October 31, 2013 at 9 PM\*

Somewhere in the middle of the woods, Rosewood

Gasping for her breath, the little girl ran into the dark arms of the auburn woods seeking a sanctuary from the abhorrent creature that lay behind. The noise of the city overshadowed her deep cry of help resonating in the forest. Her legs felt numb, for the miles she had to run away from that horror was endless. But that cry - overwhelmingly loud that can easily defeat a roaring road truck- alas, had abruptly diminished. What followed were so lurid that my words shall describe no further of the agony that the girl had have gone through. But what is understood was her abduction by a deadly creature that no man of this Rosewood city had laid their eyes upon.

\*Recounted from a later date

November 1, 2013 at 9 AM

Department of Data Science, Rosewood

Inspector Toby was at my desk this morning. His abnormally hyperactive eyes, which would otherwise wear that of condescendence, signified that he carried some serious news to be deliberated right away and started reciting them without a pause that I had heard only partially and will pursue to summarize them as below:

A little girl, Alison, was abducted in the Rosewood Forests the previous day while she was wandering near the woods. No demand had been put forth by the abductor nor was her location traced by the police, yet (Spoiler alert! I traced her though, eventually.) Fortunately, she had left breadcrumbs en route of her trail using her mobile phone; she had a GPS tracker app in her mobile phone which sends her location (within 5 meters accurate) to her parents. The inspector had got hold of these locations to retrace the route.

What did surprise me, aside from the shocking abduction, was the one standing in front of me - the pompous Inspector Toby who scorns at Data Analysts - pleading to help him find Alison. I acceded to this without any hesitation, for I know the seriousness of this case. He then drew out a map, a copy of which I’ve attached in the next page. The GPS locations of the girl had been marked with ‘+’ symbols all over the map. Apparently, the locations on the map follow a pattern, so undetectable by a human eye that you’d rather search the forest inch by inch. But, these marks of locations disappeared after a while leaving the police clueless where the abductor is headed; inspector theorized that it might be due to weak cell reception in the forests.

Perplexed by my involvement in a case so intriguing like this, I couldn’t resist but ask what assistance was required of me. Being a Data Analyst, I was requested to predict the missing trial of Alison - the life of the little girl had now lain upon my hands. With resolute grit of finding her, I scurried to the book shelves and picked up a book, the title of which the inspector and I read together as:

**‘MACHINE LEARNING FOR PREDICTIVE ANALYSIS: USING TENSORFLOW**.’

November 1, 2013 at 10 AM

‘Machine learning? I hope that they could learn that Data Science is just a joke,’ was his immediate response. Neglecting his banter and recollecting what I had learnt a few years ago, I presented him a monologue while revising through the pages of the book, which I shall present it here.

‘Brain is the most mysterious and complex organ of a human body; it can immediately recognize the pleasant face of our soul mate in a crowd; understand emotions; predict what might happen intuitively; learn from mistakes; and draw out conclusions. In spite of these, we lag severely, among others, in calculating quick and accurate mathematical results, and slower memory recollection which computers (or in a sense, machine) are good at. However, having asked a machine to recognize whether it was me or my dog in an image (I’m sure that we don’t look alike except for some of the facial features,) it wouldn’t be able to distinguish without the help of a human written algorithm. What it sees of those images are 0s and 1s and blindly follows the human commands (its dominance over us is a topic of debate for another day.) Simply put, it doesn’t have the touch and soul of a human. But what if I say that we could **combine the learning capabilities of a human brain with that of a computing capability of machine**? What if machines could learn like humans? That is **Machine Learning**! And without that Data Science which is ill-treated by your boyish immatureness, Inspector, Machine Learning wouldn’t have been possible!’ (Well, I’ll admit that I didn’t say the last part.)

