Swissgrid Data Analysis

Prepared by

K.S.N.R Ranasinghe
139177V

Faculty of Information Technology
University of Moratuwa, Sri Lanka

May 2017

Swissgrid Data Analysis

Prepared by

K.S.N.R Ranasinghe

139177V

Thesis submitted to the Department of Information Technology, University of Moratuwa, Sri Lanka for the partial fulfillment of the requirements of the Degree Master of Science in Information Technology.

May 2017

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education.

Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

Name of Student : K.S.N.R Ranasinghe

Student Number : 139177V

Student Signature: …... Date:

Name of Supervisor(s) : Dr. Lochandaka Ranathunga

Signature of Supervisor(s):....................................... Date:

Dedication

This Dissertation is dedicated to my loving husband, my loving mother and my darling son for always encouraging me and being by my side.

Acknowledgement

This project would not have been a success if not for the many people in my life. Their encouragement and guidance has helped me immensely to complete the project. I take this opportunity to express my gratitude to the people who have been instrumental in the success of the completion of this research.

My heartfelt gratitude goes to my supervisor Dr.Lochandaka Ranatunga not only for his guidance and mentoring to complete the project but also for the encouragement given all throughout the project.

I would also like to thank Prof.Asoka Karunananda for the valuable knowledge he gave us on writing a Thesis, which helped me in great scale to compose this thesis.

It is with heart felt gratitude that I would like to thank all the other Senior lecturers, Lecturers, Instructors, and staff members who helped me in many numerous ways to make this Project a success.

I would specially like to thank Ms. Dinesha Malwalage, Mrs.Chaya Atapattu and Mr.Kasun Atapattu for the encouragement they have given me throughout.

I would also like to thank all my other my M.Sc Information Technology Batch 07 batch mates for all the help given.

Last but not least, the research would have not been possible if not for my family. I would like to thank them for being understanding and for the encouragement they have given me throughout and for being the backbone of the success of this research.

Abstract

The swissgrid is the national power grid in Switzerland which is the largest in that area. Not only does it supply power to Switzerland it also exports and imports power from its neighboring countries. The power grid must be kept at a balance of 50Hz frequency. In order to help the operators maintain and take necessary action to maintain this frequency, monitoring the grid is vital. Currently, studies do not clearly show of any prediction models that the swissgrid uses. Hence this study is focused on assisting the operators monitor the grid and help them predict the energy consumption for the swiss control block.

In order to assist the operators, and interested parties, the grid data has been analyzed in order to derive real-time and batch analytics using the WSO2 Data Analytics Server. The real-time analytics computed based on the Siddi engine and batch analytics based on the Apache Spark engine is able to be viewed on a central dashboard powered by WSO2 Data Analytics Server. Moreover the solution also provides the ability to configure to detect any anomalies in the power grid and alarm any interested parties via SMS or E-mail.

The research goes on to find a model to predict the total energy consumption of the swiss control block. A model was successfully built in order to predict the energy consumption using the Liner regression with rolling window analysis. Using a 30 day window it was found that the model's optimal training data set is of 1.5 years worth data. The research expanded to find any co-relation between multiple factors that would affect the energy consumption. Using the season, and whether or not the date is a holiday, a model was built based on the multiple regression algorithm. This model was found to be trained better with the least Absolute Mean average for a data set of 2 years of data. The models built were tested against the predicting data set which proved to have predict the energy consumption easily.

The aim of the project is achieved by providing centrally viewable statistics and a tested model to predict the total energy consumption for the swiss control block.

Contents

Chapter 1 - Introduction **1**

1.1 Background & Motivation 1

1.2 . Aim and Objectives 4

1.2. 1 Aim 4

1.2.2 Objectives 4

1.3 Tools and technology 5

1.4 Structure of the thesis 5

Chapter 2 - Literature Review **7**

2.1 Introduction 7

2.2 Other's work 7

2.3 Summary 13

Chapter 3 - Using technology to analyze the Swissgrid Data **14**

3.1 Introduction 14

3.2 WSO2 DAS for Real time and Batch Analytics 14

3.3 Java for preprocessing data and publishing events 16

3.4 SciKit Learn for building the ML model to predict the Consumption 16

3.5 Using the rolling window analysis of time series with Linear regression 16

3.6 Summary 17

Chapter 4 - Turning Swissgrid Data into Information **18**

4.1 Introduction 18

4.2 System Design & Development 18

4.3 Language for Implementation 18

4.4 Apache Cassandra No SQL DB 18

4.5 Writing a Java client to publish events to WSO2 DAS 19
4.5 Real-time analysis using WSO2 Data Analytics Server 19
4.5.1 Reading the Data 20
4.5.2 Persisting the Data 20
4.5.3 Computing the statistics 20
4.5.4 Creating Alerts 21
4.6 Batch Analysis using WSO2 Data Analytics Server 22
4.6.1 Using Apache Spark for Batch Analytics 22
4.7 Displaying the statistics in the WSO2 DAS dashboard 23
4.8 Forecasting the end user consumption 23
4.8.1 Preprocessing the data 23
4.8.2 Using SciKit Learn to build the model 24
4.9 Summary 24

Chapter 5 - System Design 26

5.1 Introduction 26
5.2 High level design 26
5.3 WSO2 DAS high level architecture 27
5.4 The Event flow 28
5.5 The Machine Learning model design 29
5.5.1 Absolute Mean Error (MAE) 30
5.6 Summary 31

Chapter 6 - Implementation 32

6.1 Introduction 32
6.2 Real-time Analytics 32
6.2.1 Publishing data to the WSO2 DAS 32
6.2.2 Configuring the event stream 32
6.2.3 Writing the Execution Plan 33
6.2.4 Writing the output stream 34
6.3 Batch Analytics 35

6.3.1 Persisting the events to a data store 35
6.3.2 Using Spark SQL to Query the database 35
6.4 Publishing and viewing data in the WSO2 dashboard 36
6.5 Implementing the ML model 36
6.5.1 Reading the data for analysis 37
6.5.2 Using Linear Regression 37
6.5.3 Calculating the Absoluter Error Mean 38
6.5.4 Using rolling window analysis 38
6.6 Summary 38

Chapter 7 - Evaluation 39

7.1 Introduction 39
7.2 Evaluating the Preprocessed data 39
7.3 Evaluating the Machine Learning Model 39
7.3.1 Rolling Window test results. 39
7.3.2 Multiple Feature Test 45
7.4 Functional testing - Real Time and Batch analytics 49

Chapter 8 - Conclusion 53

8.1 Introduction 53
8.2 Analysis of the size of the data set Size. 53
8.2.1 Rolling window analysis with Linear Regression for one factor 53
8.2.2 Multiple Linear Regression analysis 54
8.3 Retrospect 54
8.4 Limitations 55
8.5 Future Work 55
8.6 Summary 56

References 57

Appendix B - Screen shots of the WSO2 DAS configuration 65

Persisting the Event 65
Creating the Spark SQL script 66
Appendix C - Data preprocesser 67
Appendix D – Source for building the model to predict energy consumption – with one factor 71
Appendix E – Source code for building the prediction model with multiple regression 73

List of Figures

Figure 1.1 - Swiss Electricity Market in 2010 2
Figure 2.1 - Wide Area Monitoring High Level Architecture 10
Figure 2.2 - Monthly End User Consumption 11
Figure 2.3 - Monthly Energy Consumption 11
Figure 3.1 - Using WSO2 DAS for Real time and Batch Analytics 14
Figure 3.2 - Linear Regression 17
Figure 4.1 - Real time event flow 20
Figure 4.2 -Spark Components 22
Figure 5.1 - High level design of the proposed system 26
Figure 5.2 - WSO2 DAS high level architecture 27
Figure 5.3 - Event Flow 28
Figure 5.4 - Rolling Window Analysis 30
Figure 7.1 - 1 year data training set results 40
Figure 7.2 - 1.5 years training set results 41
Figure 7.3 - 2 years data training set results 42
Figure 7.4 - 2.5 years data training set results 43
Figure 7.5 - 3 years training data set results 44
Figure 7.6 - Training data set results of 1 year 46
Figure 7.7 - 1.5 years training data set results 47
Figure 7.8 - 2 years training data set results 47
Figure 7.9 - 2.5 years training data set results 48
Figure 7.10 - 3 years training data set results 49
Figure 8.1 - Rolling windows data set analysis 53
Figure 8.2 - Multiple regression data set analysis 54

List of Tables

Table 1.1 - Swissgrid facts 2
Table 2.1 - Summary of previous studies 12
Table 7.1 - Functional test case1 50
Table 7.2 - Functional test case2 51
Table 7.3 - Functional test case3 52

Chapter 1

Introduction

The Swissgrid is the largest electricity transmission grid in Switzerland. It's operations are mission critical for the countries power stability as well as its neighboring countries. Swissgrid has faced losses in the past due to lack of a proper alerting system and proper forecasting system. With the aid of real time analytics Swissgrid will be able to make timely decisions more efficiently and effectively.. This study mainly has focused on building a machine learning model to predict the energy consumption and to create a dashboard to view real time and batch analytics on energy consumption and production. The data sets that are made public by the Swissgrid were refined and used for this purpose.

1.1 Background & Motivation

Being the national electricity transmission grid, it is not only responsible for the secure, reliable and cost-effective operation of the grid but also responsible for the coordination and usage in the cross-border exchange of electricity in Europe. It is critical that the operations remain smooth and continuous in the Swiss grid in order to avoid any power imbalances and in turn avoid any shortages for all countries that are a part of the Swissgrid.

The Swiss grid has two main tasks [1]:

- Transportation of Energy to the end users via the transmission system
- Trading of Electricity with bordering countries.

The transmission system as a whole is implemented in a ground distance of 6700 kilometers. The cross border exchange of electricity is done at over 40 points with the neighboring countries as illustrated in the below Figure 1.1. Monitoring all of these 40 points is essential for the Swissgrid. It is also critical that any catastrophic event such as a power imbalance in one of these points is brought to notice of the Swissgrid operators as soon as possible in order for prompt action to be taken to mitigate any damages or losses to Swissgrid.

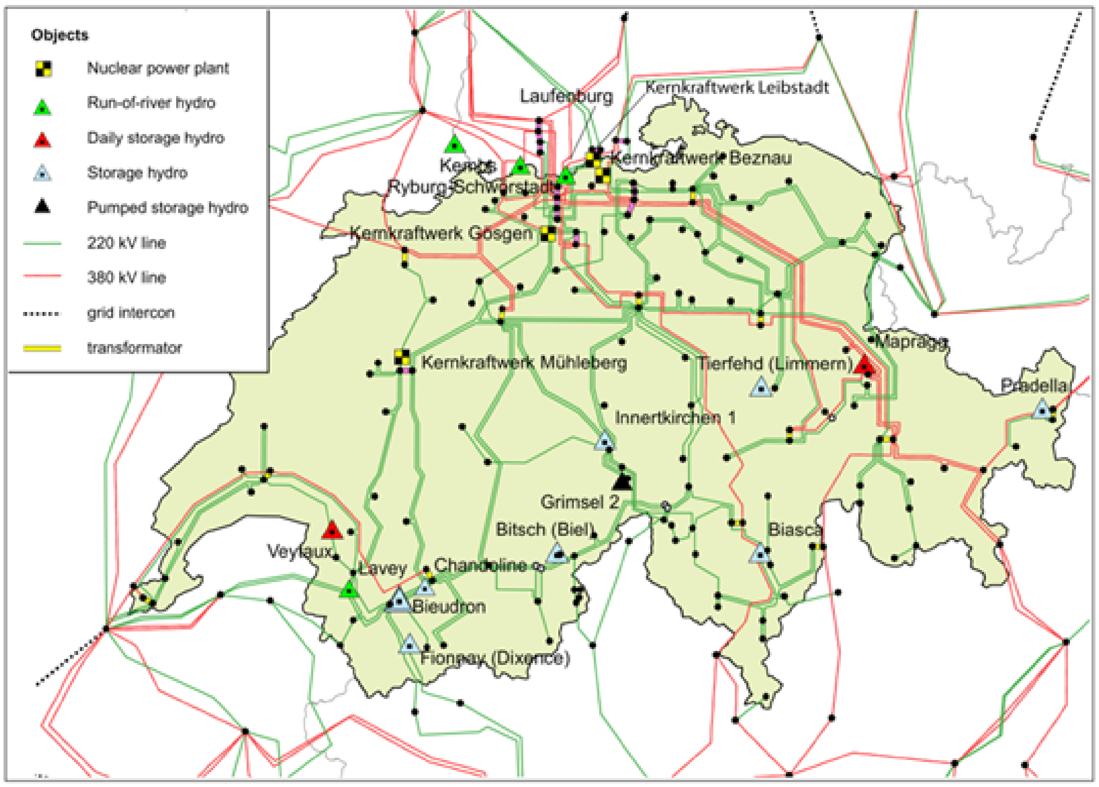


Figure 1.1 - Swiss Electricity Market in 2010 [2]

In a compact view, the swiss grid could be detailed out as below in Table 1.1. 2

Total length of transmission system	6,700 km
Electricity pylons	12'000
Substations	141
Total energy production	58'988 Gwh
Import	33'505 GWh
Export	29'091 GWh
Transit	23'887 GWh

Table 1.1: Swissgrid facts

For Swissgrid, the pressure is on the security of supply. This is because the Swissgrid can not tolerate any power outages and must be running at all times. Hence, it is considered to be one of the most stable transmission systems in the world.

The Swissgrid has made its grid data open and free for analysis. This data includes the grid-related data for the transmission grid and the Swiss control area. The data on the energy overview for Switzerland includes the aggregated quarter-hourly or hourly energy data figures.

There are events in history where countries had blackouts or the electricity suppliers had to pay customers to consume energy because the electricity usage/exchange was not properly monitored and action was not taken spontaneously. Italy once had a blackout of 2min and 30seconds due to a line tipping and the export/import was not handled in a timely manner, hence when the line was balanced with a 24 minuet delay it was too late to correct the electricity flow which caused the blackout to the entire country except Sardinia [3]. At one instance Germany had to pay their consumers to use the excess energy generated [4]. This too is a lack of awareness of generated electricity and could have been managed more efficiently if there was a monitoring system.

The motivation of this project is to be able help the Swissgrid to timely identify issues regarding the analyzed data and be able to provide solutions by taking correct decisions and avoid any shortages. Thereby being able to provide energy to its consumers in the most secure, reliable and cost-effective manner. Being able to visualize real time analytics will help in making real time decisions for operators much quicker and accurate. This is turn will make a positive impact in the service provided. In an age that Swissgrid is moving to smart grids and super grids making use of data analytics is a must in order to ensure a quality service supply and make the lives of the consumers and operators easier.

1.2 . Aim and Objectives

1.2. 1 Aim

The Aim of this project is to analyze the Swissgrid Data and produce useful statistics to assist in timely quality decisions, which would ease the lives of the operators and the consumers.

1.2.2 Objectives

1. Read the data and identify the data required for effective analysis

The Swissgrid data consists of various types of data. These data could be used in a number of combinations to bring out various statistics. Hence it was required to study the data and derive the statistics that has been worked on in this project, for example, the average energy consumption or production during the last hour.

By deriving these data, and coming up with statistics, the grid operators could take effective decisions timely and thereby maintain and improve the quality of the service at all times.

2.Deriving a model to predict and forecast events such as increase in consumption or production

Derive a model using various data mining techniques such as clustering, windowing etc and derive a working model to predict and forecast for grid operations. For example:-

- Consumption
- Identify patterns in energy consumption

By being able to forecast certain statistics such as average energy consumption, the grid operators could thereby be prepared for any situations that could arise, and/or take decisions based on it.

3. Create a centralized dashboard to view the real-time and batch
analytics

Create a dashboard to view statistics, so that the end users such as grid operators could have one place to view all the statistics and make timely and efficient decisions to improve and maintain the quality of the service provided.

4. Provide alerts on alarming events such as sudden increase of electricity consumption

For events such as consumption is greater than production, if the threshold value is challenged WSO2 DAS[12] can create real-time alerts (email, SMS, push notification, physical sensor alarms, etc.) for instant condition reporting. This will help in notifying all involved parties automatically, in parallel, and thereby assist prompt action to be taken on such events.

1.3 Tools and technology

This project is aimed in carrying out batch and real time analysis as well as training a data model using this data and visualizing them. The WSO2 Data Analytics Server[12] has been used to achieve the following.

- Analyzing data in Real-time
- Analyzing data in batch
- Visualizing the analytics in a centralized location.

The WSO2 Data analytics server is a data analytic platform in which batch and real time analytics could be performed. The WSO2 DAS[12] is designed to handle millions of event per second, hence it is capable handling big data volumes such as the Swissgrid data.

In order to build the model for predictive analysis, SciKit[17] built upon python has been used in this project. The predictive model has been created upon training an algorithm with per-structured data from the swiss grid data downloads.

