Big Data, Smart Energy, and Predictive Analytics

Time Series Prediction of Smart Energy Data

Rosaria Silipo Rosaria.Silipo@knime.com
Phil Winters Phil.Winters@knime.com

Table of Contents

Summary

One of the key topics surrounding the concept of “big data” is the availability of massive time-based or telemetry data. With the appearance of low cost capture and storage devices, it has now become possible to get very detailed data to be used for further analysis. The very high detail resolution concerns mainly time. Nowadays, time streaming data can be recorded from almost any device, calling for interpretation to know more about the underlying system or to predict future events with higher accuracy.

This work focuses on smart energy data from the Irish Smart Energy Trials, where the electricity usage of circa 6000 households and businesses was monitored over time via meter IDs. The goal of the project was two-fold: create the ability to define custom tailored contract offers and on the other side to predict future electricity usage to shield the electricity companies from power shortage or power surplus.

The definition of 6000 (and many more in real life) customized contract offers is unrealistic. The goal then becomes to identify a few groups with common electricity behavior to make it worth the creation of customized contract offers. Therefore, the first step in this project, after importing, cleaning and transforming the data, was the definition of a few measures to describe each meter ID in terms of electricity usage behavior. Such measures were then used to group the original 6000 meter IDs into a maximum of 30 clusters including meter IDs with similar behavior in terms of electricity usage. The cluster’s average time series of electricity usage values was adopted as the cluster time series prototype.

As far as the second goal is concerned, i.e. the prediction of electricity consumption, a few options are available. The prediction could concern the total amount of energy usage; however, that might not be as accurate and might be too general to understand the underlying usage pattern. At the other end of the scale, the future energy usage could be predicted for each meter ID. This, though, might be overshooting the problem, because it uses an excessive amount of computational effort to reach a very difficult interpretation of the final results. As a compromise, we focused on predicting only the clusters’ prototype time series as partial values of the total electricity usage at some point in time. An auto-regressive model was adopted to predict each time series future based on its past.

This project also offered a “big data” opportunity as a side effect. Indeed, the energy usage of 6000 meter IDs, sampled every half an hour for longer than a year, produced a considerable amount of data, whose processing has taken quite a long time even on a dedicated and powerful machine, especially during the data transformation steps before clustering. To compare a classic approach with a big data approach, the first part of the analysis was run using also the big data engine available with KNIME.

The analysis described in this paper uses both publicly available data and the open source KNIME platform to transform the massive quantity of data, cluster the time series, apply time series analysis, and then draw both predictive analytics and business conclusions from the data. While the big data components are not open source, they are also available for a trial if required. All examples are available for download from the www.knime.com website.

Setting the Scene: Big data over Time

One of the key topics surrounding the concept of “big data” is the availability of massive time-based or telemetry data. With the appearance of low cost capture and storage devices, it has now become possible and easy to actually capture this detailed data for further analysis.

Generally, telemetry based data has a time-stamp element, which calls for the application of advanced predictive time series analysis techniques. Indeed, forecasting new values into the future alone usually produces an added commercial value to the business. However, with the right data analytics tool, it is also possible to combine time series forecasting with other predictive techniques, such as clustering or classification, generating even more insight about the data.

This "big data" opportunity exists in manufacturing, chemical and life science, transportation, automotive, energy, as well as in those industries where cyber security is an issue. This paper focuses on a smart energy example for the energy industry and is based on publicly available data and on the open source data analytics KNIME platform.

The energy industry is an industry currently going through change. With extremely complex networks, the opening up of competitors in what were earlier monopoly markets, the increased regulation of energy delivery and consumer pricing, and also the growing requirements for "green" and "safe" energy, the energy industry can not simply operate as of old.

One positive trend is the introduction of smart meters. Smart meters are a way for energy companies to both help its customers understand and manage their energy usage, but also a way for energy companies to better predict energy requirements down to at least the meter but in some cases down to the appliance or device level. The challenge is that each device can produce thousands if not millions of data per day. When multiplied by the number of smart meters being tracked, this can quickly enter what everyone would agree is a "big data" challenge. One of the main challenges for sharing techniques around data such as smart meter data is the lack of a publicly available source until now.

KNIME is an open source platform for data analytics, providing a user-friendly graphical workbench for the entire analysis process: data access, data transformation, initial investigation, powerful predictive analytics, visualization, and reporting. The open integration platform provides over 1000 modules (nodes), including those of the KNIME community and its extensive partner network.

KNIME can be downloaded onto the desktop and used free of charge. KNIME products include additional functionalities such as shared repositories, authentication, remote execution, scheduling, SOA integration, and a web user interface as well as world-class support. Robust big data extensions are available for distributed frameworks such as Hadoop. KNIME is used by over 3000 organizations in more than 60 countries.

The workflows developed in this project are based on KNIME and transform the original massive quantity of energy data, cluster the time series, apply time series analysis techniques, and then draw both predictive analytic, sensible big data processing, and business conclusions from the data.

The Irish Smart Energy Trials

In Ireland, the Commission for Energy Regulation (CER) initiated a Smart Metering Project in 2007 with the purpose of undertaking trials to assess the performance of Smart Meters, their impact on consumers' energy consumption, and the economic case for a wider national rollout. It was a collaborative energy industry-wide project managed by the CER and actively involving energy industry participants including the Sustainable Energy Authority of Ireland (SEAI), the Department of Communications, Energy and Natural Resources (DCENR), ESB Networks, Bord Gáis Networks, Electric Ireland, Bord Gáis Energy, and other energy suppliers. This group ran trials on both gas and electricity smart meters. For purposes of this white paper, we focused only on the electricity data.

The Smart Metering Electricity Customer Behaviour Trials (CBTs) took place during 2009 and 2010 with over 5,000 Irish homes and businesses participating. The purpose of the trials was to assess the

impact on consumer's electricity consumption in order to form the basis of a cost-benefit analysis for a national rollout. Electric Ireland residential and business customers and Bord Gáis Energy business customers, who participated in the trials, had an electricity smart meter installed in their homes/premises and agreed to take part in research to help establish how smart metering can help shape energy usage behaviours across a variety of demographics, lifestyles, and home sizes. The trials produced positive results, the reports for which are available from CER along with further information on the Smart Metering Project at
http://www.ucd.ie/issda/data/commissionforenergyregulation/ .

The detailed data underlying the electricity customer behaviour trial was made available in anonymised format, in order to facilitate further research and the development of competitive products and services, following the anticipated rollout of Smart Meters in Ireland. No personal or confidential information is contained in the data set. This is the data set we used for this project.

The data is from almost 6000 homes and businesses that participated in the trials - one smart meter per entity. The smart meter generated information every half hour, 24 hours a day, and there is one year of data available. In addition, pre-trial and post trial surveys were done for both households and businesses. All data is available in Excel or CSV format.

Note: While a small sample of the data is included in the downloaded examples to ensure the correct running of the workflows, anyone interested in further analysis on the entire dataset should request the data from the Irish Social Science Data Archive reported below:
http://www.ucd.ie/t4cms/CER_energy_electric_issda-data-request-form.docx

Overview of the Approach

The final goal of this work is to predict energy usage for specific groups (clusters) of meter IDs.

Thus, after reading, cleaning, and transforming the data, we need a clustering strategy to group together similar meter IDs. As usual, "similar" depends on the quantitative features used to describe the energy consumption and from the distance measure adopted in the clustering procedure. While many well established clustering algorithms and distance measures are available, the quantitative description of the input patterns is always the challenging part. Here, we were interested in the trend over time of electricity usage. Thus, we measured the amount of used energy at different times of the day and of the week.

After describing each meter ID time series by means of the appropriate quantitative measures and after grouping them together in a manageably small number of clusters, we moved to the prediction part. This is a relatively new area of analytics for any platform, including KNIME. Methods, like linear regression or neural networks, have already been available for quite some time. However, there has been only a short history of applying them to time series analysis. In particular, the prediction of one value at time `t` based on N past values at times `t-1,…, t-N` requires a specific data transformation as to have past and future on the same data row. Data mining modeling can then be applied to such an input data table to train a predictive model.

This is also a topic that lends itself to a big data implementation. Indeed, the time series of the half-hourly electricity usage, spanning a bit more than a year, for circa 6000 meter IDs consists of circa 176 millions rows. Aggregations, filtering, sorting, and measure calculations can take quite a long time even on a relatively well equipped laptop. Would that be a good chance then to use big data? For demonstrational purposes, we ran the first part of the analysis - the one where the descriptive measures of electricity usage are calculated - on a big data platform as well, to see how much in cases like this a big data implementation might speed things up.

Summarizing, we implemented the following steps:

1. Import the time series for each meter ID, clean it, and aggregate it at a daily or hourly level.
2. Define and calculate some behavioral measures of electricity usage for each meter ID on the imported time series.
3. Cluster all meter IDs on the basis of these behavioral measures, so reducing the analysis space from ca 6000 meter IDs to a maximum of 30 clusters.
4. Move on to time series prediction: for each cluster create a predictive model to know future energy usage based on past values.
5. Evaluate the predictive models, by measuring the prediction error for all values or only for energy peaks.
6. Re-implement and re-run the behavioral measures calculation at steps 1 and 2 using a big data platform.

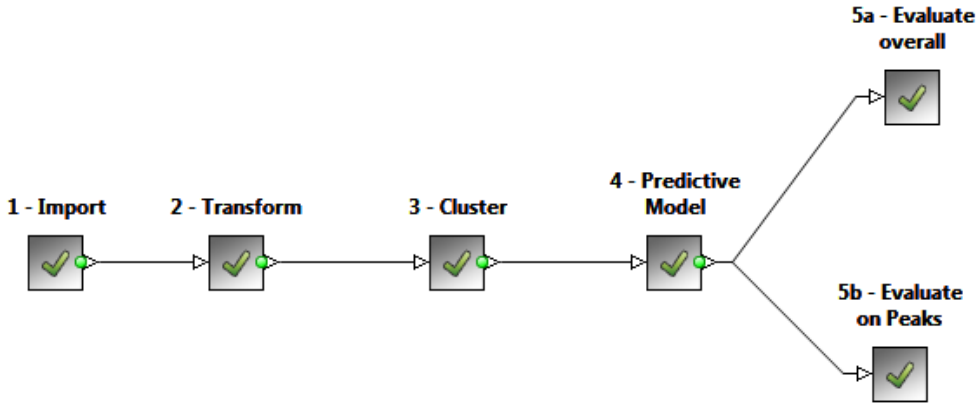

Figure 1: Overview of the Approach

In practice, we split the work over 4 workflows that can be downloaded together with this whitepaper from http://www.knime.com/white-papers .

1. The "PrepareData" workflow implements step 1 and 2; that is, it imports, cleans, transforms the data, and calculates some descriptive measures of electricity usage per meter ID.
2. The "k-Means" workflow groups together meter IDs with similar electricity usage behavior and calculates the prototype time series
3. The "AR - *" workflows and the "NN - *" workflow all implement predictive models. First the seasonality pattern is detected and then removed off the time series; then the number N of backward steps in time is selected; and finally a predictive model is trained using the N past samples at the net of the seasonality pattern.
4. Finally, the "PrepareData – DataRush" workflow re-implements exactly the same steps of the "PrepareData" workflow on the KNIME big data platform, provided by Actian (http://bigdata.pervasive.com/Products/Analytic-Engine-Actian-DataRush.aspx).

In the next sections, we describe each step and each workflow more in details, both in terms of implementation and results.

Import and Transform Data

Reading the Smart Meter Data

The smart meter data is contained in 6 well defined CSV files. There we find the Smart Meter ID, the data collection time (every half an hour), and the amount of electricity in KW used in the half an hour

previous to timestamp. There is no column name in the original files. The reading and transformation of the data is implemented in the "PrepareData" workflow.

Sub-workflow to read all or a subset of the data

After looping on the 6 files, concatenating all the data, and renaming the data columns appropriately, we obtain the data table reported in figure 2, where "Iteration" is the iteration number in the reading loop on the 6 files.

Row ID	Meter ID	datetime	KW/30	Iteration
Row0#0	1392	19503	0.14	0
Row1#0	1392	19504	0.138	0
Row2#0	1392	19505	0.14	0
Row3#0	1392	19506	0.145	0
Row4#0	1392	19507	0.145	0
Row5#0	1392	19501	0.157	0
Row6#0	1392	19502	0.144	0

Figure 2: Structure of the Smart Meter Data

The first three digits of column "datetime" count the number of days starting from 1 as January 1st 2009. The last 2 digits of "datetime" count the timestamp number updated every thirty minutes: that is 01 = 00:30 and 11 = 05:30.

After reading and concatenating the files, we had 176 million rows of smart meter data information. As it often happens in data analysis, we corrected and run the same data manipulation/data analysis a few times. Reading 176 millions rows every time would have slowed down our work and tried our patience considerably. So, we decided to introduce the option between reading a small subset of the data for debugging purposes and reading the whole data set looping across the 6 files.

This option has been implemented via an "IF Switch" block controlled via a flow variable. The flow variable was created and set in a "String Radio Button" quickform node assuming "partial" and "full" as possible values.

The switch node is fed with the list of file paths containing the smart meter data and produced by a "List Files" node. The flow variable value "partial" enables the bottom branch of the switch block to read only the first N rows of the first file of the list. On the opposite the flow variable value "full" enables the top branch of the switch block to loop on all 6 files, read their content, and concatenate the results in the "Loop End" node. This sub-workflow implemented to read the smart meter data is shown in figure 3.

Accessing the same data from the KNIME Server

The data used for this project is quite hard-disk consuming. Copying that from personal laptop to personal laptop is then not only time but also resource consuming. In cases like this, holding the data on a shared platform is advisable. In this particular case, we kept the data on a KNIME Server to be available to all KNIME users. Accessing the data from the KNIME Server rather than from the local file system required substituting the "List Files" node in figure 3 with the little sub-workflow shown in figure 4.

The "List Files" node indeed cannot access URLs, that is, it cannot access the KNIME Server URL either. To define the data file URL on the KNIME Server, we need an "Explorer Browser" node that explores all connected KNIME Servers and outputs the selected path using the "knime:" protocol. For example, to read our files from the KNIME Server named "KNIME05" and located in workflow group "Rosaria/timeseries", the URL "knime://KNIME05/Rosaria/timeseries/" was used.

Combining such URL with the list of file names allows to refer to the data file URLs on the shared KNIME Server and to feed the switch node with the same list of file paths as from the "List Files" node.