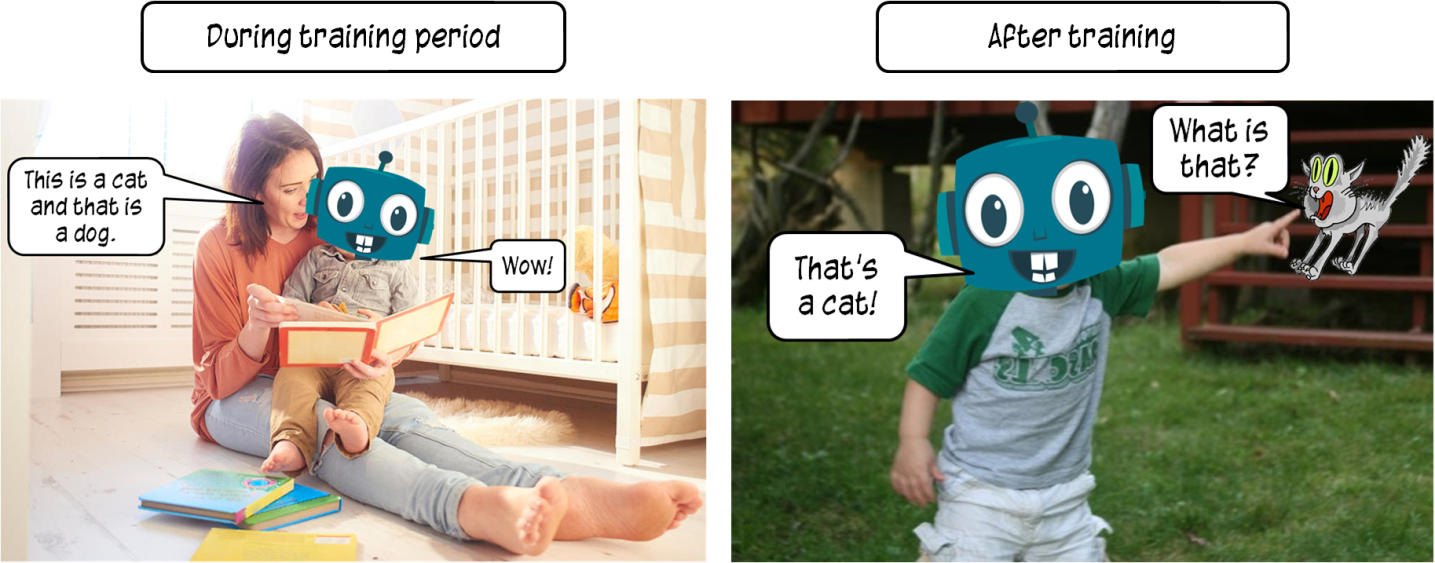
Without hesitating to stop at the puzzled face of the inspector, I continued further, as the feeling of a human company brought out my inner nerd. I’d summarize it as below:

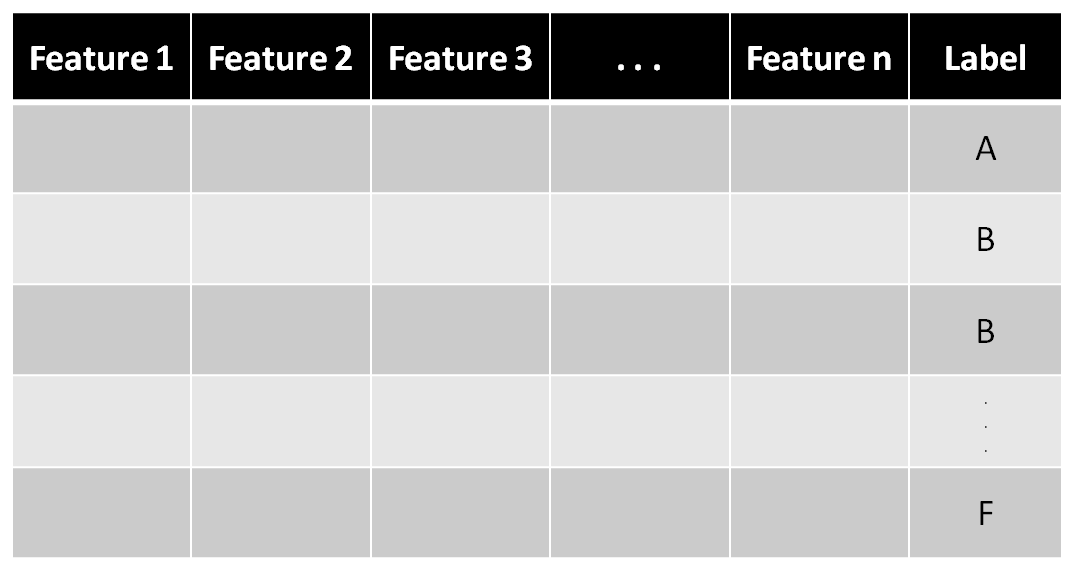
Machine Learning is a subset of a much larger domain, the Artificial Intelligence. A machine can learn using one of the following:

1. Supervised Learning (predict what might happen, intuitively)

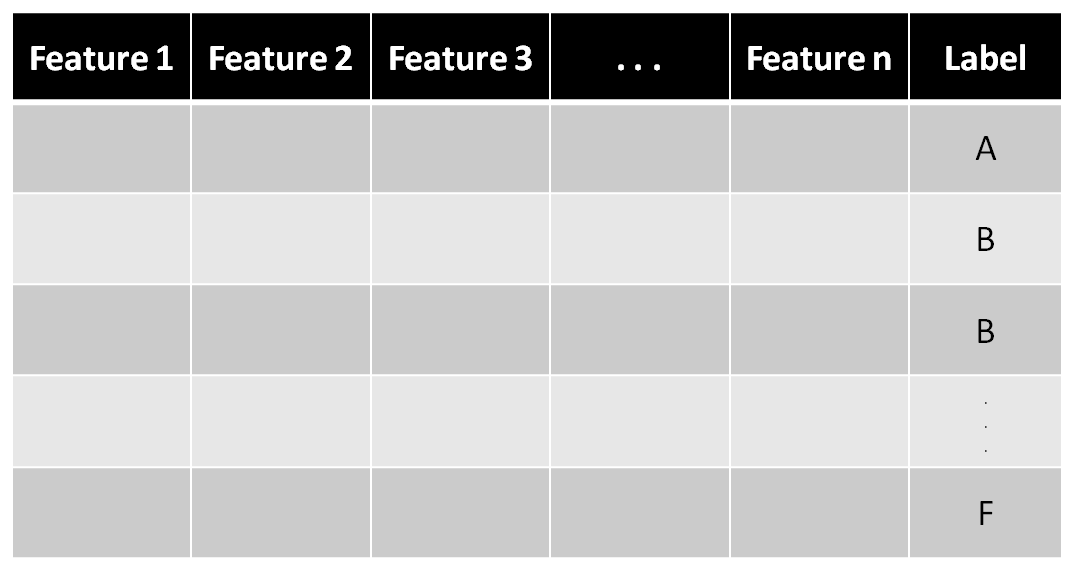
2. Unsupervised Learning (draw out conclusions)

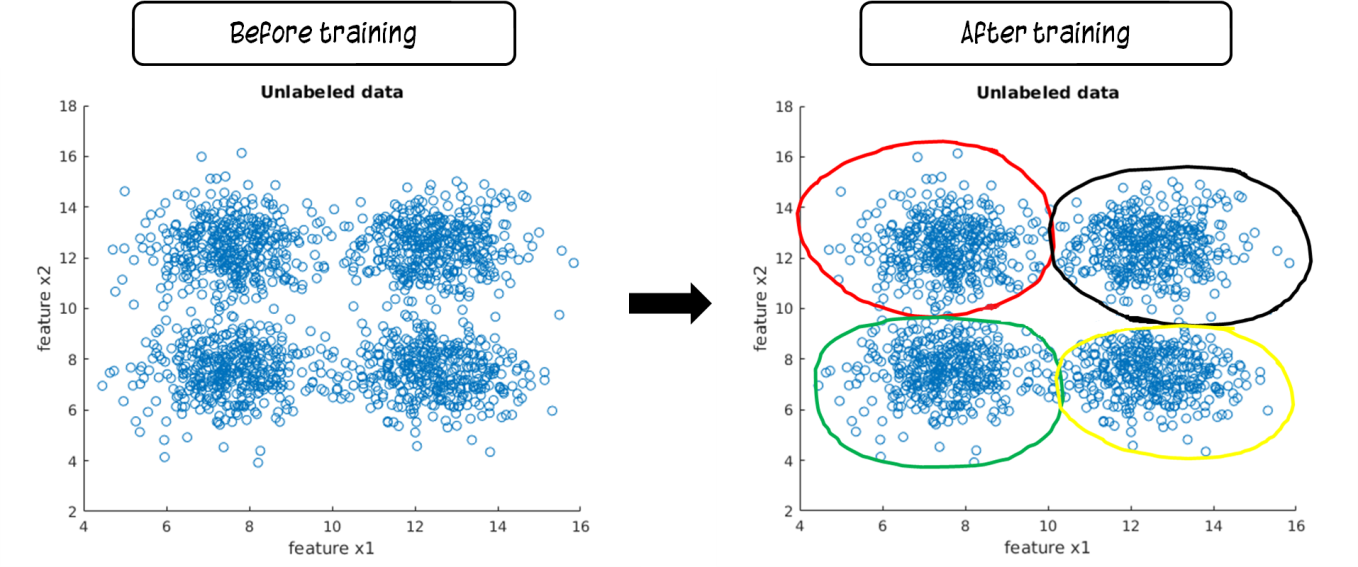
3. Reinforcement Learning (learn from mistakes)

