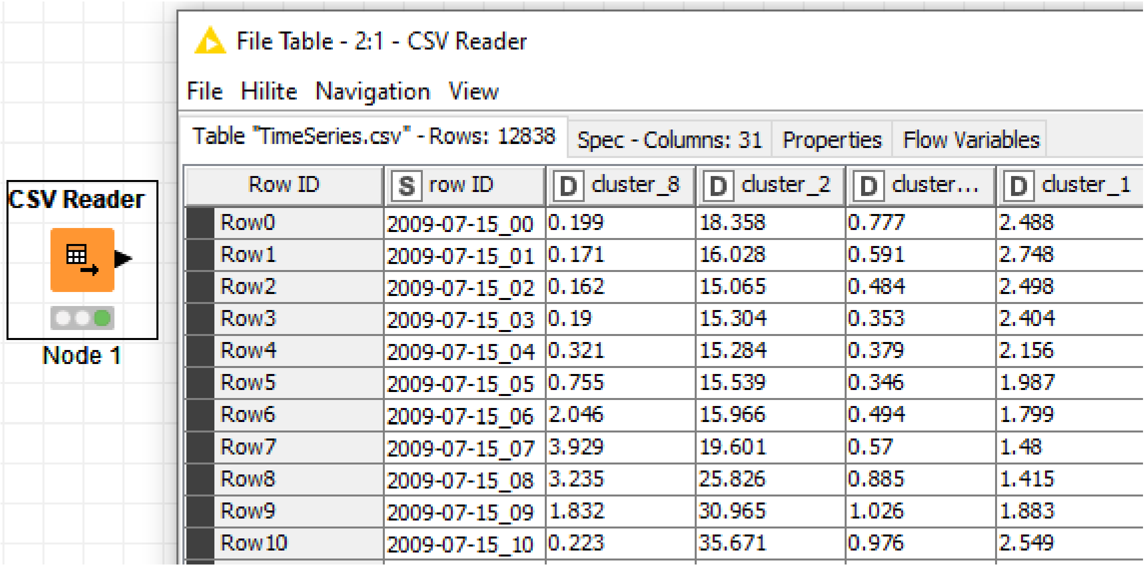
Regression on Time Series Data

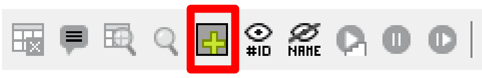
In this demonstration, we will apply linear regression on time series data.

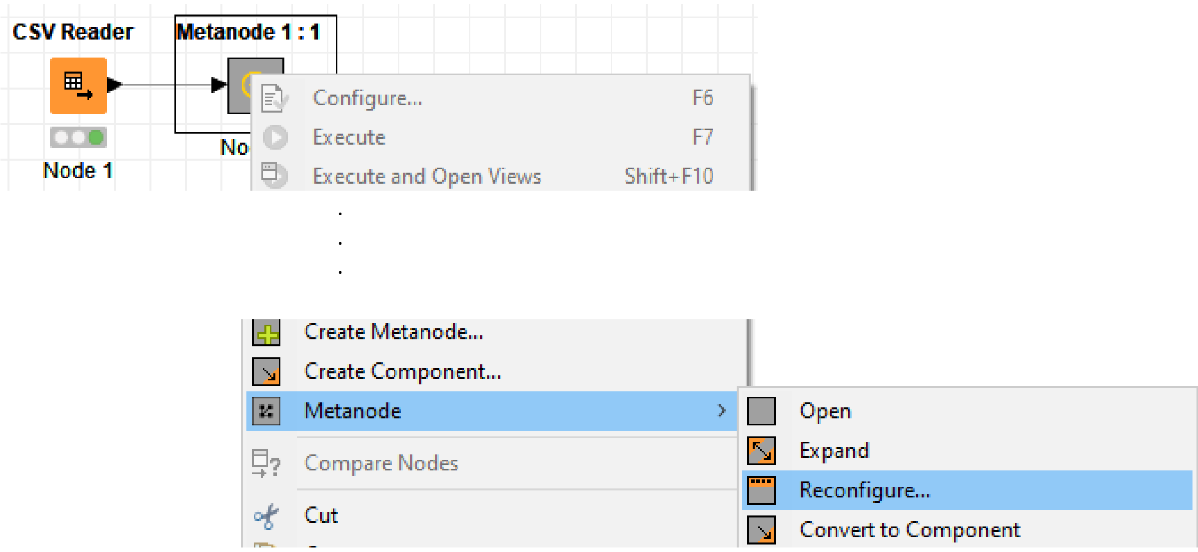
1. Firstly, let us read in the data. Use a *CSV Reader* node to read in the datasets/TimeSeries.csv file.



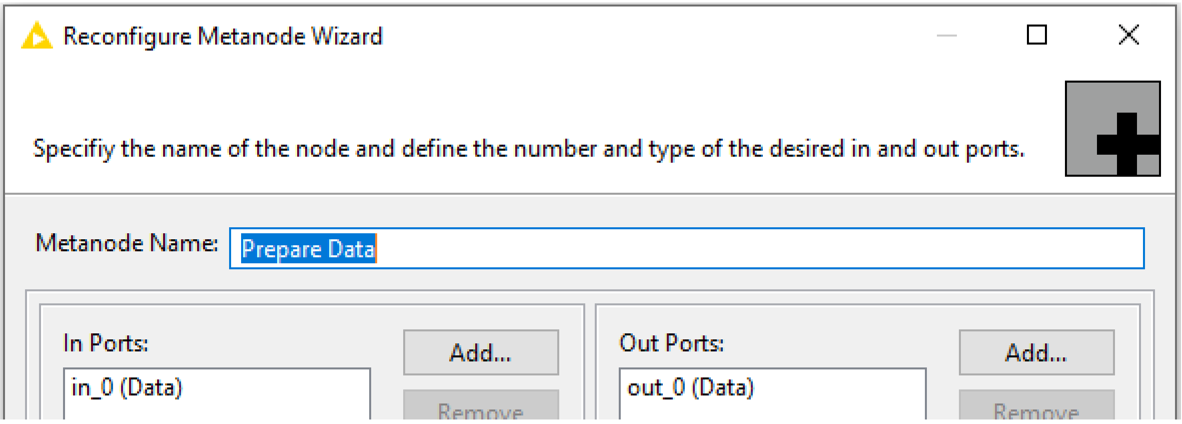
Row ID	S row ID	D cluster_8	D cluster_2	D cluster...	D cluster_1
Row0	2009-07-15_00	0.199	18.358	0.777	2.488
Row1	2009-07-15_01	0.171	16.028	0.591	2.748
Row2	2009-07-15_02	0.162	15.065	0.484	2.498
Row3	2009-07-15_03	0.19	15.304	0.353	2.404
Row4	2009-07-15_04	0.321	15.284	0.379	2.156
Row5	2009-07-15_05	0.755	15.539	0.346	1.987
Row6	2009-07-15_06	2.046	15.966	0.494	1.799
Row7	2009-07-15_07	3.929	19.601	0.57	1.48
Row8	2009-07-15_08	3.235	25.826	0.885	1.415
Row9	2009-07-15_09	1.832	30.965	1.026	1.883
Row10	2009-07-15_10	0.223	35.671	0.976	2.549