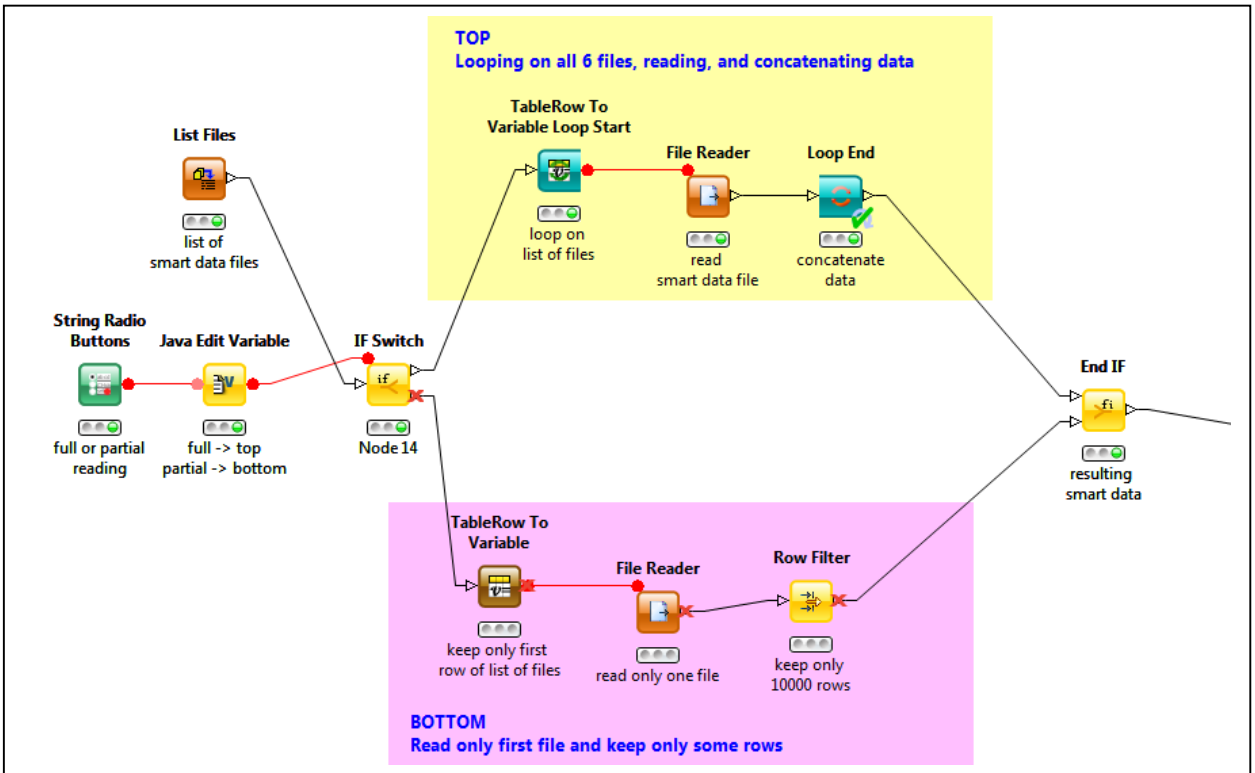

Figure 3: Sub-workflow to read the smart meter data

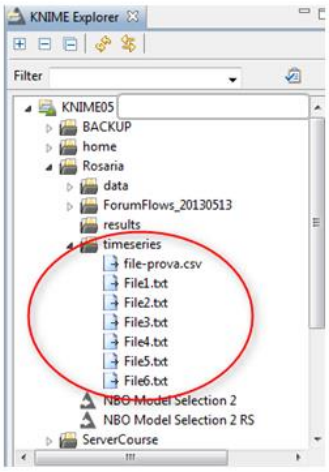

Figure 4a: The data on the KNIME server

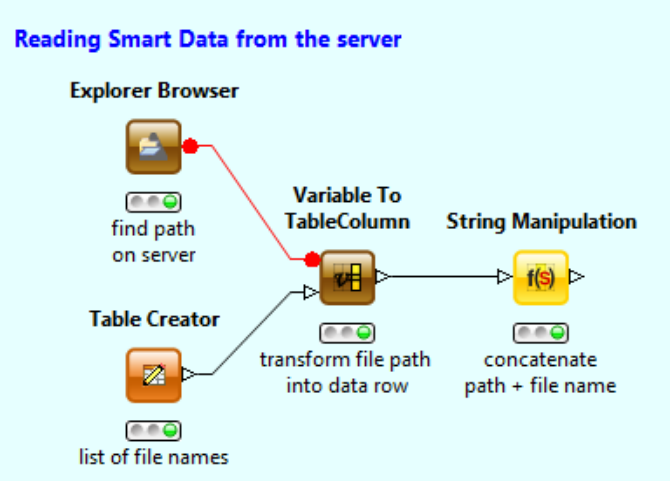

Figure 4b: Sub-workflow to read the smart data from the server

The sub-workflows shown in figures 3, 4a, and 4b were collapsed into the first metanode of the "PrepareData" workflow, named "Read all data".

Transforming the Smart Meter Data

The files contain a proprietary date/time format, as described in the previous section. The first three digits of column "datetime" count the number of days starting from 1 as January 1st 2009. The last 2

digits of “datetime” count the timestamp number updated every thirty minutes: that is 01 = 00:30 and 11 = 05:30.

Converting proprietary format datetime into Date and Time objects

The first step in transforming the data is then to convert the “datetime” data column to a standard datetime format. First the “datetime” string is split into the time and the date part. Then, from the time part, two “Math Formula” nodes extract respectively the hours and the minutes and a “Java Snippet” node concatenate the two values together to reach the “hh:mm” format.

A second “Java Snippet” node uses the Java Date library to convert the date part of the “datetime” data column into a Date object, starting from January 1st 2009, with format “dd.MM.yyyy”. The configuration window of this “Java Snippet” node, including the java code used for the date conversion, is reported in figure 5.

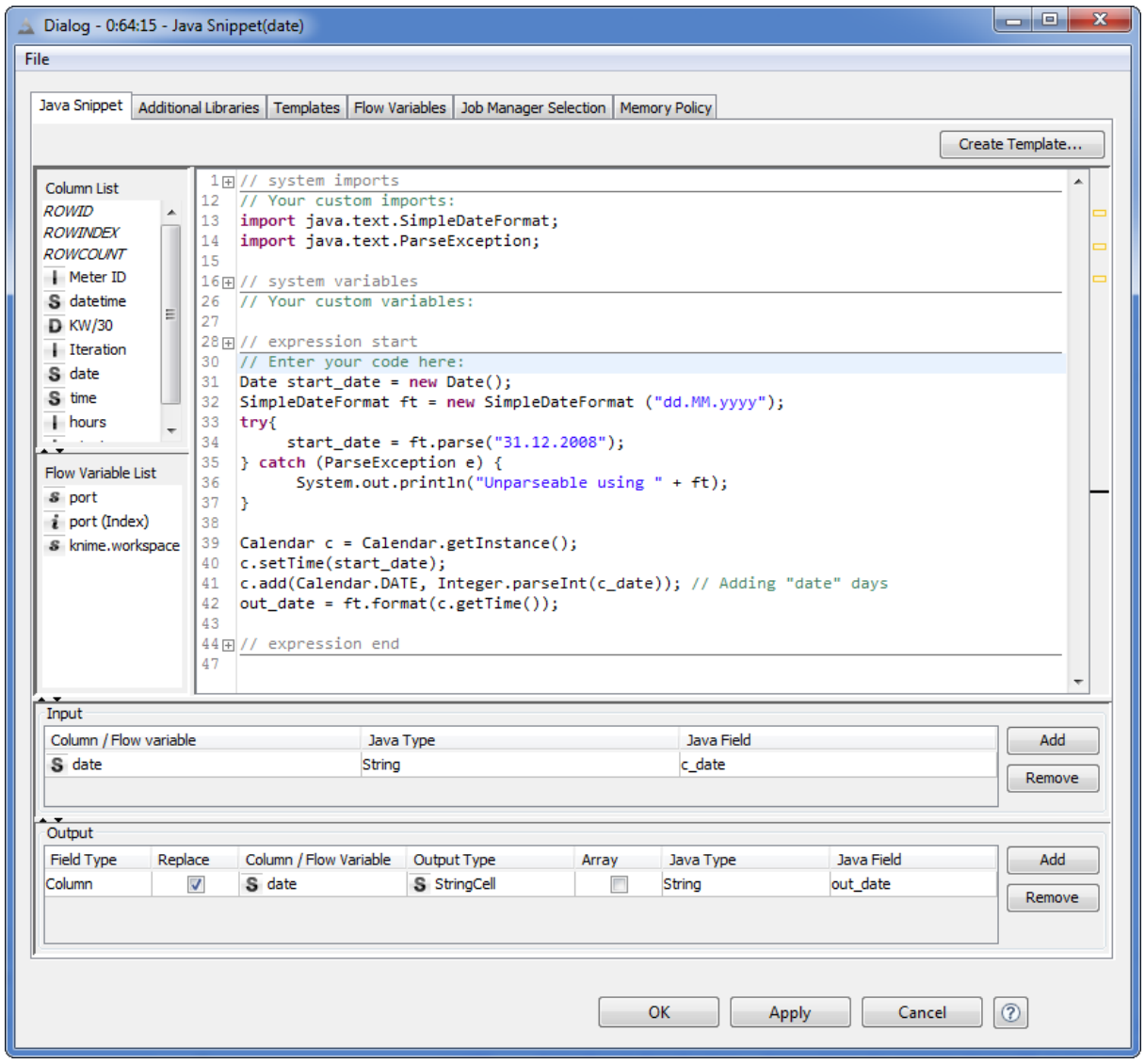

Figure 5: “Java Snippet” node to convert the progressive number of days into a date value formatted as “dd.MM.yyyy”

After this transformation, two new data columns appear: one named “date” and one named “time”, containing respectively the date and time of the smart meter measure. Finally, all data rows are

sorted by meter ID, date, and time, creating the time series of the used energy for each half an hour ("KW/30") for all meter IDs.

This is, so far, the minimum transformation required, affecting only the datetime format and sorting the resulting rows. This transformation can be found in the metanode named "String to datetime".

Aggregating at different time scales

At this point, we still needed to transform the data, to create some descriptive measures of the smart meters electrical behavior and of the times series values to use for prediction.

In order to exhaustively describe the energy usage of each meter ID, we would like to produce an energy measure on many different time scales: hourly, monthly, daily, yearly, etc … So, the next step in data transformation is the extraction of time key metrics such as day, day of week, hour, month, year, and so on. The "Time Field Extractor" and the "Date Field Extractor" nodes have been used to perform these operations on time and on date respectively (Fig. 6 and 7).

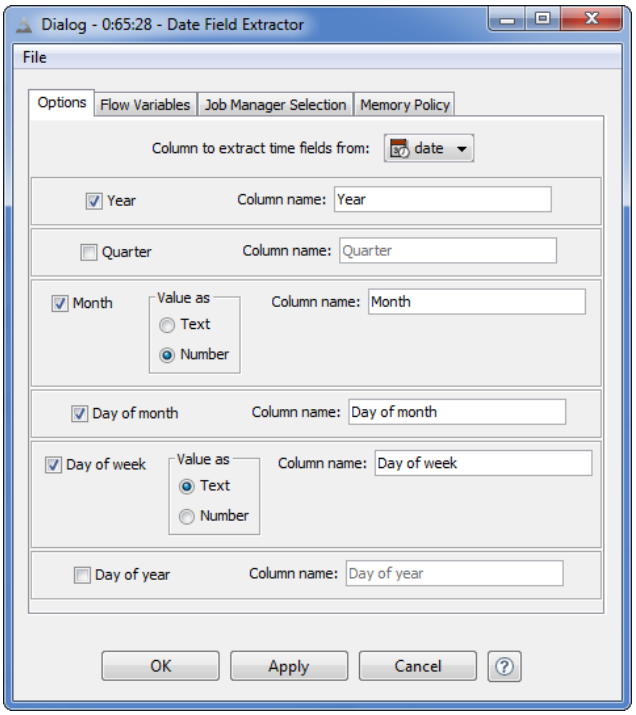

Figure 6: Configuration window of the "Date Field Extractor" node, set to get year, month, day of month, and day of week from the data column "date".

Now, having produced all the necessary time scales, we aggregated the energy values ("KW/30") by hour, day, week, month, and year. This produced new time series with the energy used each hour, day, month, year, etc … by each meter ID. These time series can be used to predict the energy consumption for that meter ID for the next hour/day/month given the energy used in the previous N hours/days/months.

At the same time, we calculated some average measures to quantify how much energy each meter ID uses per hour, per day, per month, per year in average. The overall average electricity usage by day, week, month, and year was then calculated for each meter ID.

Both operations are implemented in the "Daily, Monthly, Yearly, Weekly" and "Hourly, Intra-Day" metanodes. At the end of all these parallel transformations (segmentation of time series and calculation of average energy values at different time scales), the resulting average values are joined together.

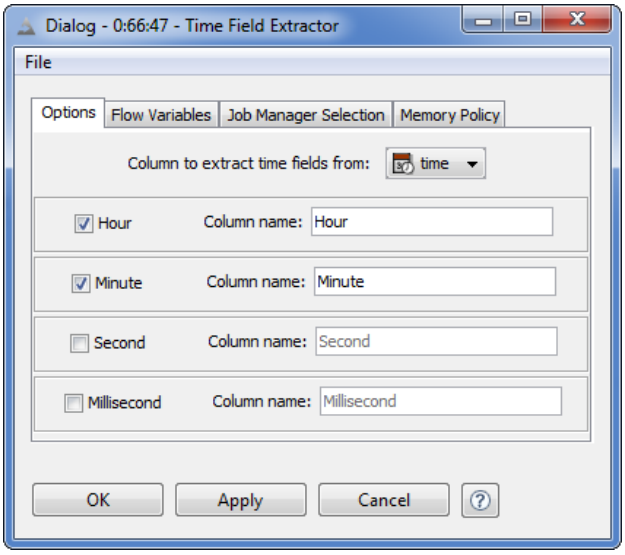

Figure 7: Configuration window of the “Time Field Extractor” node set to get hours and minutes from the data column ”time”.

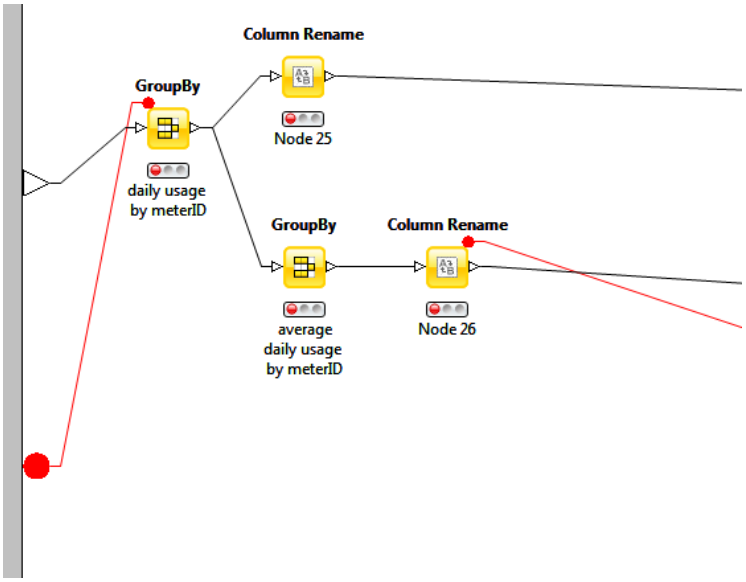

Figure 8: This sub-workflow creates the time series with the daily electricity usage values and their average daily value (in “Daily, Monthly, Yearly, Weekly” metanode). The time series with the daily values can be used for prediction, while the average of the daily values can be used as a behavioral measure to cluster the meter IDs.

However, even if we know that a meter ID “x” uses an average energy amount “y” per week, this still does not tell us how this energy is distributed across the week; for example, some meter IDs might show a difference between business days and week end days in energy usage. The same is true for the average daily value of energy consumption: the average daily value does not tell us how the energy usage is distributed across the daily 24 hours. Some meter IDs might use more energy at night, while others might show a more clear business hours trend.

Another descriptive measure can be the percentage of energy usage in different time slices of the day with respect to the energy used in the entire day (Fig. 9). Similarly, the percentage of energy usage at different week days with respect to the energy usage during the whole week can represent another behavioral measure. One additional meta-node, named “% values”, calculates the intra-day and intra-week percents of kW usage by meter ID (Fig. 11).

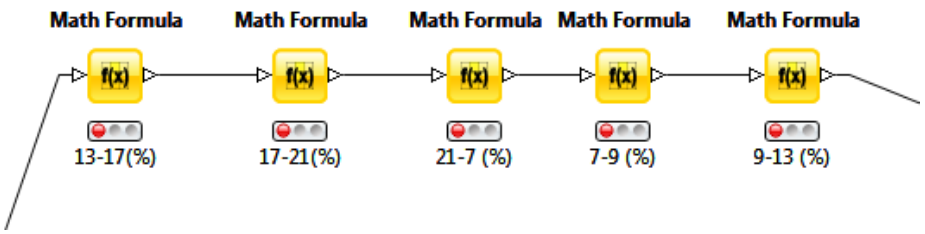

Figure 9: This sub-workflow, contained in the "% values" metanode, calculates the percentage of the daily electricity usage in the different day slices: early morning, morning, afternoon, late afternoon, and night.

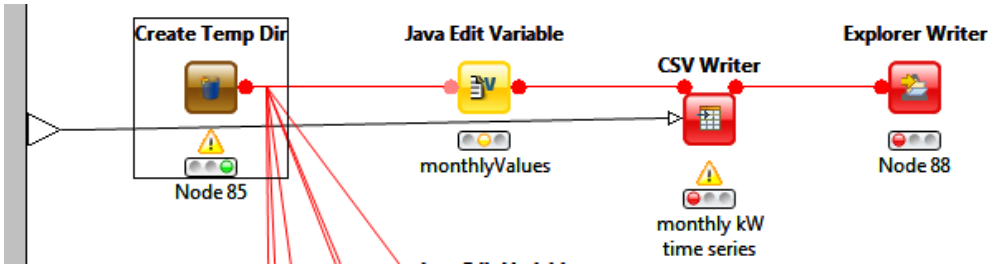

Figure 10: Sub-workflow to write resulting data to a temporary directory first and to the final folder on the KNIME server afterwards ("Write to Server" metanode).

