

**MET CS688**

**Web Analytics and Mining**

ZLATKO VASILKOSKI

# Defining IoT

- The Internet of things (IoT) is relatively recent term and refers to interconnecting of any physical devices (things) embedded with electronics which enables these devices to collect and exchange data across existing network infrastructure. Typically, IoT refers to gateways and protocols to talk to devices that might be in cars, supermarkets, university campuses, nuclear powerplants or anywhere else. IoT gateways and protocols enable connectivity of these devices to the cloud, from where the cloud applications can directly talk to these devices.
- In 2013 the United Nations specialized agency for information and communication technologies, the Global Standards Initiative on Internet of Things (IoT-GSI) defined the IoT as:

"A global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies." (<http://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx>)

# Defining IoT

- Examples of “things” can be found almost in any aspect of our life. From sensors in facilities, such as the buildings where we work or live, powerplants that generate the energy we consume, municipal drinking water and wastewater management systems, to smart vehicles, and smart devices that surround us. All of these “things” allow the equipment they are connected with to be sensed or controlled remotely, in a streaming fashion, using the existing network - the Internet.
- This is an automation in nearly all the aspects of our lives, enabling advanced large-scale application, such as smart power grids or smart cities, that have been contemplated for some time already. There are much more “things” than people, so the scale of these changes is expected to be larger than the expansion of the internet over the last two decades, when large part of world’s population got connected to the internet.

# IIoT – Industrial Internet of Things

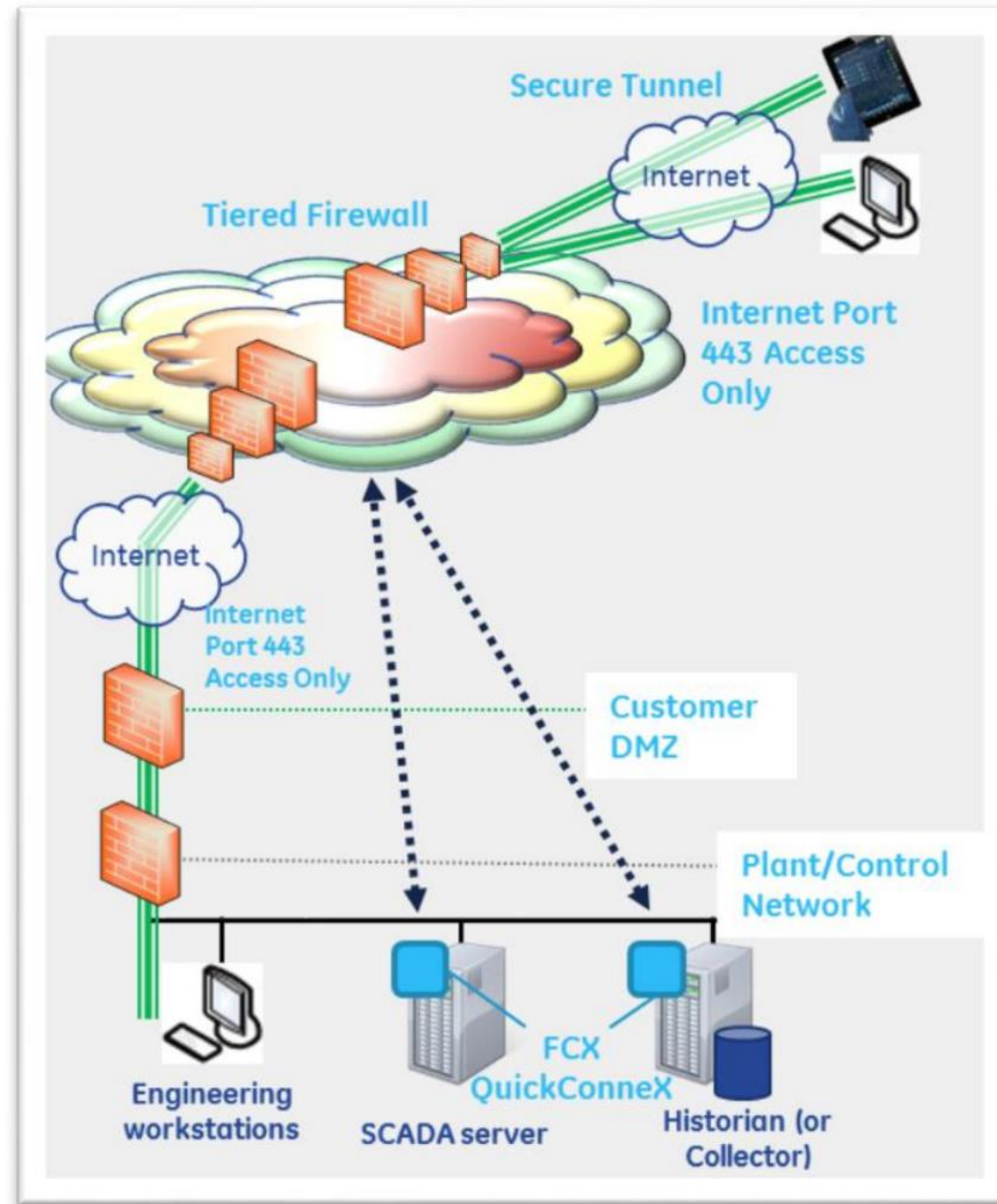
- The IIoT refers to university campuses, production plants, powerplants, hospitals etc. which with one word can be termed facilities. All of the facilities have automatic and centralized control of a building's heating, ventilation and air conditioning, lighting and energy consumption. This system is called building automation system (BAS). It may control any other aspects, besides the mentioned ones, that might be specific to a particular type of facility. IIoT is an extension to the facilities building automation system.
- Nearly every building, plant or powerplant today is designed with energy efficiency in mind. They need to stay optimized over a very long term, which is their lifetime. For example, the lifetime of a university building can be few hundred years. It becomes apparent that system degradation and facility's operation falling from peak efficiency is a constant challenge in a need for optimization. This ongoing process to resolve facility's operating problems, improve comfort, and optimize energy use is termed continuous commissioning.