2. To make workflows look cleaner and to consolidate related nodes together for easier reusability, metanodes are sometimes used. Create a new 1:1 metanode by clicking on this symbol in the top menu bar and reconfigure it:

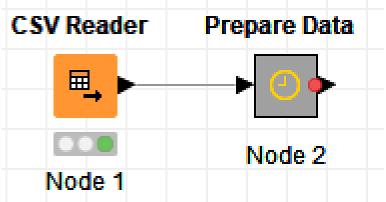




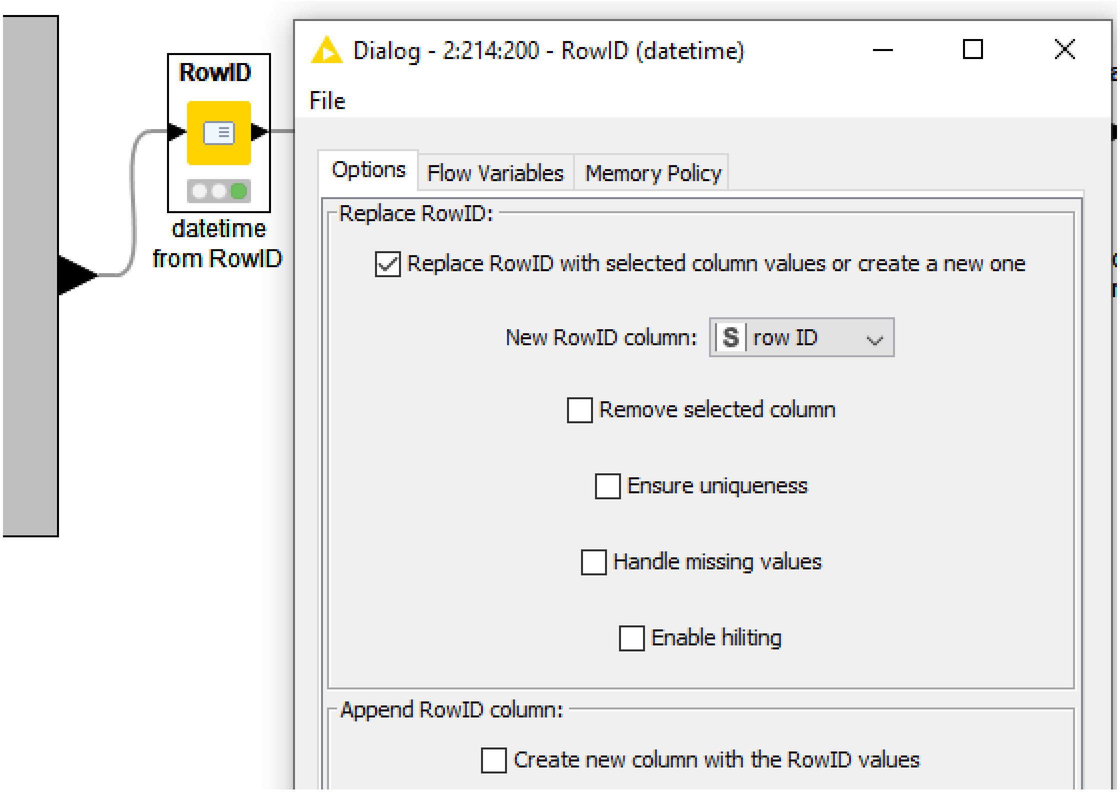
3. Rename the metanode to “Prepare Data”:



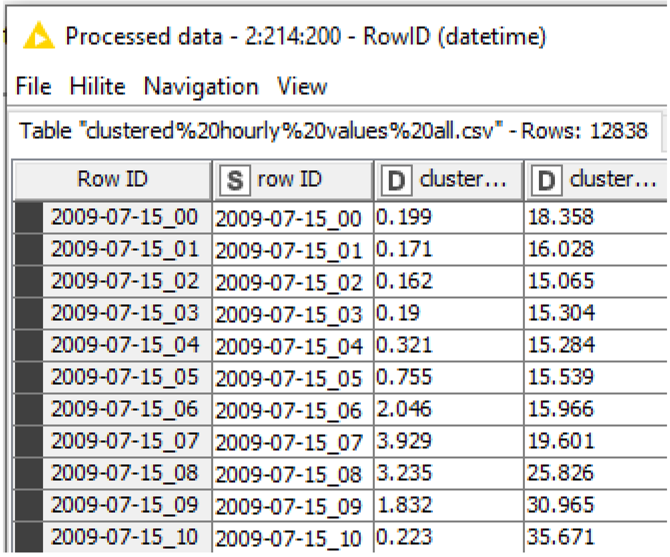
Renamed node:



4. Double-click into the node and create a RowID node to make a row id using the datetime “row ID” column:



5. You can see that the datetime column is copied as a row id:

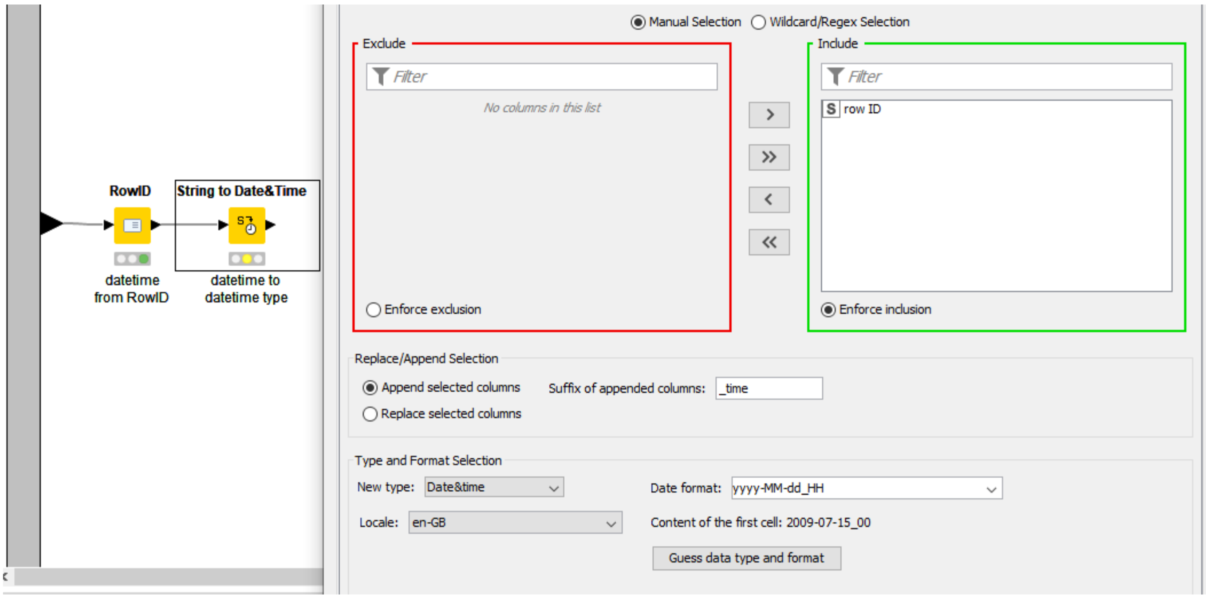

Processed data - 2:214:200 - RowID (datetime)

File Hilite Navigation View

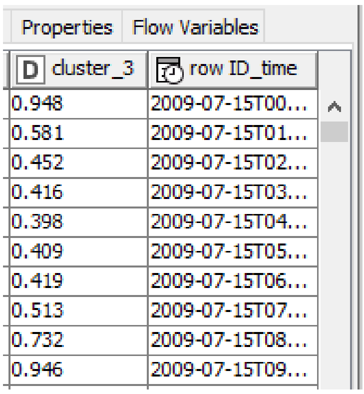
Table "clustered%20hourly%20values%20all.csv" - Rows: 12838

Row ID	S row ID	D cluster...	D cluster...
2009-07-15_00	2009-07-15_00	0.199	18.358
2009-07-15_01	2009-07-15_01	0.171	16.028
2009-07-15_02	2009-07-15_02	0.162	15.065
2009-07-15_03	2009-07-15_03	0.19	15.304
2009-07-15_04	2009-07-15_04	0.321	15.284
2009-07-15_05	2009-07-15_05	0.755	15.539
2009-07-15_06	2009-07-15_06	2.046	15.966
2009-07-15_07	2009-07-15_07	3.929	19.601
2009-07-15_08	2009-07-15_08	3.235	25.826
2009-07-15_09	2009-07-15_09	1.832	30.965
2009-07-15_10	2009-07-15_10	0.223	35.671

6. We will next change the datetime column (not the newly created row id) into an actual datetime variable.

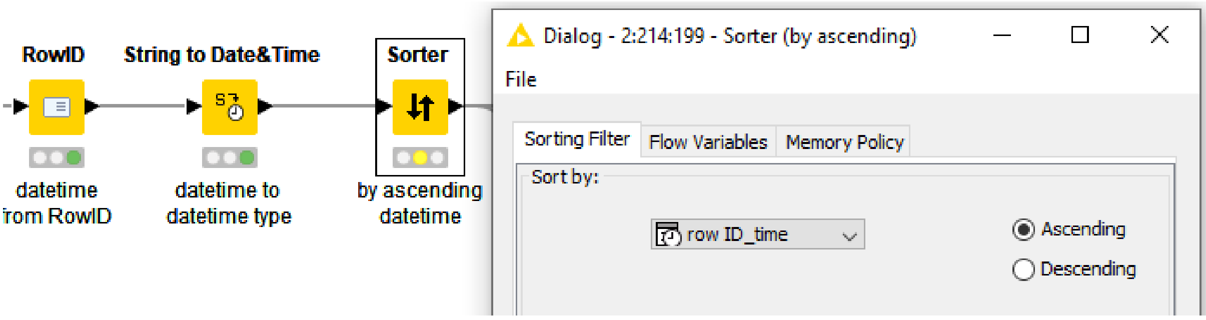


Scroll to the end of the dataset to see the datetime variable:

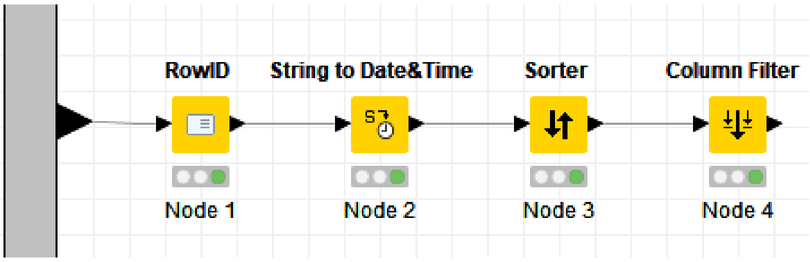

Properties Flow Variables

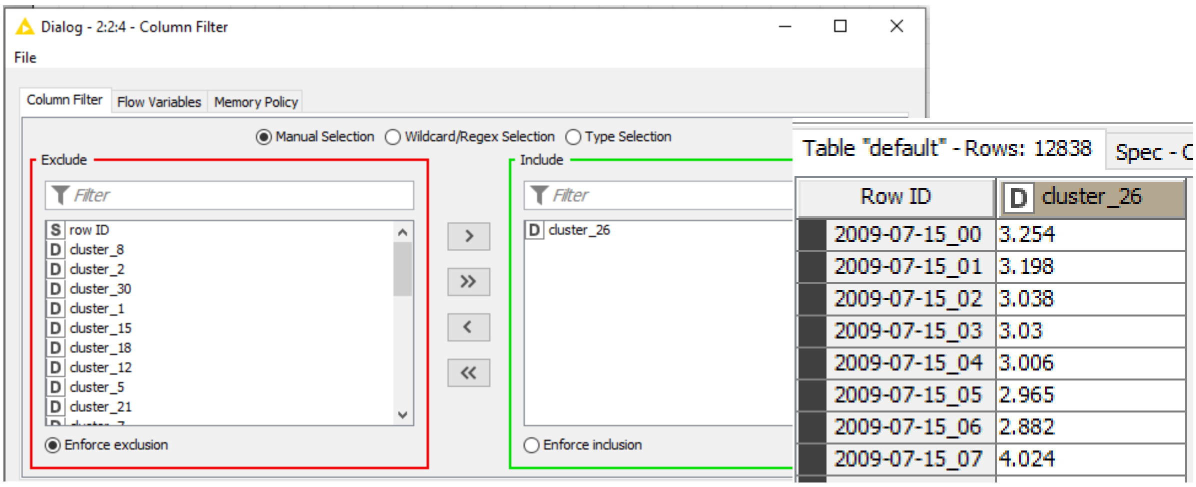
D cluster_3	row ID_time
0.948	2009-07-15T00...
0.581	2009-07-15T01...
0.452	2009-07-15T02...
0.416	2009-07-15T03...
0.398	2009-07-15T04...
0.409	2009-07-15T05...
0.419	2009-07-15T06...
0.513	2009-07-15T07...
0.732	2009-07-15T08...
0.946	2009-07-15T09...