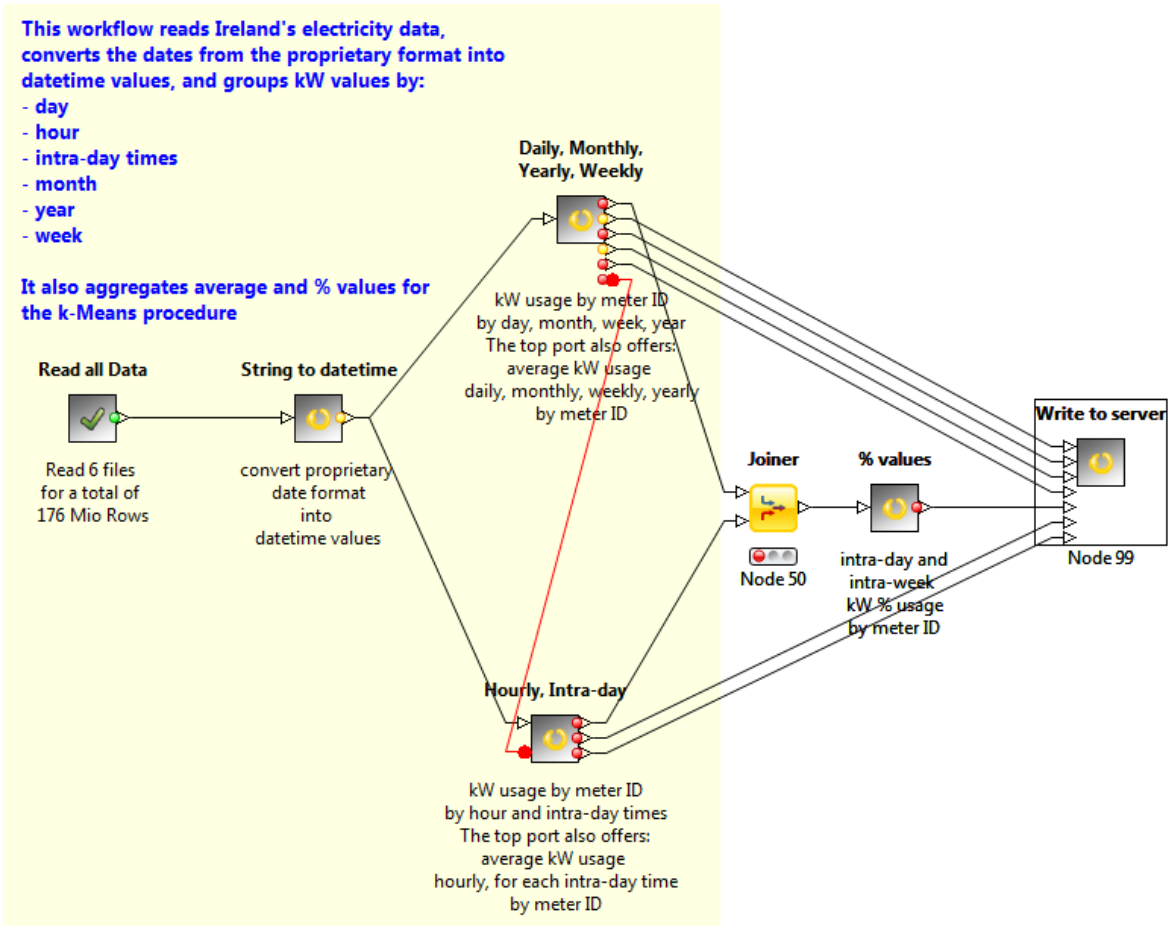

Figure 11: The complete workflow "PrepareData"

Allthough not required, all resulting data tables are written out as CSV files to facilitate further learning and exploration. Even here, data can be written to the KNIME Server using the KNIME URL

protocol. In this case, we write the data to a local temporary directory using the "Create Temp" node, and afterwards we use the "Explorer Writer" node to transfer this local temporary file to a URL location on the KNIME Server (Fig. 10). This sub-workflow is collapsed into the "Write to Server" meta-node.

The final "PrepareData" workflow, implementing the data reading and the described transformations, is reported in figure 11.

Clustering Meter IDs with similar Behavior

One of the goals of this project was to define customized contract offers based on different behaviors in electricity usage. However, the definition of 6000 (and many more in real life) customized contract offers is unrealistic. The goal, then, was changed into the identification of a few groups with some particular electricity usage behavior to make it worth it the tailoring of a customized contract offer.

After importing, cleaning and transforming the data, a few average and percentage measures were defined to describe each meter ID in terms of electricity usage behavior (see previous sections). Such measures were then used to group the original 6000 meter IDs into a maximum of 30 clusters, each one including all those meter IDs that behave similarly in terms of electricity usage.

The k-Means algorithm

To group together meter IDs based on similar behaviors, the k-Means algorithm was used. The k-Means algorithm is by now a very well established clustering technique and KNIME has a k-Means node set which is easy to train and to apply.

Feature Selection and Normalization

Of the created input features only some have been fed into the k-Means algorithm. Indeed, the goal was to cluster together meter IDs on the basis of average values and daily and weekly distribution values of the electricity usage. The features available were:

- Average and percentage values of used energy on each week day from Monday through Sunday
- Average and percentage values of energy used during five different day sements: early morning (7h-9h), morning (9h-13h), early afternoon (13h-17h), late afternoon (17h-21h), night (21h-7h)
- The total energy used over the test time in kW
- The average yearly, monthly, weekly, daily, and hourly used energy in kW
- The average energy used over the weekend (WE) and over business days (BD)

In general, if we used a percentage value, we excluded the corresponding average value. For example, if the percentage of electricity usage on Mondays has been fed into the k-Means algorithm, the average electricity usage on Mondays has not been used. This is to avoid that the Monday values influence the clusterization process more than the other variables. The final set of features for the k-Means algorithm excluded the average values of energy used during week days and the average values of energy used at different times during the day. All other features have been kept for the clusterization process.

Since the k-Means algoritm is a distance based technique, all input features need comparable ranges: that is, they need to be normalized. A "Normalizer" node was introduced and a "min-max Normalization" into the [0,1] interval was applied on all data columns.

k-Means operates on numerical values. Possible nominal values need to be discretized if to be used in a k-Means algorithm. This is not our case, since the data set we are working on is just numerical. Also, k-Means cannot work on missing values. We changed the missing numerical values, if any, into 0. It is highly unlikely that a meter ID has a 0 energy consumption. So, the 0 value can be used to represent missing values. The risk here is that low energy usage households can be grouped together with other entities with many missing values.

k-Means Settings

After normalization, the k-Means node could be applied with the configuration settings reported in figure 12.

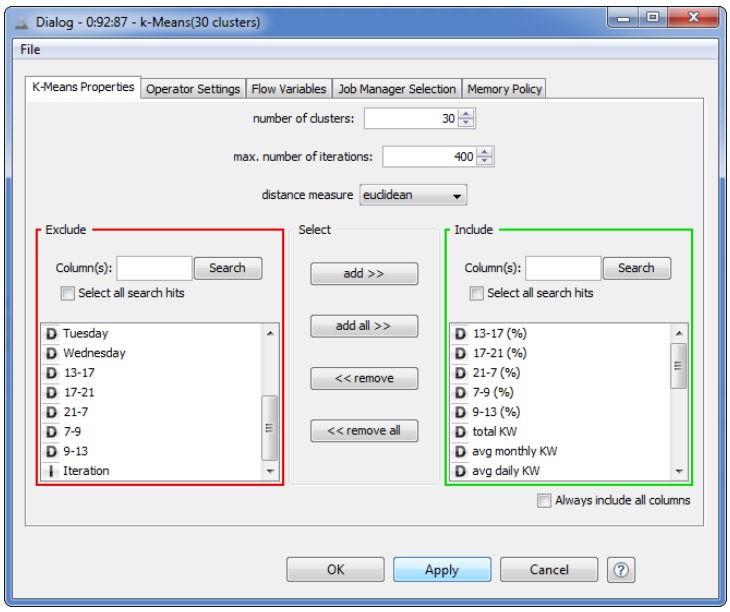

Figure 12. Configuration settings of the “k-Means” node

A maximum of 30 clusters was set. Indeed, more than 30 profiles would make the number of customized contracts unmanageable. On the other side, hopefully, “30” is a high enough number to capture most similarities among meter IDs electricity behaviors.

The distance chosen was the Euclidean distance which produces round clusters with patterns equally spaced from the cluster center.

The maximum number of iterations was fixed to 400. This is a stop criterion just in case the algorithm does not converge in a reasonable time.

After execution, if we open the resulting clustering model (right-click k-Means node and select “View: Cluster View”), we can see the 30 clusters that were created (Fig. 13): some of them cover a large number of meter IDs, some only a few. Probably the large clusters are the most interesting ones, since they group a large number of meter IDs with similar properties.

Each cluster is represented by a prototype and a prototype consists of the average features calculated across all meter IDs assigned to that cluster. After applying a k-Means algorithm, the cluster view of the k-Means node shows the clusters in the normalized feature space. It is always useful to extract the prototypes and denormalize their features back into the original ranges in order to investigate the detected data groups.

The “k-Means” node can be found in the “first n clusters” metanode in the “k-Means” workflow.

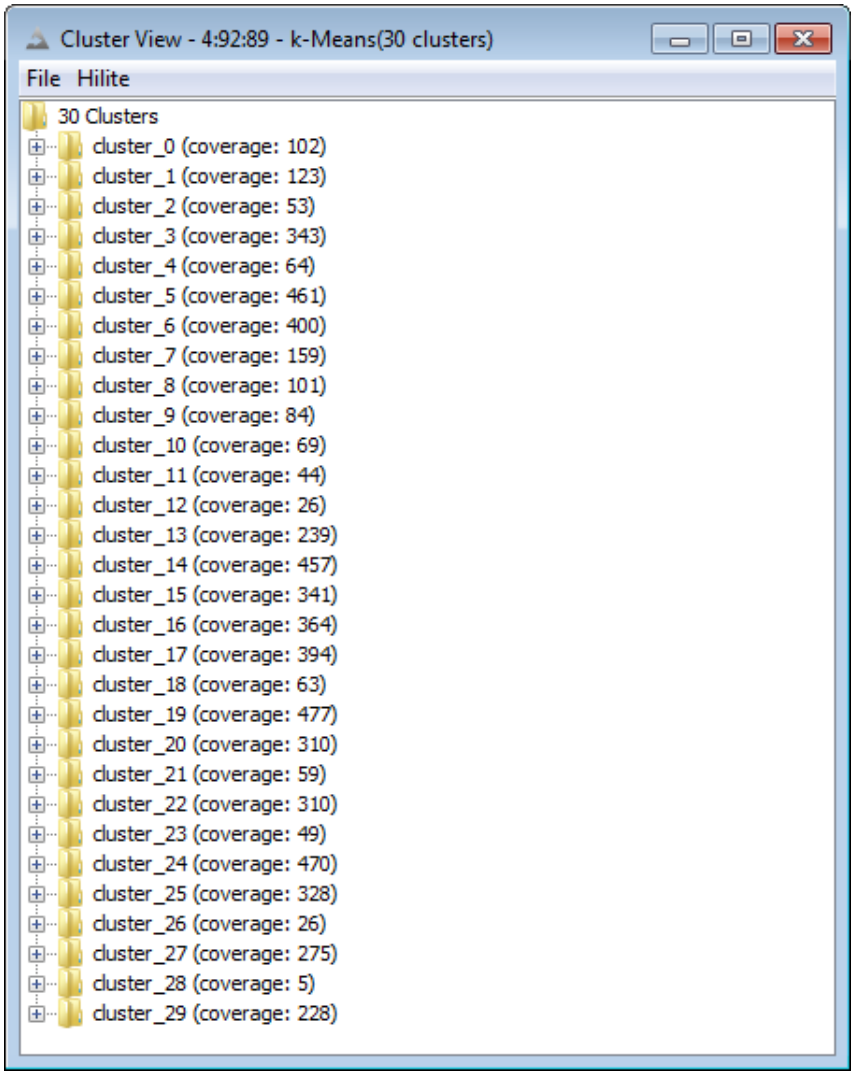

Figure 13. The clusters resulting from running the k-Means algorithm

k-Means Prototypes

In order to extract the prototypes features, we relied on the "XML" extension of the KNIME platform. The XML extension includes a number of nodes and a new data type to deal with XML structures. The new data type is an XML cell, containing a full XML structure. The XML nodes allow to read, write, convert, combine, and of course parse XML data columns.

In addition, all data mining models produced by KNIME nodes are exported in PMML format, which is a variation of the XML format. Besides the XML extension, KNIME offers also a number of PMML related nodes, to convert, read, write, and manipulate PMML structures. Since the cluster model, like all other models, is exported in PMML format - that is in XML format - we used a combination of the PMML and XML nodes to extract the prototype features from the cluster model.

First, a "PMML to Cell" node imports the cluster model into a data table cell. Subsequently a number of "XPath" nodes, from the XML category, extract cluster name, size, and prototype feature values from the model. The feature values of the cluster prototypes are then denormalized via the "Denormalizer" node to go back to their original range values.

The whole sub-workflow contained in the “first n clusters” metanode and implementing the k-Means algorithm, including normalization, denormalization, and prototype feature extraction, is shown in figure 14.

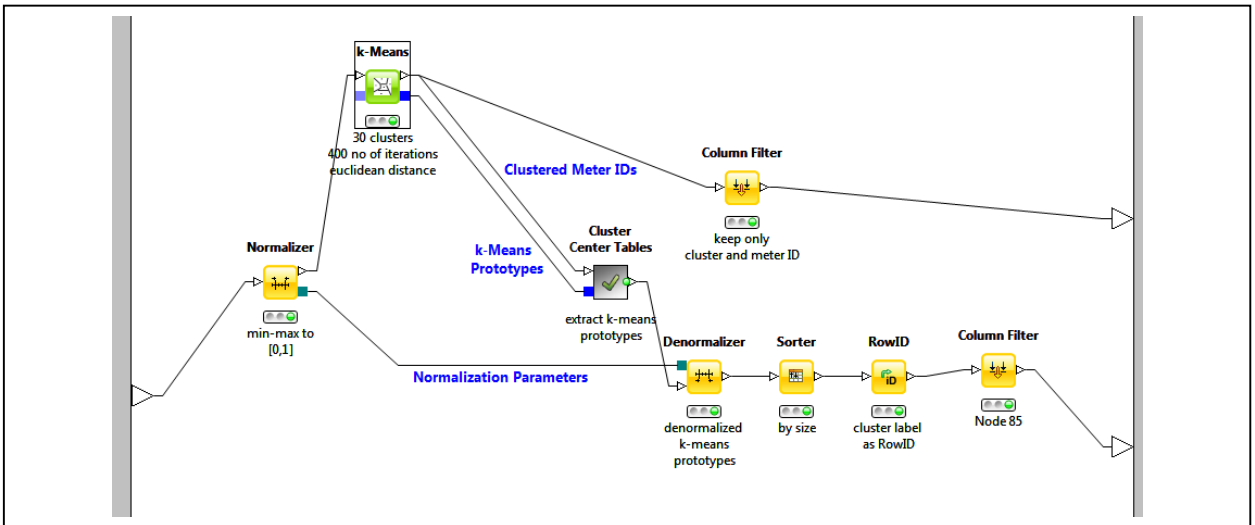

Figure 14: Sub-workflow to cluster the meter IDs via k-Means in the “first n clusters” metanode