1.4 Structure of the thesis

In this thesis, the Chapter 2 issues and challenges encountered in history and discusses previous work done by others. Chapter 3 discuses in detail the technology used and justifies reasons to use the technology in order to achieve the objectives stated in Chapter 1. Chapter 4 explains how the technology mentioned in Chapter 3 will be used in a methodical manner to achieve the objectives and aims of this research. Chapter 5 will describe the design used to implement the solution and the analysis. Chapter 6 explains in details the implementation of the system adhering to the design. Chapter 7 details on how the solution was evaluated is discussed in this chapter. Chapter 8 will gracefully conclude the results obtained by this research and discuss on the probable future work. The list of references and appendixes will be listed at the end completing the thesis

Chapter 2

Literature Review

2.1 Introduction

This chapter will discuss in detail the previous work done and validate on why the current problem requires a solution.

2.2 Other's work

The swissgrid being the national electricity grid, critically requires well defined analytics and predictions to make their operations smoother and to maintain power supply. Poor analytics and poor monitoring has resulted in many problematic situations in history. One most recent incidents was when in 2016, Germany had to pay their consumers to use the excess energy they had, which was generated from renewable energy sources [5]. They had no clear predictions of the consumption and generation. Charlton reports that the reason for this incident was the weather, and the sun and wind had contributed immensely to the generation of electricity so much that at the end of the day they had excess energy generated unexpectedly. More precisely the renewable energy, the solar plant and wind turbines had generated a 87 percent of the electricity consumption. Even though forecasting renewable energy is not an easy task, a lot of research around it has been and is continuously been carried out in order to forecast the renewable energy production based on weather. If Germany did have a prediction mechanism for the renewable energy, they could have easily avoided this situation and avoid the energy wastage.

Similarly in 2003 September, Italy had a historical blackout [3]. According to Lukszo et al., Italy had imported of 24% of the total consumption from the neighboring countries, which was 300MW above the agreed import level. This high usage of the swissgrid was unable to be gracefully controlled by the Swissgrid controller. This caused most of Italy to have a blackout for 2 minuets and 30 seconds. Lack of communication and awareness among grid operators had lead to this situation Lukszo et al. states in the book.

Another example Lukszo et al. brings out in his book to show how important monitoring of the grid is , the 2003 USA blackout [3]. The USA blackout was due to mainly human error and electricity loss caused by a tree flash-over. Though they had a proper control room, the control room was not functioning properly and for over an hour the operators did not know that the control room machines were not functioning properly. Not having real-time data to monitor and not having a proper alerting system affected highly on the blackout to occur, leaving some parts of USA in darkness.

By analyzing and learning about the historical facts, it is evident that a proper monitoring and alerting system is a very vital aspect for a energy management institution to maintain and make sure the supply of energy is consistent.

The Swissgrid energy consumption, production and transfer data is available for public freely. This data is captured every 15 minuets and logged in. While the data has many interesting fields, in this study. The main point of concentration was on the end user consumption for the Swiss control block.

Energy consumption forecasting have been a topic in research for sometime. Although, many studies have been conducted on energy consumption/ energy production , and analytics of energy production data, No research could be found on the topic of modelling the energy consumption in the Swiss control block, depending on the season and the whether or not it is a public holiday.

In the research paper [6] written by A.Borion et al. they have derived a model to predict hourly energy loads for a US utility. In this model too they have made use of the same factors that's used in this research. According to the research paper [6] they have made use of weather data from 11 weather stations and consumption data in 20 zones for 4 years. As A.Borion et al. mentions about the multiple regression models they developed and was able to reliably predict the energy load. One such model is the Full model. He also explains that the weather data in correlation with other data, for example Month/Temp, Hour/Day played a major role in selecting the model. The Indicator model eq.(1) with which he came up earlier, it did not prove to be reliable because of the complexities of using temperature.

$$Y_t = \theta_0 + \theta_1 Trend + \theta_2 T_t + \theta_3 T_t^2 + \theta_4 T_t^3 + \theta_5 Month + \theta_6 Day + \theta_7 Hour \qquad (1)$$

Since they encountered issues with direct weather data, they resorted to dividing data into seasons and going ahead to build the model. Hence they derived the Full Model. This is where the data was divided into smaller sets and tuned the model into accuracy. According to A.Borion et al., forecasting would not be accurate if the temperature is not used. Hence for forecasting he suggest the usage of the temperature model in conjunction with the load model to create a forecasting model. Since the load model was proven to be accurate by using seasons, In this research it was decided to use seasons in place of the temperature as this [6] A.Borion et al. 's research paper suggests.

Big data analysis and machine learning techniques usage for predictions have been in research for some time. P.Dagnely et al. in his research paper talks about a autoregressive model.[7]. In this the energy consumption is for casted hourly for an environment such as a medium-scale office environment. In this paper P.Dagnely et al. says that an auto regressive model is the best way to predict energy consumption. The autoregressive baseline was computed by taking the time t-7 days when time was needed to know[7]. Though the research could not find an algorithm that would outperform the autoregressive baseline, the baseline here does not seem to be sensitive to the weather or seasonal information as it checks for a period of 7 days. In the study they have assumed that the energy consumption on one Monday is the same as the next Monday, and created the baseline upon this. However, in this study the main point of concentration was on the season and the holidays and hence the baseline would not work well. Hence it was required to build a model on regression which would take in season and if the day is a holiday or not as factors in order to predict a energy consumption for a future date precisely as possible.

Monitoring the grid is critical for Swissgrid in real-time and non real time as well in order to maintain smooth operations and avoid interruptions power to the end users. The Swissgrid is said to have a monitoring and modeling system to model and monitor the oscillations in the power system[8]. The Swissgrid monitors the grid using their wide area monitoring (WAM) system. The high level architecture of the WAM system is illustrated in Figure 2.1.

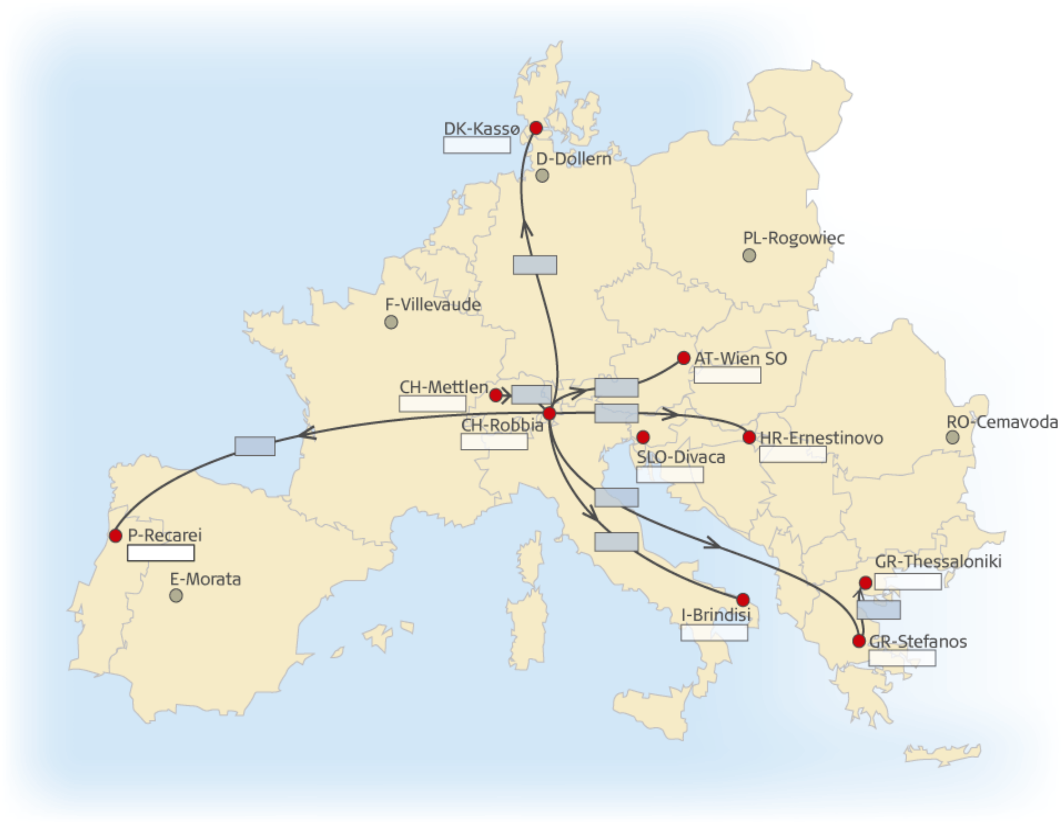
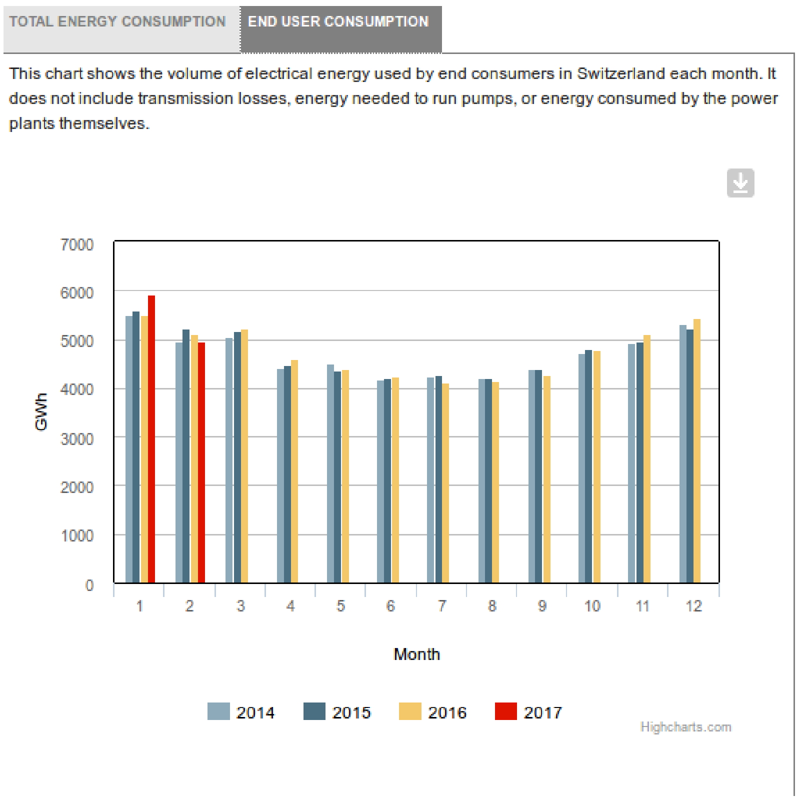


Figure 2.1: Wide Area Monitoring High Level Architecture [9]

Information about Swissgrid's monitoring system is not available freely. No prior studies or research conducted on forecasting the energy consumption, real-time and batch analytics on the Swissgrid was found. However it is reported that the Swissgrid does have a control room which does monitor the power grids [10].

Siemens also reported that they have built a self-learning software system that can stabilize power grids [6]. This program is able to forecast the renewable energy over a 72 hour period with over 90 percent accuracy. However, though this sophisticated system is able to predict the renewable energy generation, it does not mention about forecasting the energy consumption.

On the Swissgrid site , it only shows the consumption data for a month in a graph but does not show real-time and as a prediction. This is illustrated in Figure 2.2 & 2.3 [11]. It also does not mention about the type of monitoring done in the control room.



This chart shows the volume of electrical energy used by end consumers in Switzerland each month. It does not include transmission losses, energy needed to run pumps, or energy consumed by the power plants themselves.

Figure 2.2: Monthly End User Consumption [19]

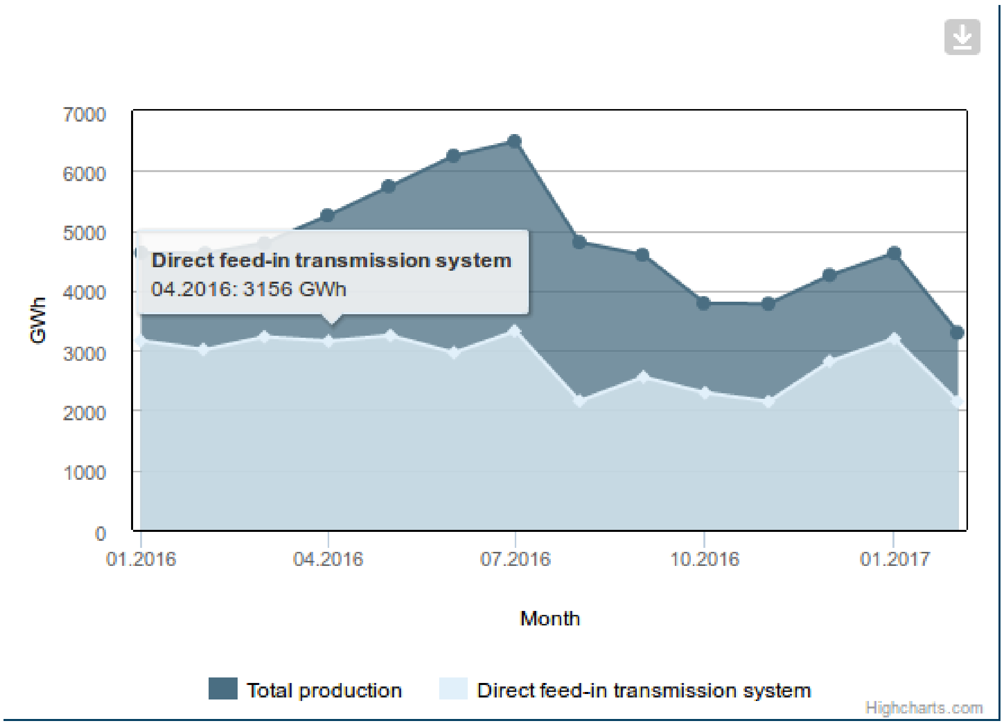


Figure 2.3: Monthly Energy Consumption [19]

However according to Quantum analytics [10], Swissgrid collects information about the weather, consumption, production , reservoir water levels in order to predict the future and avoid shortages. The Quantum team was able to predict a probable shortage and timely address the issue. However, this article does not mention about how the model was created nor if it even has a model to predict the consumption. From the Literature of the Swissgrid's current system to monitor the consumption, and production and predict the consumption, It was evident that a study to predict the consumption based on the weather and nature of the day (if its a holiday or not) on the Swissgrid data would be required and would help Swissgrid and the customers immensely. Also having real time analytics and batch analytics to view on a single screen would be beneficial to the operators and would help them take decisions without much delay.

Limitations and Issues with the previous studies in summary is listed in Table 2.1 as follows.

Limitation/Issues/Learning points	Reference
The study was done on US data and not on Swiss grid's data, The seasons are very different.	[6]
Using seasons instead of temperature yield in a better forecast	[6]
imitation of using baseline prediction with factors	[19]
Swissgrid consumption monitoring is only found to be for a month	[11]
Does not mention on any models to predict the power consumption	[10]
Using regression would be a better option for forecasting	[19]

Table 2.1: Summary of previous studies

2.3 Summary

History proves that monitoring a power grid is a must for any power supplier, and having real-time and batch analytics on the data is vital to provide smooth operations. Though Swissgrid has a control room and some studies on other forecasts have been done for example renewable energy production, the Swissgrid however does not publicly mention about the type of monitoring done in the control room and what models they use (if they use) to predict the future consumption/productions. Also looking at previous studies done on predicting the power consumption, Since no studies were found on Swiss grids' data, and since regression is found to be an effective way to model forecasts, this research will focus on building a model that would take the date, season, and nature of the day (If it is a holiday or not) and build a model on the Swiss grid data to predict the energy consumption.

Chapter 3

Using technology to analyze the Swissgrid Data

3.1 Introduction

This chapter will discuss about the technologies that was used to develop the solution. It will also discuss on why these technologies suit the best.

3.2 WSO2 DAS for Real time and Batch Analytics

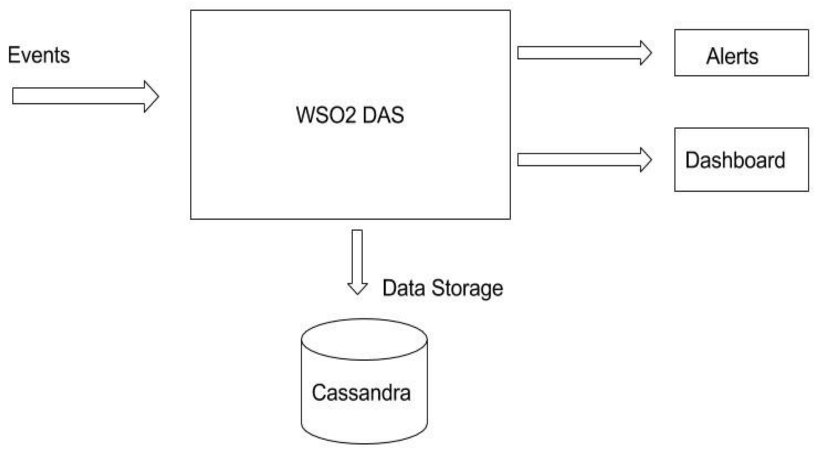


Figure 3.1 Using WSO2 DAS for Real time and Batch Analytics[12]

The WSO2 Data Analytics Server [12] is built upon the award winning Carbon platform. It is able to combine real-time, batch, interactive & predictive analysis into one platform. As illustrated in the Figure 3.1 the WSO2 DAS[12] allows the collection, persistence, analysis and communication of data to happen for both real-time and batch analytics seamlessly. More over, the WSO2 DAS[12] comes shipped with a dashboard where the analytics can be viewed. This product acts as the middleware and will help us focus on building the statistics while it will handle the load of handling the data.