7. Sort the data into an ascending order using the *Sorter* node:

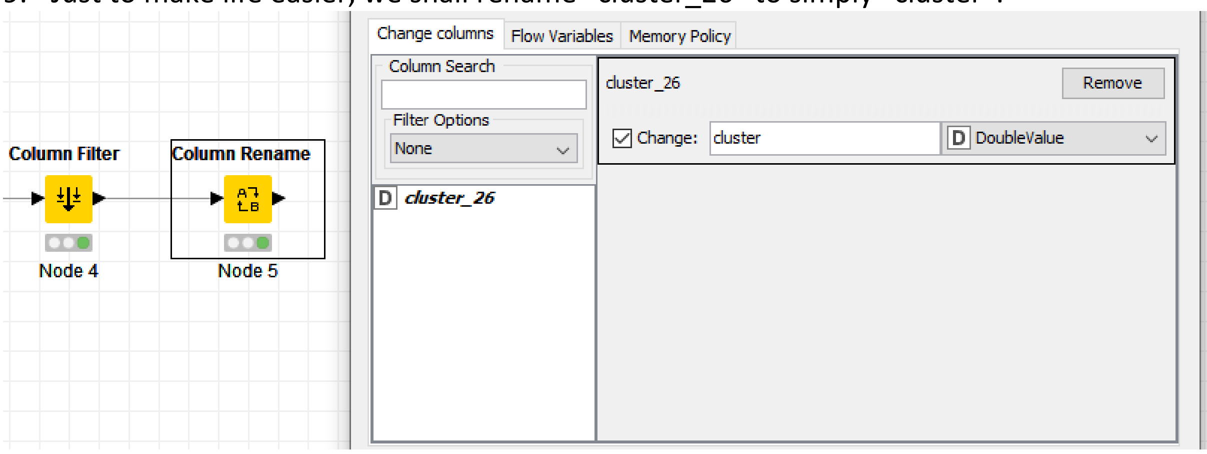


8. We shall now arbitrarily choose one of the cluster i.e. cluster_26, as our data of interest:

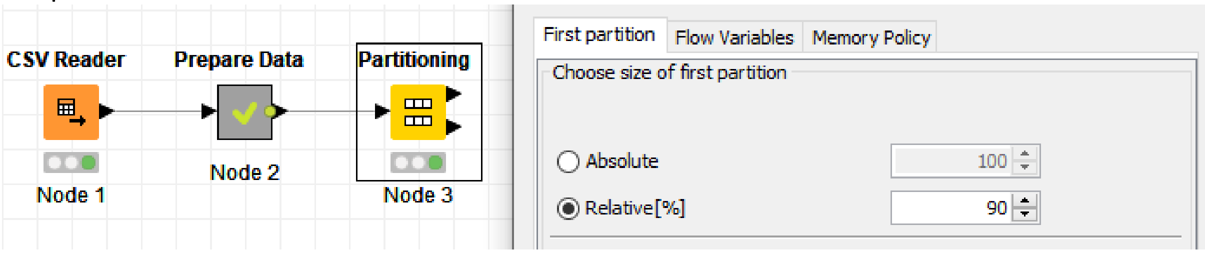




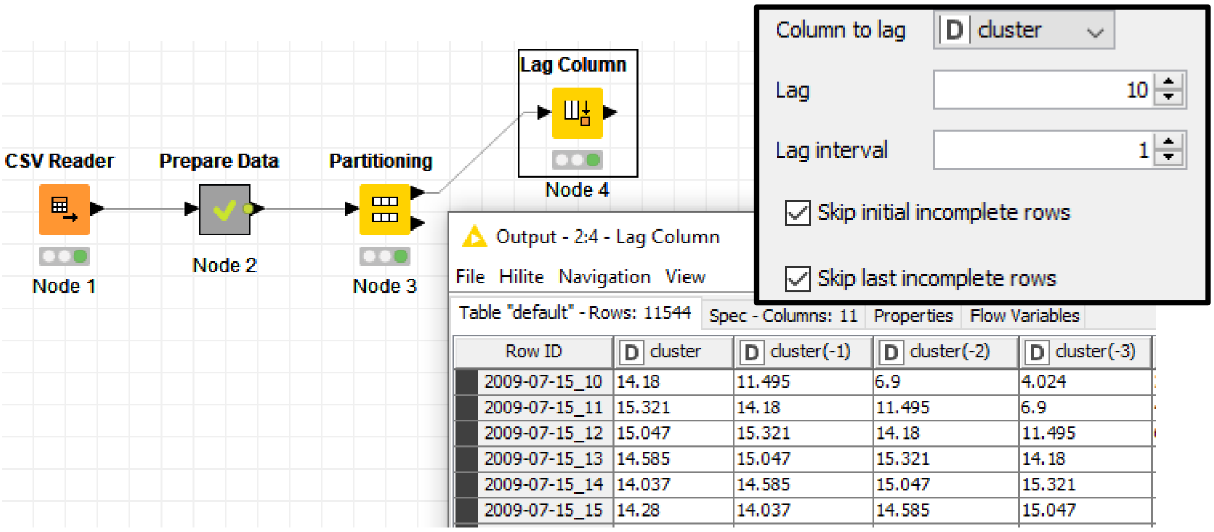
9. Just to make life easier, we shall rename "cluster_26" to simply "cluster":