# IIoT – Industrial Internet of Things

- The development of IIoT over the last few years was designed to collect and analyze equipment and environmental data to provide notification and operational recommendations about facility's optimal operation. With other words to provide continuous commissioning.
- Just as an illustration, a large university campus such as BU typically spends \$30 million per year on fuel. A continuous commissioning implemented over the IIoT typically can bring 3% to 5% savings in fuel consumption. This accumulates to yearly savings in millions of dollars on fuel only, which is not to be ignored. If not continuously commissioned, losses of a university campus for example can accumulate to hundreds of millions of dollars over a period of 10 years. On country level, the opportunities for savings are even larger. For example, US spends \$29 billion each year on air-conditioning, consuming 6% of all the electricity produced in the United States. That is why great savings can be achieved by implementing an IIoT analytics.
- The power of continuous commissioning is the ease with which analytics can be implemented over IIoT to large set of facilities to achieve the desired improvements. Typically, the goal of the implemented continuous commissioning analytics is to
  - Detect abnormal condition scenario and report on it - fault detection and diagnostics (FDD).
  - Forecasting – predict certain scenarios based on collected data.

# Connecting to the Edge with FCX

- Let us illustrate the IIoT's infrastructure on the example how FacilityConneX (FCX) platform is connecting to the Edge. FacilityConneX ([www.facilityconnex.com](http://www.facilityconnex.com)) is a cloud-based monitoring and analytics service, leveraging aspects of General Electric (GE) platform for the Industrial Internet, Amazon Web Services (AWS) for the cloud services and EMC for security technology. The basic IIoT infrastructure used by FacilityConneX (as described in one of their white papers at <https://www.facilityconnex.com/white-papers/>) is shown to the right.





# IIoT – Industrial Internet of Things

- The data, in real time, is streaming over internet from the edge devices (things), such as BMS, to the cloud platform remotely. FCX is using aspects of the GE platform as the base of its IoT capabilities. The raw streaming data is stored on the cloud into a Historian database. All the other data used for any analytics, that also resides on the cloud, is derived from this database source. This includes analytic results such as analytic alarms and notifications that are being sent over the internet to customers to analyze using the FCX user interface.
- The Historian is the central hub for all of FCX's customer raw data. Historian is a "Database" for storing "Time-Series" data points of varying data types from things. It is well suited for things like equipment sensor data. Unlike, for example SQL, Historian is not a relation database. Every data point from a sensor tag is represented in a form of three components with the following data types
- Sensor tag name (string)
  - Timestamp (ISO format)
  - Value (double)
  - Quality (integer)
- Depending on the sensor type, the value component may have any value, including negative values or zero. This is why the quality component of the data point is needed. It is used to flag if the streaming data from the sensor is available or not. The bad quality in this flag can indicate situations such as lost connection to the sensor, broken sensor or some event of similar nature preventing the flow of data to the Historian database.
- Historian database was designed for high speed and efficient data ingestion. In addition, it efficiently compresses the data. For example, for each consecutive time stamp it does not store millions of data points that are all the same.

# IoT – Internet of Things

- The Internet of things (IoT) refers to interconnecting of any physical devices (things) embedded with electronics which enables these devices to collect and exchange data across existing network infrastructure.
- Typically, IoT refers to gateways and protocols to talk to devices (cars, refrigerators etc.) enabling connectivity of these devices to the cloud, from where the cloud applications can directly talk to these devices.
- “Things” can be found almost in any aspect of our life. From sensors in facilities, such as the buildings where we work or live, powerplants that generate the energy we consume, municipal drinking and wastewater management systems, to vehicles, and smart devices that surround us.
- This is an automation in nearly all the aspects of our lives, enabling advanced large-scale application, such as smart power grids or smart cities.
- The scale of these changes is expected to be larger than the expansion of the internet from two decades ago, when large part of world’s population got connected to the internet.



# IIoT – Industrial Internet of Things

- System degradation and facility operation falling from peak efficiency is a constant challenge in a need for optimization.
- **Continuous Commissioning** - ongoing process to resolve operating problems, improve comfort, and optimize energy use.
- IIoT - an extension to facilities building automation system (BAS). IIoT encompasses university campuses, production plants, powerplants, hospitals etc. It is designed to collect and analyze equipment and environmental data to provide notification and operational recommendations resulting in a continuously commissioned facility.
- Nearly every building, plant or powerplant today is designed with energy efficiency in mind. They need to stay optimized over a very long term, which is their lifetime.
- If not continuously commissioned, losses of a university campus for example can accumulate to hundreds of millions of dollars over a period of 10 years.

# IIoT

Continuous commissioning - ongoing process to manage facility's operating problems, improve comfort, and optimize energy use.

Machine Learning (ML) refers to the capability to learn without explicitly being programmed, and get better with experience. In terms of the machine learning language the continuous commissioning analytics falls into the supervised learning techniques which require training.

- Classification (discrete set of predictions from inputs)
- Regression (continuous set of predictions from inputs)

Once trained, for any given set of inputs the model predicts the outputs.

Artificial neural networks - a family of machine learning models capturing the feature of learning.