For this research, among the objectives were to view the batch analytics and real-time analytics in a centralized dashboard. As mentioned above, by using this product it's possible to do so with minimal effort. Also as the literature proves [3], it is vital to have a updated and a clear dashboard so that the operators can have a clear idea and is able to take decisions in time.

Alerting the users in an alarming situation is also an important objective of this research. As per the literature, proper alarming has caused delays in responding towards the situation and have caused devastating results, for e.g. the historical Italy blackout[3]. This product has inbuilt support for alerts. Hence, It is easy to configure the alarming to notify people either via SMS or Email regarding a situation and notify them in order to take action in the least possible time.

Since WSO2 DAS[12] has a real-time analytic engine in built, it is very convenient to create the real time analytics. The real-time engine is powered by the Siddi engine. The Siddi query language is SQL like and hence querying for real-time analytics is very convenient. With a execution plan written it is able to easily analyze the data on the fly by keeping the data in a window. By using this it was easily possible to achieve the projects objective.

WSO2 DAS[12] has inbuilt support for Cassandra NoSQL database also, therefore, it can easily store the data which is read from the Swissgrid data sheet and store it in the Cassandra database. The Cassandra database is designed to handle large number of data. Swissgrid also produces a large number of data per day ~ 95 records which has ~65 fields per each record. This is because they collect the energy related information every 15 minuets. Hence by storing them in a Cassandra DB it will allow the batch analytics to be done very efficiently.

WSO2 DAS[12] also has a batch analytic engine in built. This is powered by Apache Spark. Apache Spark also has a SQL like query language for spark queries to be written with. It is called the Spark SQL. Spark is said to be a fast, powerful, open-source engine that is built around speed, ease of use and sophisticated analysis. Querying the data on the Cassandra DB, the analytics can be created. These Analytics can then be pushed to be displayed on the dashboard.

3.3 Java for preprocessing data and publishing events

Java is a powerful open-source language designed to run on any platform. Java is object oriented hence it is very user friendly. The solution required data pre-processing to build the model. Since the Swissgrid has very large data sets, doing it manually was impractical. Hence a Java program was made used to pre-process the data as it will be used for the model. The WSO2 DAS[12] is also written in Java. As it's required to publish events to the WSO2 DAS[12] for demo purposes, a publishing client was needed. This client was also written using Java because the integration with WSO2 DAS[12] would be seamless and because Java is Object Oriented.

3.4 SciKit Learn for building the ML model to predict the Consumption

One major objective of this research was to be able to build a model to predict the power consumption of end users in the swiss control block. For this purpose the SciKit Learn [17] tool is used. SciKit Learn is a machine learning tool that is built upon Python. Python is fast and is scalable. SciKit covers most of the machine learning tasks and has regression tools in built. Sci Kit also has very good documentation that helps using the product.

3.5 Using the rolling window analysis of time series with Linear regression

The Rolling Analysis of a time series [16] is beneficial to asses the Machine Learning model's stability over time. It assumes that the parameters in the model is constant over time. However if this is an incorrect assessment that too can be identified using this model. Back testing a statistical model is a common use case for which the rolling analysis is used. It evaluates the stability and the predictive accuracy of the model. Since the Swissgrid data is a time series data and since the aim is to predict the energy consumption over time, Rolling Analysis will be used.

Linear Regression allows us to build a relationship between two variables. It attempts to draw a straight line in the graph where the data points are plotted so that the most number of data points fall into the straight line as illustrated in Figure 3.2.

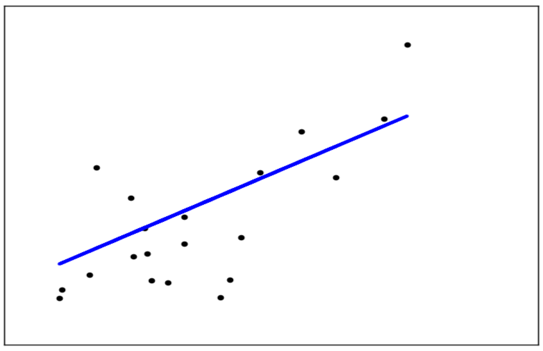


Figure 3.2 Linear Regression

However with regression only, in a time series the outcome might be misleading. It is because in a time series the each value might be affected by the preceding value which is referred to as autocorrelation. For example, the consumption will be high at 5:45p.m but also will be high at 5:46 pm. Hence it's essential to consider these information as well. Due to that Linear Regression with the above rolling window analysis have been used.

3.6 Summary

This chapter describes the technological choices made in order to produce a solution to solve the problem. WSO2 DAS[12] was chosen for the real time and batch analytics as they have in built support for both of them. A client was written in Java because of its Object oriented nature and due to the ease of integrating with WSO2 DAS[12]. SciKit Learn was decided to be used because it was written in python and because it already had support for linear regression with rolling window analysis inbuilt. The next chapter will describe the methodology and how the technologies selected are used in order to build the solution.

Chapter 4

Turning Swissgrid Data into Information

4.1 Introduction

This chapter will explain how the technologies discussed in Chapter 3 will be used to turn the Swissgrid data into interesting and useful statistics.

4.2 System Design & Development

The system was designed as a Big data analytics System. As this involves a large collection of data, and requires to be carefully analyzed the best fit for this was designing this as a Big data System. To model the system Unified Modeling Language (UML) will be used.

4.3 Language for Implementation

This research uses open source third party products mainly for real-time and batch analytics, it is required that the data is pushed to this system via an event stream. In a real world this would be connected to the input stream and the stream would act upon it. To model the behavior the input stream was written in Java. Java is a Object Oriented Programming Language. The language is a free and open source language was designed to run on any platform shining its flexibility. The main reason for selecting Java was because the third party products used also is written in Java and integrating would be cleaner and easier. Java was also written to construct the large data into meaningful data for analysis. To create the model using Machine Learning it was required that the data be organized as date, month, year, end user consumption , season and if holiday or not. This was a tedious task to be done manually. Hence it was done using a Java program.

4.4 Apache Cassandra No SQL DB

Cassandra DB is designed to handle large volumes of data efficiently and intended to used in big data analysis scenarios. WSO2 DAS[12] has inbuilt support for Cassandra. Cassandra has a 100% availability model, which is very important for swissgrid. Also, Since Cassandra has a multiple master model, the writes to the server is scalable. In

Swissgrid the data is very important to be recorded for monitoring as it can be beneficial for crucial decisions. Hence having a guaranteed write model fits perfectly. More over, since it's needed to query on the database for batch analytics, query language support would be required. Cassandra supports CQL a SQL like query language.

4.5 Writing a Java client to publish events to WSO2 DAS

A simple Java client will be written to read the Swissgrid data set from the CSV, and create events for the WSO2 DAS[12] to process. The Java client has to be written in a way such that the event is created according the the stream definition. Once the events are created it was published to the WSO2 DAS[12] as a sequence of events. Specifically here, it read the CSV and sent an event every 15 minutes. This has been done because the data was collected every 15 minutes in the Swissgrid power system. If the system was connected to the power system the data would be pumped into WSO2 DAS[12] every 15 minutes. Hence to mimic the scenario events needed to be pumped into WSO2 DAS[12] as mentioned above.

4.5 Real-time analysis using WSO2 Data Analytics Server

The WSO2 Data Analysis Server (WSO2 DAS[12]) combines real-time, batch, predictive and interactive analysis into one platform. It is capable of collecting data in real-time persisting the data and analyzing the data in order to create meaningful statistics. Figure 4.1 illustrates in high level how the real time event flow is architecture d in WSO2 DAS[12]. Moreover, it also provides a centralized dashboard to view the statistics in. As this project intents in producing real time analytics, and since WSO2 DAS[12] is reliable, easily configurable, easily extensible and open source, it has been selected as the platform to implement the suggested solution. Additionally WSO2 DAS[12]'s real-time analytic engine is powered by Siddi which can process multiple event streams in real-time. Siddi syntax is also like SQL which is user friendly.

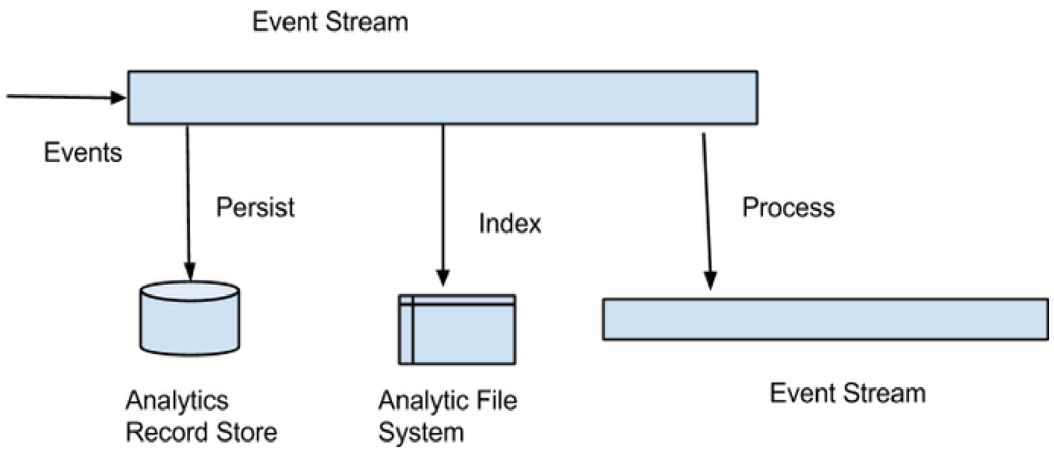


Figure 4.1 Real time event flow [12]

4.5.1 Reading the Data

Using the event publisher which is written in Java, the events will be published in 15 min time span to the WSO2 DAS[12]. Within the WSO2 DAS[12], according to the architecture (appendix A) the WSO2 DAS[12] requires an event stream to be configured. The event stream can be configured via the WSO2 DAS[12] management console. In order to analyze the data finely was required that the events are broken down into smaller events with a fewer columns that has been used for the analytics. For example, to compute the average consumption only the date, month, year and end user consumption is required. Hence it was required to extract the required fields and create relevant events for the statistics and publish in separate streams.

4.5.2 Persisting the Data

Once the event stream is received it will be configured to persist the data in a NoSQL database meant for Big data. In this research Cassandra DB has been used. The swissgrid produces a record with nearly 65 columns. All the fields in the of a record will be persisted in the database as documents which have a structure similar to a JSON .

4.5.3 Computing the statistics

The events will then be analyzed using a Siddhi Query. WSO2 DAS[12] provides an execution plan editor, to write the event processing logic using Siddhi Query Language. Using Sidhhi the event processing logic to create the statistics was written. WSO2 DAS[12] received the complete data stream which consists of 65 fields. Then an execution plan was used to extract out data fields to different streams .The extracted data was then passed through to different execution plans to carry out various analysis. For example to create end user consumption real time statistics that was extracted out the date, year, month and end user consumption. Once these data was extracted and was pumped through a separate stream, DAS[12] computed the statistics in a separate execution plan using the Siddi Query Language. To obtain the average end user consumption a window of a year was used, and compute the average end user consumption for every event that was pushed. After the computation has been done , data such as average end user consumption, the date, year and month was collected and arranged into an event and written to a preconfigured output stream. This output stream was read by the WSO2 DAS[12] Dashboard.

4.5.4 Creating Alerts

It is important to be alerted in a situation where an anomaly has occurred. This helps especially in a power plant to avoid power outages. An impacted reason for Italy's blackout[3] is because the operators were not alerted at the correct time. To avoid situations as such alerting is important. The WSO2 DAS[12] allows creating alerts based on the statistics derived and the information grasped out of it.

Through the event receivers in the WSO2 DAS[12] receives events published by the Java client that was implemented, these event was passed into an execution plan deployed within the WSO2 DAS[12]. This execution plan has been written to process that event and perform analysis to detect any anomalies. In this case one such alert is to identify a gap between the total production and total end user consumption is greater than 1. This alert is crucial because the standard frequency within Europe is 50Hz. If consumption is higher than production then frequency is lower, and higher if vice versa. Hence to maintain and monitor the frequency this alert will be very valuable. Once the query was written to identify the anomaly, the execution plan was published. Then every event was queried upon this event and if an event occurs, the execution plan would generate an event and pass it on to the preconfigured event publisher. The event publisher then would look into the events type, and either go on to send an Email or an SMS as preconfigured to alert the responsible parties.

4.6 Batch Analysis using WSO2 Data Analytics Server

WSO2 Data Analytic Server allows batch analysis to be performed on data that is persisted. As mentioned in section 4.5 the data that would be received by the event receivers of the WSO2 DAS[12] would be be persisted in a Cassandra DB. Batch Analytics will be done on top of the Cassandra DB.

4.6.1 Using Apache Spark for Batch Analytics

The WSO2 DAS[12] batch analytics engine is powered by Apache Spark. The spark high level architecture is illustrated in Figure 4.2 below. Apache Spark is designed to deliver fast, user friendly sophisticated analytics.

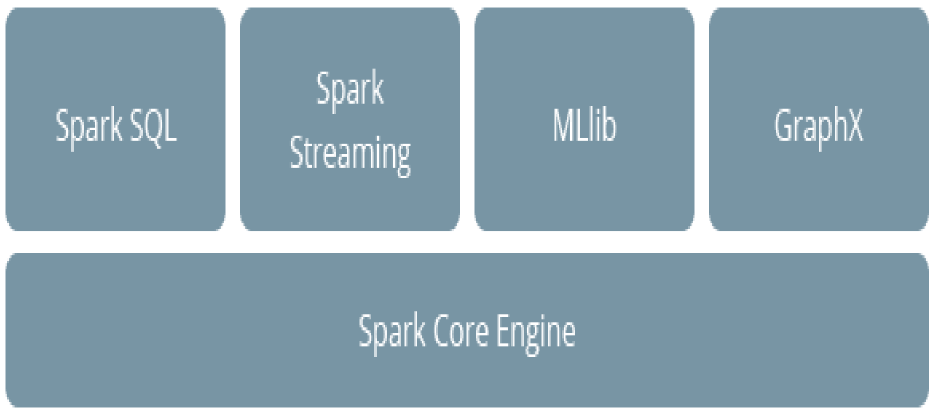


Figure 4.2 Spark Components [13]

The SQL like query language that Apache Spark provides as Spark SQL makes it easy for queries to be written to extract analytics. For example statistics such as Average energy consumption in last month. Once the computation is done this data will also be stored in a temporary table in the database. By using this query Language you can publish the events to the WSO2 DAS[12]. A Spark SQL query would be written to create a virtual table in the Spark table space in order to store the published events. This would also publish the rows of it into the predefined event stream as events. An

Event stream was predefined with the required attributes in order to publish these events. It is also required that a Event receiver to be created of type WSO2Event in order to receive these events from Spark.

The EventStreamProvider class is the interceptor between the existing Spark table and DAS[12] event stream storage. This class will fetch the data from the existing Spark table and publish them as events to the pre defined event stream.

The Batch analytic scripts will be scheduled to run periodically. This will be done via a cron job.

4.7 Displaying the statistics in the WSO2 DAS dashboard

The WSO2 Data Analytics Dashboard is able to create customized dashboards to view the statistics created. It has been required to create a dashboard to view the statistics. Once the dashboard was created, it was possible to create gadegts via DAS[12] to view the statistics. There are many pre included gadgets in the WSO2 DAS[12] dashboard server. Based on the type of the statistic you need to select the gadget. Once you select the gadget you need to tell the gadget from which data publisher to receive data from. It is also required to tell which data it needs to display and how. For e.g the difference in the consumption and production to be showed as a bar chart. The Y-axis to be the production, consumption difference and the X-axis to be the date. Once the gadget is defined it is required to publish the gadget. A dashboard would be created by organizing the gadgets created in the dashboard page. Once the dashboard is viewed the gadget included will be displayed, and the real time data analytics can be displayed.

4.8 Forecasting the end user consumption

This last sub section that the research mainly focuses on is building a model to predict the end user consumption.

4.8.1 Preprocessing the data

The Swissgrid data set consists of 65 columns of data which is collected over every 15 minuets throughout the year. That generates thousands of data records. However, to predict the energy consumption, It required a few columns of data. Identifying this data was done by going through the data set. It was identified that the date, end user energy consumption for the swiss control block and the public holidays were required as input. Furthermore it is only required to get 1 record per day. In order to do this, data records of one day was summarized and the average consumption was obtained. Since pre processing the data manually is a tedious task, writing a java program to do so was the most effective option. Hence the data was read by a Java program which would then extract the date, month, year, total end user consumption for the swiss control block, the season which was based on the date, and whether or not the date is a holiday. Four years of data was created as mentioned above. The data was stored in CSV format.

4.8.2 Using SciKit Learn to build the model

SciKit Learn [17] provides build in support for well known machine learning and statistical models. The support to such models are made available as python modules. Users can import these modules and then train and fine tune these modules to match to their use case via the provided APIs by these modules.