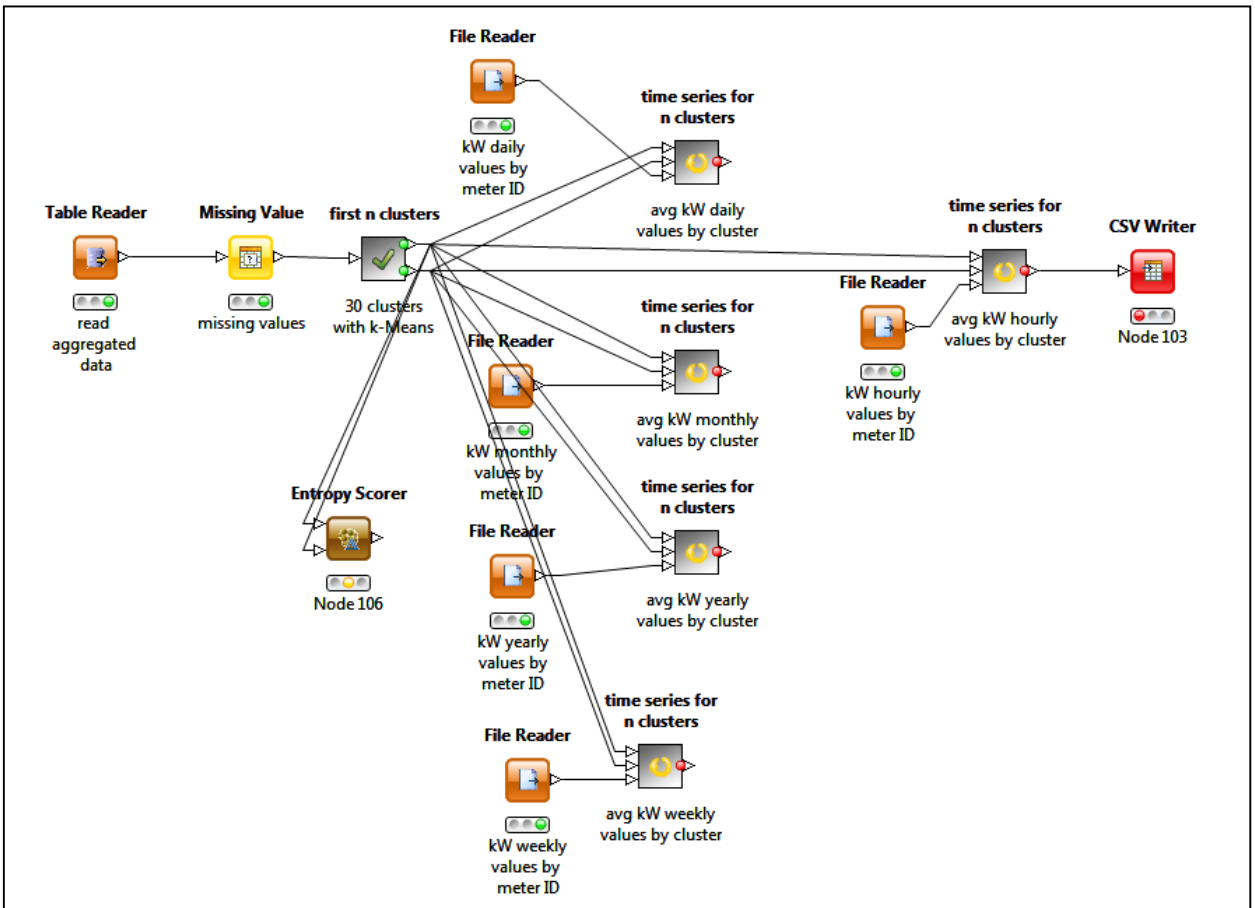

Figure 15: Workflow “k-Means”: running k-Means and creating prototype time series for each cluster

The final goal is to predict future energy usage based on past energy usage. However, we do not want to implement a predictive model down to the meter ID level. It would be enough to detect temporal trends in some of the major clusters detected with the k-Means algorithm. For each cluster, then, the average time series was calculated and used as the cluster representative time series, the

cluster time series prototype. The final "k-Means" workflow, including the k-Means cluster detection and time series prototype generation, is shown in figure 15.

Cluster Learnings

Some interesting clusters emerge from the analysis described in the previous section with 4 main groups, as shown in the following tables.

The Night Owls

The night owls use electricity mainly at night and just a little during the day. Meters in cluster 1, in particular, spend more than half of their (high) energy during the night. From cluster 19 to cluster 8, the percentage of electricity usage decreases at night to become higher in other segments of the 24 hours.

A low average energy usage per day and per hour indicates a household or a very small business. On the opposite a high average energy usage per hour and per day indicates a larger business. The higher the average used energy the larger the business. Clusters in this group have a various composition: some consist of many private, late evening active households, probably with teenagers or students (cluster 19 to 8); cluster 1, on the opposite, covers a lower number of meter IDs all identifiable as medium sized night active businesses.

	Avg daily kW	Avg hourly kW	21-07	07-09	09-13	13-17	17-21	size
Cluster 1	77	3	**65%**	9%	10%	9%	7%	53
Cluster 19	21	0.9	**50%**	6%	12%	12%	19%	177
Cluster 25	28	1.2	**41%**	5%	13%	17%	25%	424
Cluster 13	23	1.0	**41%**	5%	9%	11%	32%	96
Cluster 8	20	0.8	**40%**	7%	17%	17%	19%	364

Table 1. Average and percentage values of electricity usage over the 24h for the "night owls" clusters

	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Avg daily KW	Avg hourly kW	size
Cluster 1	15%	16%	16%	16%	15%	11%	11%	77	3	53
Cluster 19	14%	14%	14%	14%	14%	15%	15%	21	0.9	177
Cluster 25	14%	14%	14%	14%	14%	15%	15%	28	1.2	424
Cluster 13	15%	15%	15%	15%	14%	13%	15%	23	1.0	96
Cluster 8	14%	14%	14%	14%	14%	14%	14%	20	0.8	364

Table 2. Average and percentage values of electricity usage over the week days for the "night owls" clusters

The hypothesis of businesses (cluster 1) vs. households (other clusters) is confirmed by the percentages of energy usage per week day. Cluster 1 is mainly active during business days and consumes less energy over the weekend. On the opposite, the other cluters do not really show a lot of difference amongst week days: every day they use more or less the same amount of electricity. If small businesses are included in cluster 19 to cluster 8, those must be businesses working 7 days/week.

Clusters in this group also show a different daily rhythm. Cluster 1 is really active only at night, cluster 19 and cluster 13 extend their late evening activity into the late night, and cluster 25 and cluster 8 show some activity during the day as well. These different daily rhythms probably reflect the household composition.

Notice that cluster 13 has a low usage of energy on Saturdays. The difference is very small and might be not significant. However, it is a fact that might deserve a little deeper investigation.

The Late Evening Clusters

A second group of clusters consists of "late evening" smart meters. These clusters contain meter IDs that are considerably active in the late afternoon, after working hours. Some of these clusters extend their electrical activity into the night (clusters 3 and 4).

This group seems to include only households (low average daily and hourly electricity usage). Only clusters 17, 9, and 29 show a greater energy usage during the day. All other clusters seem to activate only after 17:00. Table 4 with the percentages of used energy over the week days confirms the hypothesis that all those meter IDs refer to households. Indeed there is no difference in used energy among weekdays and just a little increase over the weekend.

	Avg daily kW	Avg hourly kW	21-07	07-09	09-13	13-17	17-21	size
Cluster 6	21	0.9	28%	5%	13%	17%	**37%**	178
Cluster 20	24	1.0	29%	6%	17%	16%	**31%**	356
Cluster 4	26	1.1	**35%**	6%	13%	15%	**31%**	360
Cluster 17	25	1.0	24%	4%	18%	23%	**31%**	255
Cluster 0	21	0.9	29%	18%	14%	14%	**30%**	137
Cluster 9	28	1.2	32%	4%	14%	21%	**28%**	443
Cluster 22	25	1.0	27%	7%	21%	19%	**27%**	417
Cluster 7	32	1.3	29%	8%	17%	19%	**26%**	370
Cluster 3	24	1.0	**35%**	6%	17%	16%	**26%**	533
Cluster 29	26	1.1	26%	13%	19%	28%	**24%**	167

Table 3. Average and percentage values of electricity usage over the 24h for "late evening" clusters

Finally, this group includes only very general clusters, with at least 100 meter IDs each. Households represent probably the most frequent type of energy entity associated with a meter ID. Clusters grouping households with similar electrical usage have to include a high number of meter IDs.

Comparing with clusters in the previous group, it is interesting to notice that while businesses seem to reduce, though slightly, the electricity usage over the weekend, most households seem to increase electricity usage, even if just a bit, over the week end.

	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Avg daily KW	Avg hourly kW	size
Cluster 6	14%	14%	14%	14%	14%	15%	15%	21	0.9	178
Cluster 20	14%	14%	14%	14%	14%	15%	15%	24	1.0	356
Cluster 4	14%	14%	13%	13%	13%	16%	16%	26	1.1	360
Cluster 17	14%	14%	14%	14%	14%	15%	15%	25	1.0	255
Cluster 0	14%	14%	14%	14%	14%	14%	15%	21	0.9	137
Cluster 9	14%	14%	14%	14%	14%	15%	15%	28	1.2	443
Cluster 22	14%	14%	14%	14%	14%	14%	15%	25	1.0	417
Cluster 7	14%	14%	14%	14%	14%	15%	15%	32	1.3	370
Cluster 3	14%	14%	14%	14%	14%	15%	15%	24	1.0	533
Cluster 29	14%	14%	14%	15%	14%	14%	14%	26	1.1	167

Table 4. Averages and percentages of electricity usage over week days for “late evening” clusters

All Rounders

This third group is made of clusters with meter IDs with a more equally distributed usage of electricity throughout the 24 hours. Smart meters in this group use more energy during the day if compared to smart meters in the previous two groups. Only a few clusters (clusters 14 and 11) use as much energy at night as during the day. All other clusters show a clear higher activity during the day than at night. Some use more energy in the late evening (cluster 14), some more during the day (cluster 10), some even in the early morning (cluster 11), and some in every segment of the day indistintly.

	Avg daily kW	Avg hourly kW	21-07	07-09	09-13	13-17	17-21	size
Cluster 14	23	1.0	**39%**	10%	15%	14%	23%	287
Cluster 16	22	0.9	**26%**	6%	21%	18%	23%	178
Cluster 15	29	1.2	**33%**	5%	18%	21%	23%	497
Cluster 24	18	0.8	**38%**	6%	17%	18%	22%	43
Cluster 12	**297**	12.4	**35%**	7%	18%	20%	20%	60
Cluster 10	21	0.9	**31%**	7%	24%	18%	20%	246
Cluster 11	23	1.0	**29%**	26%	16%	**10%**	19%	15
Cluster 5	**130**	5.4	**38%**	7%	18%	18%	19%	97

Table 5. Average and percentage values of electricity usage over the 24h for “all rounders” clusters

Considering the average energy usage per day and per hour and the energy usage temporal pattern, these clusters are certainly covering large (clusters 5 and 12) businesses active during the day and still running machines at night as well as households (all other clusters) with household members with different electrical habits. Cluster 12, for example, contains very large businesses, with more than 500 kW used in average per day. The other clusters more likely represent private households.

In this group we have big and small clusters. Cluster 11 for example, the one with the early morning people, contains only 15 meter IDs. On the opposite, cluster 15 with energy distributed equally across the 24h collects more than 400 meter IDs.

Considering now the distribution of electricity usage over the week, we see again that only businesses show a considerable reduction of energy usage over the weekend. Cluster 24, on the opposite, shows a very low usage of energy during business days to increase it considerably over the weekend.

	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Avg daily KW	Avg hourly kW	size
Cluster 14	14%	14%	14%	14%	14%	15%	15%	23	1.0	287
Cluster 16	14%	14%	14%	14%	14%	15%	15%	22	0.9	178
Cluster 15	14%	14%	14%	14%	14%	15%	15%	29	1.2	497
Cluster 24	11%	11%	11%	13%	17%	23%	20%	18	0.8	43
Cluster 12	15%	15%	15%	15%	15%	13%	12%	**297**	12.4	60
Cluster 10	14%	14%	14%	14%	14%	14%	14%	21	0.9	246
Cluster 11	15%	15%	14%	15%	15%	15%	15%	23	1.0	15
Cluster 5	14%	14%	15%	15%	15%	14%	13%	**130**	5.4	97

Table 6. Average and percentage values of electricity usage over week days for “all rounders” clusters

Daily Users

This last group contains meter IDs with a high daily usage of electricity, both in the morning and afternoon, and a reduced usage of electricity at night.

	Avg daily kW	Avg hourly kW	21-07	07-09	09-13	13-17	17-21	size
Cluster 28	**584**	24.4	**28%**	6%	23%	24%	19%	29
Cluster 18	51	2.1	22%	5%	**29%**	**28%**	15%	135
Cluster 27	25	1.0	22%	5%	26%	24%	23%	188
Cluster 26	**173**	7.2	22%	6%	**29%**	**28%**	17%	83
Cluster 2	23	1.0	21%	7%	**35%**	18%	19%	69
Cluster 23	51	2.1	**8%**	4%	**40%**	**37%**	11%	77
Cluster 21	53	2.2	**8%**	2%	**34%**	**40%**	17%	89

Table 7. Average and percentage values of electricity usage over the 24h for “daily users” clusters

Besides a day vs. night pattern, most smart meters in this group show a clear business days vs. week end pattern. For those clusters, the used energy is concentrated during the business days, while it is heavily reduced during the weekend. Cluster 2 and cluster 8 are the exceptions to this trend. By now, we have learned that inflection in used energy over the weekend describes a business more than a household. While the high average used kW per hour and per day of clusters 28 and 26 clearly indicates large businesses, the relatively low average used energy per hour and per day of clusters

18, 21, and 23 indicates small businesses meter IDs, active during business hours from Monday to Friday and closed at night and during weekends.

	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Avg daily KW	Avg hourly kW	size
Cluster 28	15%	15%	15%	15%	15%	14%	12%	**584**	24.4	29
Cluster 18	16%	18%	18%	17%	16%	**8%**	**7%**	51	2.1	135
Cluster 27	14%	14%	14%	14%	14%	14%	14%	25	1.0	188
Cluster 26	16%	17%	17%	17%	16%	**11%**	**7%**	**173**	7.2	83
Cluster 2	14%	14%	14%	14%	14%	14%	14%	23	1.0	69
Cluster 23	18%	21%	20%	19%	17%	**5%**	**3%**	51	2.1	77
Cluster 21	12%	17%	16%	18%	18%	**17%**	**4%**	53	2.2	89

Table 8. Average and percentage values of electricity usage over week days for “daily users” clusters

Cluster Conclusions

What is fascinating is that, without any a priori knowledge, we are able to categorize types of users based on energy usage across the week and day as well as their average daily and hourly KW usage.

In general, the low average used energy indicates a household rather than a business. Households have different hbits: from late evening energy usage to early morning and daily energy usage, probably depending on the household members habits. A household with students or teenagers would probably belong to the “nightly owls” group, a household with more mature working people would take us into the “daily users” group, and a household with mixed occupants would probably end in the “all rounders” group.

A sudden drop in energy usage at night and especially over the weekends indicates a business. The amount of average daily and hourly energy is a strong indicator of the business size: the bigger the average used electricity the bigger the size of the business. Business clusters are found in all groups, besides the “late evening” group.

Time Series Forecasting

Time series forecasting in the energy industry is an extremely important asset. It is used hourly to forecast peaks in demand and redundancy in production. It is used daily to forecast energy usage and therefore optimize scheduling and allocation. It is used weekly to customize energy purchase policies and to tweek maintenance routines. It is used on a monthly and yearly basis to drive production, network balancing, and strategic planning.

Bunn and Farmer’s report shows that a 1% increase in forecasting accuracy could yield estimated savings in operational costs of approximately 10 million pounds. Exactly to get that 1% improvement or more in forecasting the total energy usage, in this section we start a journey into time series prediction by investigating a few different models, an auto-regression and a neural network model, different seasonality indeces, and trying to identify the best time lag for prediction.

In order to predict the global value of energy usage at time t in Ireland, we have many options. We can work on the one time series of total energy usage. Alternatively, we can fragment our prediction problem into many smaller problems, like predicting each single meter ID value of energy usage at time t. One approach might be too approximate, since predicting the energy usage for the whole

nation might be a too complex problem. The other approach though might be too detailed. In fact, while it is likely that predicting single time series is easier, predicting millions of time series at the same time might be overkilling the problem with an excessive amount of computational power.

From the k-Means algorithm applied in the previous section, however, we have a few clustered time series available, representing meter IDs with similar behavior. A good compromise between the two approaches above then could be the prediction of the clustered time series: the forecasting problem is broken into 30 easier sub-problems and forecasting 30 time series is not as computationally expensive as forecasting millions of them at the same time. Forecasting a time series representing a cluster of homogeneous meter IDs is probably easier than forecasting the total time series coming from all possible and dishomogeneous meter IDs in Ireland. At the same time, predicting the clustered time series is hopefully almost as accurate as predicting each time series by itself, since the clustered time series is the result of many time series with similar trends.