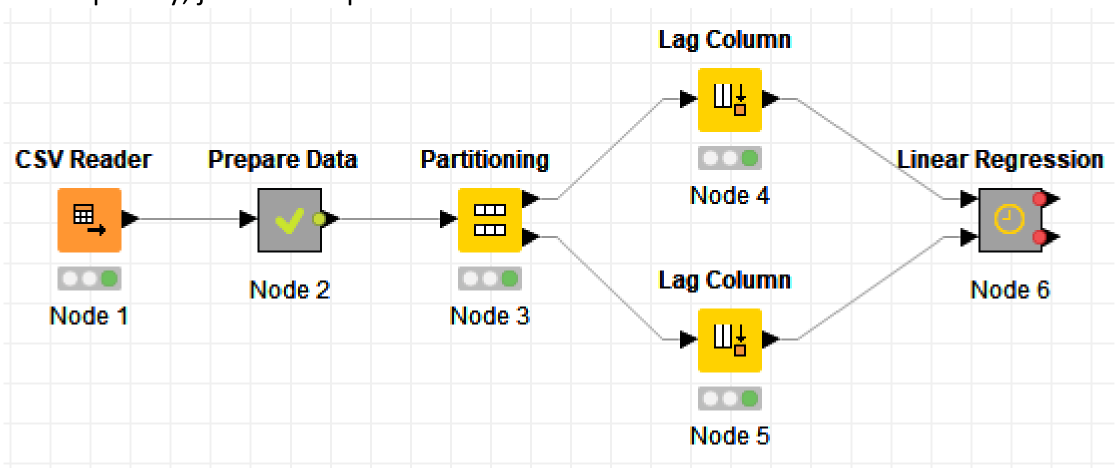
10. Let's go back to the main workflow (by clicking the correct tab) and introduce a Partitioning node to split the data. We are going to learn from 90% of the past data to predict the latest 10%:



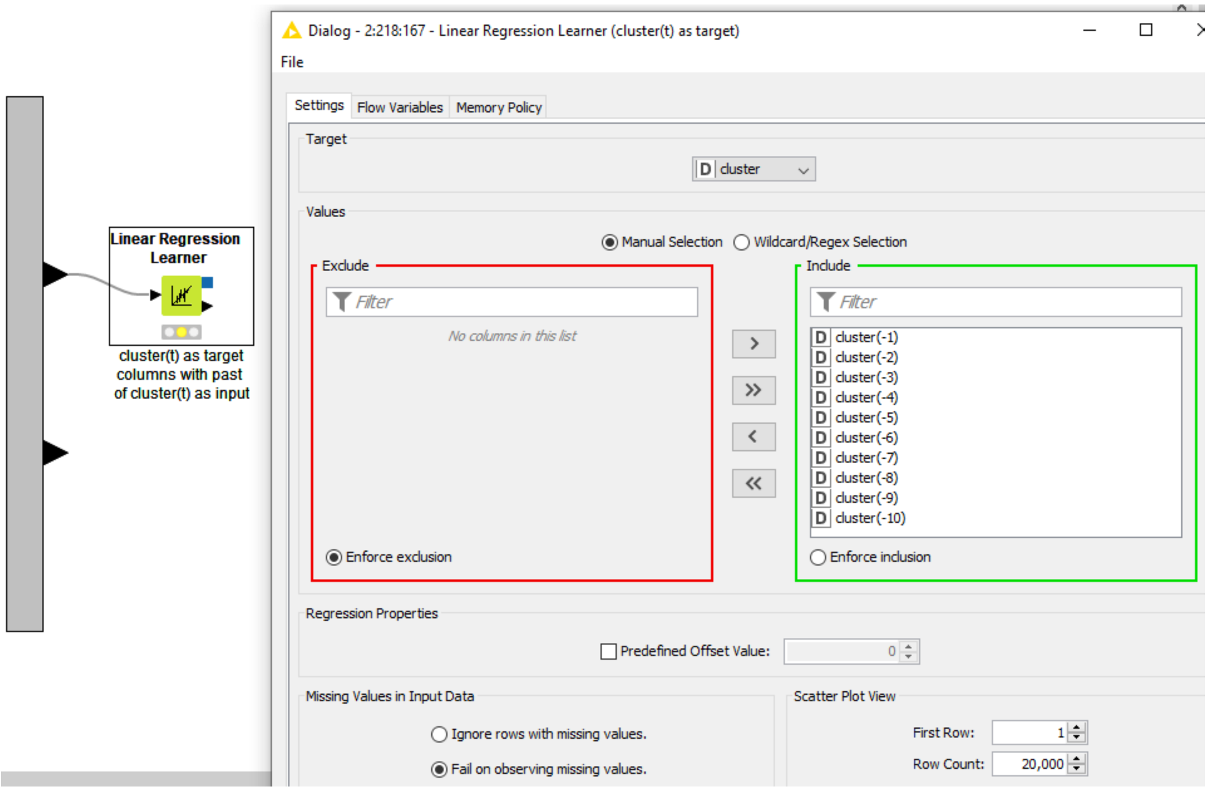
11. We have to properly structure the data to make current data (i.e. cluster) dependent cluster data that comes before it in time (i.e. cluster(-1), cluster(-2) etc…). The *Lag Column* node is used for this purpose. Note how in the final output, the first 10 rows are skipped as they do not have complete lag data, as we have checked the corresponding checkbox.



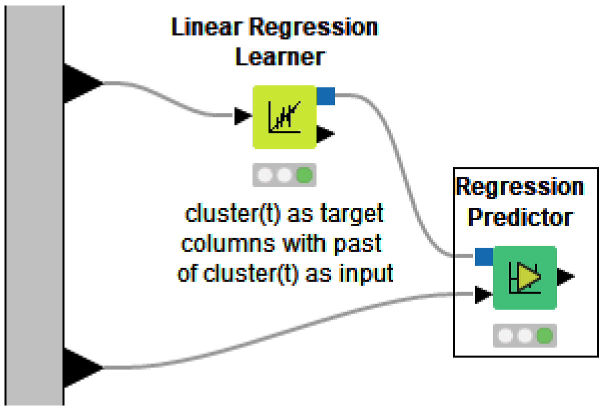
12. Use the same Lag Column node for both the training 90% data and testing 10% data. Subsequently, join them up with a 2:2 metanode:



