A Practical Guide for Stock Prediction using News Text Mining

Technical Report

Marcelo Beckmann
beckmann.marcelo@gmail.com

Nelson Ebecken
nelson@ntt.ufrj.br

Beatriz S. L. Pires De Lima
bia@coc.ufrj.br

Federal University of Rio de Janeiro, Civil Engineering Program/COPPE, Rio de Janeiro, Brazil.

Abstract

This article explains how to reproduce experiments and make new developments using the methodology developed in (Beckmann, et al., 2017). This previous work presented a computational framework that demonstrated evidences about the efficacy to use only new articles to predict the changes of stock prices.

So far, the presented results of experiments in terms of classification measures and the cumulative return obtained through investment simulation outperformed the other results found after an extensive review in the related literature.

Recently the methodology developed in this previous work became available as an open source project, and together with the source code, data, and experiments, a public Amazon Machine Image (AMI) with all these components ready to use is also available. This technical report explains how to use these resources.

Keywords

Financial Markets, Stock Market Prediction, Predictive Analytics, Natural Language Processing, Text Mining, Sentiment Analysis, Data Mining

DOI: 10.13140/RG.2.2.32112.71685

Introduction

The price forecasting in the Stock Market and other markets is a challenging task, and is still an open problem in science, as ultimately it deals with the unpredictability nature of human behaviour. In order to contribute with this branch of research, (Beckmann, et al., 2017) presented a computational framework that demonstrated evidences about the possibility to predict intraday stock prices, given the occurrence of external sources, in this case, the news articles related to the companies listed in the Down Jones Index. It is recommended the reading of this previous work for a better understanding of the next steps in this paper.

With the purpose to reproduce the experiments published in (Beckmann, et al., 2017), henceforth referred as Thesis, and provide an honest baseline for new developments in the branch of text mining and sentiment analysis applied to financial markets, in March of 2018 the Thesis's authors made its methodology comprising of source code, data, and experiments open source. All this set of software artefacts are packaged in a platform named TradeMiner. The TradeMiner platform is a RapidMiner extension (Mierswa, et al., 2006) developed in Java, and provides a scalable and robust environment for a recommendation system applied to any kind of financial markets (Stocks, Foreign Exchange, Derivatives, etc.).

What is available?

- The open source code of TradeMiner RapidMiner extension [1]
- The repository of TradeMiner experiments [2]
- The public Amazon Machine Instance (AMI) with the all set environment to run the TradeMiner experiments and make new developments [3]

The Amazon Web Services (AWS) provides on-demand cloud computing platforms on a paid subscription basis. One of the features of AWS is the Elastic Compute Cloud (EC2), that provides a pre-saved TradeMiner environment ready to run, through a public AMI named "TradeMiner AMI".

This article is divided in two parts: how-to reproduce experiments, and how-to make new developments with TradeMiner. These sections provide all the steps and tools to validate the published results, understand the source code, experiments, and database to make new developments in this platform. All the explanations and tutorials are based on the public TradeMiner

[1] https://github.com/marcelobeckmann/trademiner
[2] https://github.com/marcelobeckmann/trademiner/tree/master/REPOSITORY
[3] https://aws.amazon.com

AMI, and the description of parameters, tables, main directory structures, and files can be found in the Appendixes at the end.

Reproducing the Experiments

In scientific research, the reproductivity of experiments is an important step for understanding how a methodology is implemented, and for the acknowledge of published results. Even with the purpose of new developments, it is interesting to check if everything is running correctly, and as a baseline for future results.

Technical skills required: Amazon Web Services operation, Linux command line operation, RapidMiner operation, Machine Learning.

1. Getting the TradeMiner AMI

A public AMI named "TradeMiner AMI" is available to eu-west region. To make use of this AMI, go to your AWS account, open the EC2 dashboard and AMI page, and then select "Public Images" in the drop-down box, and type "TradeMiner AMI", then the respective instance should appear as in Figure 1.