Simple Auto-Regressive Model (no seasonality adjustement)

The first and easiest algorithm available for time series prediction is the auto-regressive model. The auto-regressive model uses the past of the same time series to predict its future values: this is the "auto-" part of the model. That is, given a time series x(t), we use `x(t-1)`, `x(t-2)`, …, `x(t-N)` to predict `x(t)`. In a simple autoregressive model, we assume no seosonality and the time series to be stationary. Let's see now how we implemented an auto-regressive model with KNIME.

Reading Data and Clustered Time Series Selection

First, we accessed the file with the clustered time series via a classic "File Reader" node.

Secondly, a "Sorter" node sorted the rows by ascending timestamp value. This is to ensure the time series character with a meaningful time order.

Since we wanted to work on each clustered time series seperately, we used a "Column Filter Quickform" node to select just one clustered time series for transformation and further investigation.

The Quickform node extracts the column headers of the one or more selected data columns from the input data, though producing an empty data table. For the real column filtering, we then used a "Concatenate" node with the "Use Intersection of Columns" option enabled.

Finally, to be completely independent in future operations from the selected data column name, a "Column Rename (Regex)" node was introduced to change the current name of the data column into a more general "cluster". A "Column Rename (Regex)" node was chosen, because the search string identifying the column name to be replaced can be controlled via a flow variable, in this case the flow variable output by the Quickform node and containing the selected column original name.

All nodes were then encapsulated into a metanode named "PrepareData". Metanodes are usually not configurable, besides when they host a Quickform node. In this case, the Quickform GUI is passes into the metanode configuration window. Figure 16 shows the metanode sub-workflow and configuration window to select one or more of the available clustered time series. This meta-node was reused in all the following more refined prediction models implemented for this project.

The Lag Column Node

Once the time series had been selected, sorted, and clean, we put it's past and future on a single row in order to train the subsequent model. A new node has been made available in KNIME 2.8 to shift columns up and down: the "Lag Column" node. We used this delay node to shift and lag the data column, effectively pulling up the previous records into the current record.

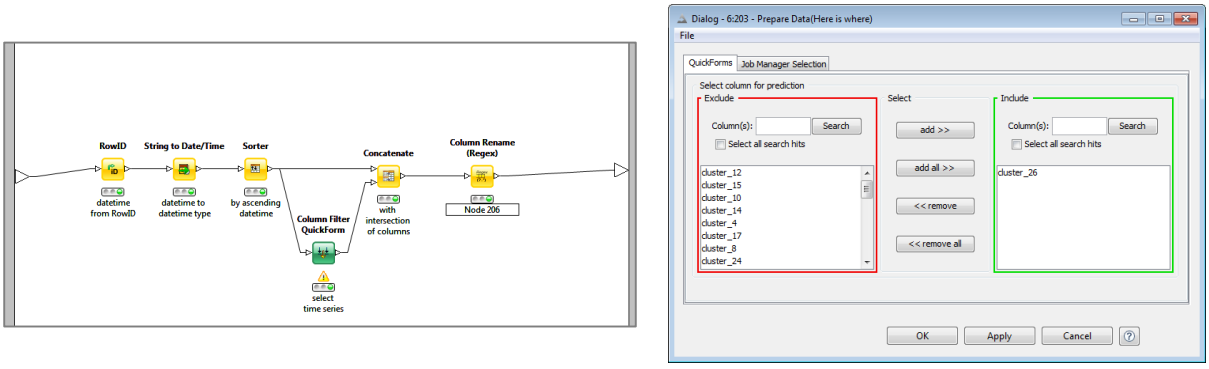

Figure 16: Sub-workflow and configuration window of the "PrepareData" metanode

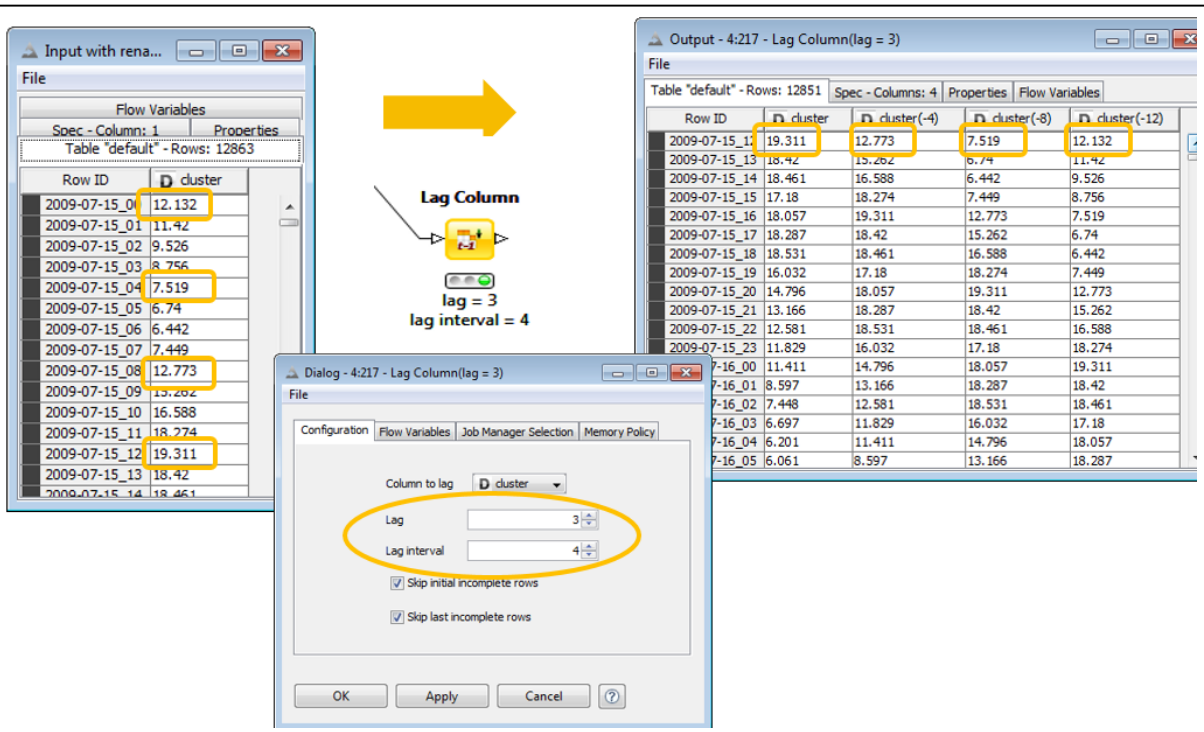

Figure 17: The „Lag Column" node

The node has three parameter settings: the column to lag, the lag, and the lag interval.

The column to lag is the time series column, for which to build the past and future record.

A lag interval P produces two data columns: `x(t)` and `x(t-P)`. `x(t-P)` is a copy of x(t) shifted in time P steps ahead. A lag interval 1 leaves the input data table unaltered. This means that only lag intervals higher than 1 make sense.

A lag N produces N data columns 1, … N steps behind in the past, `x(t-1)`, `x(t-2)`, …., `x(t-N)`, in addition to `x(t)`.

A lag Interval P and a lag N combined together produce the following data columns: `x(t-P)`, `x(t-2*P)`, …., `x(t-N*P)`, and `x(t)`. Figure 18 shows how the input time series is transformed by applying a lag interval = 4 and a lag = 3.

At this point we did not worry about seasonality, we just applied a brute force auto-regressive model to see its prediction potentials. The lag interval in the "Lag Column" node was then set to 1; this means that no periodicity or seasonality was considered for the analysis.

Having decided to ignore seasonality for now, the next question would be "how much back in the past do we need to go to have a meaningful model?". Would just the previous value be good enough to predict the current value? Good enough of course depends on the error that we are available to tolerate. That is: how big does the lag in the "Lag Column" node need to be to get a prediction error below a given threshold? Since we do not know what an acceptable error is, we have defined a measure for the prediction error and we have experimented with a few possible lags: 1, 3, 5, 7, and 10 lag values back in the past.

Linear or Polynomial Regression

After lagging the data for the selected clustered time series, a model needed to be trained using the past to predict the future. In a simple auto-regressive model we can use either a linear or a polynomial regression. First we splitted the data into a training and a testing set, then we used the lagged columns with "past values" as input to model the current value. The process is represented in figure 18.

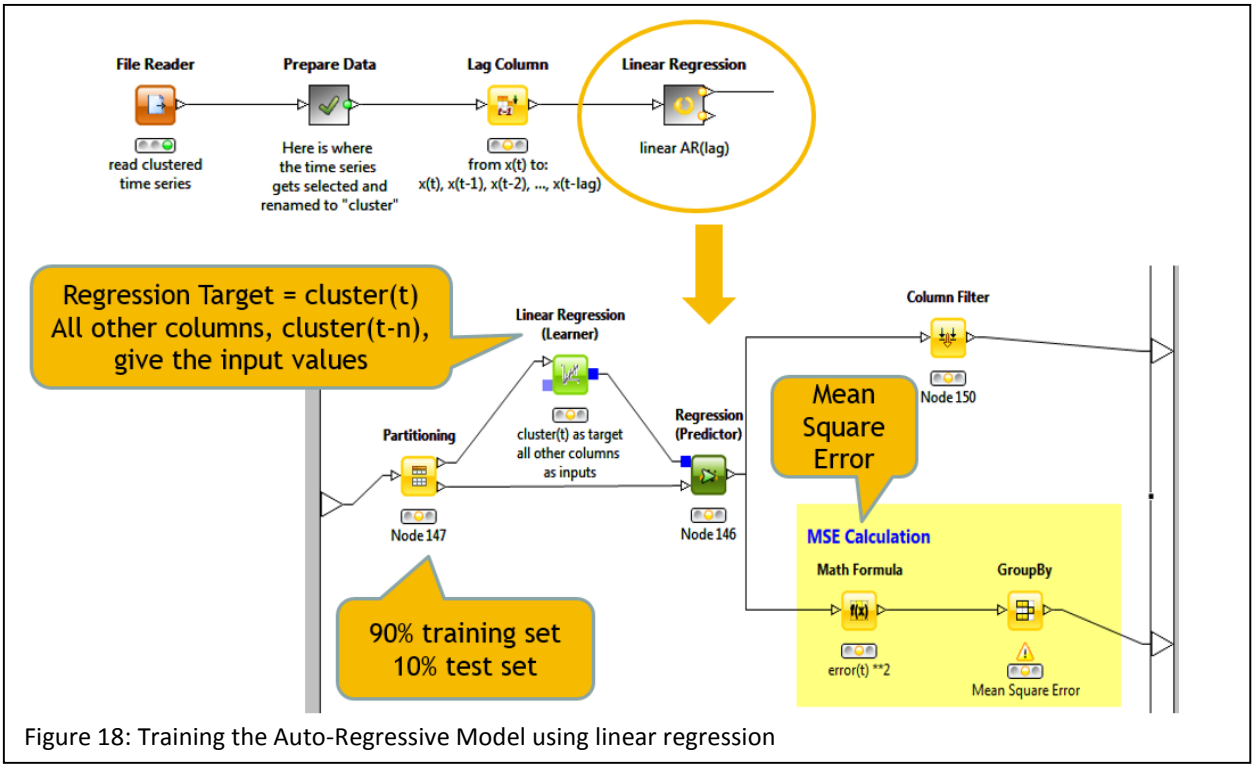

Figure 18: Training the Auto-Regressive Model using linear regression

After training, for each sequence of past values the model predicts the current value. As an example, the plot of the predicted time series coming from the regression model vs. the real values for one of the available clusters is shown in figure 19.

The prediction seems already pretty good, roughly following the original time series in its temporal trends. In order to really know how good the prediction is, we still needed an error measure. A classical error measure is the Mean Square Error (MSE). We calculated the MSE with 2 nodes. A "Math Formula" node calculated the squared difference between the original and the predicted value and a "GroupBy" node produced the average over time of these squared differences. The training process and the MSE calculation were wrapped into a metanode, named either "Linear Regression" or "Poly Regression" depending on whether the linear regression or the polynomial

regression had been used as a training algorithm. This metanode produced two output data table: the one on the top with the predicted and the original time series and the one on the bottom with the MSE calculated for the two time series and across the whole time window.

The complete workflow implementing the auto-regressive model to select and predict a time series with no seasonality adjustment is shown in figure 20.

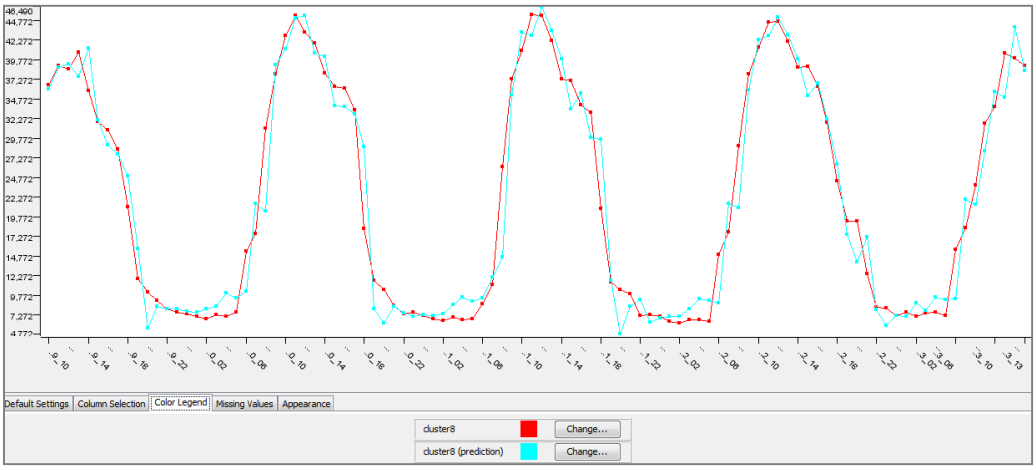

Figure 19: An Auto-Regressive Predicted Time-Series (light blue) vs. the corresponding Original Time-Series (red)

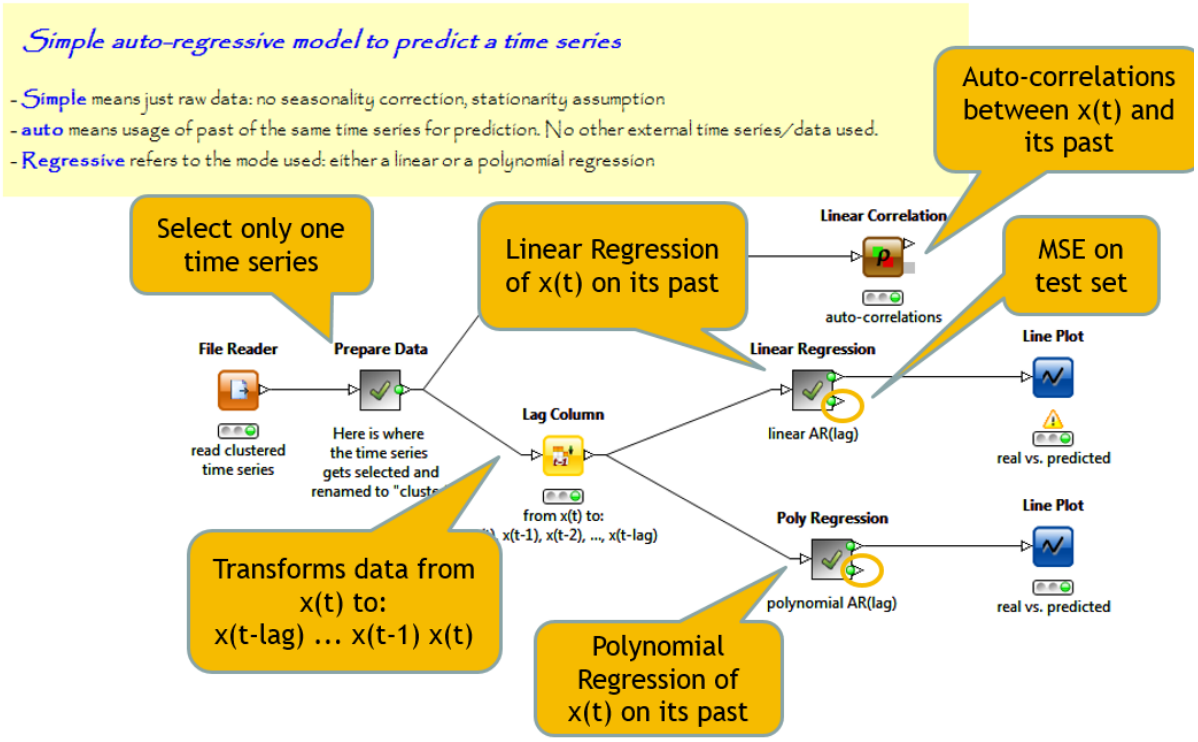

Figure 20: Workflow „AR Simple" implementing a simple auto-regressive model

Seasonality Adjustment

The auto-regressive analysis described in the previous section was just a first step. The analysis could be improved further by removing the seasonality from the time series. Energy data can include many types of seasonality: a winter vs. summer, a weekly, and even a daily seasonality.