"LinearRegression" is such module provided in SciKit which allows user to create models based on liner regression to make predictions. In this research "LinearRegression" module was used to predict end user energy consumption based on other factors using simple liner regression and with rolling window technique. The data was preprocessed and stored in CSV format to make it available to be fed into the model for training. Because SciKit Learn is based on python modules[17] such as NumPy and SciPy, it can be used in the process of training. NumPy is said to be the fundamental package for scientific computing with Python. It can also be used as a multi dimensional container for generic data. Similarly SciPy is also a library that is used for scientific computing. The usage of these modules enables easy manipulation of data and allows them to be treated as arrays. In addition, python has direct support to read CSV files into an array. By using these capability, the data stored in a CSV file was fetched into the memory and stored as an array and then it was fee into the liner regression model.

4.9 Summary

This chapter explains how the technology was used in order to solve the problem and produce a solution. It explains and justifies in detail on using the technology. The chapter talks about how the WSO2 DAS[12] fits into the solution and why it is chosen for the solution. Also it explains why Java and Sci Kit learn is also used in the solution. Apart from the technology this chapter also explains in detail the methodology used to produce the solution to solve the problem. The next chapter will detail out the System design specific information.

Chapter 5

System Design

5.1 Introduction

In this chapter the architectural design and the functional aspects will be discussed.

5.2 High level design

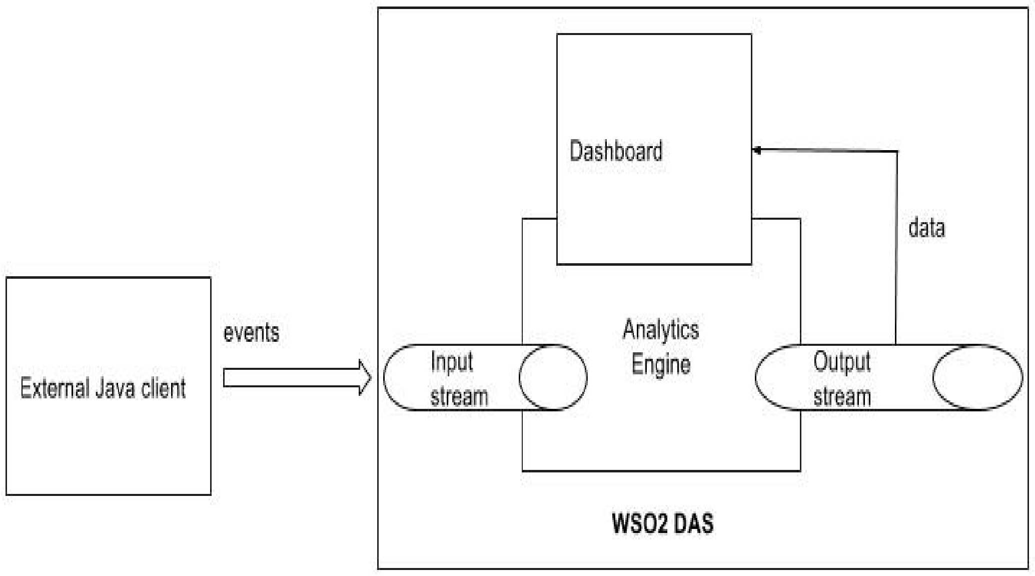


Figure 5.1 High level design of the proposed system

The proposed system as shown in the above figure Figure 5.1 is composed of mainly the External Java client and the WSO2 DAS[12] for the analytics part. The data modeling has been done by a separate third party software. The external Java client reads the data from the CSV and publishes the data to the WSO2 DAS[12]. The WSO2 DAS[12] having a configured data receiver receives these data from the input stream. Once the data is received it computes the data to build the analytics. Then it pushes the analyzed data to the output stream and to a Event Publisher. The Dashboard then reads the event publisher and output the data.

5.3 WSO2 DAS high level architecture

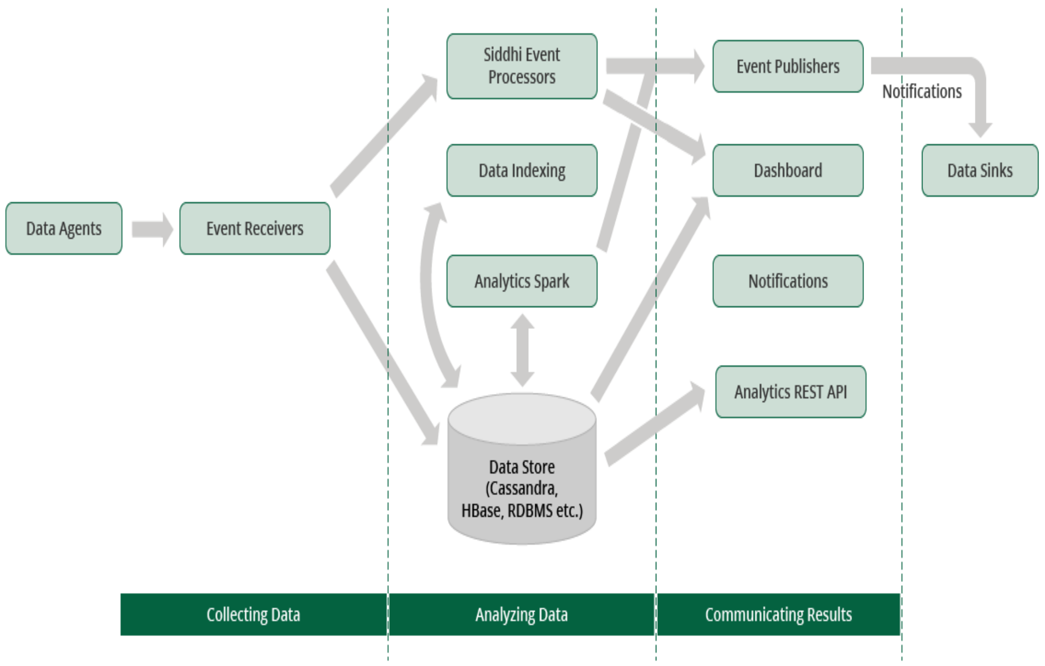


Figure 5.2 WSO2 DAS high level architecture [14]

The WSO2 DAS[12] is capable to collect data from various different data sources. As illustrated in Figure 5.2, once the data is collected , it can be persisted for batch analytics or can be computed on the fly for real time analytics. Once the analytics are created it has been needed to be published for the end users to use it. The WSO2 DAS[12] workflow consists of the following

- **Collecting Data** - Is done via a single API, for external data sources ld process the data event stream flow to generate real time analysis. The datto publish events. The data collected could be persisted for batch analysis or coua from the Swissgrid downloadable could be published to the WSO2 DAS[12] which would be configured to perform real time and batch analysis.
- **Analyzing** Data - Analyzing the data could be done as batch analysis or real time analysis. The real time analysis engine is powered by Siddhi and the

batch analysis engine is powered by Apache Spark.

- **Communicating Results** - The WSO2 DAS[12] analytics dashboard could be used to create customized dashboards for visualizing the analyzed data.

5.4 The Event flow

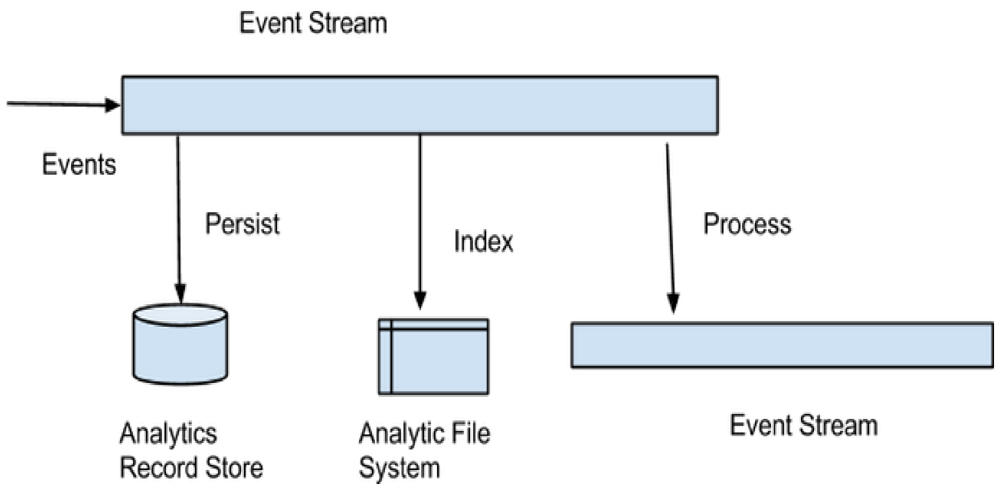


Figure5.3 Event Flow[12]

Within the WSO2 DAS[12], as shown in the above Figure 5.3 the WSO2 DAS[12] receives Events published from an external event publisher. This is fed into the input event stream. For batch analytics the data is persisted to a storage. In this research it is the Cassandra DB. Once the data is persisted, an execution plan is run for the input stream. For real time analytics this execution plan written in Siddi SQL, is used to generate the information to compose the analytics and it publishes it to an output event stream. For Batch analytics the query will be run upon the database, and the Spark SQL query stores the analytic informationn the database. This is configured to run periodically. Once the processing is done upon the data the analyzed data is published to an output event stream. Event publishers listening to the event stream will publish the data to the configured location.

5.5 The Machine Learning model design

Liner regression requires the user to provide with a dependent variable and one or more explanatory variable(s). Dependent variable are the factor which is intent to be predicted while explanatory variable(s) are the factors that decides the value of dependent variable. This research mainly focuses on predicting end user energy consumption. Therefore, end user energy consumption was selected as the dependent variable. In this study two types of models to predict the "end user energy consumption" was built.

1. Multiple Regression (Simple liner regression with multiple dependent variables)

2. Liner regression based on rolling window

For the first model while "end user energy consumption" was the dependent variable, following factors were used as the explanatory variables

- Year
- Month
- Date
- Climate Season
- If the day is public holiday or not

When training data set was fed into this model to learns how the "end user energy consumption" varies with above mentioned factors. So that it could be used to predict the future energy consumptions

In the second approach for building the model, the model was continuously assessed for it's stability, in addition to what it learns form the initial training data set. In this approach, if the window size is set to be n, then the last 'n' data points form the current data point was fed into the model for the model to adjust accordingly. This happens continuously as the data are feed into the model. For this model only the "end user energy consumption" was feed to the model. when data is fed, an initial set of data points are reserved as training set and then the model progressively trained it self based on the rolling window approach as explained above.

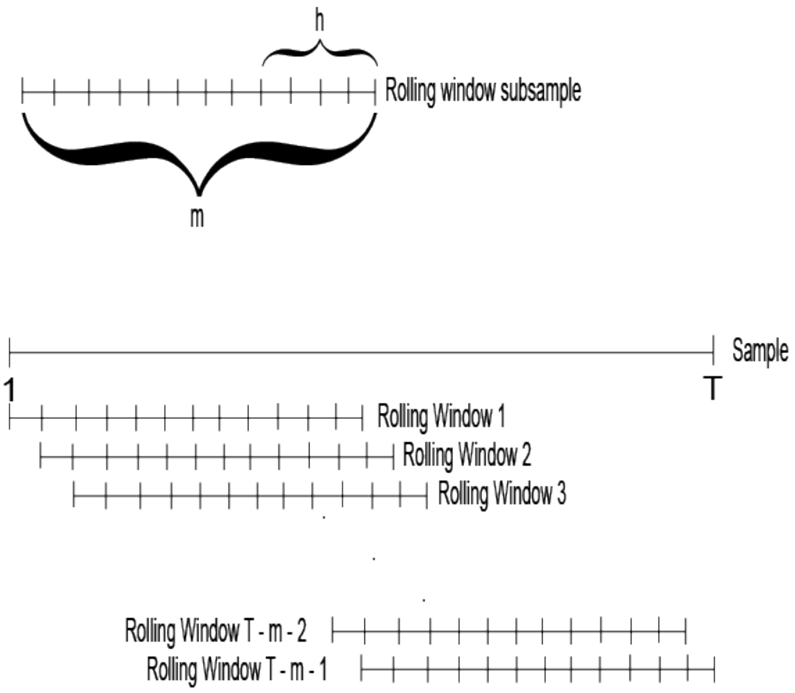


Figure 5.4 Rolling Window Analysis[15]

5.5.1 Absolute Mean Error (MAE)

The absolute mean error[19] in statistics are used to measure how the predictions or forecasts are close to the eventual outcomes as illustrated in Figure 5.4.

$\textit{if } \hat{y}_i \textit{ is the predicted value of the } i^{th} \textit{ sample}, \text{and } y_i \textit{ is the corresponding true value, then the mean absolute error } (MAE) \textit{ estimated } \text{over } n_{samples} \textit{ is defined as}$

$$MAE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |y_i - \hat{y}_i| \qquad (2)$$

The Absolute Mean Error, eq.(2) was taken into account when selecting the data set. In order to determine the data set size for the correct prediction, the least absolute mean error data set has been considered.

5.6 Summary

This chapter in detail discusses the proposed system's design. It also explains the WSO2 DAS[12]'s architectural design and how it helps solve the problem. The event flow in a high level is explained and towards the end the ML model design is also discussed. The next chapter will include the implementation specific information.

Chapter 6

Implementation

6.1 Introduction

This Chapter will discuss on the implementation details of the implemented system. It will also discuss on the techniques used and algorithms used to obtain the results.

6.2 Real-time Analytics

6.2.1 Publishing data to the WSO2 DAS

A Java client was written to publish the data as events to WSO2 DAS[12]. This Java client reads the data from the Swissgrid data sheet, and creates the events for the WSO2 DAS[12] according to the event stream defined. More specifically, the Java code reads the CSV and creates event objects. A single event has a event stream Id, time stamp, and the event payload. Via a data publisher these events are published into the WSO2 DAS[12] in 15 minuet intervals by making the current execution thread to sleep for 15 minuets. Please refer to the appendix A for the source code.

6.2.2 Configuring the event stream

On the WSO2 DAS[12] the event stream is required to be configured. The event stream should be given a name, a version, a description. The event payload has to also be defined in the stream. A stream was created in the WSO2 DAS[12] with the year, month and end user energy consumption for the swiss control block. The event stream defined is as follows

```
  "name": "consumption-production.enduser.consumption",
  "version": "1.0.0",
  "nickName": "",
  "description": "Energy used by end consumers in Switzerland each month. It does
not include transmission losses.",
  "payloadData": [
    {
      "name": "year",
      "type": "STRING"
    },{
      "name":           "month",
      "type": "STRING"
    },
    {
      "name": "endUserConsumption",
      "type": "DOUBLE"
    }
  ]
}
```

6.2.3 Writing the Execution Plan

The execution plan was written in the SQL Like Siddi Query language. In the execution plan first of all the plan name was given. Then it imports the event stream on which it needs to do the computations. The output streams were also mentioned in the execution plan. To show the real time average energy consumption, the average of the energy consumption was calculated for every event and was being inserted into a table. The information was stored in a table and then on top of that data the average had been calculated. Once the average had been calculated it was pushed into the output stream.

```
/* Enter a unique ExecutionPlan */
@Plan:name('EndUserConsumptionAnalysis')

/* Enter a unique description for ExecutionPlan */
-- @Plan:description('ExecutionPlan')

/* define streams/tables and write queries here ... */

@Import('consumption-production.enduser.consumption:1.0.0')
define stream input (year string, month string, endUserConsumption double);

@Export('perDateEnduserConsumption:1.0.0')
define stream perDateEnduserConsumption (date string, endUserConsumption
double);

@Export('enduserConsumption.stats:1.0.0')
define stream statOutput (year string, min double, max double, avg double);

define table statTable (year string, min double, max double, avg double);
```

```
from input#window.firstUnique(year)
select year, 100000.0 as min, 0.0 as max, endUserConsumption as avg
insert into statTable;

from  input  as  i  join  statTable  on
statTable.year    ==    i.year    select
statTable.year as year,
            minimum(statTable.min, i.endUserConsumption) as min,
            maximum(statTable.max, i.endUserConsumption) as max,
            (statTable.avg + i.endUserConsumption)/2.0 as avg
insert into statOutput;

from statOutput
select *
insert overwrite statTable
   on statTable.year == year;

from input
select str:concat(year, "-", month) as date, endUserConsumption
insert into perDateEnduserConsumption;
```

6.2.4 Writing the output stream

Similarly to the input stream an output stream also was defined. In the Output stream also it required to give a name and a version. Also payload format required to be defined. In this case it only required the data and the end user consumption to display on the dashboard.

```
{
  "name": "perDateEnduserConsumption",
  "version": "1.0.0",
  "nickName": "",
  "description":          "",
  "payloadData": [
   {
     "name": "date",
     "type": "STRING"
   },{
     "name": "endUserConsumption",
     "type": "DOUBLE"
   }
  ]}
```

6.3 Batch Analytics

6.3.1 Persisting the events to a data store

The WSO2 DAS[12] receives events in an event stream. This event stream is then persisted to the data store. It is required that the WSO2 DAS[12] be configured with the interested data source before hand. In this case as mentioned above it will be the

Cassandra data source. As the WSO2 DAS[12] already has inbuilt support for Cassandra, it was only a matter of uncommenting the Cassandra Data source Reader in the configuration file.

In order to persist the event, in the event configuration it was required to enable the event to be persisted, by checking a check box (appendix B).