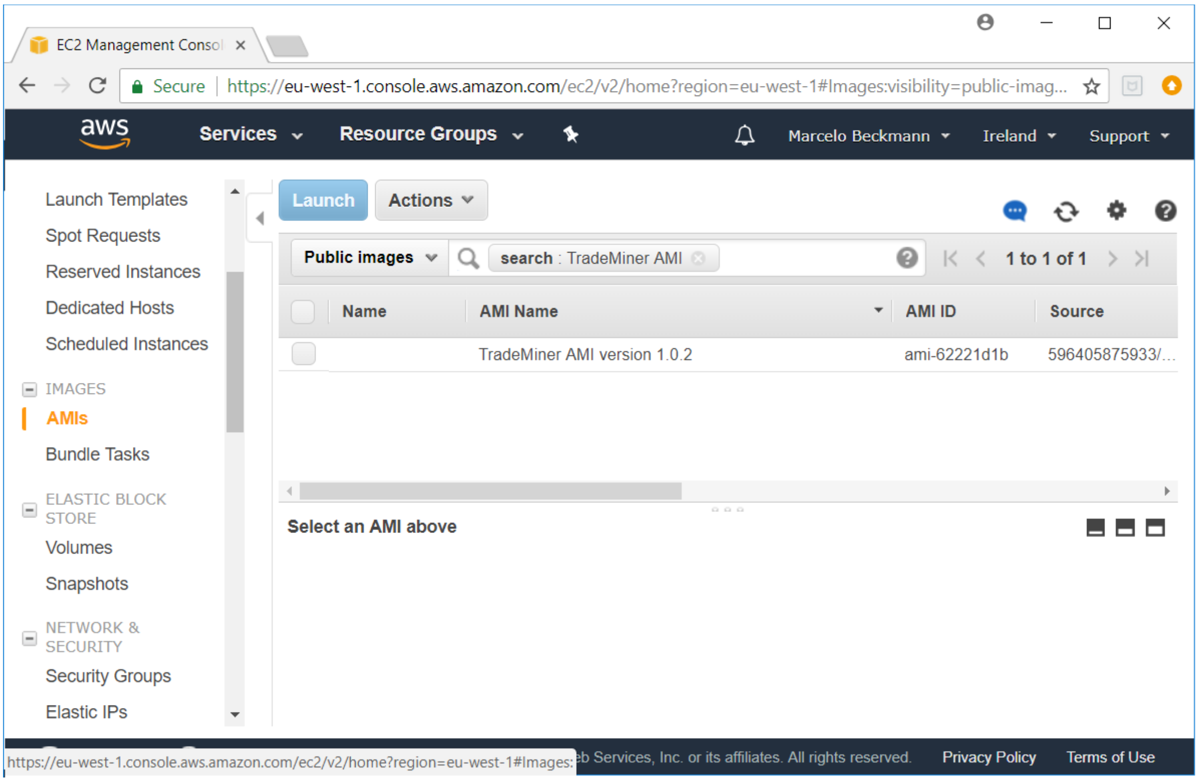


Figure 1 - Search of public TradeMiner AMI

It is possible to make a copy of this public AMI (94GB of size), to your account and region, or launch a new instance to your account, as seen in Figure 2. The minimal recommended configuration for a TradeMiner instance is 4 vCPUs and 8GB of memory.

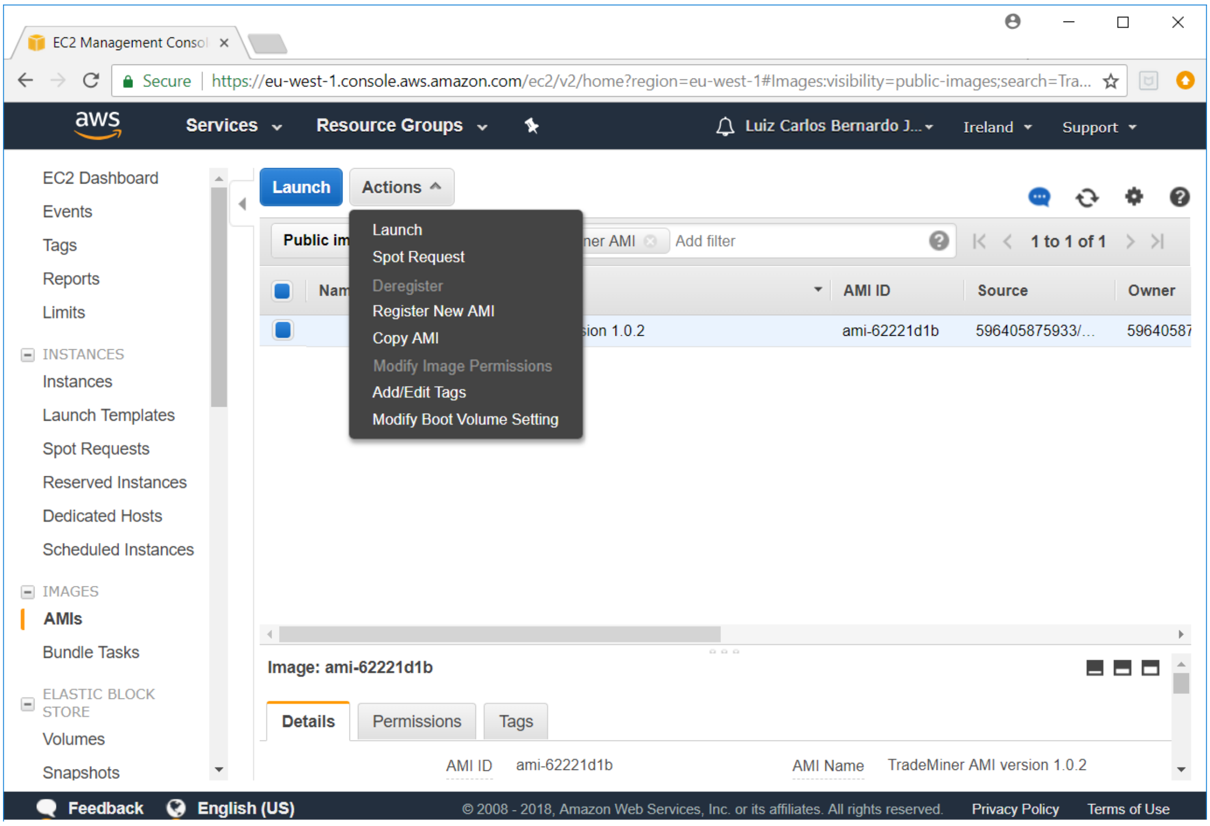


Figure 2 - Actions available for the public TradeMiner AMI.