I noticed that this was a lot for him to assimilate, so I attempted to decode each type. First, the supervised learning; it is learning under a supervisor or a teacher who knows what is right or wrong without any bias. To clarify I showed him an example as given in the book.

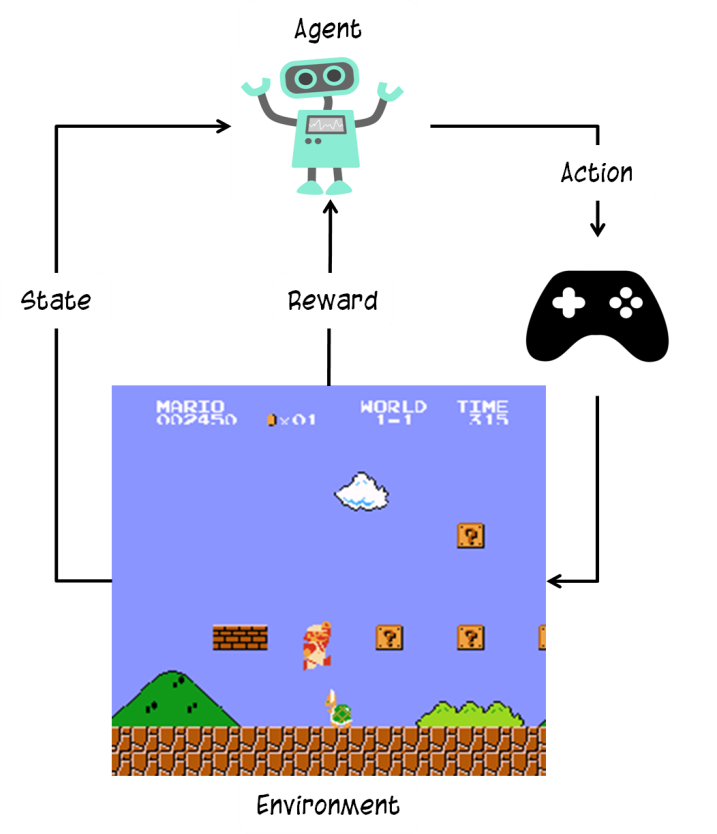
For example, we can give a large number of images of cats and dogs with correct labels (cats or dogs) as inputs to the machine. It trains on these inputs and once the training is complete, it can recognize images of cats and dogs in the future with greater accuracy. The exciting thing is no specific algorithm is hand-coded to classify these animals. These images or inputs, in general, are called features and the true outputs are called labels. After training period, the machine will be capable of labeling any other images (or features) provided to it. I have attached a sample input for supervised learning with features and labels.

Having sensed eagerness in his demeanour, I proceeded to continue my speech while extracting the coordinates of the map from the image.

Second, the unsupervised learning; where the machine learns on its own. It is provided with lots of inputs (seriously, a lot!) and trained to classify the data and draw out conclusions on its own. For example, if there are images of several objects, it can segregate them based on their colour or shape or size. There is no right or wrong answer in this type as there are no labels provided with the input unlike the supervised learning.

It comes up with interesting classifications that are quite complicated to comprehend by the human eyes. The most common approach of this type is clustering, on hearing which he pointed to the image in the book; a four blue coloured cluster spread in a graph which is segregated into groups by the machine after training.

But, I was furious when he mumbled that it’s a child’s play. Taking pity on his incomplete understanding and displaying such great aplomb, I corrected him that it is not a child’s play, especially when the number of features and data are so large for any man on this earth brave enough to track and classify them manually.