6.3.2 Using Spark SQL to Query the database

Spark is an SQL like query language written to query the Spark engine. The spark query below was the query used to display one batch analytic, which was the energy consumption of the year, in this project.

```
CREATE TEMPORARY TABLE endUserConsumptionData USING CarbonAnalytics
OPTIONS (tableName
"CONSUMPTION-PRODUCTION_ENDUSER_CONSUMPTION",schema  "year
STRING, month STRING, endUserConsumption DOUBLE");

CREATE          TEMPORARY   TABLE  yearlyendUserConsumption      USING
CarbonAnalytics OPTIONS (tableName
"yearly_endUserConsumption_summary",schema      "year     STRING,
avg_endUserConsumption DOUBLE, min_endUserConsumption DOUBLE,
max_endUserConsumption DOUBLE");

INSERT OVERWRITE TABLE yearlyendUserConsumption SELECT year,
avg(endUserConsumption) AS avg_endUserConsumption,
min(endUserConsumption)     AS        min_endUserConsumption,
max(endUserConsumption)     AS        max_endUserConsumption       FROM
```

```
endUserConsumptionData GROUP BY year;
```

Here the Spark SQL first creates a temporary table to store the end user consumption data. It stores the consumption against the month and the year. Then it also creates another temporary table to store the minimum end user consumption, the maximum end user consumption and the average end user consumption against the year. This allowed the dashboard to display the avergae, minimum and maximum end user consumption for the year.

The Spark SQL script can be scheduled to run as a cron job. The Cron expression could be configured when the Spark SQL is created (appendix B).

6.4 Publishing and viewing data in the WSO2 dashboard

To display the data in the UI, it is required that a Event Receiver was configured to read the output stream and publish data to the UI.

```
<?xml version="1.0" encoding="UTF-8"?>
<eventPublisher name="enduserUI" statistics="disable" trace="disable"
xmlns="http://wso2.org/carbon/eventpublisher">
<from  streamName="consumption-production.enduser.consumption"
version="1.0.0"/>
  <mapping customMapping="disable" type="wso2event"/>
  <to eventAdapterType="ui"/>
</eventPublisher>
```

In this instance it publishes the end user consumption data to be viewed in a gadget in the dashboard.

Once the data is published, the dashboard gadget will read the data and show the real time analytic.

6.5 Implementing the ML model

The Machine Learning models were implemented basically using SciKit. The "Linear Regression" module which is available in SciKit was mainly used in this work. The

"Linear Regression" module provides a method to pass the training data set into the model.

6.5.1 Reading the data for analysis

The preprocessed data for the model had been stored in CSV files.Please refer to appendix C for the source code of preprocessing data. These data was read from the CSV using 'panda' module in python. Panda is an open source python library that provides the capability to execute the entire data analysis workflow in Python itself [18]. The data that was read using Panda is then passed onto a instance of "Linear Regression" which is inbuilt in SciKit. NumPy and SciPy was used to manipulate the data set by treating the data as an two dimensional array .

6.5.2 Using Linear Regression

Linear Regression was used as it's required to build a relationship between two factors. This work tries to predict the end user consumption based on the date. Hence only the Simple Linear Regression is used.

Following code snippet show how LinearRegression module is instantiated and data is passed as arrays to the model for training the model

```
data_set = pd.read_csv('./data/training.csv')
y_all    =
data_set['Total_Energy_Consumed_by_end_users_Swiss_controlblock'].values
X_all = data_set[['year', 'date', 'month', 'season', 'publicHoliday']].values


lr = LinearRegression(normalize=True)
lr.fit(X_train, y_train)
```

Once the the model was trained as above, explanatory variables could have been passed on to the data set to get predictions on about dependent variable. Below code snipped illustrates how an two dimensional array of explanatory variables was passed onto the model and get the result.

```
y_pred = lr.predict(X_test)
```

6.5.3 Calculating the Absoluter Error Mean

Finally, the absolute mean error was calculated as follows to measure the accuracy against each predictions .

```
print "mean_absolute_error: %f" %(mean_absolute_error(y_test, y_pred))
```

6.5.4 Using rolling window analysis

In order to build a model with two or more factor the rolling window analysis has been used. The model is continuously assessed for it's stability, in addition to what it learns form the initial training data set. The window size has been set to the value which was re determined. The last 'n' data points form the current data point will be passed to the model so that the model could be trained. In this research implementation the "end user energy consumption" has been passed to the model. when data is feed, an initial set of data points are reserved as training set and then the model will progressively training it self based on the rolling window approach as explained above.

6.6 Summary

The proposed system was developed using the WSO2 DAS[12], a Java client and the ML model was derived using the SciKit Learn tool. The real-time and batch analytics are viewed in the WSO2 DAS[12] Dashboard. A Java client is used to pre process data in order to analyze and build the Machine Learning model to predict the energy consumption. The next chapter will discuss on the evaluation of the system and the limitations and future work.

Chapter 7

Evaluation

7.1 Introduction

This chapter is aimed to evaluate the solution in terms of achieving the objective and ultimately the aim.

7.2 Evaluating the Preprocessed data

The swissgrid data sheet has a large amount of data. Hence pre processing of that data was required in order to train the model. This data required to be for a day, where as the original data sheet has data records for every 15 minuets. Hence as explained in the Implementation details, the data was aggregated and an average for a day was created. This was done with a Java class.

This was tested using unit testing. To evaluate this, a test class was written. A pre determined value would be calculated based on 5 rows of data on a single date. This test class would use that small data set with 5 rows and same number of columns as the original data set. The test class then asserted the value obtained after pre processing with the pre determined value.

7.3 Evaluating the Machine Learning Model

7.3.1 Rolling Window test results.

In this ,the training data set kept increasing and that tried to predict the future. That is First, the future was attempted to be predicted by using 1 years worth of data as the training set. Then the training set was increased to be 1.5 years worth of data and so on.

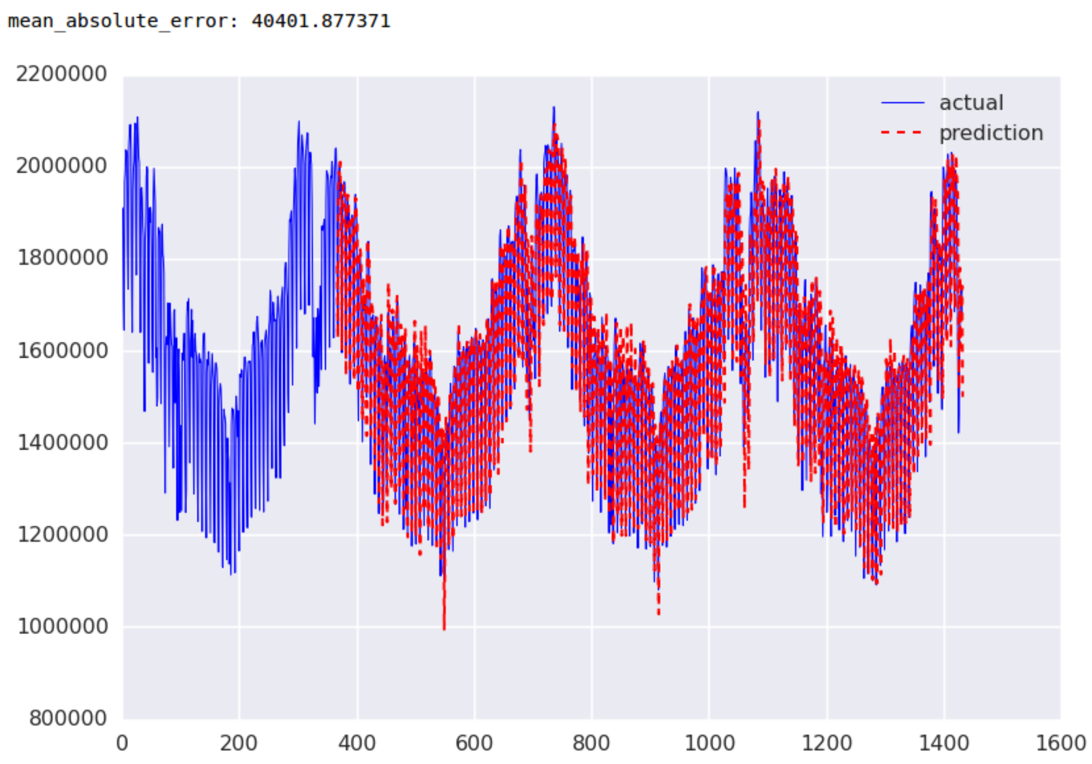
It’s important to note that in this rolling window test, it was attempted to predict the end user energy consumption only by looking at its past value, but without considering any other features. In other words in this model the past values of the same feature acts as explanatory variables. The number of past values to be used as features is decided by the window size. That is, if when there's a rolling window size of 30, the values of the last 30 days decides the value for today. So that the model can be represented in the following format

$$Y = c1*day1Value + c2*day2Value + \ldots\ldots + c30*day30Value \qquad (3)$$

For all theses tests rolling window size of 30 has been used. So that data of the last 30 days or the month is fed into the model to enable it to reassess its stability.

7.3.1.1 Finding the correct testing data size

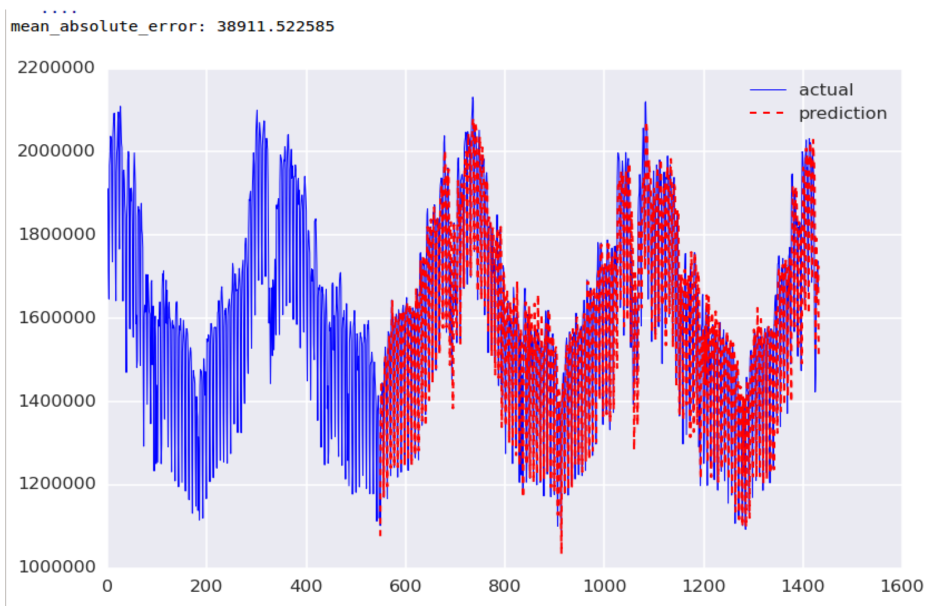
Figure 7.1 illustrates the result that has been obtained by using 1 year data set as the training set and try to predict values for 2014, 2015 and 2016. In this test it has been observed a mean absolute error of 40402 approximately



```
mean_absolute_error: 40401.877371
```

Figure 7.1 1 year data training set results

Following was the result that was obtained by using 1.5 year data set as illustrated in Figure 7.2 as the training set. This data set has been used to predict values for last 6 months of 2014, year 2015 and 2016. In this test it has been observed a mean absolute error of 38911 approximately



```
....
mean_absolute_error: 38911.522585
```

Figure 7.2 1.5 years training set results

Figure 7.3 is the result that was obtained by using 2 year data set as the training set and try to predict values for 2015 and 2016. In this test it has been observed a mean absolute error of 39555 approximately

```
mean_absolute_error: 39544.724602
```

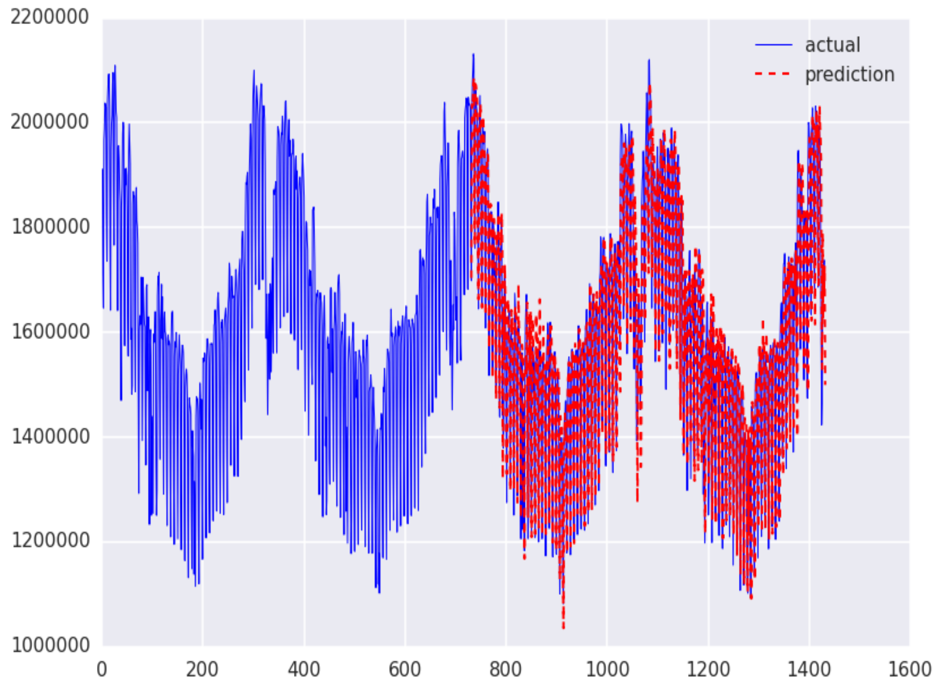
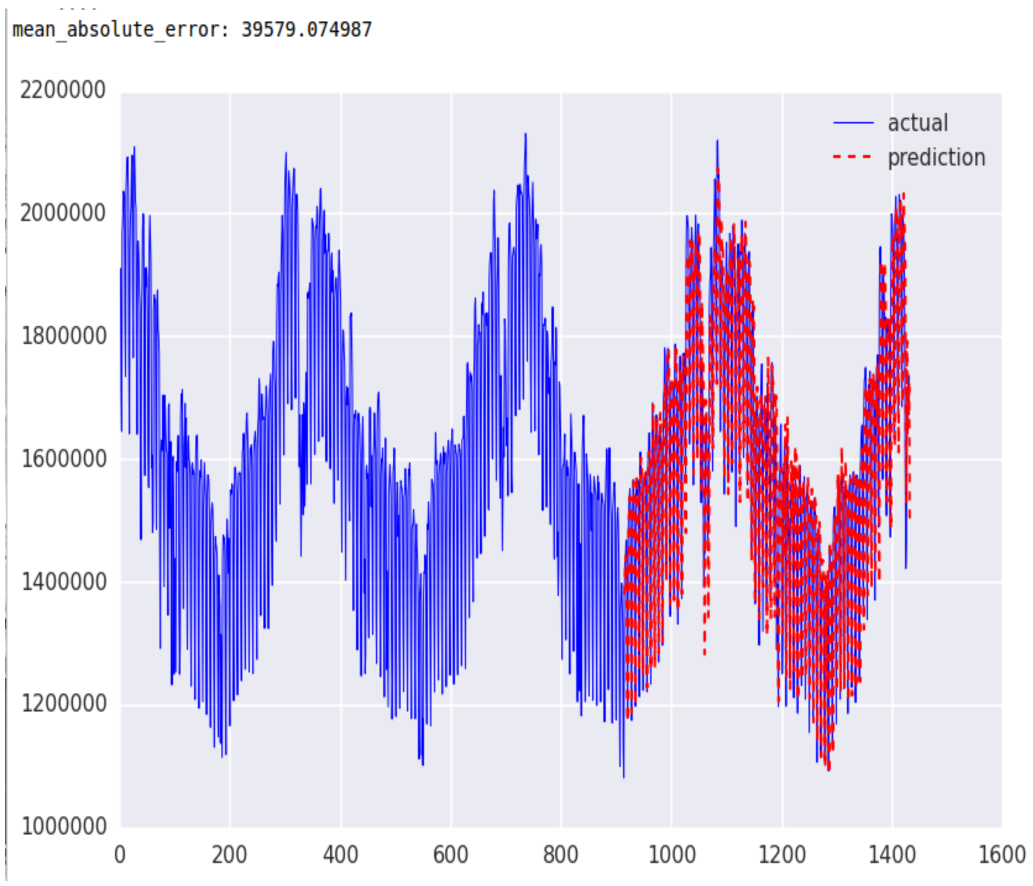


Figure 7.3 2 years data training set results

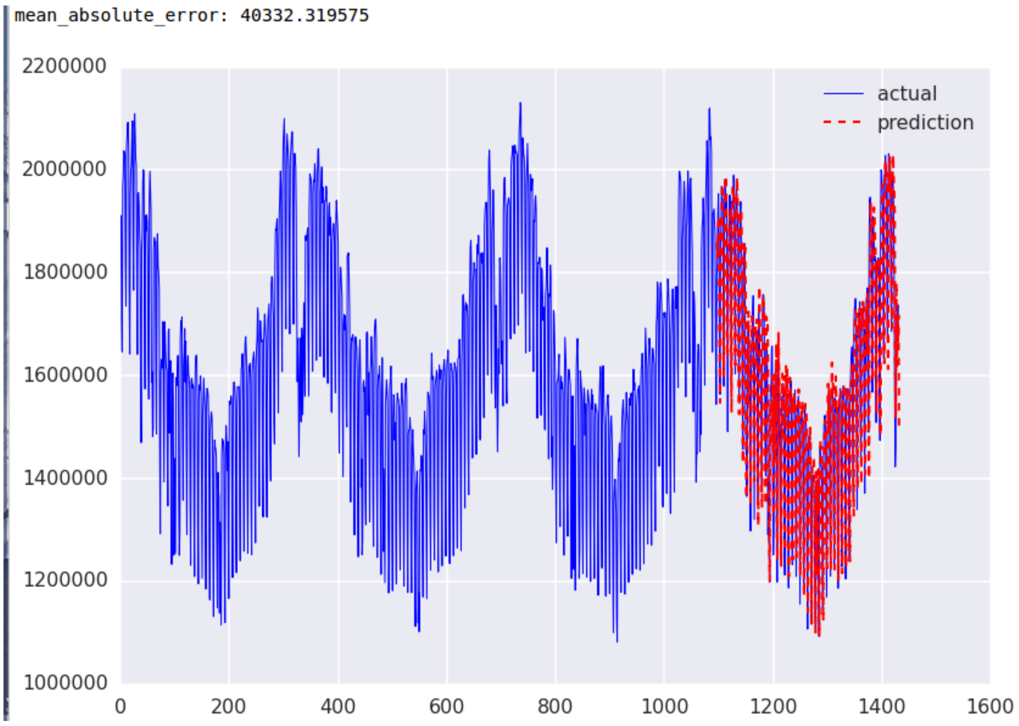
Following Figure 7.4 is the result that was obtained by using 2.5 year data set as the training set and try to predict values for last 6 month of 2015 and year 2016. In this test it has been observed a mean absolute error of 39579 approximately



```
mean_absolute_error: 39579.074987
```

Figure 7.4 2.5 years data training set results

Figure 7.5 illustrates the result that was obtained by using 3 years data set as the training set and try to predict values for 2016. In this test it has been observed a mean absolute error of 40322 approximately



```
mean_absolute_error: 40332.319575
```

Figure 7.5 3 years training data set results

By looking at the results that was observed, the least results for the mean absolute error is given when 2 years of data is used as the the training set. Therefore, it can be concluded that the best training data set size is 2 years for the rolling window based model.