If a time series has a 24-hour periodicity (or seasonality), this means that there is a 24-hour repeating pattern. If this pattern is identified and then removed from the time series, we only need to predict the deviations of the time series values from the repeating pattern, which might be easier to model. The problem now is split in two parts: we identify the repeating pattern (seasonality) and we remove it from the original time series, and then we train a model to predict the difference values.

Visually identifying the Seasonality Patterns

Specific algorithms exist to discover the best approximation for the data seasonality. However, for this first time series project, we decided to just visually inspect the seasonality trend. Let's plot, for example, the time series representing cluster 21 in the "Daily Users" group.

On a full scale (Fig. 21), from July 15th 2009 to January 1st 2011, it is easy to see the winter/summer seasonality. Notice that not all clustered time series show this winter vs. summer seasonality. Some clusters show no big difference in electricity usage between summer and winter.

The Christmas breaks are also easy to spot for this cluster. Considering these Christmas breaks and the medium-low electricity usage during business hours and business days (see tables 7 and 8), this cluster is likely to group together small businesses rather than households.

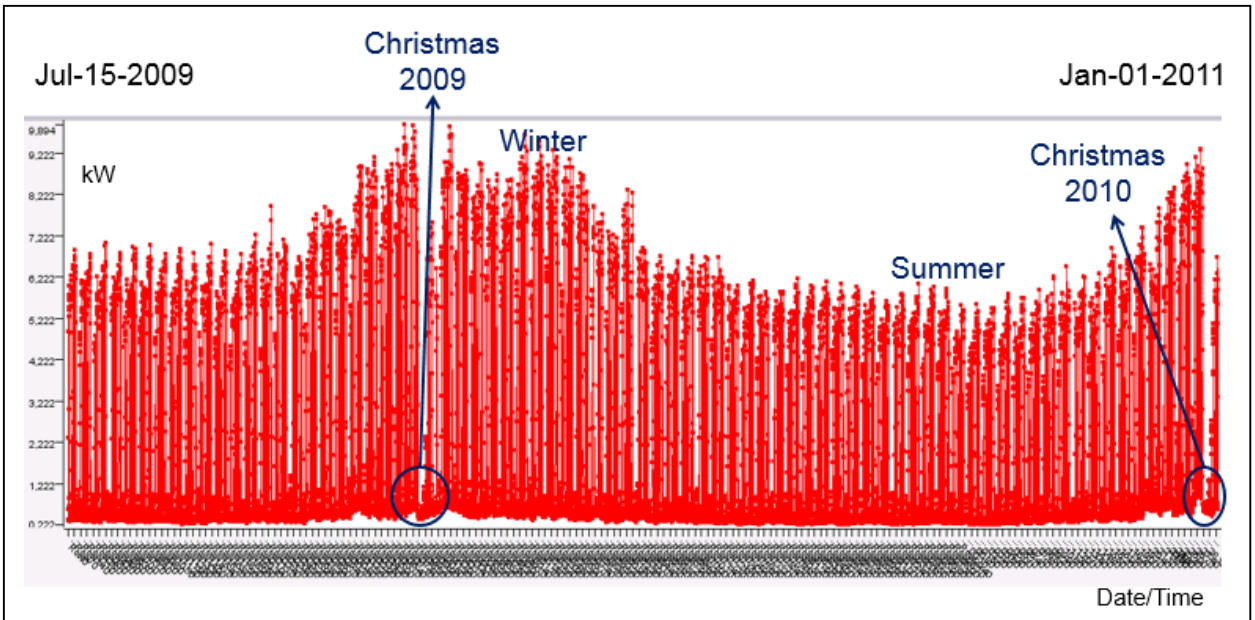

Figure 21: Cluster 21 Time-Series on a full time scale

If we look at the same plot using a smaller time window (Fig. 22), the weekly seasonality emerges clearly. Saturdays and especially Sundays are characterized by a very low usage of electricity throughout the day, while business days exhibit a higher usage of electricity and a similar usage pattern throughout the week. Almost all clusters show, more or less evidently, a weekly seasonality, that is a repeated pattern covering a week. Only a few clusters show no weekly seasonality, i.e. every day is similar to the next or there is an unpredictable pattern of electricity usage for every day of the trial period.

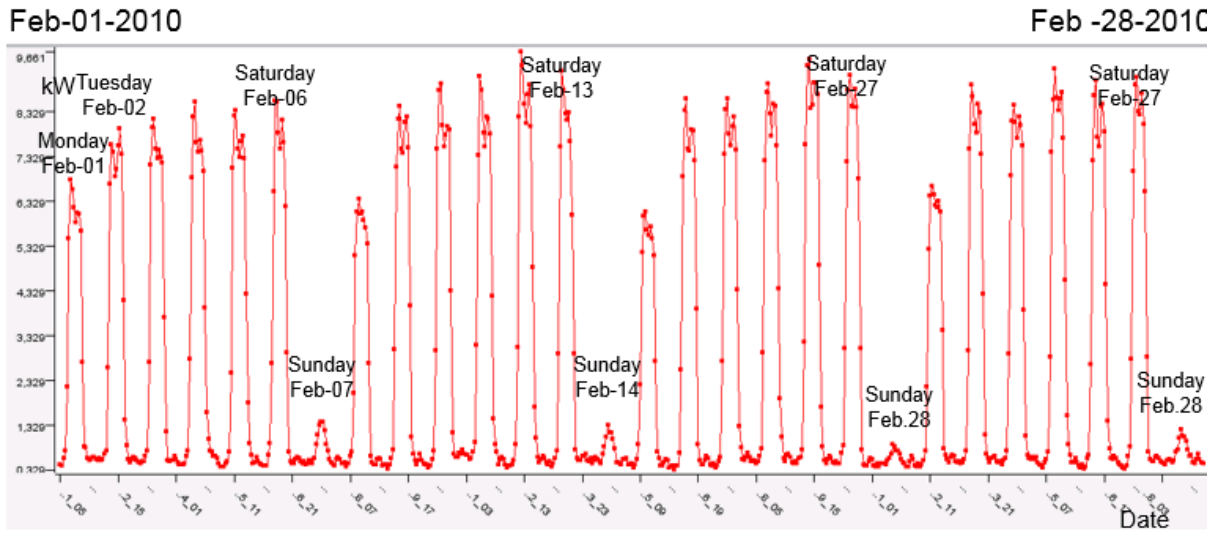

Figure 22: Cluster 21 Time-Series on a time scale of a few weeks.

Concentrating on an even smaller time window, for example on a one-week time window, the daily seasonality also emerges clearly. In this case, every business day starts around 7:00 and ends around 20:00. Saturday and Sunday the business is inactive, electrically speaking.

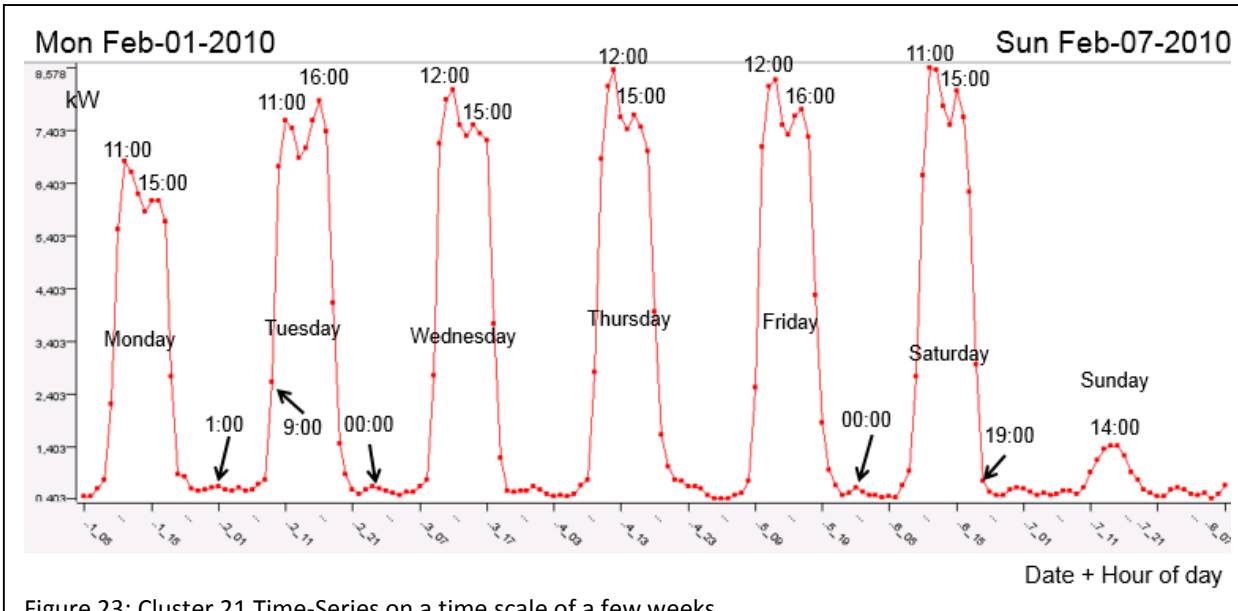

Figure 23: Cluster 21 Time-Series on a time scale of a few weeks.

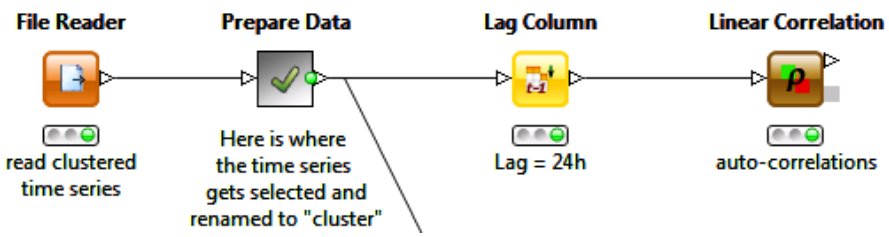

Figure 24. Workflow for the Calculation of the Auto-correlation Matrix

A 24h seasonality emerges also from the auto-correlation matrix calculated over a 24 hour window of the time series. The auto-correlation is calculated using the “Lag Column” node, with Lag Interval = 1

and Lag = 24, followed by a “Linear Correlation” node (Fig. 24). The auto-correlation matrix for cluster 21 is shown in figure 25, which shows a repeating pattern over the 24 hours.

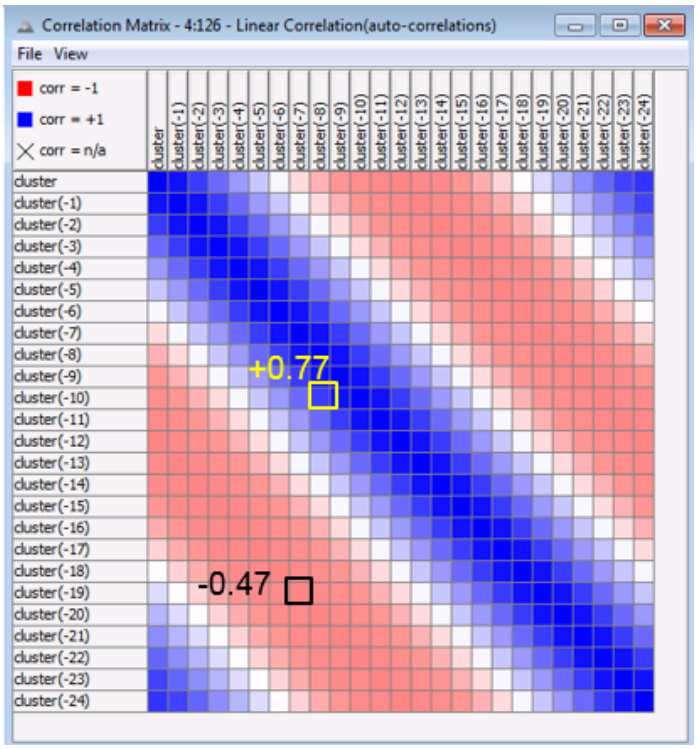

Figure 25. Auto-correlation Matrix for the time series of cluster 21

Removing 24 hour Seasonality

Our initial focus is on the 24-hour seasonality, since it is the easiest to observe and to calculate. There are many ways to introduce a 24 hours template for seasonality adjustment:

- The first 24hours in the whole time series
- The average 24 hours on the training set
- The previous 24 hours before current time t

The first 24-hour of the time serie is probably the least accurate method, because it is just a lucky guess. If the first 24 hours in the time series are not typical for the rest of the time series, the seasonality adjustment using this pattern will be less than accurate. Using the average 24 hours of the training set might also be misleading, because averaging 24 hours windows from summer and winter all together might be inaccurate. The previous 24 hours are probably the most accurate representation of the daily rhythm, since things change gradually over time.

We produced workflows to remove the 24 hour seasonality for two of the above listed seasonality patterns: the first 24 hours of the time series, the previous 24 hours for each 24 hours window in the time series. In both cases, the 24h pattern is created and removed from the subsequent day(s). The linear regression model is then trained on the differences between the current pattern and the seasonality pattern.

Figure 26 shows the workflow “AR - first 24h”, which removes the first 24 hours of the time series from every subsequent day in the time series and trains a simple linear regression model on the remaining values. After lagging and predicting the time series, a new meta-node, named “re-build signal” takes care of reintroducing the removed 24-hour pattern into the predicted signal.

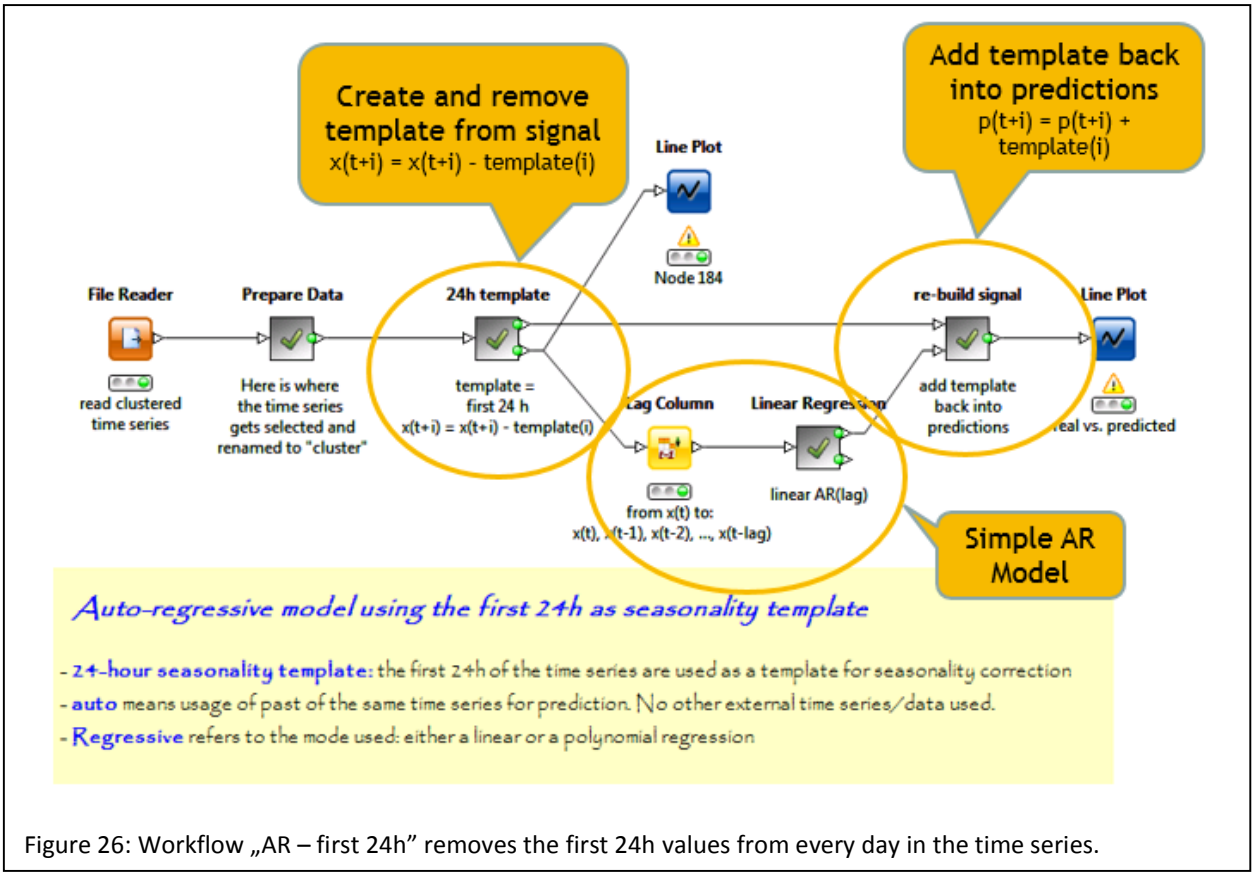

Figure 26: Workflow „AR – first 24h" removes the first 24h values from every day in the time series.