To make sure you have access to the Remote Desktop of your TradeMiner instance, it is necessary to configure the security group to open the ports 22 and 3389, as depicted in Figure 3.

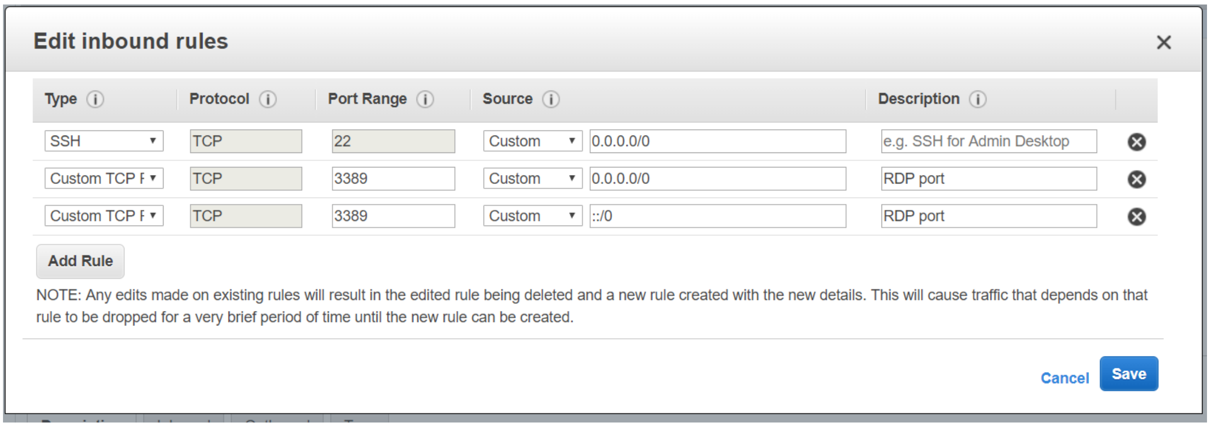


Figure 3 - Security group requirements for Remote Desktop access.

2. Run the classification experiments

Once the TradeMiner instance is started, to run the classification experiments (the test phase as described in section 5.2 of Thesis), it is necessary to connect it via Putty terminal according the security options you choose when launched the TradeMiner instance.

Once connected via SSH terminal, go to directory `/home/ubuntu/scripts` and execute `./0_2_run_all_tests.sh`, after this the classification job will start, and the classification results of the experiment for window size= 1 minute will appear at the end (Figure 4).

