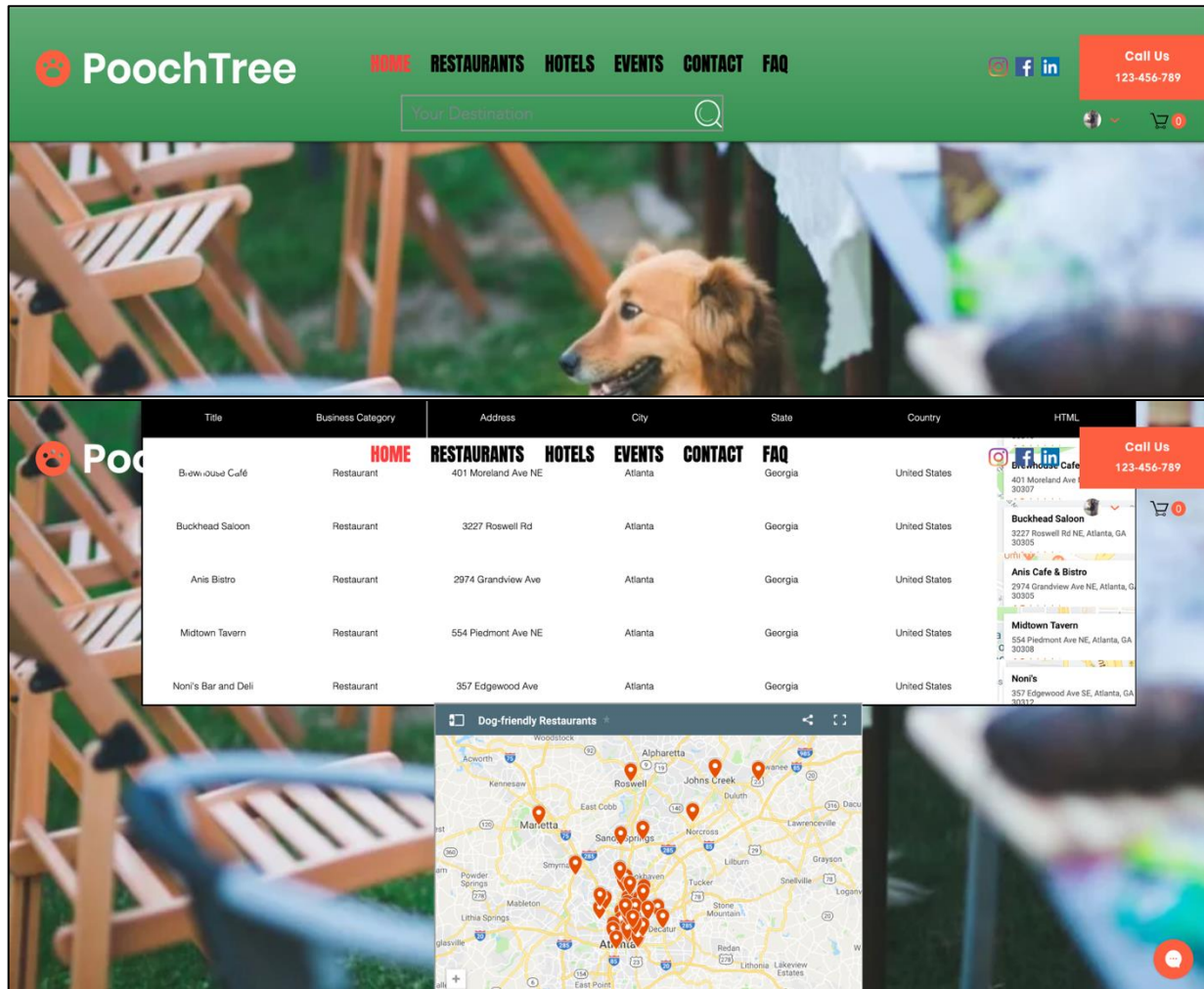
We used Power BI because of the following factors:

- **ETL/Data Discovery Suite** - Power BI has a robust set of tools for ETL and data discovery when compared to Tableau. Data prep is 90% of any reporting requirement. Reporting datamarts can be directly built into Power BI without having to go to third parties.
- **Custom Visualizations** – It lets you create your own custom visualization allowing you to explore and display your data in captivating ways.
- **Visualization** - Power BI offers a robust library of visualization types and its open framework allows users to add more visualizations to their collections via the office store. Advanced users can also implement d3.js and R visuals.