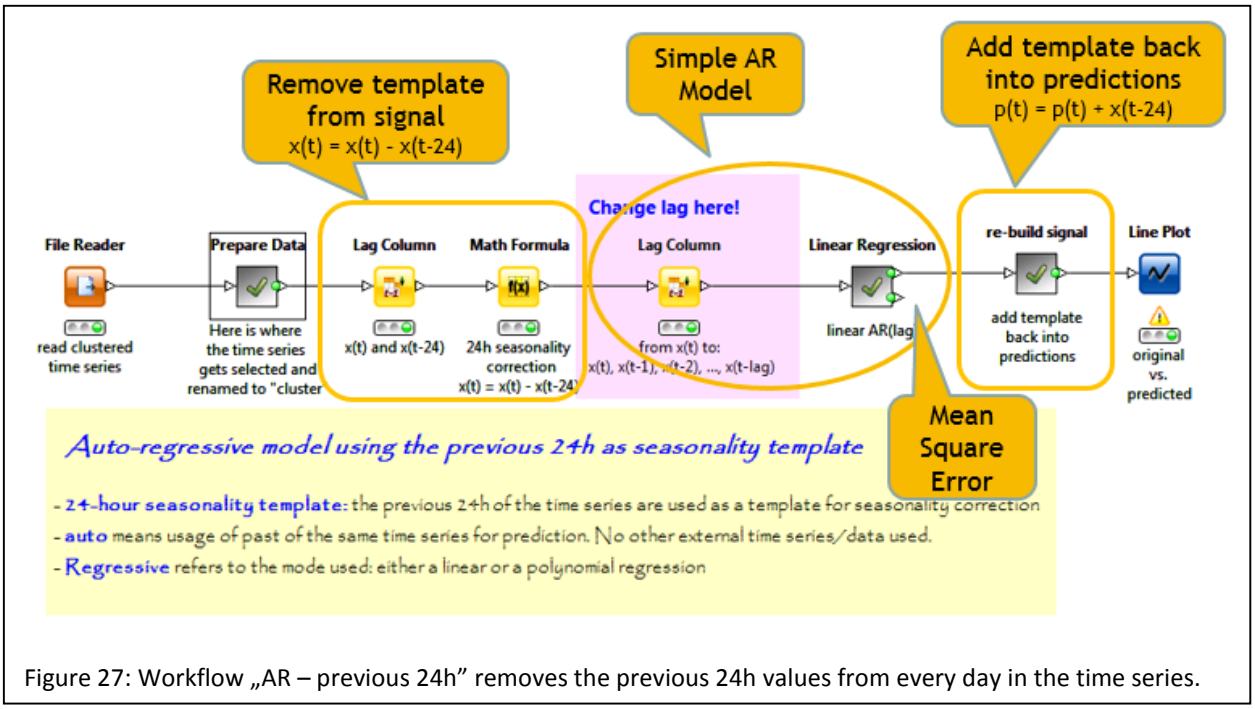

Figure 27: Workflow „AR – previous 24h" removes the previous 24h values from every day in the time series.

Similarly the workflow "AR - previous 24h" adjusts the time series values by removing the previous 24h from each 24h window (Fig. 27). Again, a simple linear regression model is then trained on the remaining values.

This workflow uses a "Lag Column" node to get a copy of the time series from the 24 hours before the current time t (Lag Interval = 24) and a "Math Formula" node to subtract the value at time t-24h from the current time series alue at time t. Also here, after lagging and predicting the time series, a

new meta-node, named “re-build signal” takes care of reintroducing the removed 24-hour pattern into the predicted signal.

Removing the weekly Seasonality

However, the 24 hours is not the only seasonality observed in the time series. There is also a weekly and a yearly seasonality. Since our data includes only a bit more than a year, we are going to ignore the yearly seasonality. We cannot ignore though the weekly seasonality.

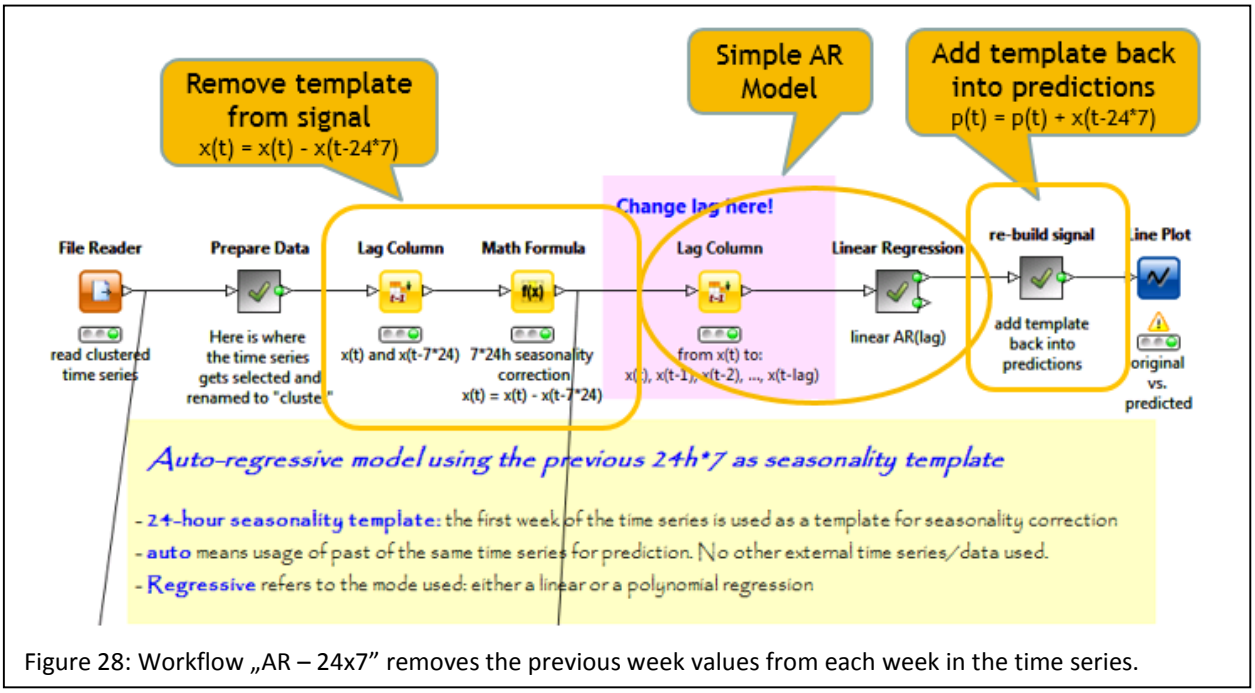

Figure 28: Workflow „AR – 24x7” removes the previous week values from each week in the time series.

In order to model the weekly seasonality, we just extend the “AR – previous 24h” workflow to remove the values from the previous 24 * 7 hours time window from the current 24 * 7 hours. The resulting workflow is shown in figure 29. Here, the “Lag Column” node copies and shifts the time series 24*7 samples backwards; then the “Math Formula” node subtracts the `x(t-24*7)` values from the current time series `x(t)` values; finally, the meta-node named “re-build signal” puts the one-week long template back into the predicted signal.

Removing the previous week value `x(t-24*7)` from the current week value `x(t)` produces the signal shown in figure 29, with the blue line being the moving average on a moving window of 21 samples. An extract of one predicted time series, using the linear regression model, is shown, together with the original time series, in figure 30.

How good this prediction is, we can only say via the Mean Squared Error implemented inside the linear regression metanode. Prediction MSE values vary between 0.01 and 30, depending on the clustered time series. However, time series showing bigger consumed energy, also show bigger MSE values. Using the relative MSE error as the MSE divided by the average hourly value of used energy expressed in percent, we get a more realistic measure of the prediction quality. Prediction errors then range between 1% and 10%.

For most clustered time series, the weekly seasonality adjustment brings the biggest improvement in prediction quality. A few clusters though show no need for seasonality adjustment; others show the best prediction improvement with a daily seasonality adjustment. Optimizing the prediction model for each clustered time series produces a better estimation of the energy used over all in Ireland. In

addition, concentrating on improving the prediction of the clusters with large usage of electricity might influence even more the overall prediction quality.

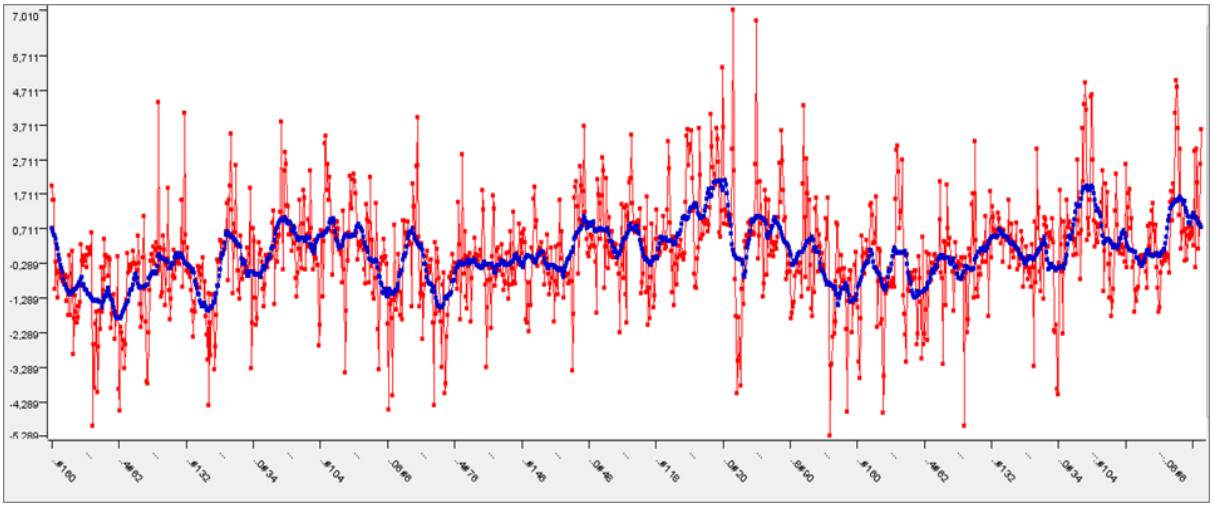

Figure 29: Time series values adjusted for weekly seasonality in red and its moving average in blue.

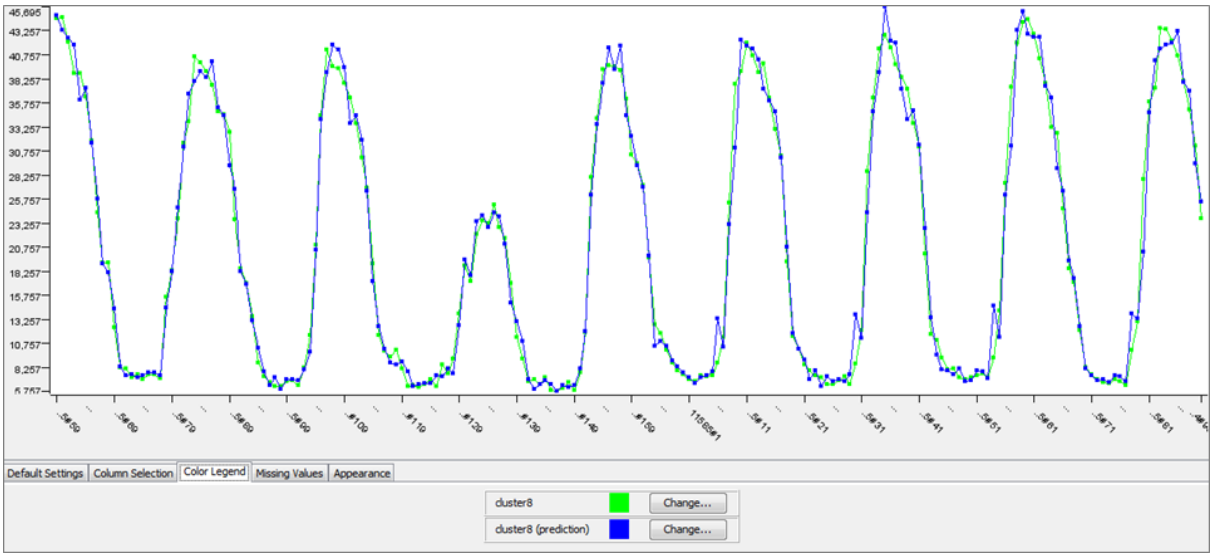

Figure 30: Original time series and predicted time series after being adjusted for weekly seasonality in green and blue respectively.

A first frame for general time series prediction

In the previous section we have seen how to implement an auto-regressive model to predict future time series values using linear regression and seasonality adjustment. However, the frame implemented derives a more general character from the KNIME modularity. Indeed, the same frame can be used with any other numerical data mining model. The linear regression model, for example, could be changed with a multilayer perceptron in the "Linear Regression" meta-node (Fig. 31).

Since the "RProp MLP Learner" node (the node that implements the Multilayer Perceptron in KNIME) accepts only numerical values between 0 and 1, a normalization and a denormalization nodes were introduced to train the model and to rebuild the time series respectively (Fig. 32). As you see, the work load necessary to update the original workflow to a neural network based predictor has been minimal. In general, this frame (workflow) could be easily extended to include more complex seasonality effects, moving averages, and more powerful prediction techniques.