Feeling content that I’ve defeated my rival, I felt no compulsion to stop my monologue. Third, the reinforcement learning; where the machine learns from its mistakes. Immediately, he pointed out an example of computers playing video games on their own. Apparently, he was right; the computer (called as *agent*) observes the gaming *environment*, explores new combinations for playing the game on each trial, gets feedback as high scores (*rewards*) and takes corrective *actions* based on that rewards – in essence, there is no supervisor throughout the process. Since the process of learning is reinforced over several iterations, this type is called Reinforcement Learning.

The extraction of coordinates from the image of the map to the computer was completed by the time my speech had ended. I felt an urge to take a break, but my responsibility had had me sprinting to the computer to start working on the case immediately. The inspector followed me and both of us stared at the monitor, fearing the torture that Alison had have endured.

November 1, 2013 at 11 AM

Having extracted the coordinates, we stared at a list of numbers separated by commas indicating the x- and y- coordinates of Alison’s trial from the reference point of the map (and I shall attach it with this memoir as ‘Alison\_locations.csv’)

Since this is a newly installed computer, I had to install the necessary applications before predicting Alison’s trial. Though this might be a memoir which does not require briefing the technicalities involved in the case, I will, however, proceed to document anything that I deem important, as these steps might help anyone to extricate from a situation like this in the future; and obviously, I do find it exciting and satisfying to boast my intellect.

I had installed Jupyter Notebook so that it helps me to cook codes quickly and deploy them with fewer errors. I shall summarize the steps involved in installing such an excellent tool.

1. I downloaded ‘Anaconda Navigator’ from this [link](https://www.anaconda.com/download/) and then, installed it. I should add that, I had checked ‘Add Anaconda to the system PATH environment variable’ option, during installation, so that I can access the same from the command prompt.

2. However, my computer seriously lacked something which is of utmost important for my deduction in the case – TensorFlow. So, I installed the same by opening a command prompt and entering the following:

*pip install tensorflow*

*or*

*pip install --upgrade https://storage.googleapis.com/tensorflow/mac/cpu/tensorflow-1.0.1-py2-none-any.whl*

I could hear the inspector enunciating the command for which I felt obliged and proud to resort to another monologue, as being reticent is not my disposition.

‘TensorFlow is an excellent library of choice, as you shall see soon Inspector, for most of the users interested in machine learning. It provides such a great abstraction and simplification of the machine learning process that it takes lesser time to implement codes and projects with fewer bugs, albeit being powerful in performance and features. TensorFlow projects can be easily implemented using the programming language Python which makes it far more compelling and exciting to use,’ said I with such great alacrity that felt impertinent to this situation.

Once I pressed the ‘Launch’ button near ‘Jupyter Notebook’ in the Anaconda Navigator, it launched in my browser. I created a new ‘Python 3’ notebook file from ‘New’ menu. All the Jupyter notebook files have an extension ‘.ipynb’ and so did mine – ‘Untitled1.ipynb’ – which I renamed it to ‘Finding Alison’s Trial’ as I am nitpicky about trivial details (I’ve attached the same with this memoir among several other notebook files, which can be opened only with Jupyter Notebook.) I entered the following codes in the boxes, technically called ‘cells,’ of the Notebook. In an act of inquisition, the Inspector, fluttering his eyelids like the wings of a boozed butterfly, came closer to the computer. He enquired about the codes that I typed into the cells while staring wide at the computer with an impassive face without having a single glance at me, which it felt like he did not want to strain his honour in front of one he considers disdainful; I explained him the codes like I would to my baby nephew which I shall proceed henceforth (Apparently, the Inspector did some basic programming, out of compulsion of course, in his school which made my job as a tutor easier.)

I’ll import ‘tensorflow’ for implementing the Machine Learning algorithm. I’ll also import ‘numpy’ and ‘pandas’ library to handle the ‘Alison\_locations.csv’ file. These libraries provide functions to make vector operations and manipulations extremely easy and are of common occurrence in Data Science. ‘matplotlib’ is another library, which I imported, for plotting the various graphs that I’d encounter in my deduction.

