SOCIETY FOR SCIENCE AND EDUCATION UNITED KINGDOM
TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE
Volume 7 No 2
ISSN 2054-7390

Demonstration of Machine Learning Capabilities on Internet of Things Devices

Abhinandan H. Patil
14 Inc, Belgaum, Karnatka, India
Abhinandan_patil_1414@yahoo.com

ABSTRACT

Since the problem definition mentioned in the title of this paper is very broad it was narrowed down to temperature sensing using the IoT device and demonstrating the machine learning capabilities using the TensorFlow with the Python libraries. The data was started collecting starting 1:45 PM and collected till 6:00 PM. As the temperature in India starts cooling down from 2:00 till the evening, we should be getting down-ward slope i.e temperature starts tapering down. It is clearly linear regression problem where the slope is down-ward as we proceed further in time line. If we start collecting the data in the morning and collect till after-noon we should again get the linear regression model however this time the temperature increases as we proceed in the time line till 2:00 PM.

Keywords: Weka; Internet of Things; Machine Learning; Arduino; Python

1 Introduction

Following pre-requisites are needed for this experiment.

Table 1 Pre-requisities for the experiment

Pre-requisites	Item name	Comments
Operating System (Software)	Kubuntu/Windows	As per the performance requirements
IDE(Software)	Visual Studio 2019 and Arduino Uno	On Kubuntu bare-metal vi editor along with command prompt performs the job much faster
Python libraries (Software)	Python3, TensorFlow, matplotlib and Tk	Problem was attempted on Kubuntu with the Java Standard Edition (JSE) and with the limited success
Weka (Software)	Weka3.9	Ease of using is much better.
IoT device (Hardware)	Arduino Uno	Available online at Amazon India and far easy to use and

DOI: 10.14738/tmlai.72.6447
Publication Date: 30th April, 2019
URL: http://dx.doi.org/10.14738/tmlai.72.6447

		inexpensive compared with raspberry pi
Temperature Sensor (Hardware)	LM35	Available online at Amazon India

2 Temperature Sensor using the LM 35 Arduino Uno

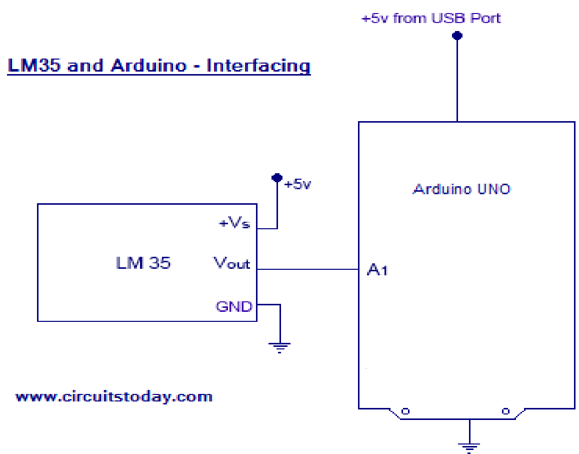

Figure 1. Circuit for LM 35

Figure 2. Arduino Interfaced with PC

Figure 3. Arduino Intefaced with PC

3 Experiment

A First interface the sensor to Arduino board using the circuit as show above. Install the Arduino IDE. Then flex the following code to the microcontroller.

============================== Code snippet ==================================

```
const int sensor=A1; // Assigning analog pin A1 to variable 'sensor'
float tempc;  //variable to store temperature in degree Celsius
float tempf;  //variable to store temperature in Fahreinheit
float vout;  //temporary variable to hold sensor reading
void setup()
{
pinMode(sensor,INPUT); // Configuring pin A1 as input
Serial.begin(9600);
}
void loop()
{
vout=analogRead(sensor);
vout=(vout*500)/1023;
tempc=vout; // Storing value in Degree Celsius
tempf=(vout*1.8)+32; // Converting to Fahrenheit
Serial.print("in DegreeC=");
Serial.print("\t");
```

```
Serial.print(tempc);

Serial.println();

Serial.print("in Fahrenheit=");

Serial.print("\t");

Serial.print(tempf);

Serial.println();

delay(60000); //Delay of 60 seconds for ease of viewing

}
```

=============================== End of Code snippet ===============================

Using the COM3 port capture the reading (Arduino Uno IDE -> Tools -> Serial Monitor).