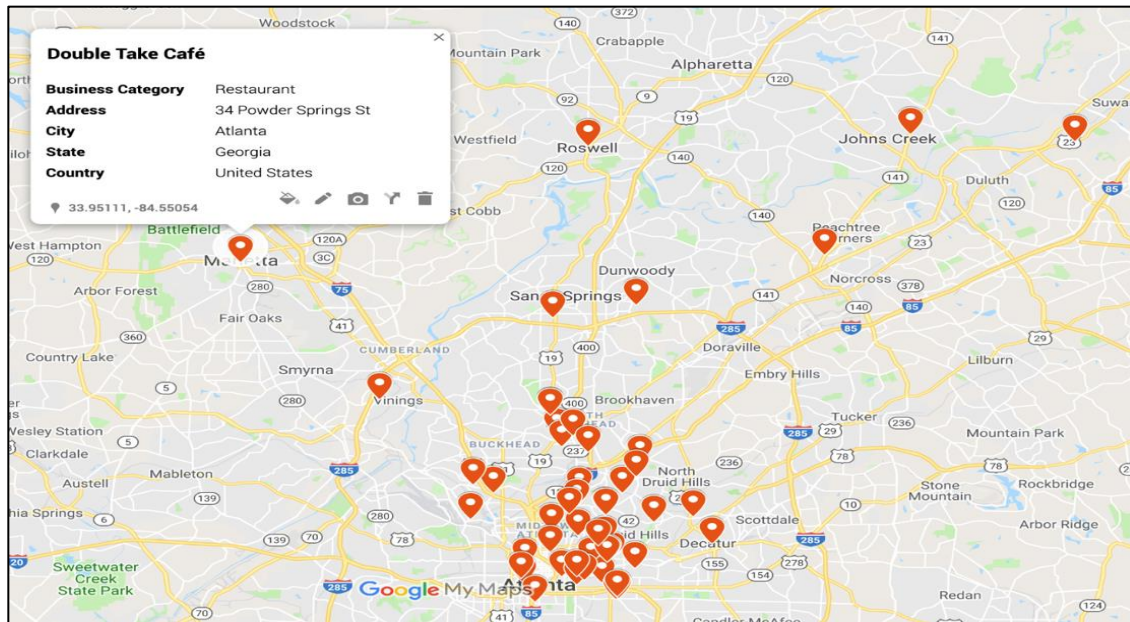


## Website Creation

We have attempted to create a prototype of the recommendation system for users to select the type of place they wish to visit with their dogs.



We have plotted all the restaurants in Atlanta using Google maps so that users can navigate through each location to get the exact address of the location.




## Future Implementation

**Yelp** – Although Yelp does have a flag for dog friendly establishments, many of the listings do not have that flag turned on or they are not accurate, but it is still a good place to check. If Yelp has the flag and a comment stating a restaurant is dog friendly or if Yelp has a flag and Instagram has a tagged picture or if Yelp has a comment and Instagram has a picture, it is very likely it is dog friendly.

Yelp Dog Friendly Flag and Other Attributes

### Yelp Comments Section



**Steve V.**  
Atlanta, GA  
470 friends  
1964 reviews  
Elite '19


★★★★★ 6/18/2019

This place is perfect for a nice day! It is located on the beltline and has a huge outdoor area. I would not even consider sitting inside. It does get crowded though so be ready to sit communally at the tables, or even stand. The service was great on this busy Saturday afternoon, with our server making the rounds about every 10 minutes.

It is **dog friendly** and offers an awesome menu, with a variety of food. The drink selection is great as well. They have a couple of family style platters on their menu, which is mostly BBQ. The group sitting next to us got it and the presentation was amazing. We ended up just getting a handful of apps, including wings, which was great and meaty. They were all wings though, with no drumsticks.

Michael S. and 1 other voted for this review

Useful 1 Funny 1 Cool 2



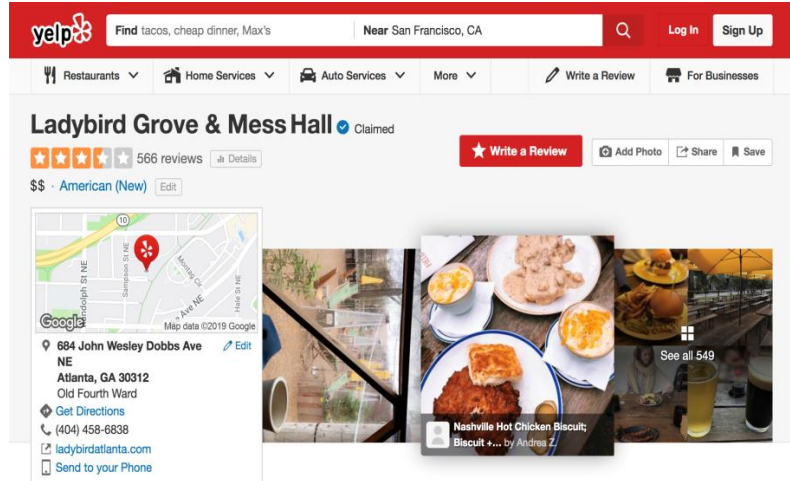
**Andrea Z.**  
Athens, GA  
0 friends  
1 review