Following are the coefficients of the model for the this test. Because the rolling window is considered to be as the set of features which decides the value, there are 30 coefficients.

[-3.27835429e-04 -1.63487346e-01   1.82017482e-01   9.39205611e-03
-3.54808874e-02   5.90628401e-02 -7.12785283e-02   4.29548412e-02
-2.06309117e-01   2.46737702e-01 -8.23642069e-02   7.01043010e-02
-5.99164900e-03 -2.95570992e-02   6.86008982e-03 -1.57872288e-01
1.77017128e-01   4.71091544e-02 -1.68748115e-01   8.35166869e-02
-1.86998142e-02 -6.32641494e-04 -1.94024897e-01   3.37142395e-01
4.19067742e-02   1.25459019e-01 -1.48166239e-01   1.33358637e-01
-7.01084360e-02   7.62576803e-01]

7.3.2 Multiple Feature Test

In this test 5 factors were used as the features or the explanatory variables. Following are the factors which that was used

- Year (y)
- Month (*m*)
- Date (*d*)
- Climate season (*s*)
- If a day is public holiday or not (*h*)

Therefore, the regression model can be presented in the following format

$$\text{EndUnderConsumption} = c1*y + c2*m + c3*d + c4*s + c5*h \tag{4}$$

In this test a similar approach to the rolling window was used to train the model. That is, the training data set size kept increasing starting from data of year 2013 and tried to predict the future till year 2016

7.3.2.1 Determining the optimal data set

Following is the result that was obtained by using 1 year data set as the training set and try to predict values for 2014, 2015 and 2016. In this test it has been observed a mean absolute error of 193854 approximately. This is shown in figure 7.6.

```
....
mean_absolute_error: 193854.382995
```

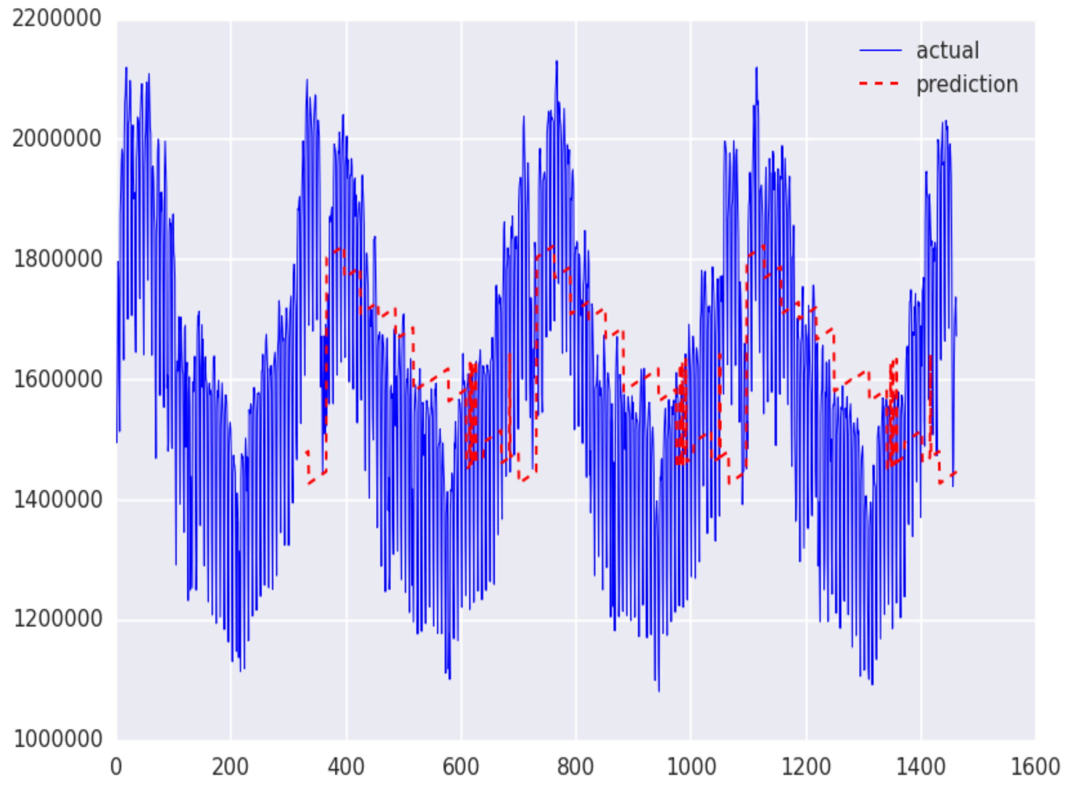


Figure 7.6 Training data set results of 1 year

Figure 7.7 is the result that was obtained by using 1.5 year data set as the training set and try to predict values for last 6 month of 2014, year 2015 and 2016. In this test it has been observed a mean absolute error of 213190 approximately.

Figure 7.8 is the result that was obtained by using 2 year data set as the training set and try to predict values for 2015 and 2016. In this test it has been observed a mean absolute error of 191538 approximately

```
mean_absolute_error: 213190.803394
```

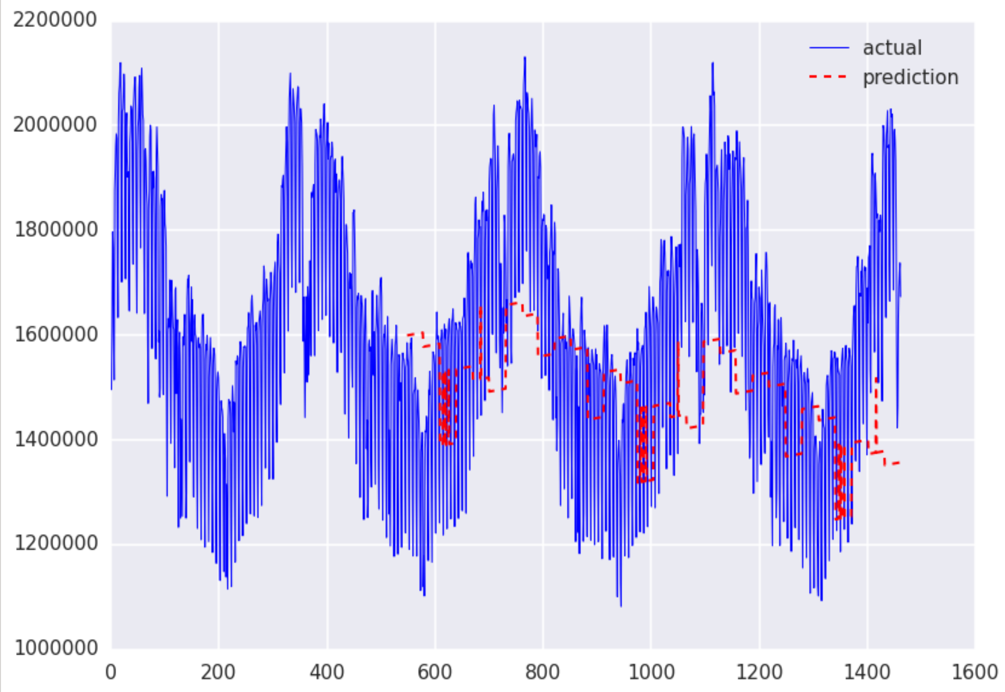


Figure 7.7 1.5 years training data set results7

```
mean_absolute_error: 191538.985520
```

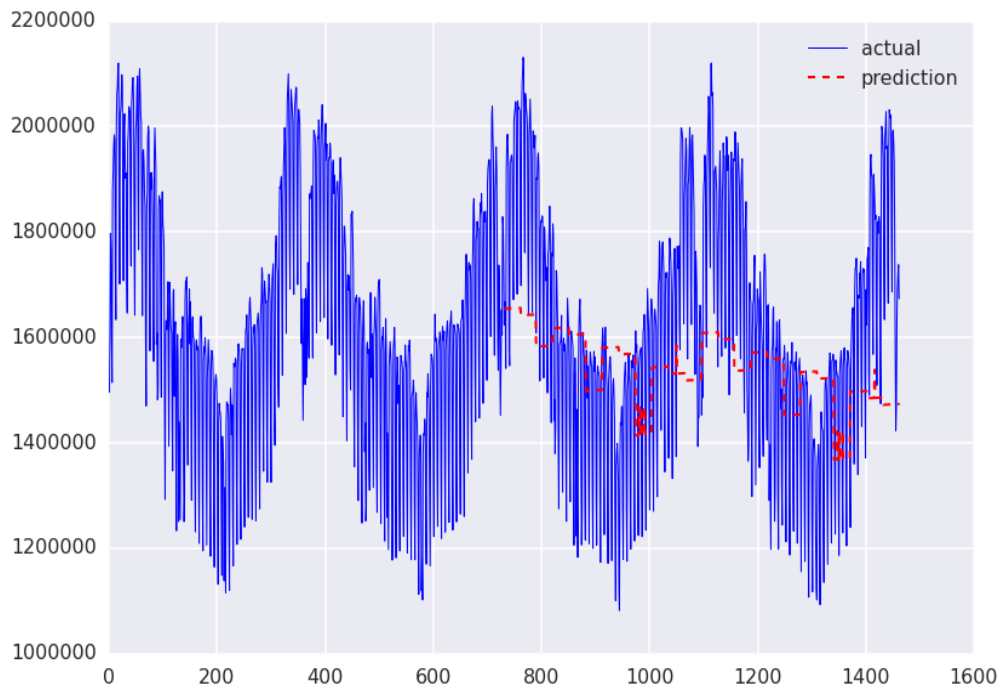
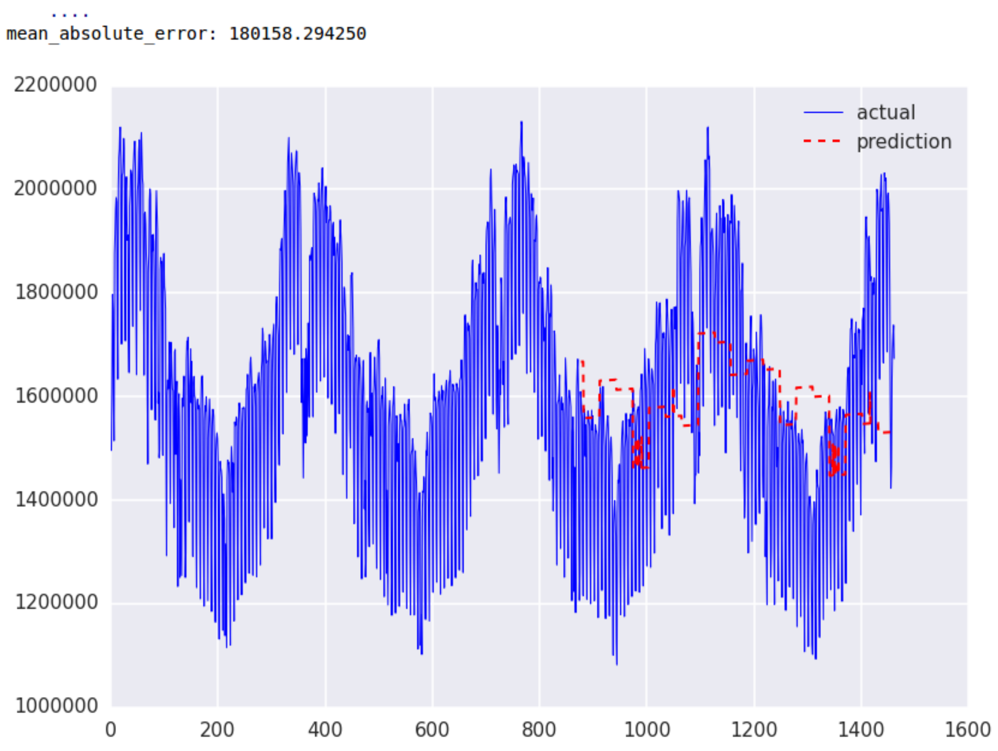


Figure 7.8 2 years training data set results

Following Figure 7.9 is the result that was obtained by using 2.5 year data set as the training set and try to predict values for last 6 month of 2015 and year 2016. In this test it has been observed a mean absolute error of 180158 approximately.



```
....
mean_absolute_error: 180158.294250
```

Figure 7.9 2.5 years training data set results

Following Figure 7.10 is the result that was obtained by using 3 years data set as the training set and try to predict values for 2016. In this test it has been observed a mean absolute error of 183428 approximately.

```
mean_absolute_error: 183428.328644
```

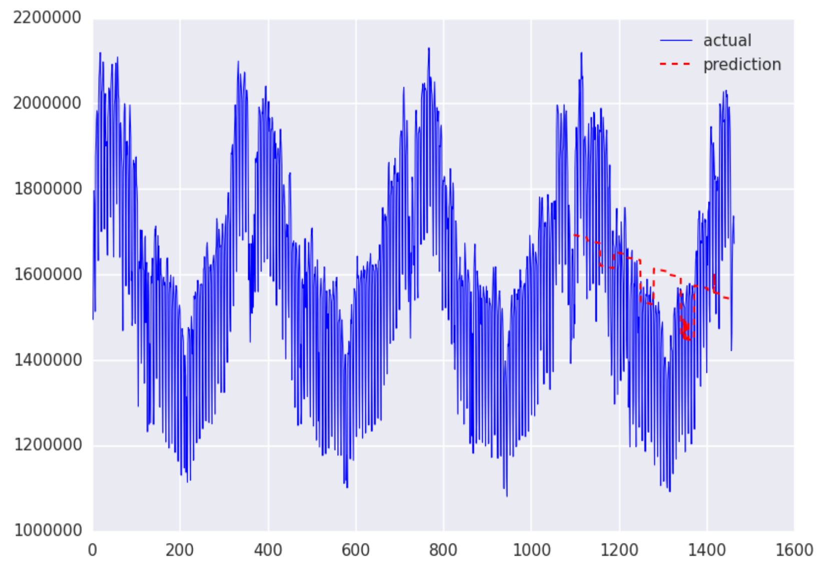


Figure 7.10 3 years training data set results

By looking at the results that was observed, the least results for the mean absolute error is given when it uses 2.5 years of data for the training set. Therefore, it can be concluded that the best training data set size is 2 years for in linear regression with multiple types of features

Following is the equation that was derived by the best case for prediction with simple linear regression

$$EndUnderConsumption = -13618.346*y + 69.31*m -17427.322*d -44847.31*s + 59239.76*h \qquad (5)$$

7.4 Functional testing - Real Time and Batch analytics

Displaying the realtime and batch analytics in a dashboard is also a requirement in this project. After separately creating the real time analytics and batch analytics. It was integrated by a single execution plan to have persist the data received and to do the real time analytics at the same time. To evaluate this a functional test was carried out. The expected output was graphs and charts showing the analyzed data.