ubuntu@ip-172-31-27-201: ~/scripts

```
May 24, 2018 7:50:57 AM com.rapidminer.Process run
INFO: Process //NewLocalRepository/I_test_separated_compute_metrics_agg finished successfully after 35 s
May 24, 2018 7:50:57 AM com.rapidminer.RapidMinerCommandLine run
INFO: Process finished successfully
####### EXPERIMENT RESULTS - CLASSIFICATION ########
+--------+---------------+---------+---------------+------------+--------+---------+--------------+
| symbol | experiment_id | avg_acc | avg_precision | avg_recall | gmean  | avg_auc | avg_fmeasure |
+--------+---------------+---------+---------------+------------+--------+---------+--------------+
| AA     |           837 | 0.9917  |        0.7479 |     0.7479 | 0.7056 | 0.7479  |       0.7479 |
| AXP    |           837 | 0.9235  |        0.5688 |     0.7167 | 0.6832 | 0.7318  |       0.5953 |
| BA     |           837 | 0.9893  |        0.6971 |     0.6793 | 0.6015 | 0.7126  |       0.6878 |
| BAC    |           837 | 0.9905  |        0.6625 |     0.5351 | 0.2671 | 0.6527  |       0.5564 |
| CAT    |           837 | 0.9840  |        0.6949 |     0.6398 | 0.5329 | 0.6507  |       0.6626 |
| CVX    |           837 | 0.9766  |        0.5853 |     0.5937 | 0.4444 | 0.6409  |       0.5893 |
| DD     |           837 | 0.9356  |        0.5661 |     0.6422 | 0.5631 | 0.6424  |       0.5867 |
| DIS    |           837 | 0.9926  |        0.9963 |     0.5625 | 0.3536 | 0.6342  |       0.6092 |
| GE     |           837 | 0.9845  |        0.5877 |     0.8677 | 0.8597 | 0.6459  |       0.6389 |
| HD     |           837 | 0.9808  |        0.6930 |     0.5973 | 0.4460 | 0.6403  |       0.6285 |
| HPQ    |           837 | 0.9861  |        0.7436 |     0.5379 | 0.2772 | 0.6271  |       0.5632 |
| IBM    |           837 | 0.9918  |        0.9959 |     0.6000 | 0.4472 | 0.6248  |       0.6646 |
| INTC   |           837 | 0.9970  |        0.9985 |     0.6250 | 0.5000 | 0.6250  |       0.6993 |
| JNJ    |           837 | 0.9820  |        0.9910 |     0.5385 | 0.2774 | 0.6161  |       0.5669 |
| JPM    |           837 | 0.9929  |        0.6641 |     0.5823 | 0.4078 | 0.6134  |       0.6093 |
| KO     |           837 | 0.9899  |        0.9950 |     0.5556 | 0.3333 | 0.6100  |       0.5975 |
| MCD    |           837 | 0.9888  |        0.7958 |     0.6652 | 0.5765 | 0.6131  |       0.7115 |
| MMM    |           837 | 0.9200  |        0.5926 |     0.9593 | 0.9584 | 0.6241  |       0.6350 |
| MRK    |           837 | 0.9249  |        0.6189 |     0.8574 | 0.8544 | 0.6440  |       0.6665 |
| MSFT   |           837 | 0.9977  |        0.9988 |     0.5833 | 0.4082 | 0.6424  |       0.6423 |
| PFE    |           837 | 0.9857  |        0.9928 |     0.5909 | 0.4264 | 0.6394  |       0.6502 |
| PG     |           837 | 0.9899  |        0.9949 |     0.5714 | 0.3780 | 0.6370  |       0.6224 |
| T      |           837 | 0.9938  |        0.7473 |     0.5710 | 0.3778 | 0.6349  |       0.6095 |
| TRV    |           837 | 0.9916  |        0.9958 |     0.6667 | 0.5774 | 0.6353  |       0.7479 |
| UNH    |           837 | 0.9815  |        0.9907 |     0.6667 | 0.5774 | 0.6358  |       0.7453 |
| UTX    |           837 | 0.8323  |        0.5123 |     0.5877 | 0.5298 | 0.6352  |       0.4896 |
| VZ     |           837 | 0.0090  |        0.2175 |     0.3195 | 0.0417 | 0.6188  |       0.0090 |
| WMT    |           837 | 0.9908  |        0.6968 |     0.6097 | 0.4707 | 0.6187  |       0.6405 |
| XOM    |           837 | 0.9407  |        0.5607 |     0.7645 | 0.7427 | 0.6256  |       0.5891 |
+--------+---------------+---------+---------------+------------+--------+---------+--------------+
ubuntu@ip-172-31-27-201:~/scripts$
```

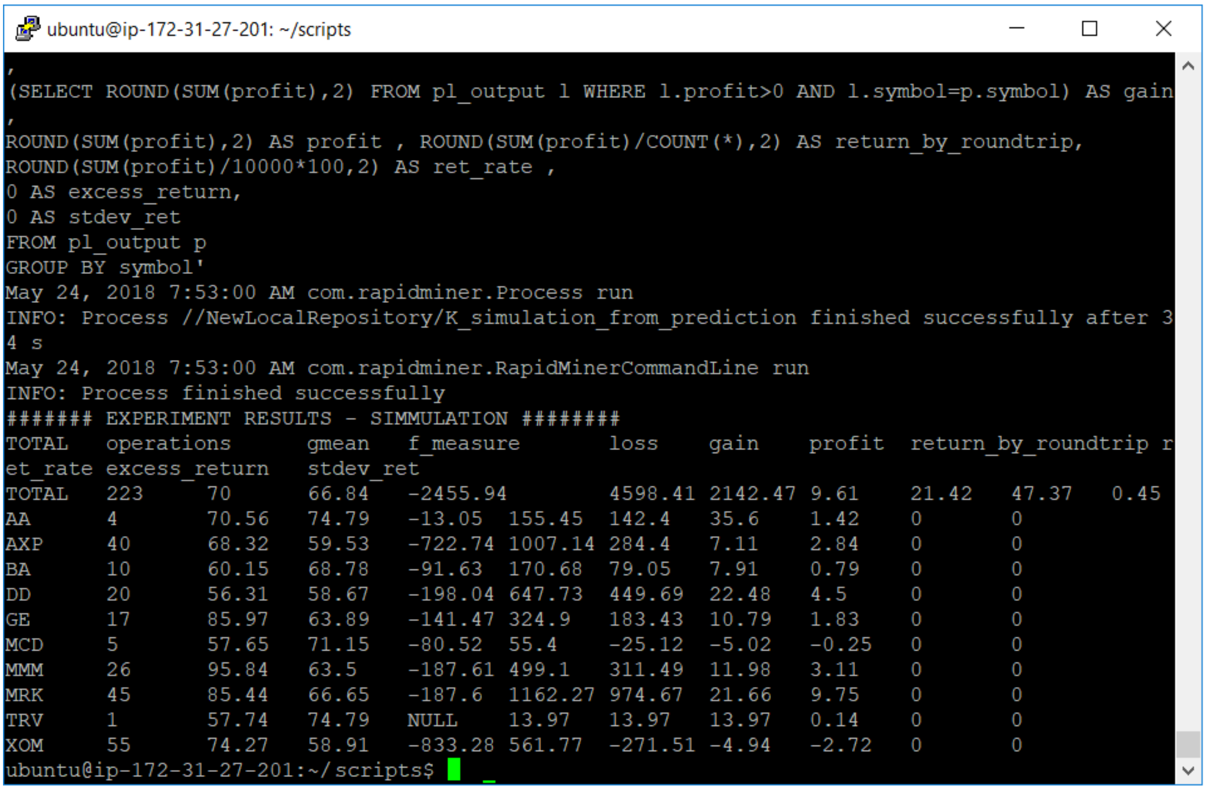
Figure 4 - Classification results for window size=1.