★★★★★ 1/29/2019

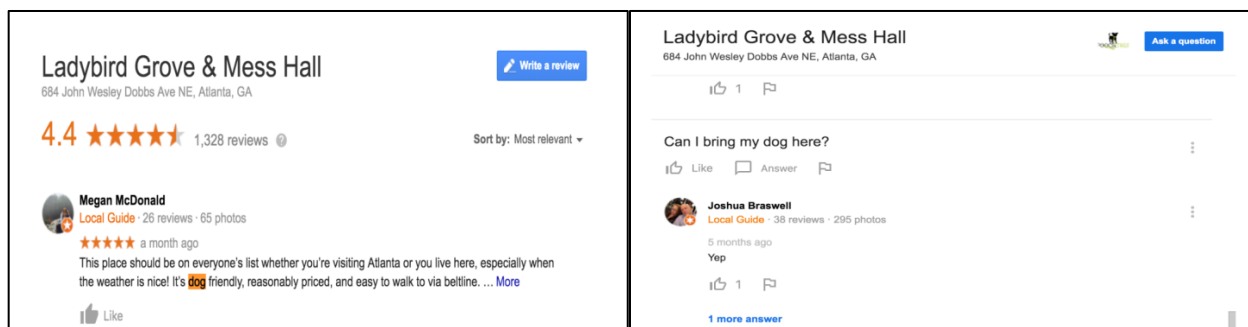
Ladybird just became one of my favorite restaurants in ATL! Staff was very attentive, the food was great and the atmosphere was even better. It's a very **dog friendly** place, right off the beltline. Highly recommend!

#### More business info

Happy Hour Specials **Yes**  
 Takes Reservations **Yes**  
 Delivery **No**  
 Take-out **Yes**  
 Accepts Credit Cards **Yes**  
 Accepts Apple Pay **No**  
 Accepts Google Pay **No**  
 Good For **Brunch, Lunch, Dinner**  
 Parking **Street, Private Lot**  
 Bike Parking **Yes**  
 Wheelchair Accessible **Yes**  
 Good for Kids **No**  
 Good for Groups **Yes**  
 Ambience **Hipster, Casual, Classy**  
 Noise Level **Average**  
 Alcohol **Full Bar**  
 Good For Happy Hour **Yes**  
 Outdoor Seating **Yes**  
 Wi-Fi **No**  
 Has TV **Yes**  
 Dogs Allowed **Yes**  
 Waiter Service **Yes**  
 Caters **No**



**Google** – When you search a listing on Google, the reviews and Q&A may detail if an establishment is dog friendly. I can purchase a Google API if needed here because we will most likely need it when development starts for the listing details.



## Challenges Faced :

As with every project, we faced many challenges in this too and tried many workarounds and different approach to solve them and move forward. Below is a list of the challenges we faced:

- Quality of data: The data that we downloaded as part of web scrapping, did not meet quality guidelines and the overall integrity of the data. While scraping we got lot of unrequired data. In addition to the Images, we got .txt , video files etc which we did not require. As part of resolution we tweaked the python code to keep only the desired data i.e. images and discard the others.
- Data required to train the Image Classification Model – As many people face, we also struggled to find enough data in order to train the image classification model. We were initially dependent on business to provide us the data, which was not enough. So, we

looked for a secondary source (Kaggle) where we could find a dataset which we were looking for. But when that was not good enough, we implemented Image Augmentation (GANS) to generate training data. This was able to solve our problem with data.

- Cloud resources while training the Image Classification Model – As we were dependent upon the free tier cloud resources, we had limitations on the storage and processing engines of the AWS infrastructure. Hence, we came up with a work around of storing the training data in the local, train the model in the local, save it and deploy it on cloud (AWS SageMaker) and use the test data from AWS S3 bucket to test the model. This also worked out efficiently for us.
- Creation of Endpoints in AWS Lambda – While creating the Lambda endpoints the free tier processors did not work. We had to raise support request to AWS in order to increase the limit of resource utilization for creating the endpoints.

## Experience & Key Learnings

The overall experience and the learning curve for this entire project was great. From the outset we were forced to think out of the box, because until now we were used to just completing projects. We got a chance to work with an entrepreneur with a real business use case. We also fetched real time or live data from the social media platforms and were able to infer from that data and use it for a business case. The concept of Product was new to us and we are so glad that we were able to achieve our goal of learning.

The learning curve for us during the entire process was very steep because of the following reasons:

Meeting up to the expectation of business to understand the difficulties they are facing right now and to be able to design an improved process and implement it was a challenge in the beginning.

We actually got to implement something which we had only studied about. The list is long such as **web scrapping, using APIs, developing image classification model (CNN), image augmentation (GANS), AWS implementation, website development** etc.

We realized that while developing individual units of the application was easy, integrating them was one of the most difficult tasks.