The raw data is transferred to the text file. The Extract transform operation is manual. The data is extracted such that only two parameters are present viz.

1. Time of data capture

2. Temperature in degree Centigrade.

Then use the following code [2] to model the data in a linear regression model. Visual studio 2019 was used as the Integrated Development environment.

=========================== Start of python ML Code snippet ==========================

```
'''

A linear regression learning algorithm example using TensorFlow library.


Author: Aymeric Damien

Project: https://github.com/aymericdamien/TensorFlow-Examples/

'''

from __future__ import print_function

import tensorflow as tf

import numpy

import matplotlib.pyplot as plt

rng = numpy.random

# Parameters

learning_rate = 0.01

training_epochs = 1000
```

```
display_step = 50

# Training Data

# Change the following data as per the requirement

train_X =
numpy.asarray([13.47,13.48,13.49,13.50,13.51,13.52,13.53,13.54,13.55,13.56,13.57,13.58,13.59,14.00,
14.01,14.02,14.03,14.04,14.05,14.06,14.07,14.08,14.09,14.10,14.11,14.12,14.13,14.14,14.15,14.16,14.1
7,14.18,14.19,14.20,14.21,14.22,14.23,14.24,14.25,14.26,14.27,14.28,14.29,14.30,14.31,14.32,14.33,14.
34,14.35,14.41,14.42,14.43,14.44,14.45,14.46,14.47,14.48,14.49,14.50,14.51,14.52,14.53,14.54,14.55,1
4.56,14.57,14.58,14.59,15.00,15.01,15.02,15.03,15.04,15.05,15.06,15.07,15.08,15.09,15.10,15.11,15.12,
15.13,15.14,15.15,15.16,15.17,15.18,15.19,15.20,15.21,15.22,15.23,15.24,15.25,15.26,15.27,15.28,15.2
9,15.30,15.31,15.32,15.33,15.34,15.35,15.41,15.42,15.43,15.44,15.45,15.46,15.47,15.48,15.49,15.50,15.
51,15.52,15.53,15.54,15.55,15.56,15.57,15.58,15.59,16.00,16.01,16.02,16.03,16.04,16.05,16.06,16.07,1
6.08,16.09,16.10,16.11,16.12,16.13,16.14,16.15,16.16,16.17,16.18,16.19,16.20,16.21,16.22,16.23,16.24,
16.25,16.26,16.27,16.28,16.29,16.30,16.31,16.32,16.33,16.34,16.35,16.41,16.42,16.43,16.44,16.45,16.4
6,16.47,16.48,16.49,16.50,16.51,16.52,16.53,16.54,16.55,16.56,16.57,16.58,16.59,17.00,17.01,17.02,17.
03,17.04,17.05,17.06,17.07,17.08,17.09,17.10,17.11,17.12,17.13,17.14,17.15,17.16,17.17,17.18,17.19,1
7.20,17.21,17.22,17.23,17.24,17.25,17.26,17.27,17.28,17.29,17.30,17.31,17.32,17.33,17.34,17.35,17.41,
17.42,17.43,17.44,17.45])

train_Y =
numpy.asarray([73.35,76.87,70.71,68.95,68.95,68.07,72.47,69.83,69.83,66.31,75.11,75.11,62.79,64.55,
64.55,75.99,62.79,75.11,68.07,71.59,64.55,75.11,66.31,75.11,71.59,61.91,59.27,72.47,72.47,61.91,74.2
3,70.71,65.43,62.79,72.47,63.67,70.71,67.19,73.35,61.91,68.07,72.47,72.47,72.47,73.35,71.59,70.71,75.
99,60.15,72.47,68.95,76.87,71.59,75.99,68.95,69.83,68.95,73.35,60.15,73.35,73.35,75.11,67.19,62.79,7
0.71,68.07,63.67,68.95,76.87,75.11,75.11,72.47,68.95,69.83,76.87,70.71,75.99,75.11,63.67,72.47,59.27,
75.99,57.51,71.59,73.35,58.39,63.67,66.31,75.99,72.47,73.35,68.07,57.51,72.47,68.95,76.87,75.99,67.1