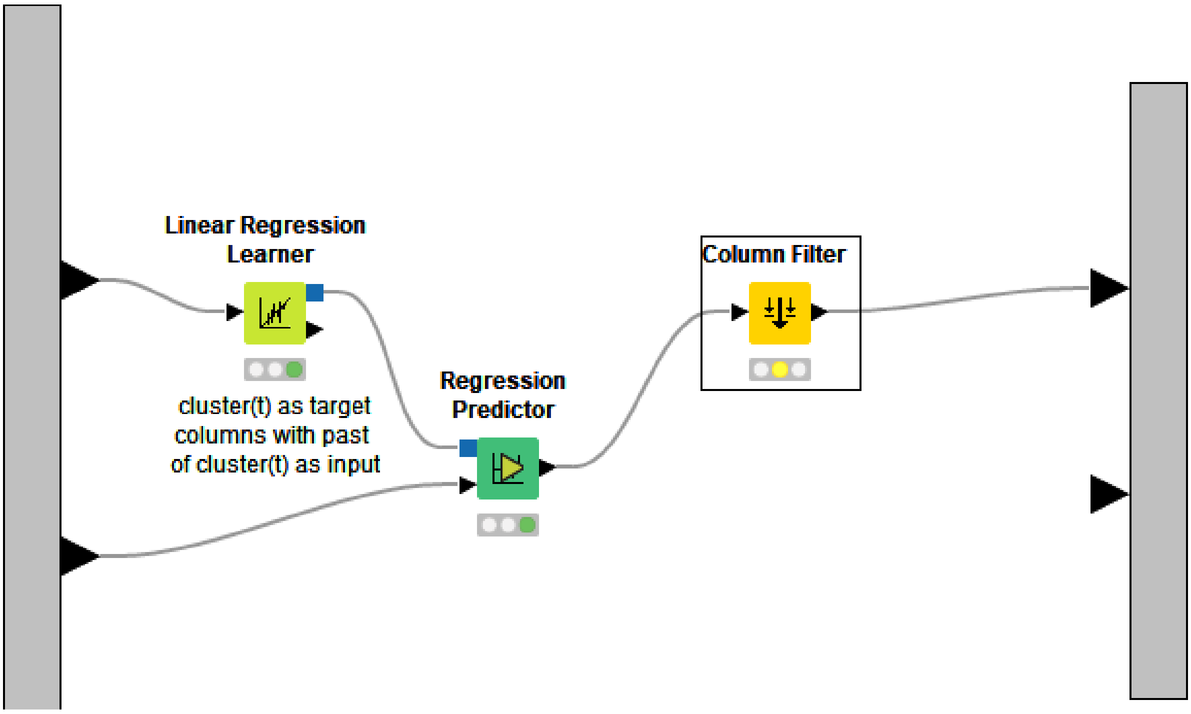
13. In this metanode, put in a Linear Regression Learner to learn from the training 90% data:

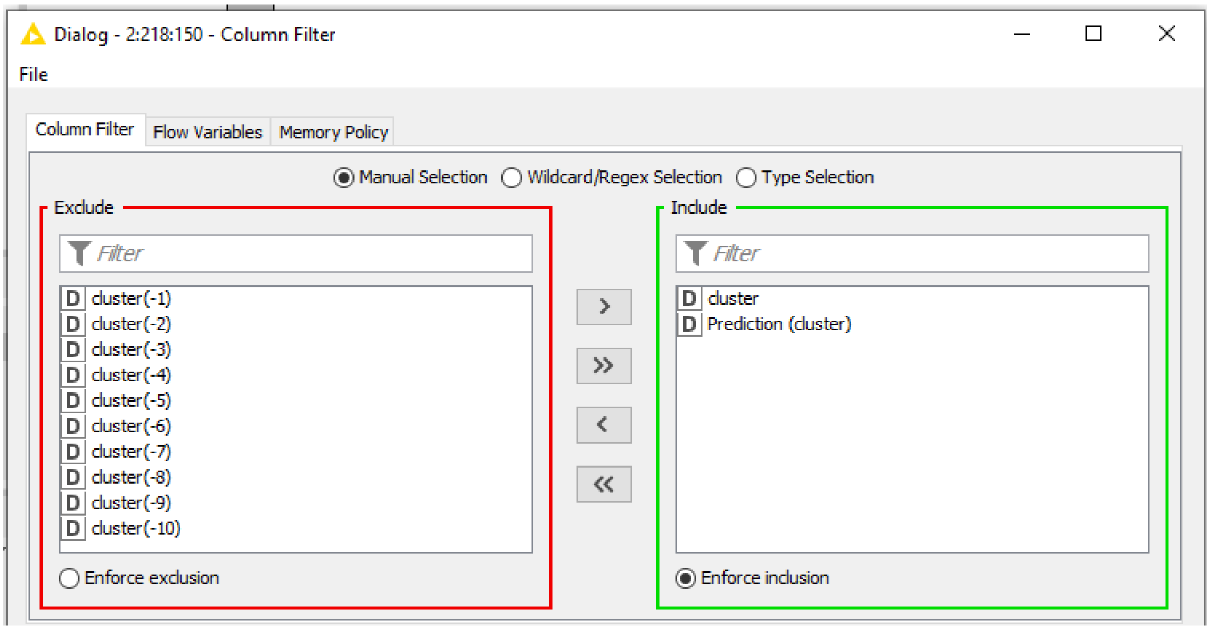


14. Next, apply the learner's model to the test 10% data:

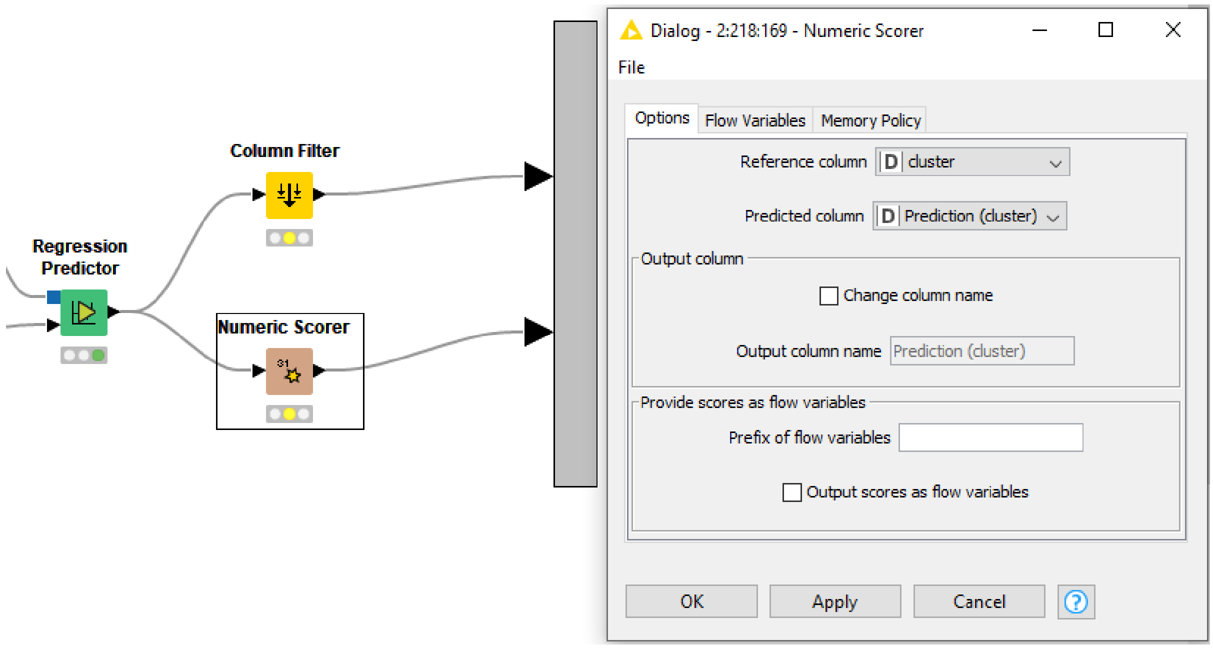


15. Now, since we are only interested in current data and its prediction (all the cluster(-i) data are redundant as the model is trained), use a *Column Filter* node to remove them.





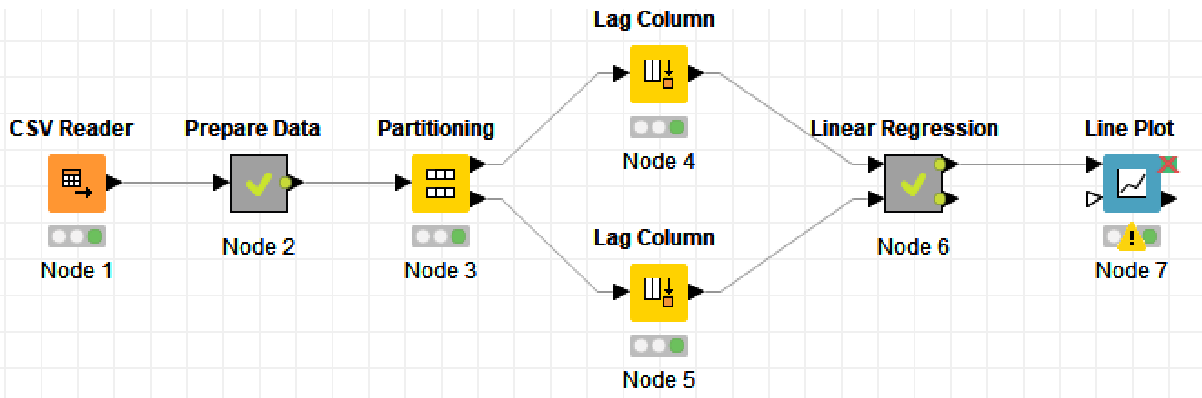
16. We can put in a *Numeric Scorer* node to assess the accuracy of the model on the test data:



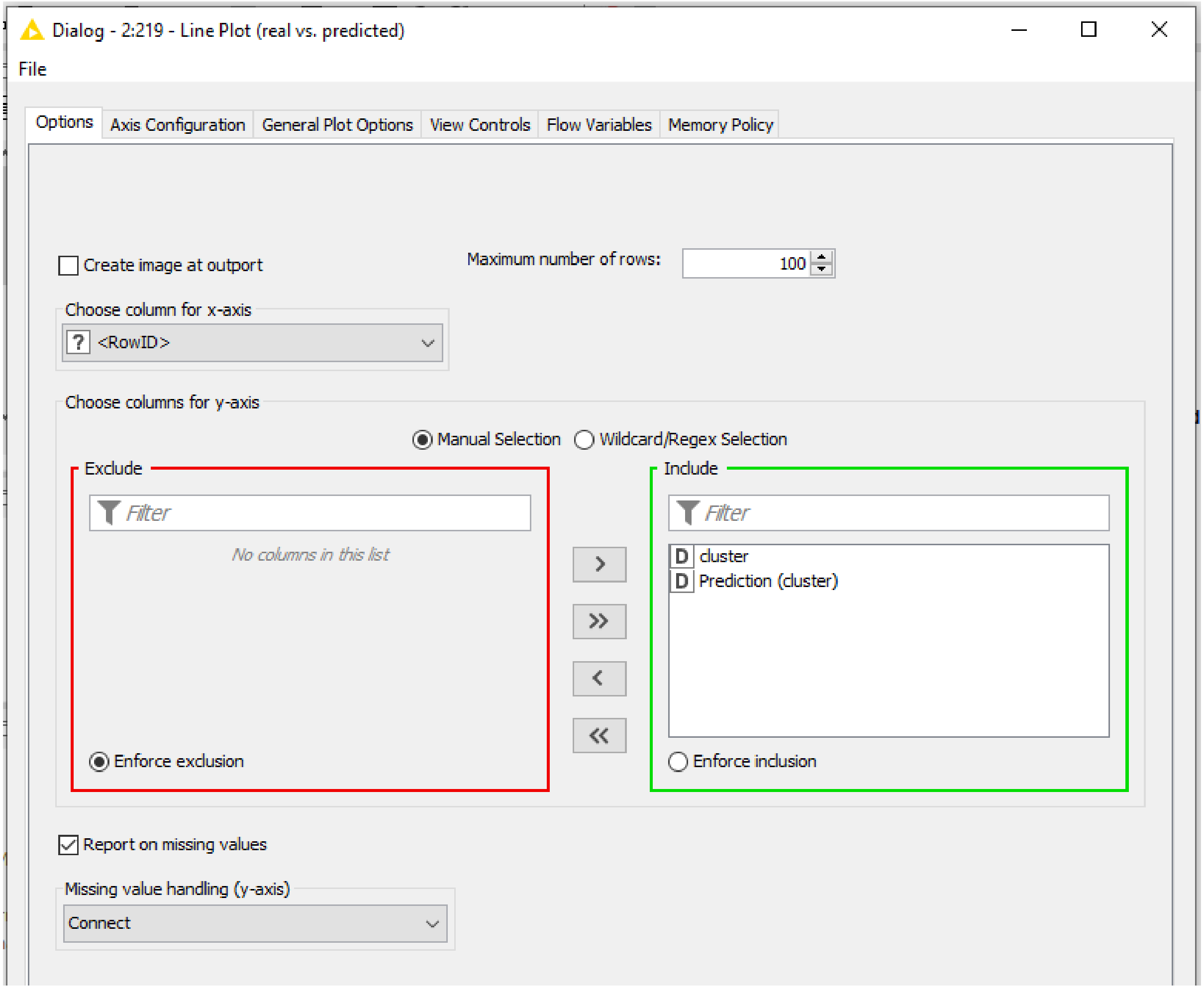
The output tatistics of the scorer tells us that the model has quite a high r-square value:

Row ID	D Predicti...
R^2	0.959
mean absolut...	0.773
mean square...	1.366
root mean sq...	1.169
mean signed ...	-0.115
mean absolut...	0.105

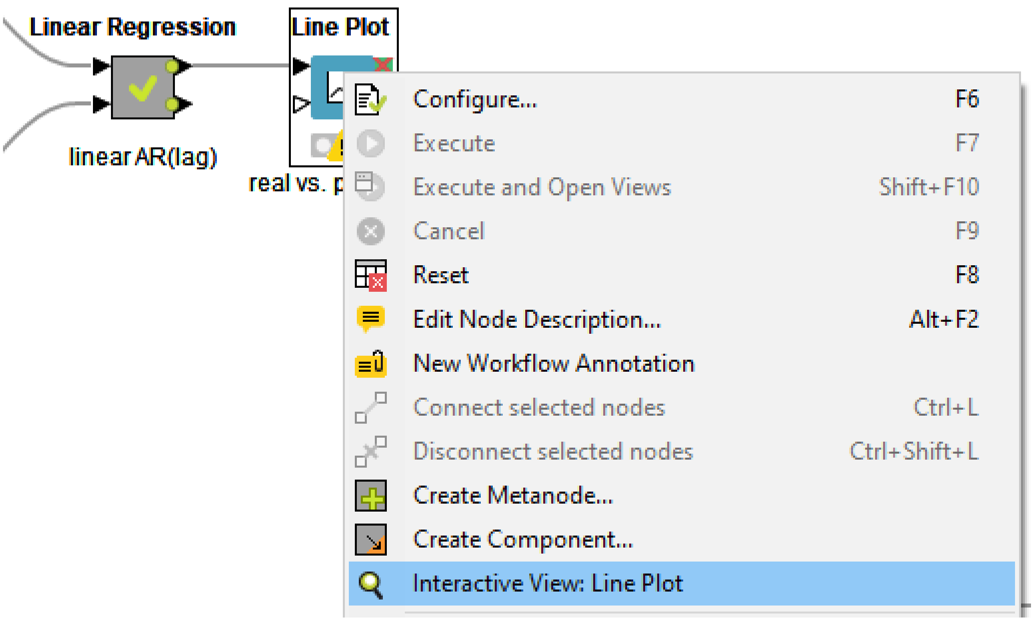
17. Let's visualize the actual time series data with the prediction using a line plot:



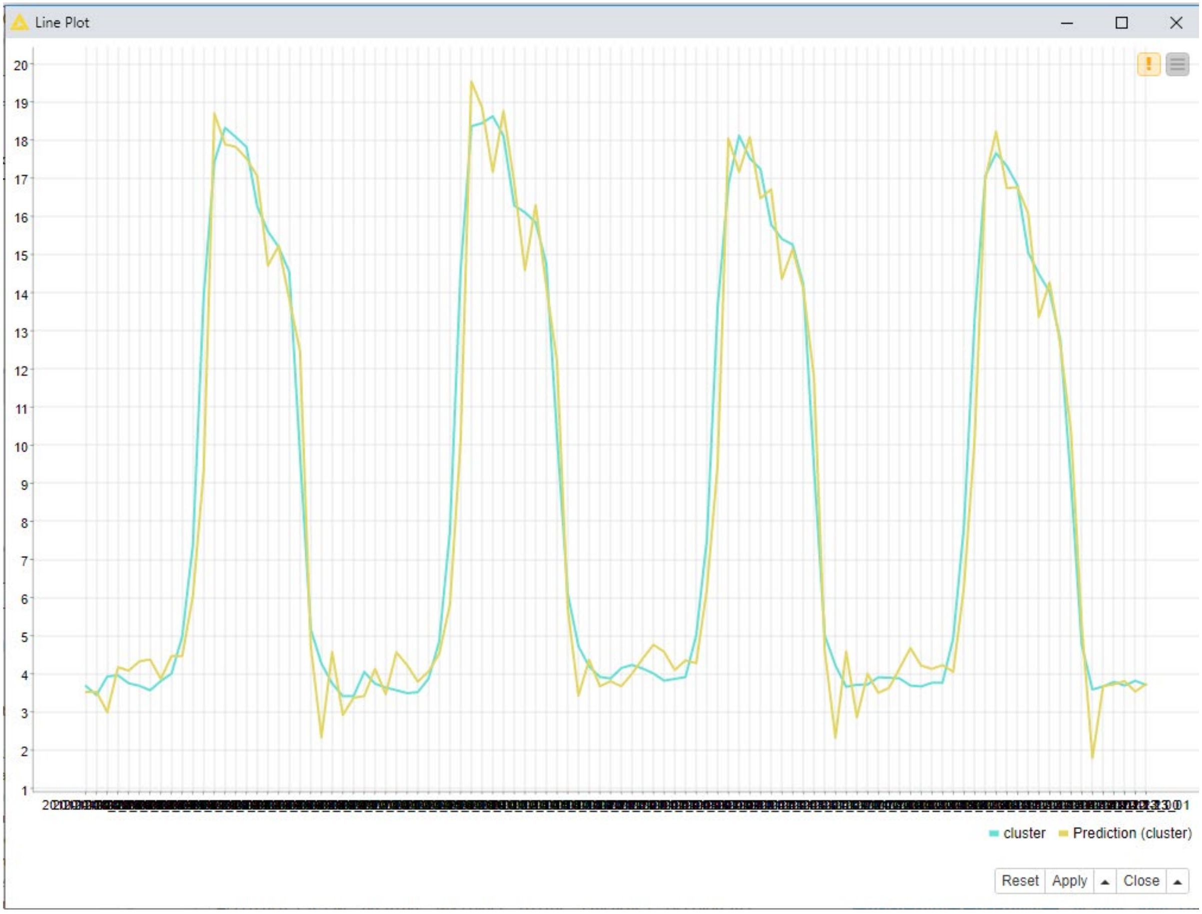
18. We can configure the *Line Plot* node as such:



19. Excecute the *Line Plot* node and open up the “Interactive View: Line Plot” once it is done.



20. The line plot shows that the prediction is quite accurate as compared to the actual cluster data. The prediction seems to have some noise, which one can try to decrease by using more past data for learning.



Conclusion

We have seen how to use KNIME to use linear regression to predict time series data.