*import tensorflow as tf*

*import numpy as np*

*import pandas as pd*

*import matplotlib.pyplot as plt*

*​* Having imported all the necessary libraries, I pressed ‘Shift + Enter’ on my keyboard to execute the codes in the cell. Notebook comes with several advantages; pressing ‘Tab,’ after typing a character or two, would list all the possible options available; pressing ‘Shift + Tab,’ after typing an inbuilt function or method, will show the official documentation of the function.​

I’ve stored Alison's GPS locations in a file named 'Alison\_locations.csv' from which I had extracted the data using the ‘read\_csv’ function of Pandas library.

*raw\_data* ***=*** *pd.read\_csv('Alison\_locations.csv')*

I, then, typed ‘raw\_data’ in a cell and executed it to display the data in a well-formatted table (called DataFrame in Pandas.) In Notebook, I can find the contents of a variable by just typing its name and executing it in a cell.

*raw\_data*

I’ll extract the x-location and y-location column of raw\_data into variables, x and y respectively.

*x* ***=*** *raw\_data['x-location']*

*y* ***=*** *raw\_data['y-location']*

Since the Notebook is an interactive one unlike a single script environment, the following line will display a plot within the notebook.

***%****matplotlib inline*

To visualize the data, I plotted x and y in a graph as a Scatter plot.

*plt.scatter(x, y)*

Obviously, one, with even a vision of a bat, could easily identify that it followed a linear trend (i.e. the plot looked like a straight line.) The coordinates were spread fairly apart in the graph and stopped at x = 10, for something terrible might have happened to that poor girl, which I didn’t want to ponder upon.

I think this might be the right time for me to explain my approach to solve this case as I had kept it as suspense so far. So, let me break the ice: Since the input locations look like a line, my prediction should also be a straight line, ‘y = mx + c’ where ‘m’ is its slope, ‘c’ is its y-intercept, and x and y are the Alison’s locations. Up to the x- location, x = 10, I knew Alison’s locations and for the locations, x > 10, my Machine Learning algorithm should predict the corresponding ‘y’ locations to determine the trial. I grabbed a pen and jotted down my algorithm in a paper as given below.

1. Assign initial values as 0 to the variables, ‘m’ and ‘c’.

2. Frame the equation: y = m \* x + c

3. Train the model to find the true value of ‘m’ and ‘c’ over several iterations using values of y for x <= 10 and make sure that the error is reduced at each step.

4. Use this predicted value of ‘m’ and ‘c’ to find ‘y’ locations when x > 10.

5. Plot the graph and find out the path traced by Alison.

Before getting my hands dirty, I should explain further about the three important flavours of data available in TensorFlow: constants, variables and placeholders.

1. Constants, as the name suggests, are values which do not change throughout the program.

2. Variables are containers of data which can automatically change its values, during the training, using the Machine Learning algorithm. In my algorithm, ‘m’ and ‘c’ are the variables, initially assigned to a value 0. In each iteration period of training, these values get updated with latest values such that the error is minimized.

3. Placeholders are the inputs and outputs of the Machine Learning model which are not assigned with any value, initially, and are initialized only when the training begins. Here, the values x and y will be injected to the input and output placeholders during training. Feeling contented that I had articulated my thoughts, I shall continue with my code.

X\_input and Y\_output are the input and output placeholders respectively.

*X\_input* ***=*** *tf.placeholder("float")*

*Y\_output* ***=*** *tf.placeholder("float")*

*m* ***=*** *tf.Variable(0.0)*

*c = tf.Variable(0.0)*

To find the values of ‘y’ using the assigned or updated values of ‘m’ and ‘c’ for corresponding x value, I framed the equation as:

*y\_model = m \* X\_input + c*

Now, to calculate the error between the calculated y\_model and true y values, I computed the following cost or error function:

*cost = tf.reduce\_mean(tf.square(Y\_output - y\_model))*

The above line takes the mean of squares of differences between the calculated and actual values.

TensorFlow provides a function called ‘Gradient Descent Optimizer’ which tries to reduce the error or minimize the value of cost function during training. ‘learning\_rate’ is the rate at which error is to be minimized. Using high ‘learning\_rate’ can cause the error to fluctuate while using a lower value can slow down the process. So, I used the recommended value of 0.01.