# IIoT Work Study Example – Energy Performance Regression Model

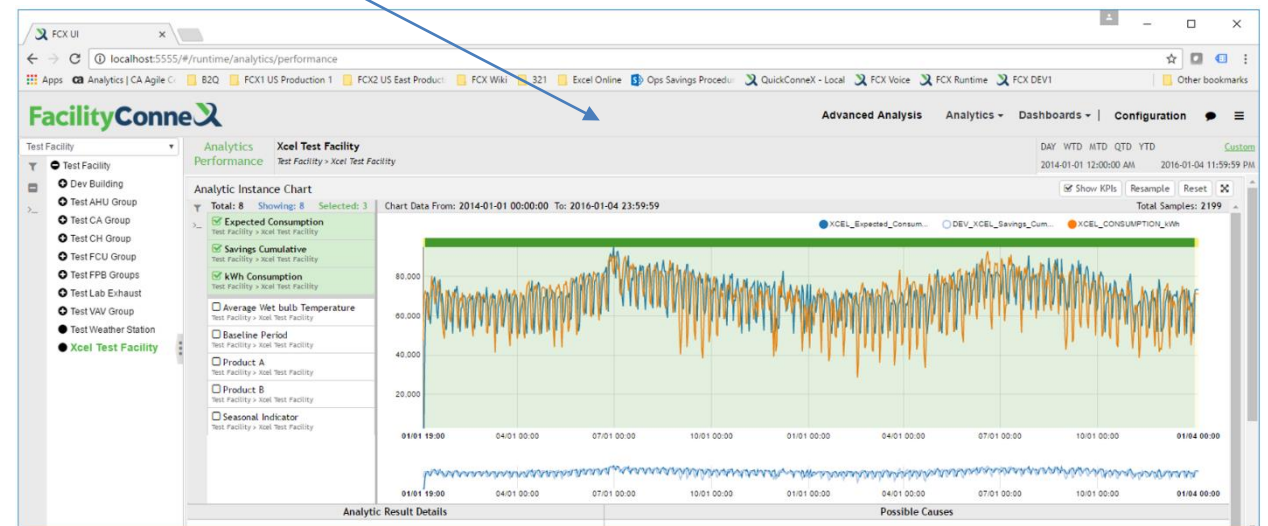
Illustrates the implementation of linear regression in creating an energy performance model (baseline) of a production plant facility.

The electric energy consumption strongly follows the temperature changes

	A	B	C	D	E	F	G	H	I	J
1	Date	Baseline Period	Avg WB Temperature (°F)	Product A	Product B	Product C	Total Production	Seasonal Indicator	Actual Consumption (kWh)	Expected Consumption (kWh)
2	1/1/2014	1000000	55	135249.76	273359.78	248526.17	657135.71	0	53757.39	65195.52
3	1/2/2014	1000000	55	0	0	16653.4	16653.4	0	47778.01	50894.19
4	1/3/2014	1000000	55	147185.53	346725.65	293444.11	787355.29	0	70153.83	68181.08
5	1/4/2014	1000000	55	249292.6	349968.46	340708.94	939970	0	72805.76	71868.32
6	1/5/2014	1000000	55	245493.57	343220.03	435672.31	1024385.91	0	70841.87	71499.16
7	1/6/2014	1000000	55	279235.74	0	61385.22	340620.96	0	63689.89	60667.44
8	1/7/2014	1000000	55	141725.3	0	115569.09	257294.39	0	57914.31	55854.57
9	1/8/2014	1000000	55	123323.54	291907.22	369535.12	784765.88	0	68507.71	65427.26
10	1/9/2014	1000000	55	253445.36	413516.11	273053.25	940014.72	0	70658.97	74237.84
11	1/10/2014	1000000	60.49	244107.89	422853.57	567444.21	1234405.67	0	80445.27	77760
12	1/11/2014	1000000	63.8	218493.89	447646.01	494010.15	1160150.05	0	74690.57	79860.05
13	1/12/2014	1000000	60.1	226429.94	361699.78	195434.83	783564.55	0	75364.6	74749.5
14	1/13/2014	1000000	61.71	211196.53	0	0	211196.53	0	65989.45	62591.82
15	1/14/2014	1000000	64.2	48819.94	0	66000.65	114820.59	0	69756.1	58507.92
16	1/15/2014	1000000	61.68	125672.13	282354.26	155879.36	563905.75	0	70383.87	69459.47

Expected Consumption (kWh) = A+ B\*Avg Temperature + C\*Total Production +D\*Seasonal Indicator

Constant	A=16066.497
Avg Temperature (°F)	B=647.173
Total Production	C=0.02
Seasonal Indicator	D=1946.042



# IIoT Work Study Example 1 – Forecasting with Neural Networks Model

Replicate the calculated linear regression (Expected Consumption) from previous slide using RNN.

Train the RNN on the input tags, for which we need to provide targets (or labels) – in this case the ones we already calculated.

We want to test if the RNN will give comparable predictions to the linear regression model.

- Selects the appropriate tags (columns) from the provided CSV data set.
- Set % of the data for training for testing.
- Scale & normalize data. NN like values [0 to 1]
- Run elaman RNN
- Plot results

	A	C	J	M	N
1	Date	Actual Consumption (kWh)	Expected Consumption (kWh)		
2	1/1/2014	53757.39	65195.52		
3	1/2/2014	47778.01	50894.19		
4	1/3/2014	70153.83	68181.08		
5	1/4/2014	72805.76	71868.32		
6	1/5/2014	70841.87	71499.16		
7	1/6/2014	63689.89	60667.44		
8	1/7/2014	57914.31	55854.57		
9	1/8/2014	68507.71	65427.26		
10	1/9/2014	70658.97	74237.84		
11	#####	80445.27	77760		
12	#####	74690.57	79860.05		
13	#####	75364.6	74749.5		
14	#####	65989.45	62591.82		
15	#####	69756.1	58507.92		
16	#####	70383.87	69459.47		
17	#####	78083.06	78756.56		
18	#####	73622.7	66903.13		
19	#####	73158.43	71546.1		
20	#####	70104.59	65217.61		
21	#####	58579.68	56276.59		

# Lab: IoT Implementation of Multivariate Regression

## Task:

Provided production plant data and multivariate regression model that represents facility's energy baselining and provides optimal energy consumption based on level of production and seasonality.

Create a Neural Network model trained on the optimal energy consumption and compare the regression model forecasting versus the neural network forecasting.