Alternatively, you can retrieve the last classification and simulation results any time, executing the script `./0_4_show_last_results.sh`. These results are stored in the pl_output and experiment_result_auc4 MySQL tables.

Note: The results published in the section 5.2 of Thesis are an average after 10 runs. It is not expected to get exactly those results, but an approximated classification performance will be obtained, and will be around the published results. The same occurs with investment simulation. Please send us an email if the expected results are not achieved, and if the initial experiment conditions were not changed.

3. Run the simulation experiments

To run the simulation experiments (as described in section 5.3 of Thesis), start the script `./0_3_run_simulation.sh` in the same directory, then the similar results like Figure 5 will appear.



```
ubuntu@ip-172-31-27-201: ~/scripts
,
(SELECT ROUND(SUM(profit),2) FROM pl_output l WHERE l.profit>0 AND l.symbol=p.symbol) AS gain
,
ROUND(SUM(profit),2) AS profit , ROUND(SUM(profit)/COUNT(*),2) AS return_by_roundtrip,
ROUND(SUM(profit)/10000*100,2) AS ret_rate ,
0 AS excess_return,
0 AS stdev_ret
FROM pl_output p
GROUP BY symbol'
May 24, 2018 7:53:00 AM com.rapidminer.Process run
INFO: Process //NewLocalRepository/K_simulation_from_prediction finished successfully after 3
4 s
May 24, 2018 7:53:00 AM com.rapidminer.RapidMinerCommandLine run
INFO: Process finished successfully
####### EXPERIMENT RESULTS - SIMMULATION ########
TOTAL   operations      gmean   f_measure       loss    gain    profit  return_by_roundtrip r
et_rate excess_return   stdev_ret
TOTAL   223     70      66.84   -2455.94        4598.41 2142.47 9.61    21.42   47.37   0.45
AA      4       70.56   74.79   -13.05  155.45  142.4   35.6    1.42    0       0
AXP     40      68.32   59.53   -722.74 1007.14 284.4   7.11    2.84    0       0
BA      10      60.15   68.78   -91.63  170.68  79.05   7.91    0.79    0       0
DD      20      56.31   58.67   -198.04 647.73  449.69  22.48   4.5     0       0
GE      17      85.97   63.89   -141.47 324.9   183.43  10.79   1.83    0       0
MCD     5       57.65   71.15   -80.52  55.4    -25.12  -5.02   -0.25   0       0
MMM     26      95.84   63.5    -187.61 499.1   311.49  11.98   3.11    0       0
MRK     45      85.44   66.65   -187.6  1162.27 974.67  21.66   9.75    0       0
TRV     1       57.74   74.79   NULL    13.97   13.97   13.97   0.14    0       0
XOM     55      74.27   58.91   -833.28 561.77  -271.51 -4.94   -2.72   0       0
ubuntu@ip-172-31-27-201:~/scripts$
```

Figure 5 - Simulation results for window = 1 minute.

Optional Steps

4. Run all the experiments

In the steps 2 and 3 the last part of the experiment was executed (i.e., the test phase and simulation), but it is also possible the entire process, since the retrieval of stored news from the MySQL database, transformation, training, test, and simulation.

To execute this, run `./0_1_run_all_gen_train_test.sh` in the scripts directory.

Note 1: this is a data and CPU consuming process that will take in average 2 to 3 days to be finished, depending the CPU and memory configuration you set for your AWS instance.

Note 2: All the intermediate and results are generated in the `/data` directory, in Comma Separated Format (CSV) files.

By default, the TradeMiner instance is set to execute experiments for a time offset = 1 minute. This means that all the intermediate and result files will be written (or re-written) in the `/data/output_1` directory.

5. Changing the time offset

In the TradeMiner experiments, the macros.csv is a centralized file that contains all the parameters for the entire process. A change in one of these parameters can change the final results. The file is self-commented with references to the thesis sections. For further details, see Appendix A - Parameter Setup with macros.csv.

There are five macros.csv in the ~/REPOSITORY (Table 1). Each macros_*.csv file represents a set of parameters for time offset 1, 2, 3, and 5 minutes respectively, and by default macros.csv = macros_#763_w1.csv.