9,61.91,69.83,64.55,58.39,75.99,70.71,76.87,64.55,61.03,70.71,76.87,76.87,61.91,72.47,72.47,70.71,61.
03,73.35,61.03,69.83,68.07,76.87,68.07,60.15,61.91,64.55,67.19,64.55,65.43,61.91,61.03,70.71,61.03,6
8.07,76.87,75.99,65.43,75.11,74.23,68.95,76.87,70.71,59.27,60.15,75.99,72.47,72.47,67.19,69.83,68.07,
69.83,62.79,69.83,75.99,69.83,72.47,72.47,72.47,73.35,75.11,70.71,68.95,61.03,61.03,71.59,70.71,71.5
9,68.95,66.31,61.03,75.11,73.35,70.71,73.35,61.03,72.47,72.47,71.59,68.95,72.47,67.19,74.23,72.47,69.
83,70.71,69.83,72.47,61.03,71.59,63.67,67.19,75.11,69.83,71.59,68.95,71.59,74.23,60.15,59.27,62.79,7
0.71,72.47,69.83,69.83,70.71,70.71,70.71,70.71,69.83,68.07,66.31,73.35,61.03,72.47,75.11,68.95,74.23,
69.83,69.83,68.07,70.71])

n_samples = train_X.shape[0]

# tf Graph Input

X = tf.placeholder("float")

Y = tf.placeholder("float")


# Set model weights
```

```
W = tf.Variable(rng.randn(), name="weight")
b = tf.Variable(rng.randn(), name="bias")

# Construct a linear model
pred = tf.add(tf.multiply(X, W), b)

# Mean squared error
cost = tf.reduce_sum(tf.pow(pred-Y, 2))/(2*n_samples)
# Gradient descent
#  Note, minimize() knows to modify W and b because Variable objects are trainable=True by default
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)

# Initialize the variables (i.e. assign their default value)
init = tf.global_variables_initializer()

# Start training
with tf.Session() as sess:

    # Run the initializer
    sess.run(init)

    # Fit all training data
    for epoch in range(training_epochs):
        for (x, y) in zip(train_X, train_Y):
            sess.run(optimizer, feed_dict={X: x, Y: y})

        # Display logs per epoch step
        if (epoch+1) % display_step == 0:
            c = sess.run(cost, feed_dict={X: train_X, Y:train_Y})
            print("Epoch:", '%04d' % (epoch+1), "cost=", "{:.9f}".format(c), \
                "W=", sess.run(W), "b=", sess.run(b))
```

```
print("Optimization Finished!")
training_cost = sess.run(cost, feed_dict={X: train_X, Y: train_Y})
print("Training cost=", training_cost, "W=", sess.run(W), "b=", sess.run(b), '\n')

# Graphic display
plt.plot(train_X, train_Y, 'ro', label='Original data')
plt.plot(train_X, sess.run(W) * train_X + sess.run(b), label='Fitted line')
plt.legend()
plt.show()

# Testing example, as requested (Issue #2) KINDLY IGNORE THIS FOR TIME BEING
test_X = numpy.asarray([6.83, 4.668, 8.9, 7.91, 5.7, 8.7, 3.1, 2.1])
test_Y = numpy.asarray([1.84, 2.273, 3.2, 2.831, 2.92, 3.24, 1.35, 1.03])

print("Testing... (Mean square loss Comparison)")
testing_cost = sess.run(
    tf.reduce_sum(tf.pow(pred - Y, 2)) / (2 * test_X.shape[0]),
    feed_dict={X: test_X, Y: test_Y})  # same function as cost above
print("Testing cost=", testing_cost)
print("Absolute mean square loss difference:", abs(
    training_cost - testing_cost))

plt.plot(test_X, test_Y, 'bo', label='Testing data')
plt.plot(train_X, sess.run(W) * train_X + sess.run(b), label='Fitted line')
plt.legend()
plt.show()
```

