

Association rule mining

#ReportRegressionTesting #Python #KNIME #pySpark #AWSGlue #MarketBasket #AssociationMining

#MarketBasketAnalysis #FPGrowth #Apriori #NaiveBayes #DataMocking #MachineLearning #UnsupervisedLearning

Summary

Differences – a **risk management** system generates a report with a million records and each record shows key properties of a financial contract and various risk sensitivities and P/L components. When the system is upgraded (data or logic), the re-generated report is (very) different: one million contracts, thirty columns, two million different cells. But we have reasons to believe that the root causes are only a handful and we want to define a model that can classify the differences and the properties that the affected contracts have in common.

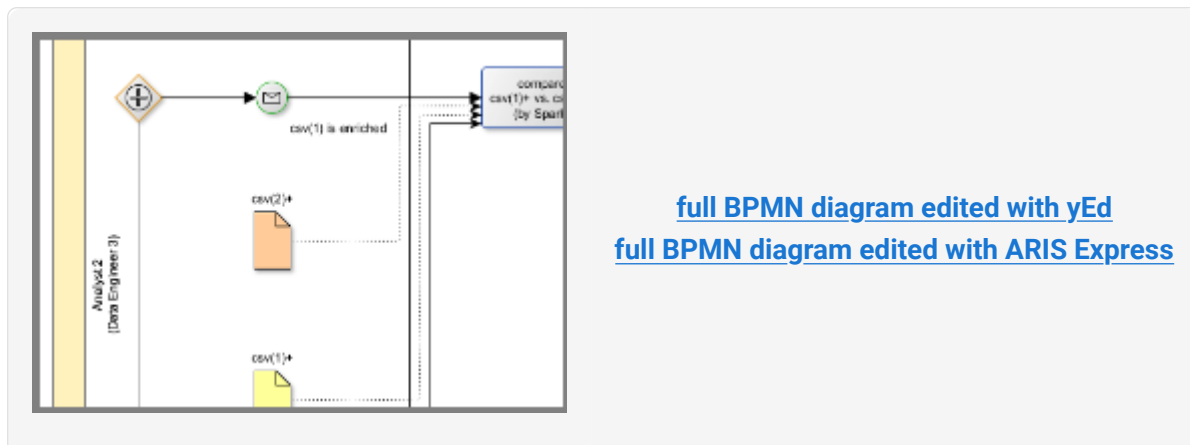
Framing – we re-frame this problem as a **Market Basket Analysis**. A difference in the PV of one contract attributable to FX delta is like a can of beer and the contract domestic currency being EUR is like a box of diapers ^[1]. We go and investigate the shopping patterns.

Complexity - The implementation of the M.B.A. is usually based on the Naive Bayes algorithm.

Unfortunately, its complexity is $O(N!)$ and the size of our report is too big for desktop hardware and time. The FPGrowth implemented in the pySpark MLLIB library is an unsupervised machine learning algorithm that is designed for big-data cases.

Infer – after preparing a mock report and applying fabricated differences by test rules, we build and run an algorithm on the AWS Glue service that infer back our rules in a couple of seconds.

Below are the links to the BPMN diagram that shows the entire process in which we create a mock report, we apply fabricated rules, we fit the model, we calibrate it and we verify the inferred rules.



Preparation

Baseline report generator

The implementation of the process **mock data generator** of the data-flow-diagram is based on the python Faker library and available here:

<https://github.com/a-moscatelli/home/blob/main/am-wiki-assets/bayesdatamining/faker-datamining.py>



Generated baseline report

This is an excerpt of the CSV file generated by the process above.

The independent variables are the columns from 'trade_date' to 'delim'.

The dependent variables are the columns from 'delim' to '* PnL' (realized PnL, unrealized PnL, total PnL = R + U).

```

1 id,trade_date,exp_date,trader,trade_ccy,product,disc_curve,fx_cut,market,PnL
2 0,2023-05-04,2023-06-09,Shannon Byrd,USDALL,fx swap,USDALL-GE,TKY1500,GE,USD
3 1,2023-05-14,2023-06-25,Ashley Wilson,USDBWP,fx spot,USDBWP-GM,TKY1500,GM,US
4 2,2023-04-29,2023-06-02,Jasmine Baker,USDFJD,fx spot,USDFJD-AG,NYC1000,AG,US
5 3,2023-05-14,2023-06-16,Ashley Gilmore,USDAZN,fx futures,USDAZN-BW,TKY1500,B
6 4,2023-04-26,2023-05-13,Roberta Wade DVM,USDTMT,fx spot,USDTMT-WS,TKY1500,WS
7 5,2023-05-13,2023-05-14,Jennifer Jones,USDZMW,fx fwd,USDZMW-GN,NYC1000,GN,US
8 6,2023-05-11,2023-06-29,Rebecca Flores,USDMNT,fx spot,USDMNT-GD,NYC1000,GD,U
9 7,2023-04-25,2023-06-02,Stephen Murray,USDPAB,fx swap,USDPAB-OM,NYC1000,OM,U
10 8,2023-05-02,2023-05-08,John Franklin,USDLSL,fx futures,USDLSL-TH,TKY1500,TH
11 9,2023-04-28,2023-05-05,Jennifer Roberts,USDSRD,fx futures,USDSRD-SK,ECB1415