Table 1- List of macros.csv parameter files.

```
macros.csv
macros_#763_w1.csv
macros_#758_w2.csv
macros_#777_w3.csv
macros_#769_w5.csv
```

To change the time offset experiment, just copy the respective macro file to macros.csv.

Note: The current AMI is set with 94GB, and around 60% of this is occupied by the intermediate files and results tor time offset 1 and 2. Because of this, more disk space needs to be added to the AWS instance, or deleting the directories `/data/output_1` or `/data/output_2`, if the purpose is to test other time offsets, or try other experiments.

The reproduction of experiments with TradeMiner demonstrates a proof of concept that it is possible to predict the Stock Markets using only text mining and sentiment analysis. Feel free audit the code and experiments to check if there is no cheating in some part of the process, and if the methodology can be applied in a real-life scenario.

New Developments

It is understandable how is difficult to assemble a large data analytics process since the beginning, even with a good methodology in mind. Beyond the experiment reproductivity, it is also important to provide to the scientific community a stable and open platform for new developments in text mining and sentiment analysis applied to Financial Markets.

To start new developments with the TradeMiner platform it is necessary to understand what is inside the TradeMiner Instance, and how to modify and create new experiments, and this will be provided in the next steps.

Technical skills required: AWS Operation, Linux command line operation, RapidMiner operation, Java language development (or the language of your preference if you don't want to create new operators in RapidMiner), MySQL query language, Machine Learning.

1. Exploring the TradeMiner instance

Table 2 lists the software installed, and Figure 6 describes the most important directories in the TradeMiner instance.

Table 2 - List of installed software in the TradeMiner AMI.

Ubuntu 14.04
Java 1.7
RapidMiner 5.2 Community Edition
Eclipse 3.8
MySQL 5.5.6
Git 1.9.1
TradeMiner source code 5.0.0
TradeMiner MySQL database

```
\
├── data                           Data directory
│   ├── logs                       Webcrawler logs
│   ├── output_1                   Output files of experiments with time offset=1 minute
│   │   ├── D0603                  Data split folder with generated CSVs between 03/Jun to 07/Jun
│   │   ├── ...                    Several weekly split folders until September/2013
│   │   └── D0902                  Data split folder with generated CSVs between 02/Sep to 06/Sep
│   └── output_2                   Output files of experiments with time offset=2, same as output_1
└── home
    └── ubuntu                     Home directory for ubuntu user
        ├── Desktop                Desktop files
        ├── RapidMiner5            RapidMiner installation directory
        ├── REPOSITORY             Symbolic link to git repository with RapidMiner experiments
        ├── scripts                Scripts to start up experiments trough command line
        │   └── run_by_step        Scripts to start up experiments step by step
        └── workspace              Eclipse workspace
            └── trademiner         TradeMiner git cloned repository
                ├── logmon         Source code of Log Monitor application
                ├── RapidMiner_Trademiner_Unuk      Source code of TradeMiner extension for RapidMiner
                ├── RapidMiner_Unuk                 Source code of RapidMiner
                └── REPOSITORY                      Repository with TradeMiner experiments
                    └── webcrawler                  WebCrawler scripts
```

Figure 6 - Directory structure of TradeMiner AMI.

2. MySQL

The TradeMiner instance comes with a MySQL version 5.5.6 installed. Inside that you have a trademiner database. Most of the tables and columns are self-commented, but a data dictionary for the trademiner database is available in Appendix B - Tables in trademiner database.

.