======================== End of python ML code snippet============================

No changes were made to this code. Only data set was modified to suit the needs. However, python3, TensorFlow, matplotlib and tk libraries were installed to make the code work. The same thing was tried on Kbuntu 18.10 without IDE and performance was much superior. The mentioned job can be done with

Weka as well [3]. The data was put in the comma separated value file. It was noticed that the Weka did the same job but with lot of ease.

4 Results

4.1 TensorFlow Output

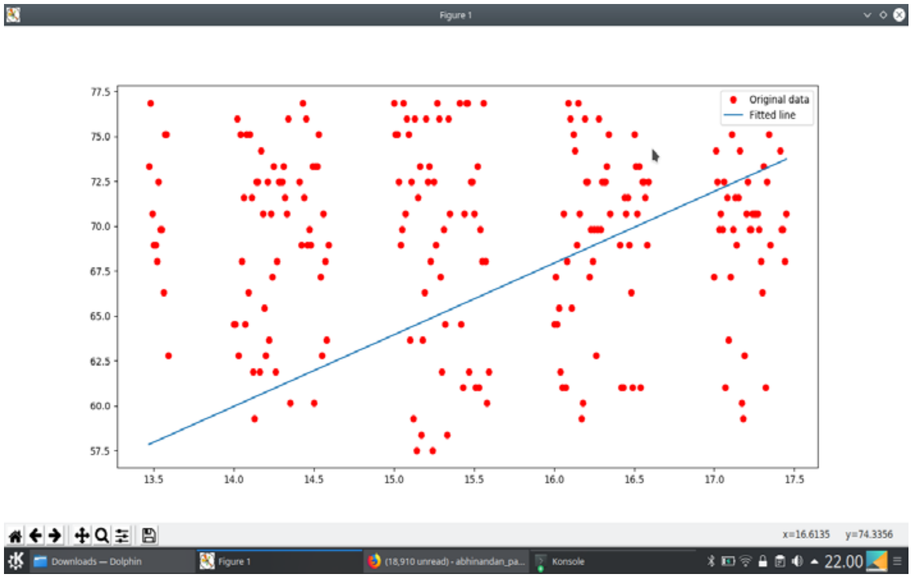

Figure 4. TensorFlow OutPut for Experiment

4.2 Weka Output

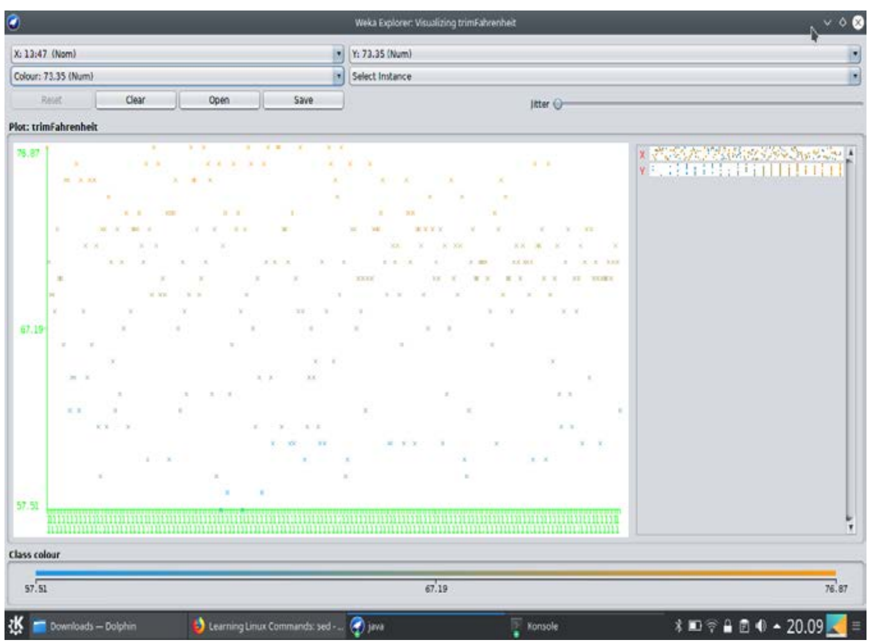

Figure 5. Weka OutPut for given Experiment

APPENDIX A

Disclaimer: The temperature data was collected at Belgaum, Karnataka, India at room-temperature with ceiling fan on and may not be the exact readings. Further the data was collected using the USB port interfaced with the Windows 10 and Arduino IDE on COM3 port. The error rate was noticed to be high.

APPENDIX FOR ETL

Sample output from Arduino COM3 port:

13:47:36.282 -> in DegreeC= 22.97

13:47:36.282 -> in Fahrenheit= 73.35

13:48:36.290 -> in DegreeC= 24.93

13:48:36.324 -> in Fahrenheit= 76.87

13:49:36.326 -> in DegreeC= 21.51

13:49:36.326 -> in Fahrenheit= 70.71

13:50:36.311 -> in DegreeC= 20.53

13:50:36.345 -> in Fahrenheit= 68.95

13:51:36.335 -> in DegreeC= 20.53

13:51:36.335 -> in Fahrenheit= 68.95