# Machine Learning Terminology

**Machine Learning** - to learn without explicitly being programmed, and get better with experience.

**Supervised Learning techniques** – Require training input and output data sets.

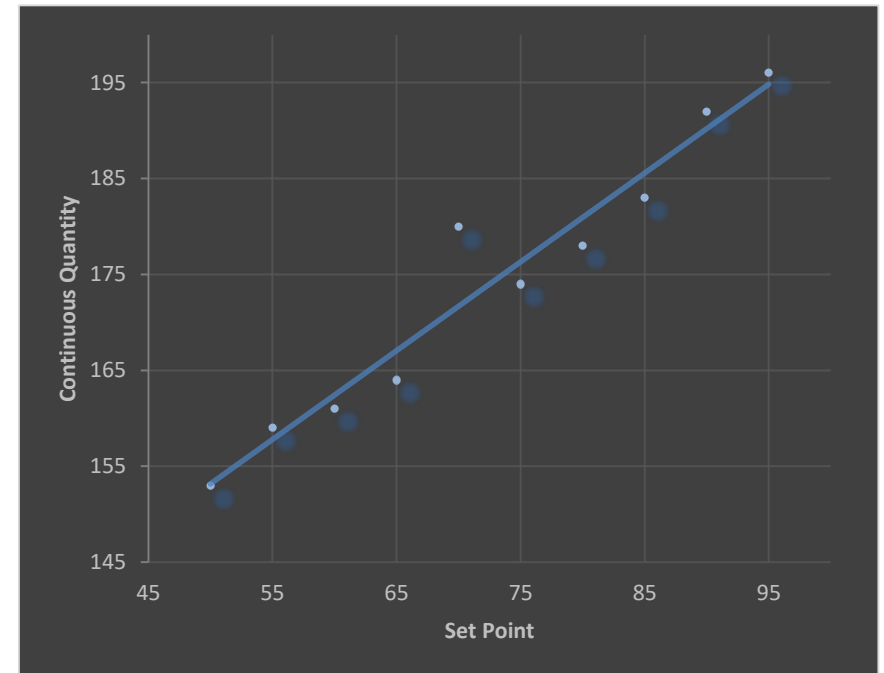
- Once trained:
  - for any given set of inputs the model predicts the outputs.
- It has two different techniques:
  1. **Classification** (discrete set of predictions from inputs)
    - IIoT fault detection and diagnostics (FDD) analytics falls into this type. It analyzes the input data (tags from sensory devices) and based on their values it outputs a label associated with each FDD scenario (discrete set) that occurred.
  2. **Regression** (continuous set of predictions from inputs) )
    - Any predictive (IIoT) analytics such as forecasting falls into this supervised learning technique.



# Regression

- The regression is used for predicting continuous-response values  $y$  (for example energy usage) based on input  $x$  (for example the sensor readings).
- The regression algorithm, will detect the tendency in the training data and make a forecast.
- Types of regression
  - Linear regression (straight line)
  - Non-linear regression (curve)
  - Multivariate regression

After establishing the correct regression model (the line) from the train data (the points) we can predict the value of  $y$  for any  $x$  (even the ones not given in the train data).



# Energy Performance IIoT Model

## **Goal:**

- Analyze past energy usage pattern and provide customers and utility representatives with insight
  - to the facility's energy performance
  - potential energy savings based on variable level of production and seasonal factors.

## **Task:**

- Develop a multi-variable regression "Expected Energy" model based on previous "Actual Energy" consumption (baseline) data.
- Based on this model calculate (potential) Cumulative Energy Savings.

# Model Description

**Facility Type:** Refrigerated Warehouse

**Energy Model Coefficients:**

Constant	A	15592.325	Dates	Start	End
Max (WB,55)	B	641.852	Baseline:	1/1/2014	8/23/2014
Total Production	C	0.035	CUSUM Start:	1/1/2014	
Seasonal Indicator	D	1714.754			

**Multi-variable (4) Regression Equation:**  $A + B * \text{MaxTemp}(\text{DB}, 55) + C * (\text{Product A} + \text{Product B}) + D * \text{Seasonal\_Indicator}$

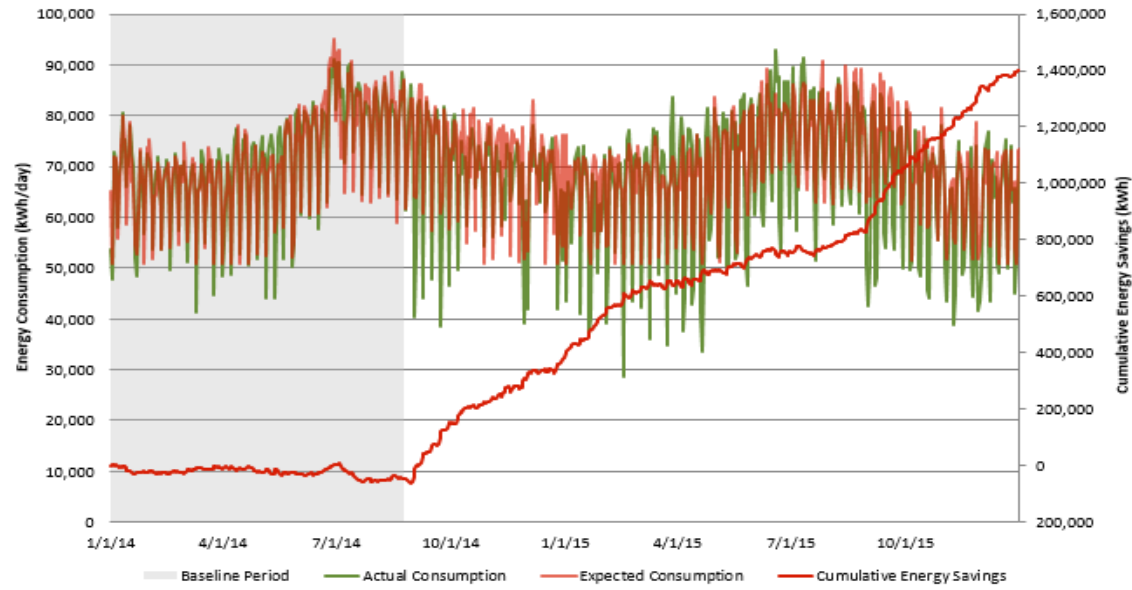
## Task:

- The solution should allow for an individual model coefficient to be applied to multiple production variables as necessary for the unique characteristics of each plant site.