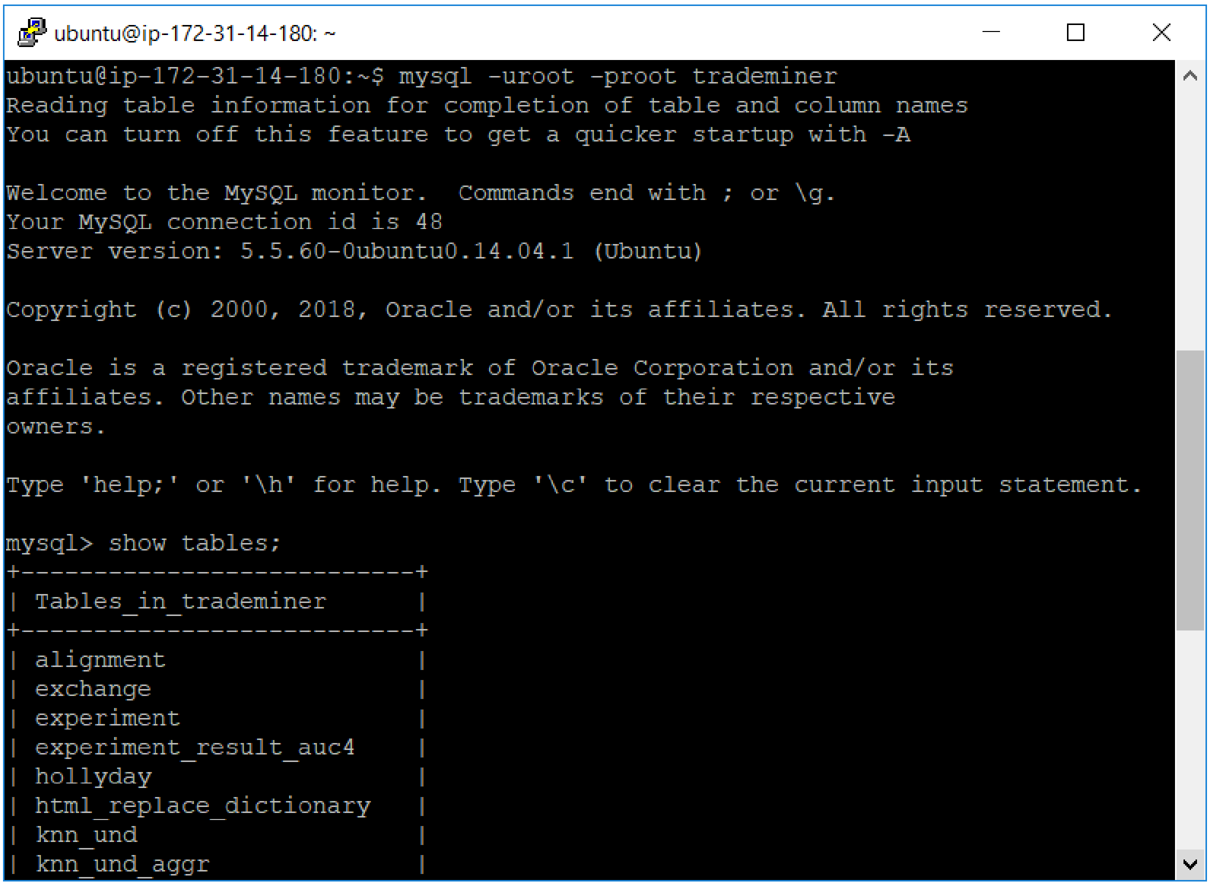


Figure 7 - The MySQL command line tool.

The user/password for MySQL is root/root, then to access the MySQL command line tool use the command `mysql -uroot -proot trademiner`, as demonstrated in Figure 7.

3. Using the remote desktop

The remote desktop is already configured in the TradeMiner Instance, and it uses the very fast xrdp protocol and xfce4 remote desktop, as recommended by Amazon Web Services in (Amazon Web Services, 2017), but it is important to remind the ports 22 and 3389 need to be open in the security group of your TradeMiner instance (Figure 3).

The only step required after launch the TradeMiner instance is to define a remote password for ubuntu user:

```
$ sudo passwd ubuntu
```

After that it is possible to open the Remote Desktop Connection and put the Public DNS from the running TradeMiner instance (Figure 8).

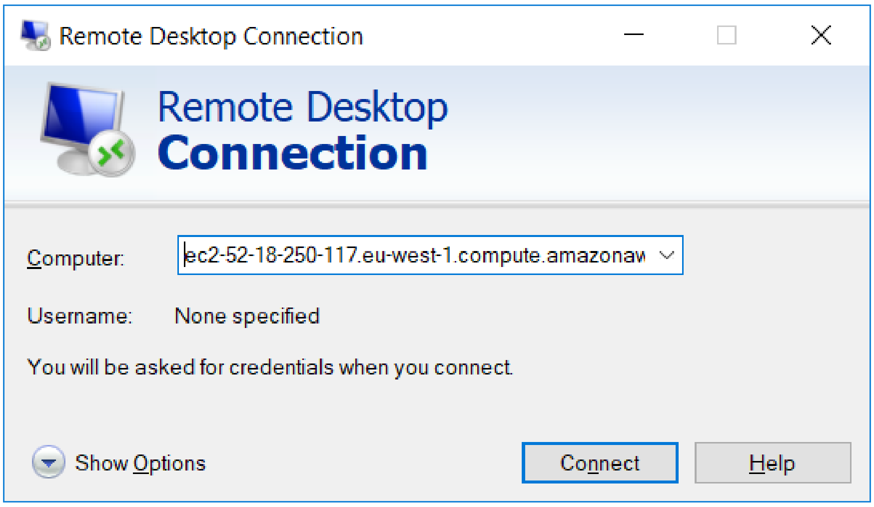


Figure 8 - Remote Desktop Connection from Windows desktop.

Then a warning message will appear, check the box and click Yes (Figure 9).