```

Execution

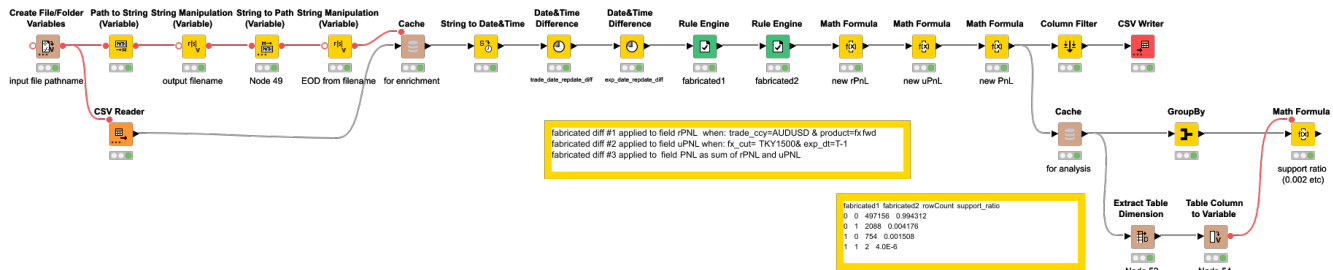
Feature engineering (i.e. rule engine and signal derivation)

With the processes **rule engine** and **signal derivation** of the data-flow-diagram, we are going to generate the CANDIDATE report through the application of a few rules in order to get a set of fabricated differences.

Then, we are going to calculate a few derived variables.



This step is done to transform our *discrete* variables (like the `time_to_expire`), into categorical ones (like `expired_today true/false`, `expired_yesterday true/false`), that are accepted by our data mining algorithm.



[Hi-res picture](#)

After running the workflow above (see [KNIME 4.6](#)) we have a CANDIDATE report and the derived signals. An example of signal expressions is as follows:

```

1 | signal_1m = (expiry_date == report date - 1)
2 | signal_1e = (expiry_date == report date)
3 | signal_1p = (expiry_date == report date + 1)
4 | signal_2p = (trade_date == report date + 1)
5 | ...

```



Providing these extra variables to the rule miner is a way of verifying further hypotheses we have in mind:
is there any systematic issue with
the price computed for a class of trades expiring tomorrow?

The data miner

In the process **data miner** we are running the FPGrowth provided with pySpark.

```

1 | # docker
2 | services:
3 |     pyspark:
4 |         image: "jupyter/pyspark-notebook:notebook-6.5.4"

```

FPGrowth is an enhancement of **Apriori** which is an enhancement of the **Naive Bayes**. It is an unsupervised machine learning implementation.

An example of Apriori implementation is available at:

https://github.com/a-moscatelli/DEV/tree/main/association_mining

The FPGrowth documentation supported by pySpark is available [here](#) 



The FPGrowth requires the specification of a minimum support and confidence. The **minimum support** we provide should be lower than the occurrence of the fabricated differences.

Results

Inferred rules

We now execute the process **p3a** of the data-flow-diagram.

Inferred rules for realized PnL: [Jupyter notebook](#) 

SN	antecedent	consequent	confidence	lift	support	lenA	lenC
1	[PnL_ccy=AUD, product=fx fwd]	[rPnL_diff=1]	1	661.38	0	2	1
2	[trade_ccy=AUDUSD, product=fx fwd]	[rPnL_diff=1]	1	661.38	0	2	1
3	[PnL_ccy=AUD, product=fx fwd, exp_date_m1=0]	[rPnL_diff=1]	1	661.38	0	3	1
4	[trade_ccy=AUDUSD, product=fx fwd, exp_date_m1=0]	[rPnL_diff=1]	1	661.38	0	3	1
5	[PnL_ccy=AUD, trade_ccy=AUDUSD, product=fx fwd]	[rPnL_diff=1]	1	661.38	0	3	1
6	[PnL_ccy=AUD, trade_ccy=AUDUSD, product=fx fwd, exp_date_m1=0]	[rPnL_diff=1]	1	661.38	0	4	1



The implied rule 2 is exactly the rule that was applied to introduce fabricated differences. The implied rule 1 is a byproduct of rule 2: in our input, $\text{trade_ccy}=\text{AUDUSD} \Rightarrow \text{PnL_ccy}=\text{AUD}$

Inferred rules for unrealized PnL: [Jupyter notebook](#) 

SN	antecedent	consequent	confidence	lift	support	lenA	lenC
1	[exp_date_m1=1, fx_cut=TKY1500]	[uPnL_diff=1]	1	239.23	0	2	1
2	[exp_date_m1=1, fx_cut=TKY1500, PnL_ccy=USD]	[uPnL_diff=1]	1	239.23	0	3	1




The implied rule 1 is exactly the rule that was applied to introduce fabricated differences.
The implied rule 2 is less general than rule 1 - but can still give clues about the root causes of the diffs.

AWS test run

The rule mining script above,
executed on AWS Glue 3.0 with the default settings,
could process our half-million-record file pair
in 1 min 55 s
at a cost of USD 0.15
(Apr-2023)

```
1 | aws glue start-job-run --job-name JOBNAME --region REGION --number-of-worker
2 | :: default settings: 10 DPUs, G.1X
3 | :: execution: 0.32 DPU hours
```

Full program: <https://ampub.s3.eu-central-1.amazonaws.com/am-wiki-js/rulemining/FPG-FPmining-adapted-for-glue.py> 

Back to [Portfolio](#)

[backlog](#) 

1. <https://tdwi.org/articles/2016/11/15/beer-and-diapers-impossible-correlation.aspx>  