13:52:36.347 -> in DegreeC= 20.04

13:52:36.347 -> in Fahrenheit= 68.07

13:53:36.344 -> in DegreeC= 22.48

13:53:36.378 -> in Fahrenheit= 72.47

13:54:36.356 -> in DegreeC= 21.02

13:54:36.389 -> in Fahrenheit= 69.83

13:55:36.368 -> in DegreeC= 21.02

13:55:36.402 -> in Fahrenheit= 69.83

13:56:36.392 -> in DegreeC= 19.06

13:56:36.392 -> in Fahrenheit= 66.31

13:57:36.403 -> in DegreeC= 23.95

13:57:36.403 -> in Fahrenheit= 75.11

1) cat veryimpdump.txt | awk '/Fahrenheit/ {print}'| sed 's/-> in Fahrenheit=/,/g' > Fahrenheit.txt

2) cat veryimpdump.txt | awk '/DegreeC/ {print}'| sed 's/-> in DegreeC=/,/g' > DegreeC.txt

3) cat DegreeC.txt | sed 's/.[0-9]\{3\}//g'

4) cat Fahrenheit.txt | sed 's/.[0-9]\{3\}//g'

5) Manual process to remove insignificant time lines

6) cat afterhours.txt | awk '/Fahrenheit/ {print}'| sed 's/-> in Fahrenheit=/,/g'| sed 's/.[0-9]\{3\}//g'

7) cat train.csv | awk '{print $1 $2}'| grep -v '^$' | paste -s -d"," | sed 's/,,/,/g'

8) cat train.csv | awk '{print $3}' | sed -e :a -e 'N;s/\n/,/;ba'

REFERENCES

[1] CircuitsToday.2019. CircuitsToday. Retrieved from CircuitsToday: http://www.circuitstoday.com/lm35-and-arduino-interfacing

[2] Daimen, A. 2019. Americ. From Americ: https://github.com/aymericdamien/TensorFlow-Examples

[3] Edu, K. 2019. Kean Edu. From Kean Edu: https://www.kean.edu/~fosborne/bstat/09rc.html

[4] Weka. 2019. University of Waikato. From University of Waikato: https://www.cs.waikato.ac.nz/ml/weka/