*train\_op* ***=*** *tf.train.GradientDescentOptimizer(learning\_rate = 0.01).minimize(cost)*

All along, the Inspector had been observing the code intently. So, I ventured to explain him the inner workings of my code: TensorFlow will not initialize, train or operate on any variables or operations unless it is instructed to. To demonstrate, I typed ‘m’ in a separate cell to display its value. Instead of displaying a value of 0 for m, the output showed the data type of the variable:

*<tf.Variable 'Variable\_1:0' shape=() dtype=float32\_ref>*

This is because all these variables, placeholders and operations are treated as objects which will not be executed until a special function ‘run()’ of the object type ‘Session()’ is called. First, I created a Session() object by typing the following line.

*sess* ***=*** *tf.Session()*

Note : to execute any object in TensorFlow, one could do the following:

*sess.run(object\_name)*

Though the values of variables ‘m’ and ‘c’ are indicated as 0 while declaring them, they aren’t initialized yet. So, the function ‘tf.global\_variables\_initializer()’ is used to initialize the values of all the variables created so far.

*init* ***=*** *tf.global\_variables\_initializer()*

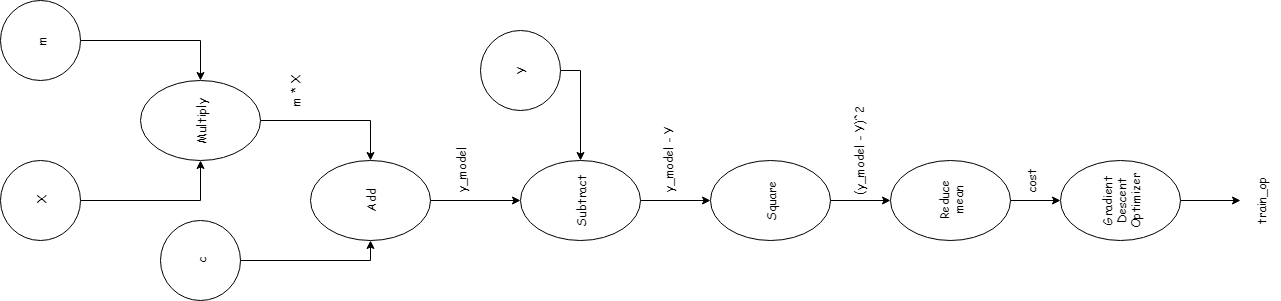
*sess.run(init)*

I decided to train the model 1000 times to get a better accuracy, where training\_epochs is the number of training iterations. In each iteration, I’d execute the object ‘train\_op’ using sess.run() function. As I’ve mentioned before that values are assigned to placeholders only during training, inputs x and y are assigned to the placeholders X\_input and Y\_output while executing the ‘train\_op’ object.

*training\_epochs* ***=*** *1000*

*for epoch in range(training\_epochs):*

*sess.run(train\_op, feed\_dict****=****{X\_input: x, Y\_output: y})*

At the beginning of my career, I wondered how executing just one object trains the entire model; this is taken care by Session() object. It automatically allocates CPU and GPU resources and calls the objects in right order. In my code, during each execution of train\_op, the following is the order of execution implicitly carried out by Session().

Now, I recovered the predicted value of ‘m’ and ‘c’ and called the sess.close() method to close all the resources.

*m\_val = sess.run(m)*

*c\_val = sess.run(c)*

*sess.close()*

Then, I calculated the learned y value using predicted values of ‘m’ and ‘c’ and plotted the same in a graph.

*y\_learned = m\_val \* x + c\_val*

*plt.scatter(x, y)*

*plt.plot(x, y\_learned, 'r')*

The Inspector was amused at my coding skills and at the straight line that exactly fitted amongst the random looking GPS points. He queried whether my code is a type of supervised learning, to which I nodded positive. I explained to him that the machine had learnt to find a continuous curve from a set of labeled inputs (hence, supervised learning) through a statistical concept called regression and since the output is a straight line, this type is often referred to as Linear Regression.

To predict the y coordinates of Alison’s trial for x > 10, I created an evenly spaced array from 10 to 20 with 100 points between them using np.linspace() method.