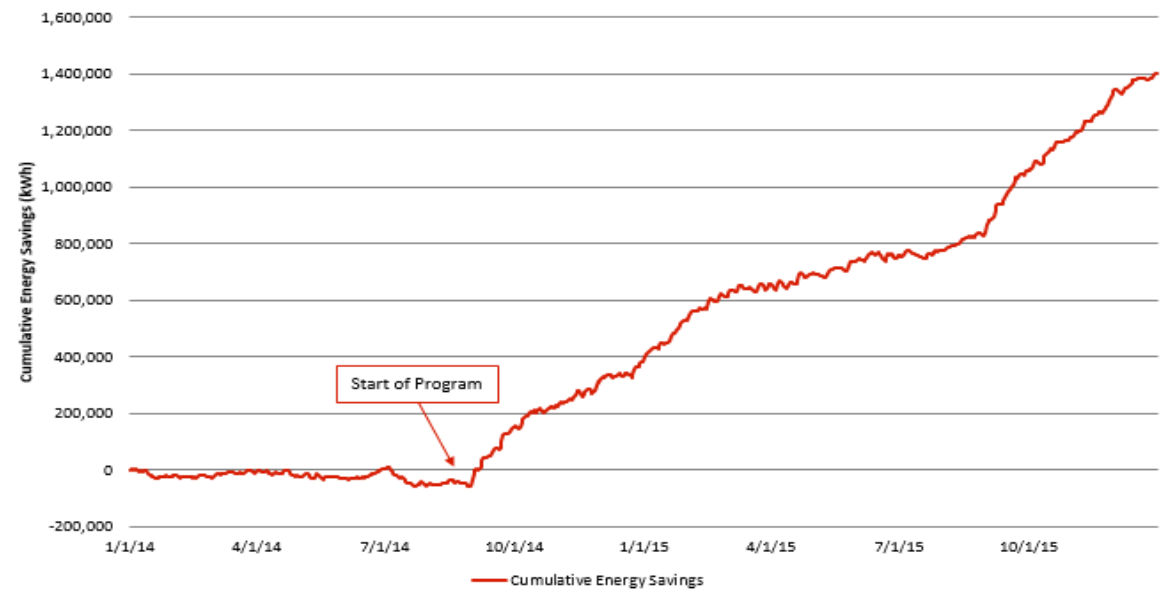
**Dataset of daily values that was used to develop the multivariate regression model.**

Raw Data							Calculated Data				
Date	Baseline Period	Actual Consumption	Avg WB Temperature	Product A	Product B	Seasonal Indicator	Max (WB,55)	Total Production	Expected Consumption	Actual Minus Expected	Cumulative Energy Savings
	(°F)	(kWh)	(°F)	(qty)	(qty)	(°F)	(°F)	(qty)	(kWh)	(kWh)	(kWh)
1/1/14	1,000,000	53,757	53.14	135,250	273,360	0.00	55.00	408,610	65,196	11,438	0
1/2/14	1,000,000	47,778	51.45	0	0	0.00	55.00	0	50,894	3,116	3,116
1/3/14	1,000,000	70,154	41.39	147,186	346,726	0.00	55.00	493,911	68,181	-1,973	1,143
1/4/14	1,000,000	72,806	35.97	249,293	349,968	0.00	55.00	599,261	71,868	-937	206
1/5/14	1,000,000	70,842	36.07	245,494	343,220	0.00	55.00	588,714	71,499	657	863
1/6/14	1,000,000	63,690	43.50	279,236	0	0.00	55.00	279,236	60,667	-3,022	-2,159
1/7/14	1,000,000	57,914	38.58	141,725	0	0.00	55.00	141,725	55,855	-2,060	-4,219
1/8/14	1,000,000	68,508	45.24	123,324	291,907	0.00	55.00	415,231	65,427	-3,080	-7,299

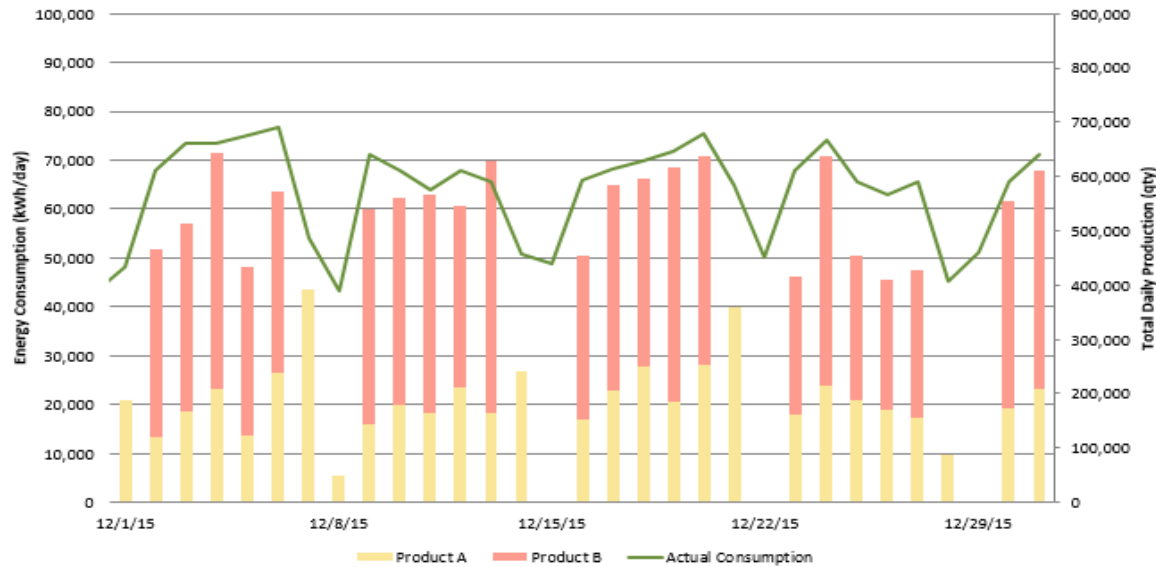
### Expected vs. Actual (w/ Cumulative Energy Savings)



### Cumulative Energy Savings



### Energy Consumption vs. Production (Last Month)



### Energy Consumption vs. Temperature (Last Month)



# IIoT Work Study Example 2 – Forecasting with Neural Networks Model

Optimize energy consumption of an office building.