The functional tests were as follows as listed in Table 7.1, Table 7.2 and Table 7.3

Test Case 1	
Test Case Name	Real time analytics viewed on Dashboard
Expected Output	Dashboard shows gadget for real-time analytics, with data changing with every event that is published
Steps to follow	•Publish the event using the Java client • Check console for event published • Open the dashboard of the WSO2 DAS[12] • Check if the real-time analytic gadget, minimum end user consumption keeps changing with every event published

Table 7.1 Functional test case1

Test Case 2	
Test Case Name	Batch Analytics viewed on dashboard
Expected Output	•Dashboard shows gadget for batch analytics • Gadget displays data changing every 1hour
Steps to Follows	• Publish Events using Java client • Configure Spark Query to run every 1 hour • Open the dashboard of the WSO2 DAS[12] • Check if the batch analytic gadget for e.g. consumption per month changes every hour

Table 7.2 Functional test case2

Test Case 3	
Test Case Name	Negative test case
Expected Output	• Dashboard shows gadget for batch analytics and real -time analytics • Gadget does not displays data
Steps to Follow	• Stop publishing events using Java client • Open the dashboard of the WSO2 DAS[12] • verify that the Gadget does not displays data

Table 7.3 Functional test case3

Chapter 8

Conclusion

8.1 Introduction

This chapter will focus on the conclusion of this research and state the outcomes. Further this chapter will mention future work and any limitations.

8.2 Analysis of the size of the data set Size.

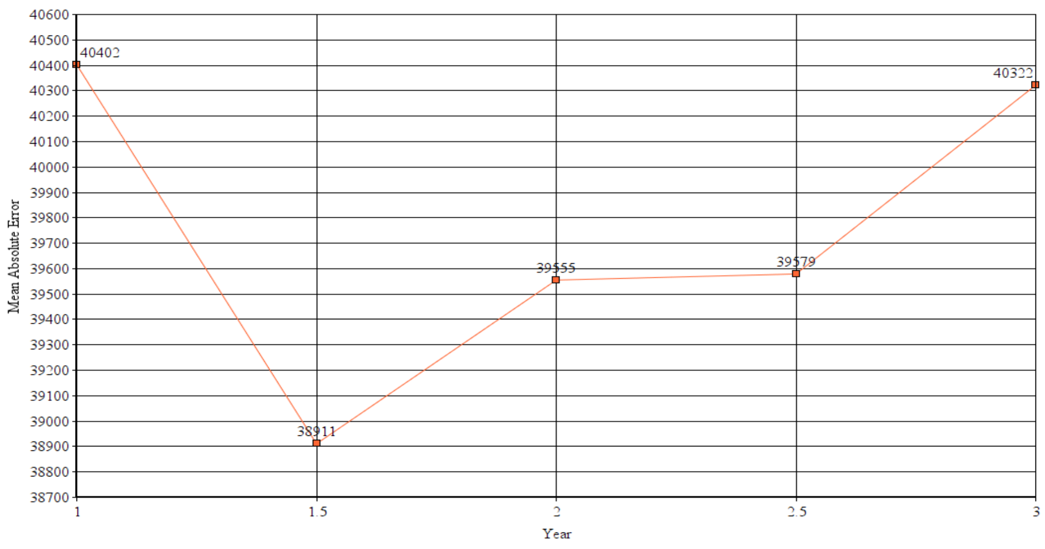


Figure 8.1: Rolling windows data set analysis

8.2.1 Rolling window analysis with Linear Regression for one factor

By analyzing the Absolute Mean Error that was seen with different sizes of data sets as illustrated in Figure 8.1, it is evident that for the rolling window analysis a data set of 1.5 years of data is best suited.

8.2.2 Multiple Linear Regression analysis

By analyzing the Absolute Mean Error that was obtained by training the data model as illustrated in Figure 8.2, it is evidents that for Multiple Linear Regression a data aset of 2 years is bets fit.

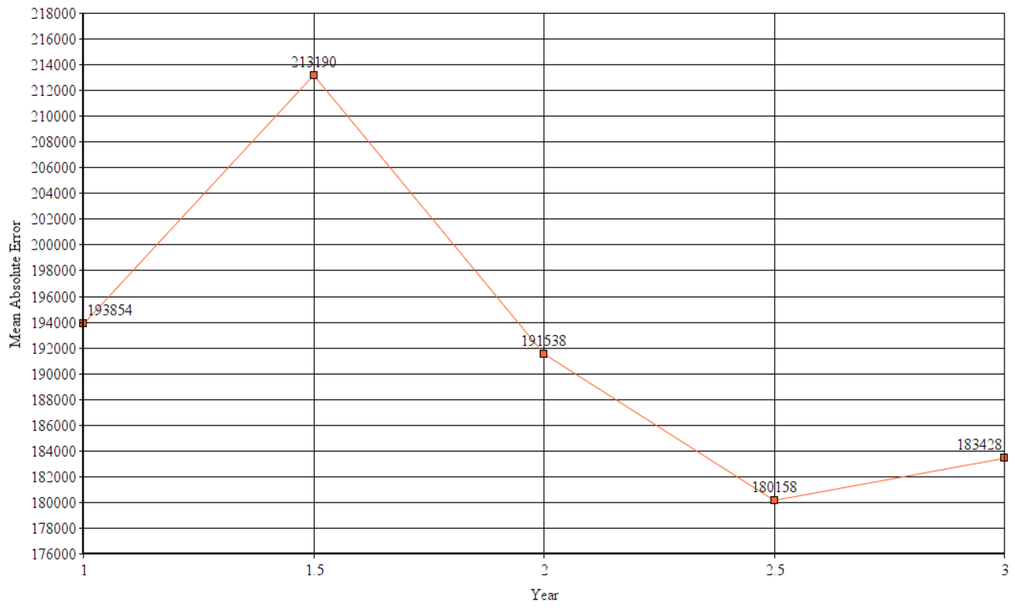


Figure 8.2 Multiple regression data set analysis

8.3 Retrospect

Building a model to predict the energy consumption was a major objective of this project. With the use of Rolling Window Analysis together with Linear regressions it was possible to build a model to predict the energy consumption. It was also found that the best data set to train the model is of 1.5 years.

$$EndUnderConsumption = c1*y + c2*m + c3*d + c4*s + c5*h \qquad (6)$$

Expanding the research, it was also learned that for multiple factors, a different algorithm was needed. And hence the multiple regression was used. After analyzing the data set and using the data set, a model to predict the energy consumption with 5 factors was derived. It was also learned that a data set of 2 years is the best fit.

$$EndUnderConsumption = c1*year + c2*month + c3*day + c4*season + c5*holiday \qquad (7)$$

One other objective for this research was to build a centralized dashboard to view real time and batch analytics in a central dashboard. With the use of WSO2 DAS[12], this was possible. The real-time analytics generated based on the Siddi real time engine and the batch analytics created upon the Spark engine was brought together to be viewed in a single dashboard.
Further more, alerting was another objective that the project set. This objective was achieved by sending alerts to configured users, for real-time and batch analytics using the WSO2 DAS[12].

8.4 Limitations

1. The data set that was downloaded , had nearly 65 columns. However in this research, only 2 columns has been used. This was because the focus was on building a model for consumption, and the end user consumption for swiss control block was the interested data.
2. The swissgrid data had to be pre processed to make meaningful information and to derive factors for analysis.
3. The study only limits to analyze 5 factors.
4. The WSO2 DAS[12] is yet not matured to handle variations of linear regression model building. Hence used SciKit.

8.5 Future Work

1. Incorporate the model building into WSO2 DAS.
2. Use more columns of the data sheet and build relationships by means of a model.

8.6 Summary

This research on the swissgrid data, has resulted in two models to predict the total energy consumption for the swiss control block. One model which depends on one factor which is date and the other model depending on five factors which are, date, month, year, season and nature of day. The selection of the optimal data set to train the algorithms were found based one the absolute mean error that was found with each data set. This chapter also mentions the limitations and the future work.

References

[1] Swissgrid - The Swiss transmission system, Swissgrid.ch, 2016. [Online]. Available: https://www.swissgrid.ch/swissgrid/en/home/grid/transmission_system.html. [Accessed: 13- Nov- 2016].

[2] “Swissmod”. [Online]. Available http://simlab.ethz.ch/swissmod.php.[Accessed: 24-Apr-2017]

[3] Z. Lukszo, *Securing electricity supply in the cyber age*, 1st ed. Dordrecht: Springer, 2010, pp. 60-62.62

[4] C. CHARLTON, "Germany PAYS people to use electricity: Excess energy created by wind and solar power meant consumers made a profit as prices were driven so low they went NEGATIVE," 10:20 BST, 11 May 2016

[5] Forecasting Electricity Demand."Forecasting Electricity Demand", Siemens.com, 2016. [Online]. Available: http://www.siemens.com/innovation/en/home/pictures- of-the-future/energy-and-efficiency/power-transmission-forecasting-electricity-demand.html. [Accessed: 12- Nov- 2016].

[6] A. Boiron, S. Lo and A. Marot, "Predicting Future Energy Consumption", 2012.

[7] P. Dagnely, T. Ruette, T. Tourwé, E. Tsiporkova and C. Verhelst, "Predicting Hourly Energy Consumption. Can Regression Modeling Improve on an Autoregressive Baseline?", in *Proceedings of 24th Annual Machine Learning Conference*, Benelearn, Delft, The Netherlands, 2015, pp. Pages 105-122.1.

[8] M. Larsson, L. F. Santos, A. Suranyi, W. Sattinger and R. Notter, "Monitoring of oscillations in the continental European transmission grid," *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, Vienna, 2013, pp. 4774-4778.

[9] “Wide Area Monitoring”. [Online]. Avaialble https://www.swissgrid.ch/swissgrid/fr/home/reliability/wam.html [Accessed: 24-Apr-2017] [Accessed: 24-Apr-2017]

[10] Predicting the price of electricity is as uncertain as the weather |

@guardianletters."Predicting the price of electricity is as uncertain as the weather | @guardianletters", theGuardian,2016. [Online].Available:https://www.theguardian.com/environment/2014/jul/14/predicting-price-of-electricity-uncertain. [Accessed: 13- Nov- 2016].

[11] Smart Learning Software Predicts Renewable Energy Output with 90% Accuracy."Smart Learning Software Predicts Renewable Energy Output with 90% Accuracy", T reeHugger,2016. [Online].Available:http://www.treehugger.com/clean-technology/learning-software-predicts-renewable-energy-output.html. [Accessed: 12-Nov- 2016].

[12]"Introducing DAS - Data Analytics Server 3.1.0 - WSO2 Documentation", Docs.wso2.com,2016.[Online].Available: https://docs.wso2.com/display/DAS310/Introducing+DAS. [Accessed: 12-Nov- 2016].

[13]"Batch Analytics Using Spark SQL". [Online]. Available https://docs.wso2.com/display/DAS310/Batch+Analytics+Using+Spark+SQL. [Accessed: 24-Apr-2017]

[14] "Architecture". [Online]. Available https://docs.wso2.com/display/DAS310/Architecture [Accessed: 24-Apr-2017] [Accessed: 24-Apr-2017]

[15]"Rolling-Window Analysis of Time-Series Modelss". [Online]. Available https://www.mathworks.com/help/econ/rolling-window-estimation-of-state-space-models.html?requestedDomain=true. [Accessed: 24-Apr-2017]

[16] S. Perera, "Rolling Window Regression: a Simple Approach for Time Series Next value Predictions", *Medium*, 2017. [Online]. Available: https://medium.com/making-sense-of-data/time-series-next-value-prediction-using-regression-over-a-rolling-window-228f0acae363. [Accessed: 03- June- 2016].

[17] scikit-learn: machine learning in Python — scikit-learn 0.18.1 documentation."scikit-learn: machine learning in Python — scikit-learn 0.18.1 documentation", Scikit-learn.org, 2016. [Online]. Available: http://scikit-learn.org/stable/. [Accessed: 12- Nov- 2016].

[18] "Python Data Analysis Library — pandas: Python Data Analysis Library." [Online]. Available: http://pandas.pydata.org/. [Accessed: 24-Apr-2017].

[19] "Mean absolute error," *Wikipedia*. 24-Apr-2017.

[20] Quantum, "FAST DATA SCIENCE HELPS SWITZERLAND PLAN FOR ELECTRICITY SHORTAGES," 01-Feb-2016. [Online]. Available: http://quantumanalytics.ch/wordpress/wp-content/uploads/2016/03/Quantum-Analytics-Swissgrid-electricity-shortage-in-CH.pdf

[21]"InnovationNewsPressSiemensChina."[Online].Available:http://w1.siemens.com.cn/news_en/frontier_technology_en/2294.aspx. [Accessed: 13- Nov- 2016].

[22] Power transmission."Power transmission", Siemens.com, 2016. [Online]. Available:http://www.siemens.com/innovation/en/home/pictures-of-the- future/energy-and-efficiency/power-transmission-electrictiy-superhighways.html. [Accessed: 13- Nov-2016].

[23] Smart Power Transmission Saves Millions."Smart Power Transmission Saves Millions",Siemens.com,2016.[Online].Available: http://www.siemens.com/innovation/en/home/pictures-of-the-future/energy-and-efficiency /power-transmission-facts-and-forecasts.html. [Accessed: 12- Nov-2016].https://www.google.lk/search?q=Data-Stream-Mining-A-Review-on-Windo&oq=Data-Stream-Mining-A-Review-on-Windo&aqs=chrome..69i57j69i60.262j0j4&sourceid=chrome&ie=UTF-8#q=PredictingFutureEnergyConsumption.pdf]

[24] "Swissgrid - Production and consumption." [Online]. Available: https://www.swissgrid.ch/swissgrid/en/home/reliability/griddata/generation.html https://www.google.lk/search?q=Data-Stream-Mining-A-Review-on-Windo&oq=Data-Stream-Mining-A-Review-on-Windo&aqs=chrome..69i57j69i60.262j0j4&sourceid=chrome&ie=UTF-8#q=PredictingFutureEnergyConsumption.pdf] [Accessed: 24-Apr-2017]

[25] Swissgrid - Smart Grid."Swissgrid - Smart Grid", Swissgrid.ch, 2016. [Online]. Available:https://www.swissgrid.ch/swissgrid/en/home/future/smartgrid.html. [Accessed: 13- Nov- 2016].

[26] Swissgrid - Supergrid."Swissgrid - Supergrid", Swissgrid.ch, 2016. [Online]. Available:https://www.swissgrid.ch/swissgrid/en/home/future/supergrid.html. [Accessed: 13- Nov- 2016].

[27] Volatile but predictable: Forecasting renewable power generation."Volatile but predictable: Forecasting renewable power generation", Clean Energy Wire, 2016. [Online]. Available: https://www.cleanenergywire.org/factsheets/volatile- predictable-forecasting-renewable-p ower-generation. [Accessed: 13- Nov- 2016].

[28] "Swissgrid -Production and consumption." [Online].Available: https://www.swissgrid.ch/swissgrid/en/home/reliability/griddata/generation.html. [Accessed: 24-Apr-2017]

[29] E. Zivot and J. Wang, *Modeling financial time series with S-Plus*, 1st ed. New York: Springer, 2006, pp. 299-346.