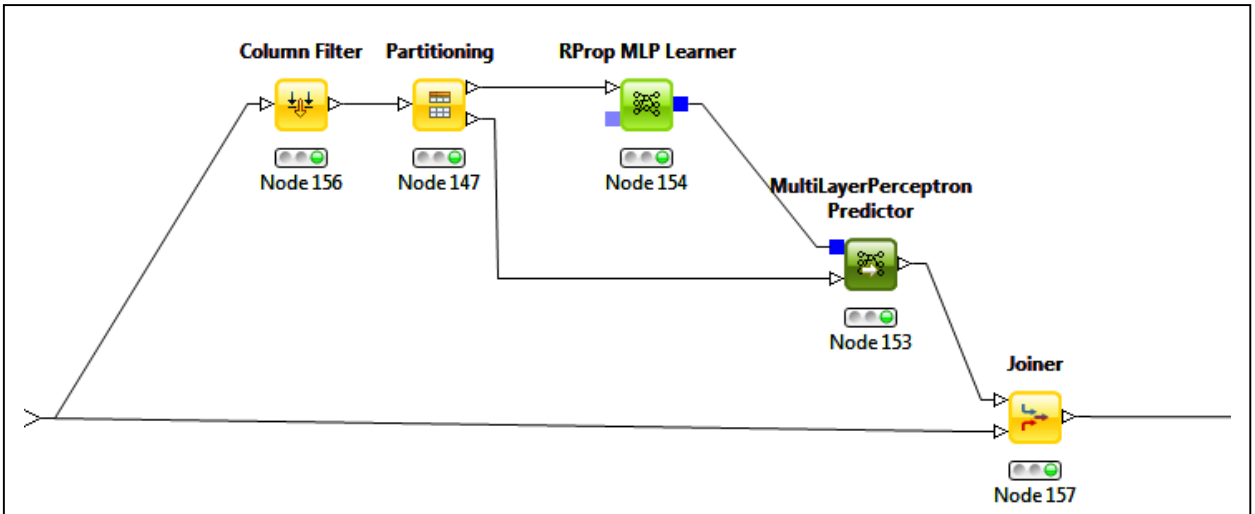

Figure 31: Time Series Prediction using a MultiLayer Perceptron

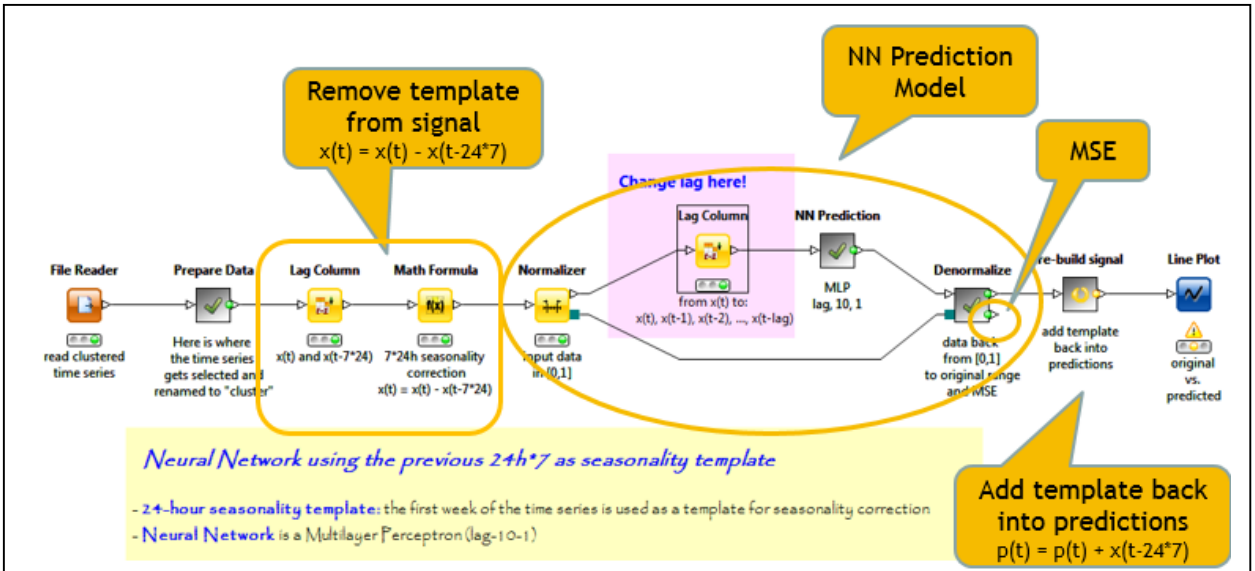

Figure 32: Time Series Prediction using a MultiLayer Perceptron

Big Data Effect

This entire example was run using KNIME open source (www.knime.com) on a 4 core laptop with 8GB RAM. All workflows used in this project contain an incredible amount of data manipulation and transformation. In particular, the first workflow, named "PrepareData", performs a number of aggregations, conversions, and sorting operations on the original 176 millions rows. This, while done effectively, did take hours to process. For example, only the "Sorter" node, sorting rows by time in an increasing order, was executing in 5 hours.

With the advent of low cost hardware and infrastructure such as Hadoop, we have now the option of saving a considerable amount of time. It is, of course, possible to implement KNIME on Hadoop and combine together these two pieces of the open source world. However, if you do not have the time or the inclination to proceed with this new in-house implementation, you can turn to the Actian DataRush (soon to be DataFlow) engine (http://bigdata.pervasive.com/Products/Analytic-Engine-Actian-DataRush.aspx). The DataRush engine is a high performance parallel engine designed for use

on distributed platforms, such as Hadoop. The DataRush engine has also a KNIME extension, which includes a set of proprietary nodes for high speed parallel data processing.

While KNIME Desktop is free of charge, DataRush is one of the commercial third-party extensions of KNIME, i.e. a commercial license is required. A temporary test license might be available on demand from Actian's web site. DataRush extension for KNIME can be installed as any other KNIME extension via the "Help/Install New Software" or the "File/Install KNIME Extension" options. Installation is quick and easy and produces a new KNIME category in the Node Repository panel named Actian RushAnalytics.

Figure 33 places KNIME and DataRush on a scatter plot according to their respective strengths, i.e. speed vs. analytics. So, while KNIME is positioned at a high value for analytics sophistication, DataRush products occupy the lower part of the graphic with higher execution speed. This graphic shows the complementarity of the two engines.

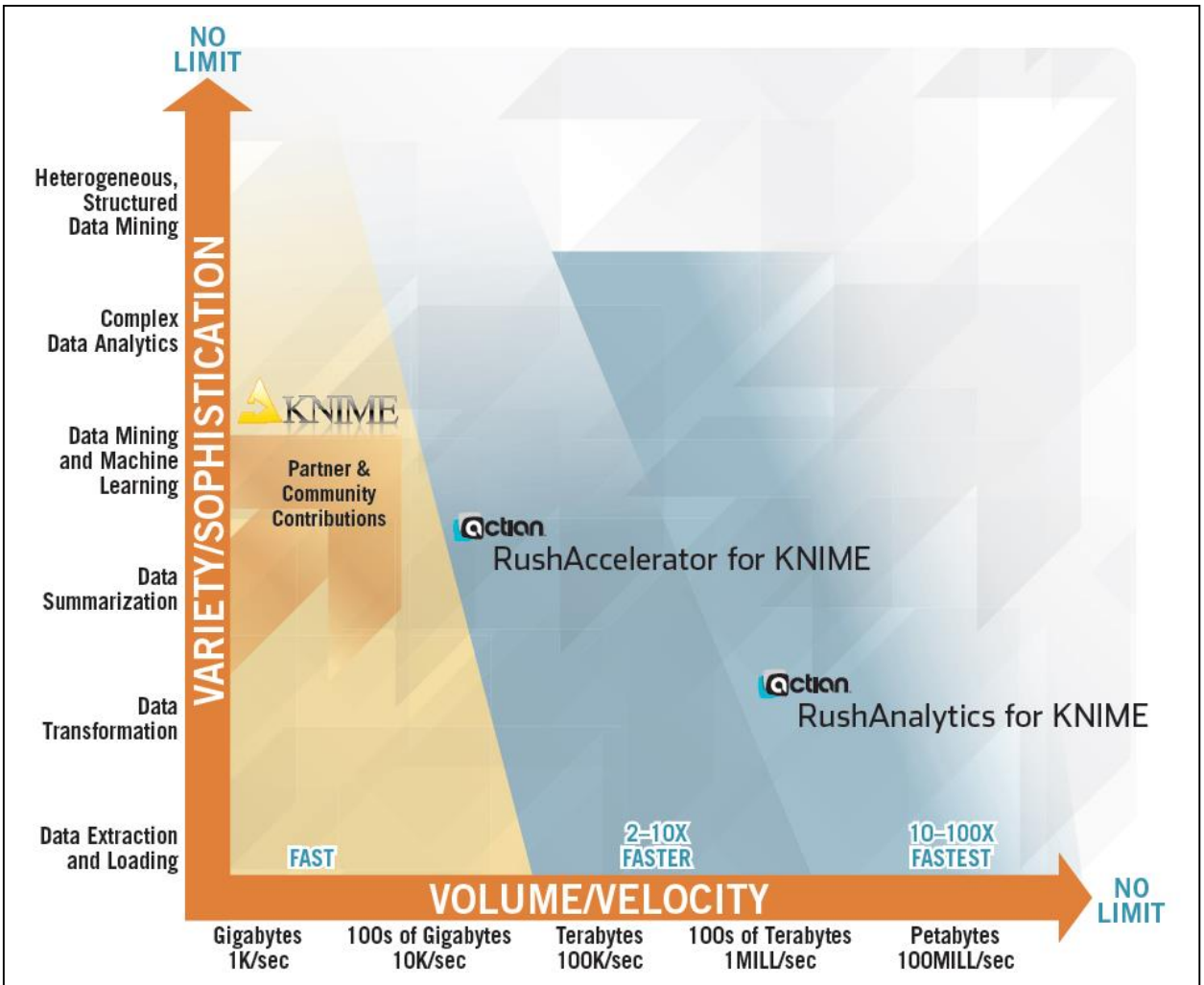

Figure 33: A high level overview of a big data workflow architecture

The blue/brown separation in the figure also indicates what is worth to move onto a big data platform. In general, the first transformations in a workflow are aimed at reducing the data space dimensionality mainly through aggregation, conversion, filtering, sorting, etc…. Those are the most computationally intensive transformations, possibly dealing with a very high number of data rows. These operations, when happening on very large data sets, are worth a thought about parallelization to speed up execution. On the opposite, subsequent operations, like Data Mining, Machine Learning,

and Statistical algorithms, usually deal with already processed data, i.e. with a reduced size data set. For these operations, usually and so far, a big data approach has not been necessary.

Our three sets of workflows are no exception. The first workflow "PrepareData" has a high computational load, requiring a few days to execute on a 4-core with 8GB RAM laptop. The following workflow, named "k-Means", requires just a few hours and the execution time is concentrated mainly on the aggregating operations to create the prototype cluster time series. Finally, the last workflow group implementing the time series prediction deals with such a reduced data set to take only a few minutes to execute. Based on these considerations, we remodeled the first workflow, the one named "PrepareData", using the DataRush extension's nodes, i.e. effectively moving portions of the original workflow onto Hadoop. We renamed the new workflow "PrepareData – DataRush" (Fig. 34).

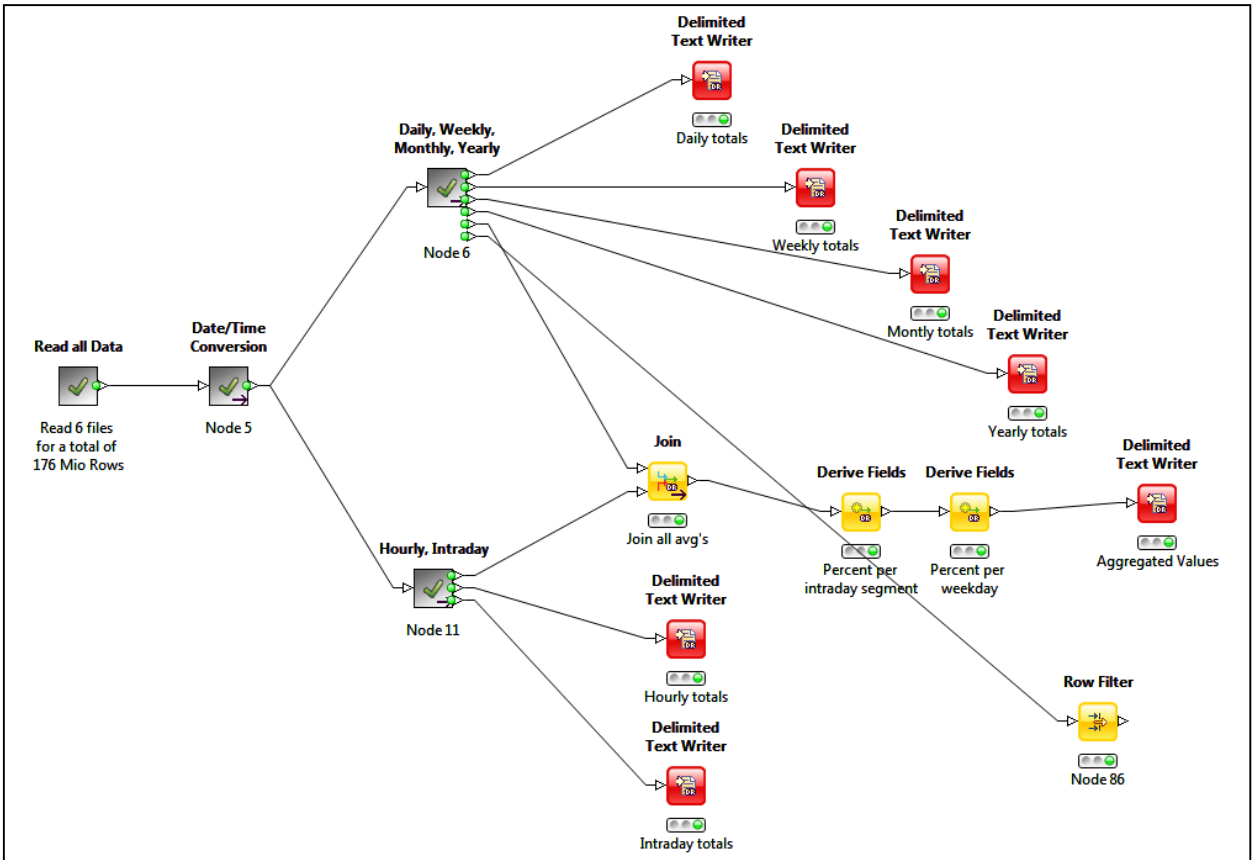

Figure 34: Workflow "PrepareData – DataRush" implemented following the workflow "PrepareData" in figure 11 and moving some nodes onto the DataRush big data platform.

By introducing the DataRush nodes into the "PrepareData" workflow, the execution time for data extraction and particularly for data transformation got reduced by a factor of 30.

By introducing the DataRush nodes in other portions of the subsequent workflows, speed improvements were not as great, as the previous processing already reduced the data down to a much smaller size for the clustering and time series prediction workflows. For our work, the need for a "big data" processing option can be summarized as follows:

Accessing Data: Reading even very large amounts of data does not seem to be prohibitive even on a KNIME Desktop.

Manipulating Data: Here the big data option can be very beneficial when dealing with very large quantities of data.

Data Visualization: For simply visualizing data, big data options are not needed. After all, you can only visualize so many data points on a screen anyway.

Predictive Mining: This is case dependent. In our case, big data was not necessary, since clustering and the previous data manipulation steps reduced the data so significantly (ie: one time series per cluster) before running the time series prediction.

Execution: When using the models for prediction, for a batch execution against even larger quantities of data or, if required, for a real time execution, the big data option could be very useful.

However, the most important feature of this approach is that, with the right platform such as KNIME combined with Actian DataRush, you can mix and match your techniques within the workflow to achieve both the optimal point in flexibility, processing, speed, and investment based on your requirements.

Conclusions

Data Science Conclusions and next Steps

In this project we implemented a first attempt to time series forecasting using KNIME. This first step basically consisted of an autoregressive model with seasonality correction. The key to the implementation a time series forecasting workflow was given by the new "Lag Column" node, distributed with the latest version 2.8 of KNIME. This node indeed allows for the shifting of a selected data column, effectively building data rows with past and future of the time series with respect to a given time t. After this data transformation, the application of any numerical data mining algorithm implements a time series forecasting task.

The complete implementation of an arima model still requires a moving average. The next step in the project would then be the introduction of a moving average into the timeseries forecasting workflow, in order to implement an arima(p,d,q) model. The implementation of a node for the automatic detection of the optimal arima orders p, d, and q would also be useful.

Moving away from the arima model, in future developments of this project we could experiment with different time series forecasting techniques. We started here by using a neural network with satisfactory results. However, using the same framework, we could apply any other numerical data mining or statistical technique.

As we know, results can be improved by using different more specialized techniques as well as limiting the investigation area. For example, we could implement a stepwise prediction technique or we could categorize, and therefore simplify, the target, which suddenly makes a much bigger range of data analysis algorithms available.

From a more engineering point of view, it would be useful for the whole community to build a time series meta-node. This meta-node could then provide a pre-packaged menu of options for time series forecasting. This metanode would also include a sub-metanode for the calculation of the mean square error between the predicted time series and the original time series.

Finally, we should compare the results of this approach of predicting each cluster time series to the current global forecasting methods. This is to determine whether a lower resolution approach leads to a higher degree of accuracy in forecasting.

Business next Steps

From the business side, combining clustering and forecasting techniques has surfaced new fact-based insight that has allowed us to further tailor the offerings while controlling the costs.

For the future, it would be useful to continue investigating each cluster more in detail, combining the clusters with the original survey data, which will provide more demographics and fact based information about each cluster. One could then determin which clusters are suitable for pro-active pricing approaches and which clusters can receive an offer when in combination with others.

Probably the most exciting area for further investigation is the definition of a very few “golden questions” in pre-purchase questionaires and online forms to pre-identify the cluster of energy usage for each new meter ID. In this way, a more accurate tailored pricing plan can not only be offered to existing customers but can be used to entice new customers where their energy usage profilie is not known.

These techniques have wide implications for all other industries that have massive quantities of time-based or telemetry data. The costs of storage are now low, the expertise is available in white papers such as this, and - most importantly - open source options such as KNIME can bring the data together with the tools and the science to benefit the business.

Note. This white paper along with a sample of all workflows is available for download from http://www.knime.com/white-papers. The KNIME Desktop open source software can be downloaded from www.knime.com.