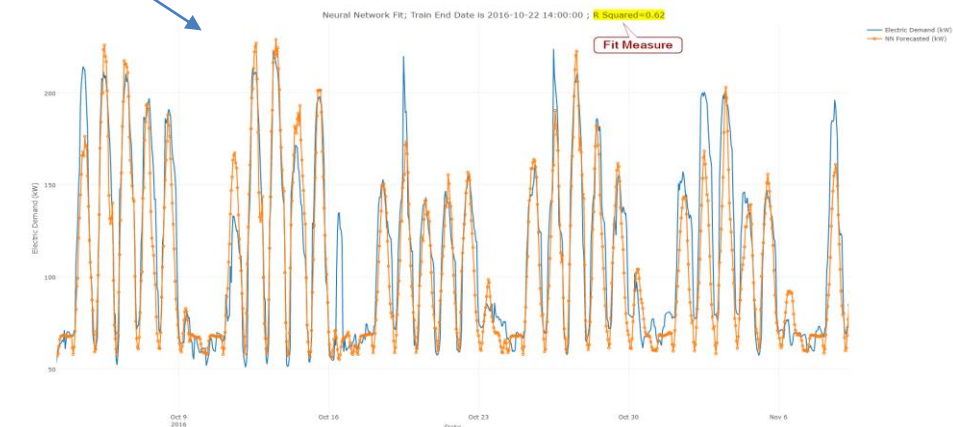
“Electric Demand” is the target, 70/30 train/test ratio.

Same RNN steps as before, but now there are 5 inputs.

Note how the energy consumption varies with “Hour of the day” and “Day of Week” columns.

How do we measure the RNN forecasted consumption - calculate the r-squared measure between the actual and predicted electric demand.

	A	B	C	D	E	F	G
1	Date/Time	Hour of Day	Day of Week	Electric Demand (kW)	Outdoor Drybulb	Outdoor Dewpoint	Relative Humidity
2	3/1/2012 1:00	1	4	90.816	33.98	32	0.92
3	3/1/2012 2:00	2	4	84	33.98	32	0.92
4	3/1/2012 3:00	3	4	82.752	35.06	33.08	0.92
5	3/1/2012 4:00	4	4	82.92	35.6	33.8	0.93
6	3/1/2012 5:00	5	4	82.776	35.06	33.08	0.92
7	3/1/2012 6:00	6	4	93.24	35.06	33.08	0.92
8	3/1/2012 7:00	7	4	139.2	35.06	33.98	0.96
9	3/1/2012 8:00	8	4	161.952	35.06	33.98	0.96
10	3/1/2012 9:00	9	4	192.36	35.06	33.08	0.92
11	3/1/2012 10:00	10	4	195.216	33.98	32	0.92
12	3/1/2012 11:00	11	4	199.536	35.06	33.08	0.92
13	3/1/2012 12:00	12	4	200.664	35.96	33.98	0.92
14	3/1/2012 13:00	13	4	204.24	37.04	35.06	0.92
15	3/1/2012 14:00	14	4	207.384	37.04	35.06	0.92
16	3/1/2012 15:00	15	4	198	37.04	33.98	0.89
17	3/1/2012 16:00	16	4	195.192	35.96	33.98	0.92
18	3/1/2012 17:00	17	4	185.496	35.96	33.08	0.89
19	3/1/2012 18:00	18	4	170.304	35.06	32	0.88
20	3/1/2012 19:00	19	4	155.808	28.94	26.06	0.89
21	3/1/2012 20:00	20	4	146.4	28.94	26.06	0.89
22	3/1/2012 21:00	21	4	143.712	28.04	24.98	0.88
23	3/1/2012 22:00	22	4	119.16	28.04	24.08	0.85
24	3/1/2012 23:00	23	4	82.176	28.4	24.8	0.86
25	3/2/2012 0:00	0	5	82.104	26.96	23	0.85



```
# ### ===== Claculate Residuals =====  
Y <- Actual.Electric.Demand; Yp <- Predicted.Electric.Demand  
R2 <- 1 - sum( (Yp-Y )^2 ) / sum( (Y-mean(Y) )^2 ) # Coeff of determination (R-squared)
```

# IIoT Work Study Example 3 – Modified Data

Observe from data (this is on office building in RI) the drastic consumption difference during working days (working hours) vs weekend (night hours).

In this example the “Hour of the day” and “Day of Week” columns are modified to produce better forecasted consumption (r-squared).

This is done by extracting the information from the first column (time stamp).


## 1) Extract

- Extract hours from time stamp
- Extract days from time stamp

## 2) Modify


- working days are labeled 1, Saturday 2, Sunday 3. I made the hours go up and down to 12.