Appendix A *- Implementation of the Event Publisher & Data Publisher*

Event Publisher

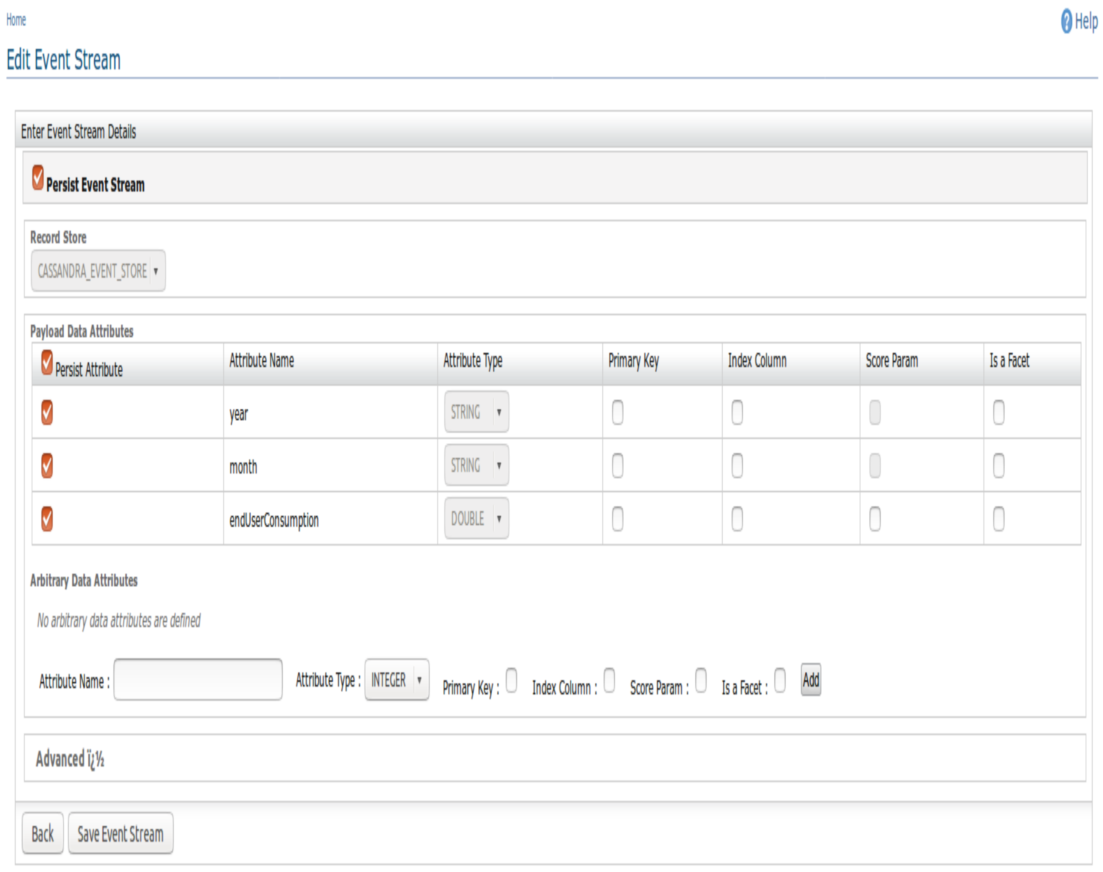
```
public class ResearchEventPublisher{


    private static Log log = LogFactory.getLog(ResearchEventPublisher.class);
    private   static   DataPublisher   privateDataPublisher;
    private static DataPublisher currentDataPublisher;
    private static int count = 0;
    public static void main(String[] args) {
                                                            System.setProperty("org.xml.sax.driver",
"com.sun.org.apache.xerces.internal.parsers.SAXParser");
        System.setProperty("javax.xml.parsers.DocumentBuilderFactory","com.sun.org.
apache.xerces.internal.jaxp.DocumentBuilderFactoryImpl");
        System.setProperty("javax.xml.parsers.SAXParserFactory","com.sun.org.apache
.xerces.internal.jaxp.SAXParserFactoryImpl");
               System.out.println("Starting WSO2 Event ResearchEventPublisher Stream
Client");
           AgentHolder.setConfigPath(publisher.schedular.util.DataPublisherUtil.filePat
h + "/src/main/java/files/configs/data-agent-config.xml");
          publisher.schedular.util.DataPublisherUtil.setTrustStoreParams();
          String protocol = "thrift";
          String singleNodeHost = "tcp://localhost:7611";
          String   username   =   "admin";
          String password = "admin";
       try{
            privateDataPublisher = new DataPublisher(protocol, singleNodeHost, null,
username, password);
          currentDataPublisher = privateDataPublisher;
                                                                  List<Object[]>    eventsList         =
StatisticsInputReaderTask.readCurrentValuesFromFile("/home/shani/MSC/research/s
wiss/resources/swissDataGridData2016.csv");
            for (Object[] eventpayload : eventsList){
                publishEvent(eventpayload, "ControlBlock:1.0.0");
            }
        }catch(Exception e) {
             log.error("Exception occurred while Publishing data",e);
        }
    }
      public static void publishEvent(Object[] eventPayload, String streamId) throws
InterruptedException {
              Event event = new Event(streamId, System.currentTimeMillis(), null, null,
eventPayload);
        currentDataPublisher.publish(event);
        Thread.sleep(900000);
    }
}
```

Creating the data publisher with the Stream definition

```
public class DataPublisherUtil {
    private static Log log = LogFactory.getLog(DataPublisherUtil.class);
                                            public static String filePath =
"/home/shani/MSC/research/swiss/eventPublisherMsc/";
    static File securityFile = new File(filePath + "src/main/java/files/configs");
    public static void setTrustStoreParams() {
        String trustStore = securityFile.getAbsolutePath();
        System.setProperty("javax.net.ssl.trustStore", trustStore + "" + File.separator +
"client-truststore.jks");
        System.setProperty("javax.net.ssl.trustStorePassword", "wso2carbon");
    }
    public static Map<String, StreamDefinition> loadStreamDefinitions() {
        String directoryPath = filePath + "/src/main/java/files/streamDefinitions";
        File directory = new File(directoryPath);
            Map<String, StreamDefinition> streamDefinitions = new HashMap<String,
StreamDefinition>();
        if (!directory.exists()) {
                                        log.error("Cannot load stream definitions from " +
directory.getAbsolutePath() + " directory not exist");
            return streamDefinitions;
        }
        if (!directory.isDirectory()) {
                                        log.error("Cannot load stream definitions from " +
directory.getAbsolutePath() + " not a directory");
            return streamDefinitions;
        }
        File[] defFiles = directory.listFiles();
        if (defFiles != null) {
            for (final File fileEntry : defFiles) {
                if (!fileEntry.isDirectory()) {
                    BufferedReader bufferedReader = null;
                    StringBuilder stringBuilder = new StringBuilder();
                    try {
                        bufferedReader = new BufferedReader(new FileReader(fileEntry));
                        String line;
                        while ((line = bufferedReader.readLine()) != null) {
                            stringBuilder.append(line).append("\n");
                        }
                                                        StreamDefinition streamDefinition =
EventDefinitionConverterUtils.convertFromJson(stringBuilder.toString().trim());
                                          streamDefinitions.put(streamDefinition.getStreamId(),
streamDefinition);
                    } catch (FileNotFoundException e) {
                        log.error("Error in reading file " + fileEntry.getName(), e);
                    } catch (IOException e) {
                        log.error("Error in reading file " + fileEntry.getName(), e);
} catch (MalformedStreamDefinitionException e) {
       log.error("Error in converting Stream definition " + e.getMessage(),
} finally {
bufferedReader.close();
    try {
        if (bufferedReader != null) {
        }
    } catch (IOException e) {
log.error("Error occurred when reading the file : " +e.getMessage(), e);
                        }
                    }
                }
            }
        }
        return streamDefinitions;
    }
}
```

Persisting the Event


Home

Help

Edit Event Stream

Enter Event Stream Details

✓ Persist Event Stream

Record Store

CASSANDRA_EVENT_STORE

Payload Data Attributes

✓ Persist Attribute	Attribute Name	Attribute Type	Primary Key	Index Column	Score Param	Is a Facet
✓	year	STRING	☐	☐	☐	☐
✓	month	STRING	☐	☐	☐	☐
✓	endUserConsumption	DOUBLE	☐	☐	☐	☐

Arbitrary Data Attributes

No arbitrary data attributes are defined

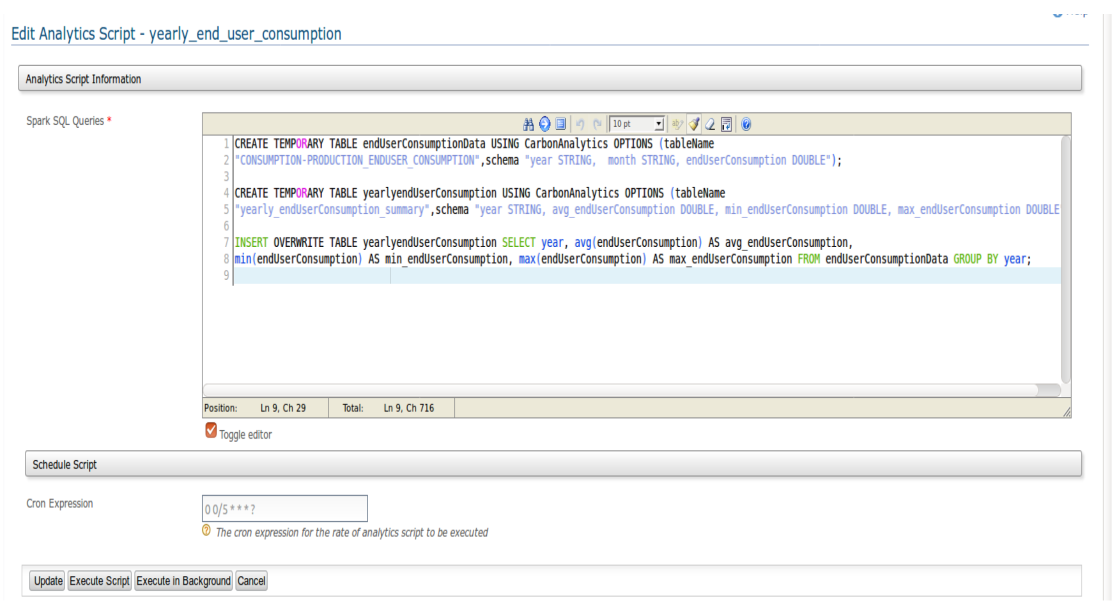
Attribute Name : Attribute Type : INTEGER Primary Key : ☐ Index Column : ☐ Score Param : ☐ Is a Facet : ☐ Add

Advanced ï¿½

Back Save Event Stream

Appendix B Figure 1 persisting events

Creating the Spark SQL script



Edit Analytics Script - yearly_end_user_consumption

Analytics Script Information

Spark SQL Queries *

```
CREATE TEMPORARY TABLE endUserConsumptionData USING CarbonAnalytics OPTIONS (tableName
"CONSUMPTION-PRODUCTION_ENDUSER_CONSUMPTION",schema "year STRING,  month STRING, endUserConsumption DOUBLE");

CREATE TEMPORARY TABLE yearlyendUserConsumption USING CarbonAnalytics OPTIONS (tableName
"yearly_endUserConsumption_summary",schema "year STRING, avg_endUserConsumption DOUBLE, min_endUserConsumption DOUBLE, max_endUserConsumption DOUBLE

INSERT OVERWRITE TABLE yearlyendUserConsumption SELECT year, avg(endUserConsumption) AS avg_endUserConsumption,
min(endUserConsumption) AS min_endUserConsumption, max(endUserConsumption) AS max_endUserConsumption FROM endUserConsumptionData GROUP BY year;
```

Position: Ln 9, Ch 29 Total: Ln 9, Ch 716

Toggle editor

Schedule Script

Cron Expression: 0 0/5 * * * ?

The cron expression for the rate of analytics script to be executed

Appendix B Figure 2 Spark SQL script

Appendix C - Data preprocesser

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
public class DataCruncher {
   private static final String datapath = "home/shani/MSC/research/swiss/datasets";
   private static final List<Datapoint> data = new ArrayList<Datapoint>();
   public static void main(String[] args) throws IOException {
      BufferedReader br = new BufferedReader(new FileReader(datapath +
"/data2015.csv"));
      boolean isHeaderSkipped = false;
      try {
         StringBuilder sb = new StringBuilder();
         String line = br.readLine();
         while (line != null) {
            if (isHeaderSkipped == false){
               isHeaderSkipped = true;
            } else {
               dailyAverage(line);
            }
            line = br.readLine();
         }
      } finally {
         br.close();
      }
      outputResults();
   }
```

```
    private static void outputResults(){
        data.sort(Comparator.comparing(datapoint -> datapoint.hashCode()));
        data.forEach(datePoint -> {
            System.out.println(datePoint.toString());
        });
    }

    private static void dailyAverage(String line){
        final String[] columns = line.split(",");
        int year = Integer.parseInt(columns[0]);
        int month = Integer.parseInt(columns[1]);
        int date = Integer.parseInt(columns[2]);;
        double consumption = Double.parseDouble(columns[3]);
        int season = Integer.parseInt(columns[4]);
        int holidayFlag = Integer.parseInt(columns[5]);



        Datapoint dataPoint = getDataPoint(year, month, date, season, holidayFlag);
        dataPoint.adjustAverage(consumption);
    }

    private static Datapoint getDataPoint(int year, int month, int date, int season, int
holidayFlag){
        Datapoint newDatapoint = new Datapoint(year, month, date, season,
holidayFlag);

        for (Datapoint d : data){
            if (d.equals(newDatapoint)){
                return d;

            }
        }
        data.add(newDatapoint);
        return newDatapoint;
    }

    static class Datapoint{
        public int month;
        int date;
        int year;
        double totalConsumption = 0.0;
        int dataPoints = 0;
        int season;
        int holidayFlag;
        static NumberFormat formatter = new DecimalFormat("#0.00");

        public Datapoint(int year, int month, int date, int season, int holidayFlag){
            this.year = year;
            this.month = month;
            this.date =  date;
            this.season = season;
            this.holidayFlag = holidayFlag;
        }

        public void adjustAverage(double newConsumptions){
            totalConsumption += newConsumptions;
            dataPoints++;
        }

        @Override
        public String toString(){
            return Integer.toString(year) + "," +             ((month < 10) ? "0" +
Integer.toString(month) : Integer.toString(month)) + "," +
                    ((date < 10) ? "0" + Integer.toString(date) : Integer.toString(date)) + ","
+
                    formatter.format(totalConsumption/dataPoints) + "," +
                Integer.toString(season) + "," +
                Integer.toString(holidayFlag);
        }
        @Override
        public boolean equals(Object obj){
            if (obj == null) {
                return false;
            }
            if (!Datapoint.class.isAssignableFrom(obj.getClass())) {
                return false;
            }
            final Datapoint other = (Datapoint) obj;
            if (this.year == other.year && this.month == other.month && this.date ==
other.date){
                return true;
            } else {
                return false;
            }
        }
        @Override
        public int hashCode(){
            String hashString = Integer.toString(year) +
                ((month < 10) ? "0" + Integer.toString(month) : Integer.toString(month))
+
                ((date < 10) ? "0" + Integer.toString(date) : Integer.toString(date)) ;
            return  Integer.parseInt(hashString);
        }
    }
}
```

Appendix D – Source for building the model to predict energy consumption – with one factor

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pylab as plt

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error

import seaborn as sns
sns.set_style("darkgrid")
sns.set_context("poster")

def rolling_univariate_window(time_series, window_size):
    shape = (time_series.shape[0] - window_size + 1, window_size)
    strides = time_series.strides + (time_series.strides[-1],)
    return np.lib.stride_tricks.as_strided(time_series, shape=shape, strides=strides)

def build_rolling_window_dataset(time_series, window_size):
    last_element = time_series[-1]
    time_series = time_series[:-1]
    X_train = rolling_univariate_window(time_series, window_size)
    y_train = np.array([X_train[i, window_size-1] for i in range(1, X_train.shape[0])])

    return X_train, np.hstack((y_train, last_element))

def train_test_split(no_of_training_instances, X_all, y_all):
    X_train = X_all[0:no_of_training_instances, :]
    X_test = X_all[no_of_training_instances:, :]
    y_train = y_all[0:no_of_training_instances]
    y_test = y_all[no_of_training_instances:]
    return X_train, X_test, y_train, y_test

def print_graph(X_all, X_test, y_all, y_test, y_pred):
    training_size = X_all.shape[0] - X_test.shape[0]
    x_full_limit = np.linspace(1, X_all.shape[0], X_all.shape[0])
    y_pred_limit = np.linspace(training_size+1, training_size + 1 + X_test.shape[0],
X_test.shape[0])
    plt.plot(x_full_limit, y_all, label='actual', color='b', linewidth=1)
    plt.plot(y_pred_limit, y_pred, '--', color='r', linewidth=2, label='prediction')
    plt.legend(loc=0)
    plt.show()

data_set = pd.read_csv('/home/sajith/shani-project/datasets/consumptions.csv')
data_set = data_set.values.flatten()

window_size = 16
training_set_size = 30

X_all, y_all = build_rolling_window_dataset(data_set, window_size)
X_train, X_test, y_train, y_test = train_test_split(training_set_size, X_all, y_all)

lr = LinearRegression(normalize=True)
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
print "mean_absolute_error: %f" %(mean_absolute_error(y_test, y_pred))
print_graph(X_all, X_test, y_all, y_test, y_pred)
```

Appendix E – Source code for building the prediction model with multiple regression

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pylab as plt

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error

import seaborn as sns
sns.set_style("darkgrid")
sns.set_context("poster")



def train_test_split(no_of_training_instances, X_all, y_all):
    X_train = X_all[0:no_of_training_instances, :]
    X_test = X_all[no_of_training_instances:, :]
    y_train = y_all[0:no_of_training_instances]
    y_test = y_all[no_of_training_instances:]

    return X_train, X_test, y_train, y_test


def print_graph(X_all, X_test, y_all, y_test, y_pred):
    training_size = X_all.shape[0] - X_test.shape[0]
    x_full_limit = np.linspace(1, X_all.shape[0], X_all.shape[0])
    y_pred_limit = np.linspace(training_size+1, training_size + 1 + X_test.shape[0], 
X_test.shape[0])
    plt.plot(x_full_limit, y_all, label='actual', color='b', linewidth=1)
    plt.plot(y_pred_limit, y_pred, '--', color='r', linewidth=2, label='prediction')
    plt.legend(loc=0)
    plt.show()
```

```
data_set = pd.read_csv('/home/sajith/shani-project/datasets/summarized.csv')
training_set_size = 30

y_all =
data_set['Total_Energy_Consumed_by_end_users_Swiss_controlblock'].values
X_all = data_set[['year', 'date', 'month', 'season', 'publicHoliday']].values
X_train, X_test, y_train, y_test = train_test_split(training_set_size, X_all, y_all)

lr = LinearRegression(normalize=True)
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
print "mean_absolute_error: %f" %(mean_absolute_error(y_test, y_pred))
print_graph(X_all, X_test, y_all, y_test, y_pred)
```

Illustration Index

Figure 1.1 - Swiss Electricity Market in 2010 [2] 2
Figure 2.1: Wide Area Monitoring High Level Architecture [9] 10
Figure 2.2: Monthly End User Consumption [19] 11