*predict\_x = np.linspace(10,20,100)*

*predict\_y = m\_val \* predict\_x + c\_val*

*plt.scatter(x, y)*

*plt.plot(x, y\_learned, 'r')*

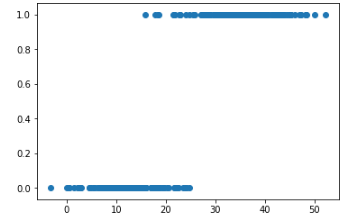
*plt.plot(predict\_x, predict\_y, 'g')*

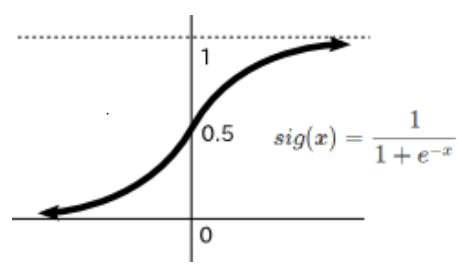
Finally, I plotted the various graphs: input graph, learned graph, and the most important of them all – the predicted graph which indicates ALISON’S TRIAL. I took a printout and handed it over to the Inspector. A sudden resilience emerged in his face, so austere that sent chills in my spine. Within seconds, he sprinted out of my room and drove towards the forest with the rescue team to retrieve Alison from that anonymous abductor.

November 1, 2013 at 3 PM

I sat down near my computer to introspect my thoughts on the events that were set in motion today and didn’t even bother to have my luncheon as I considered it an act of perfidy against Alison. Suddenly, I glanced upon a file on my computer’s desktop. It was a data set (a collection of data, e.g., census) of time period after which people, who entered the forest, went missing (it is the file named ‘Rosewood\_forest\_dataset.csv’.)

I quickly made some modifications in the ‘Finding Alison’s Trial.ipynb’ file as follows and I will reveal in doing so later (I’ve attached the same as ‘Finding Alison’s Trial Part 2.ipynb’ file.) Since I had already explained the approach, I shall brief only the major modifications that I did in the file. For clarity, one can always refer the Notebook file provided. After importing the ‘Rosewood\_forest\_dataset.csv’ using Pandas library, I extracted the DataFrame table. The table provides the time in hours after they had entered the forest and reported whether they were missing or not. The first column is the feature column or the input column and the second column represents the labels (whether the person is missing or not - '1' indicates that the person is missing and '0' denotes that he/she had returned successfully from the forest.)



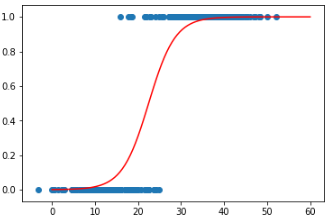
I, then, plotted the input and output columns as a scatter plot. I observed that the output has only two values: 1 or 0. For bi-labeled classification of this kind, it is recommended to use Sigmoid Functions (called Logistic Regression) instead of Linear Regression which looks like below:

I had classified values of Sigmoid function greater than or equal to 0.5 as '1' and lesser than 0.5 as '0'. Since, binary labels are used (1 or 0, A or B, cat or dog, etc.) with Sigmoid function, this type is called as Binary Classification.

If in linear regression, *y\_model = m \* x + c = linear(x),* then in logistics regression, *y = sigmoid(linear(x))*

*y\_model = tf.sigmoid(w[1] \* X\_input + w[0])*

where w[1] and w[0] are called weights, similar to ‘m’ and ‘c’ respectively. I’ll train this equation into our model to find the values of w[0] and w[1].



From the curve, 0.5 of Sigmoid curve is close to 22.5 hours. So, I checked whether the value of the sigmoid function is lesser than 0.5 or greater than or equal to 0.5 for the time period = 22 hours. Implementing this condition using the following,

*if sess.run(tf.sigmoid(now\_time \* w\_val[1] + w\_val[0])) < 0.5:*

*missing\_or\_not = 0*

*else:*

*missing\_or\_not = 1*

I obtained missing\_or\_not as 0 for now\_time = 22 and as 1 for now\_time = 23. Apparently, at 22 hours a person is less probable to be missing. So, I need not worry. Alison had been lost for only 18 hours. The Inspector will definitely find her using the trial I have predicted.

It might sound cliché, but, fate had other plans that day.

November 1, 2013 at 4 PM

Inspector Toby and his rescue team searched every inch and square tracing the predicted line using the map I provided. ‘Over here, Inspector! I found a bag!’ shouted a man in the uniform. Inspector Toby scurried to the spot. It was the girl’s bag. But she was nowhere to be found.