This improved r-squared. You may use the same or come up with another way



```
# Extract Hours field from Time.Stamp
Hours <- as.numeric(format(Time.Stamp, '%H'))
# Extract Days field from Time.Stamp
Day.Date <- as.numeric(format(Time.Stamp, '%d'))
Day.Number <- as.numeric(format(Time.Stamp, '%w'));
Day.Number[Day.Number==0]=7 # Mon=1,...,Sun=7
Day.Name <- weekdays(Time.Stamp)

# Modify Hours & Days
temp <- 12 - Hours; temp[temp >= 0] = 0 #
Hours.Modified <- Hours + 2*temp
Day.Number.Modified <- Day.Number
Day.Number.Modified[Day.Number<6]=1;
Day.Number.Modified[Day.Number==6]=2;
Day.Number.Modified[Day.Number>6]=3;
```



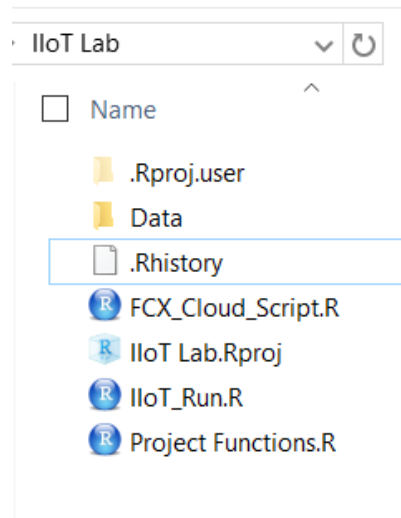
```
Browse[3]> Hours.Modified[1:25]
[1] 0 1 2 3 4 5 6 7 8 9 10 11 12 11 10 9 8 7 6 5 4 3 2 1 0
```



# IoT Lab - Running it Locally

Modify the provided IoT R project to create predictive IIoT analytics and deploy it on the FCX cloud platform. The dataset on the FCX cloud contains the same data for Temperature, Dewpoint and Humidity as the attached CSV file, but it does not contain the “Hour of Day” and “Day of Week” data.

- Modify the given code for “Project Functions.R” to use the time stamp from the “DATE” column to extract the information about the “Hour of Day” and “Day of Week”.
- Implement the provided labeling from the “IIoT Work Study Example 3 – Modified Data” slides or come up with your own. Training the neural network without the “Hour of Day” and “Day of Week” will result in very bad fit ( $R\text{-squared} \approx 0$ )
- Run your code locally to verify it and optimize it to get at least  $R\text{-squared} > 0.5$ .
- Once satisfied with your R-squared result, deploy your analytics to the FCX cloud.



The screenshot shows an Excel spreadsheet titled "OfficeBuildingData.csv". The data is organized in a table with columns A through F. The first row (row 1) contains headers: "DATE", "Outdoor D", "Outdoor D", "Relative H", "Electric De", and "Optimal (kW)". The subsequent rows (rows 2 through 7) contain numerical data for each column. The "DATE" column shows timestamps from 1/1/2016 0:00 to 1/1/2016 5:00. The "Optimal (kW)" column shows values ranging from 72.4 to 106.82.

	A	B	C	D	E	F
1	DATE	Outdoor D	Outdoor D	Relative H	Electric De	Optimal (kW)
2	1/1/2016 0:00	8.96	5	0.84	96.124	106.82
3	1/1/2016 1:00	5	1.04	0.83	87.276	97.13
4	1/1/2016 2:00	3.92	-0.04	0.83	86.304	86.85
5	1/1/2016 3:00	3.92	-0.94	0.8	84.428	77.64
6	1/1/2016 4:00	3.02	-0.94	0.83	86.572	72.4
7	1/1/2016 5:00	3.02	-2.02	0.79	88.96	74.22

# IoT Lab - Running it on the FCX Cloud

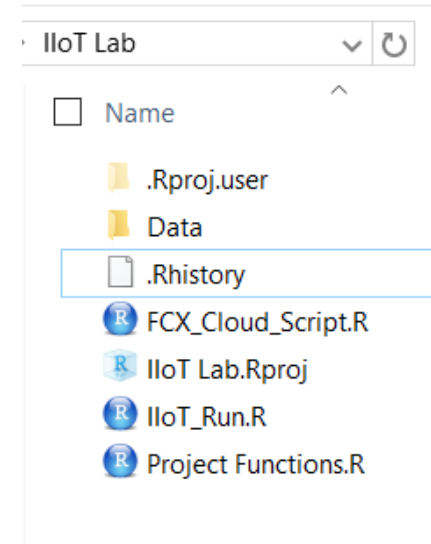
- Run your code locally to verify it and optimize it to get at least  $R\text{-squared} > 0.5$ .
- Once satisfied with your  $R\text{-squared}$  result, deploy your analytics to the FCX cloud.
- A template shell R script “FCX\_Cloud\_Script.R” is included with the attached project.
- Login to the FCX UI will be assigned to you.
- **Re-name the script “FCX\_Cloud\_Script.R” as your provided login name**, insert your code modifications and upload your IIoT analytics to the FCX UI as described in class.
- Download the results from FCX UI as CSV file and compare them with your local results.

# Cloud Implementation

# Step 1.

You have all the code and the components given. You just need to put them together, run them on the FCX cloud platform, download the results and submit them.

- Download “IIoT Lab.zip” from Blackboard and open (double click) the R project file.
- You run R script “IIoT\_Run.R” that contains all the code you need.
  - The call to the function “Forecast.Electric.Demand” in script “Project Functions.R”
  - Calculation for R-squared measure.
  - Plot the results (**Note: To Plot type "p" in console**)
- Run or debug “IIoT\_Run.R” to see how it works. (download needed packages if needed)
- Note that the CSV data does not contain the day and the hour columns. In the function “Forecast.Electric.Demand()” these fields are set to 1, thus the fit (r-square) is not good.
- Task: Extracts this information from the time stamp.



```
17 # Extract Hours field from Time.Stamp
18 Hours <- rep(1,length(TrainRange)) # Replace this Line
19 # Insert your code here
20
21 # Extract Days field from Time.Stamp
22 Day.Number <- rep(1,length(TrainRange)) # Replace this Line
23 # Insert your code here
24
25 # Modify Hours & Days
26 Hours.Modified <- Hours # Replace this Line
27 Day.Number.Modified <- Day.Number # Replace this Line
28 # Insert your code here
29 print("Extracting Hour_of_Day & Day_of_Week fields from the DATE field Time Stamp ")
30
```

## Step 2.

- In R studio you can see that the file “Forecast\_kWh\_Demand.R” contains all the functions that Pre-Process Data & Call the Neural Network.
- You need to add the code described in slide IIoT Work Study Example 3 – Modified Data that extracts the information from the time stamp. Once it works you can come up with your own way if you like.
- Once satisfied with your modifications, you have a good r-squared, move to Step 3

```
2
3  ### Pre-Process Data & Call Neural Network
4
5  Forecast.Electric.Demand <- function (Raw_Data)
6  {
7    print("2. Inputs sent to function: Forecast.Electric.Demand()")
8    # Extract Time Stamps from Data
9    Num.Data.Points <- dim(Raw_Data)[1]
10   Time.Stamp <- strptime(Raw_Data$DATE,"%m/%d/%Y %H:%M")
11
12   # Select Training Range
13   StartTime <- 1 # which(Time.Stamp=="2014-03-01 01:00:00 EST")
14   TrainRange <- StartTime:Num.Data.Points
15   print(paste0("Training data start date: ",Time.Stamp[StartTime]))
16
17   # Extract Hours field from Time.Stamp
18   Hours <- rep(1,length(TrainRange)) # Replace this Line
19   # Insert your code here
20
21   # Extract Days field from Time.Stamp
22   Day.Number <- rep(1,length(TrainRange)) # Replace this Line
23   # Insert your code here
24
25   # Modify Hours & Days
26   Hours.Modified <- Hours # Replace this Line
27   Day.Number.Modified <- Day.Number # Replace this Line
28   # Insert your code here
29   print("Extracting Hour_of_Day & Day_of_Week fields from the DATE field Time Stamp ")
30
31   # Choose Data to Process
32   Dependent.Ix <- c(2:4) # Select dependent columns
33   Dependent.Data <- cbind(Hours.Modified, Day.Number.Modified, Raw_Data[TrainRange,Dependent.Ix]); # X ()
34   Independent.Ix <- c(5) # Select Independent columns
35   Independent.Data <- Raw_Data[TrainRange,Independent.Ix]; # Y (Actual Electric Demand )
```

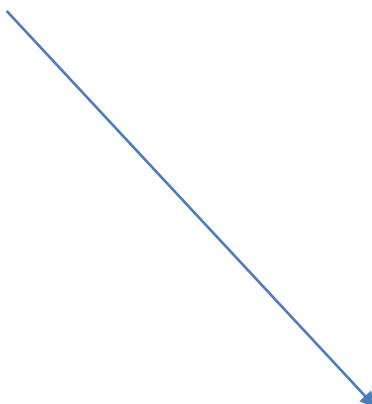
# Step 3.

- Attached code on Blackboard contains “FCX\_Cloud\_Script.R” Script.
- **Before you forget, rename this script with the username I emailed you.**
- This script is equivalent to “Project Functions.R” and produces the same result as the code you started with . The only difference is that this script is formatted so it can run on the FCX cloud. You can follow the instructions how to run it. It will produce the same bad fit chart that you can see and analyze in the FCX UI.
- To complete this assignment you need to
  - Carefully insert the modification you did locally on your computer for “Forecast\_kWh\_Demand.R” script into your renamed “FCX\_Cloud\_Script.R” script.
  - Log in to FCX cloud with the information I provided you in my email.
  - Select your modified script (this just copies the path to your script in the FCX UI)
  - Don’t forget to click on Upload button (this actually moves your script to the FCX cloud).
  - Click Run and once finished (30-40 sec), follow instructions how to download the results as CSV.
  - Finally submit your renamed “FCX\_Cloud\_Script.R” script and the downloaded results CSV.



# Your renamed “FCX\_Cloud\_Script.R” script

Carefully insert the modification you did locally on your computer for “Forecast\_kWh\_Demand.R” script into your renamed “FCX\_Cloud\_Script.R” script.



```
12 ix <- grep("Outputs.Predicted_Electric_Demand",inputs$parameters$Name); InputData <- cbind(InputData, input
13 ix <- grep("Outputs.Predicted_Electric_Demand",inputs$parameters$Name); InputData <- cbind(InputData, input
14 colnames(InputData) <- c("DATE","Relative_Humidity","Outdoor_Dewpoint","Outdoor_Temperature","Electric_Dem
15 return (InputData) # Returned object
16 }
17
18 Forecast.Electric.Demand <- function (Raw_Data)
19 {
20   library("RSNNS")
21   print("2. Inputs sent to function: Forecast.Electric.Demand()")
22   # Convert Time Stamps
23   Num.Data.Points <- dim(Raw_Data)[1]
24   Time.Stamp <- strptime(Raw_Data$DATE,"%Y-%m-%dT%H:%M:%S")
25
26   # Select Training Range
27   StartTime <- 1 # which(Time.Stamp=="2014-03-01 01:00:00 EST")
28   TrainRange <- StartTime:Num.Data.Points
29   print(paste0("Training data start date: ",Time.Stamp[StartTime]))
30
31   # Extract Hours field from Time.Stamp
32   Hours <- rep(1,length(TrainRange)) # Replace this Line
33   # Insert your code here
34
35   # Extract Days field from Time.Stamp
36   Day.Number <- rep(1,length(TrainRange)) # Replace this Line
37   # Insert your code here
38
39   # Modify Hours & Days
40   Hours.Modified <- Hours # Replace this Line
41   Day.Number.Modified <- Day.Number # Replace this Line
42   # Insert your code here
43   print("Extracting Hour_of_Day & Day_of_Week fields from the DATE field Time Stamp ")
44
45   # Choose Data to Process
46   Dependent.Ix <- c(2:4) # Select dependent columns
47   Dependent.Data <- cbind(Hours.Modified, Day.Number.Modified, Raw_Data[TrainRange,Dependent.Ix]); # X ()
48   Independent.Ix <- c(5) # Select Independent columns
```



# FCX Cloud Platform