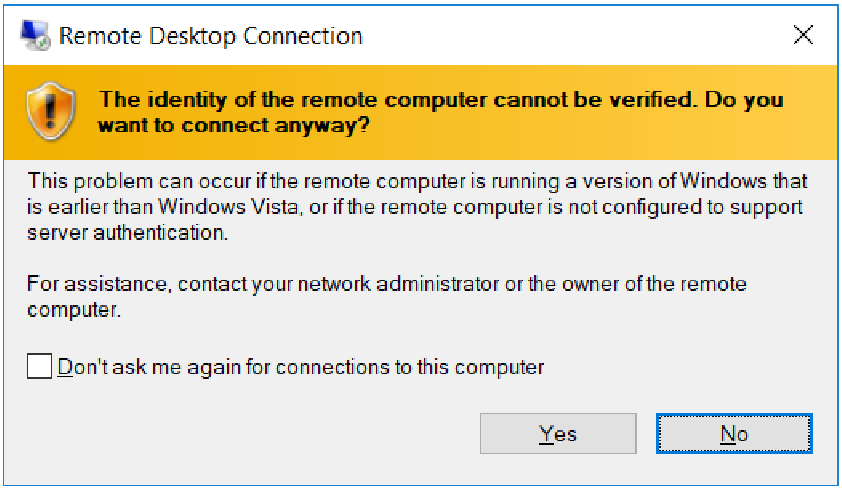


Figure 9 - Warning message to skip after discovering the TradeMiner instance.

In the authentication dialog (Figure 10), select sesman-Xvnc as Module, use ubuntu as user and put the password defined previously, and use -1 for port, then press Ok.

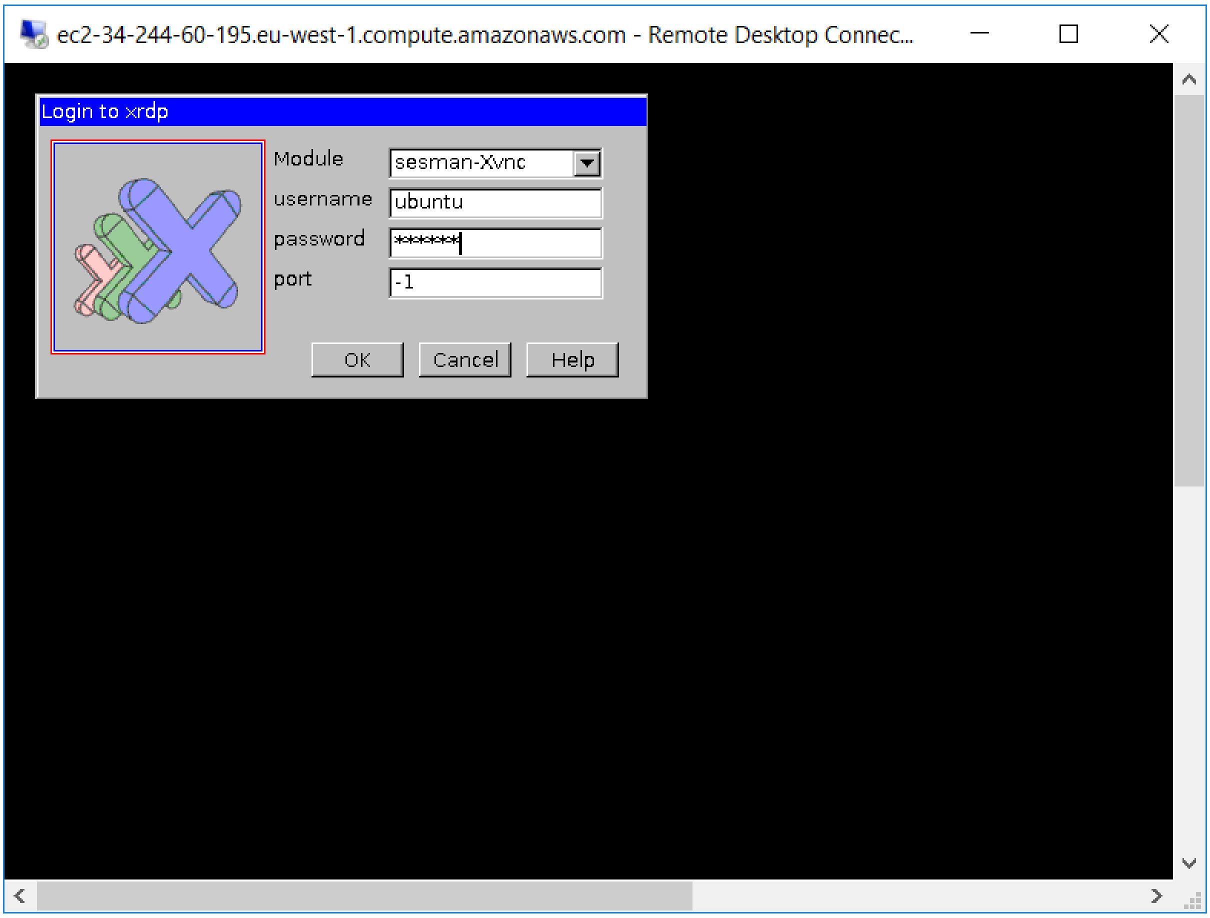


Figure 10 - Authentication dialog to access the Remote Desktop of TradeMiner instance.

Once authenticated it is possible to see a screen like in Figure 11. Take note of the connection port in order to return to the same desktop session later, by using this port number in the login dialog, instead of -1.

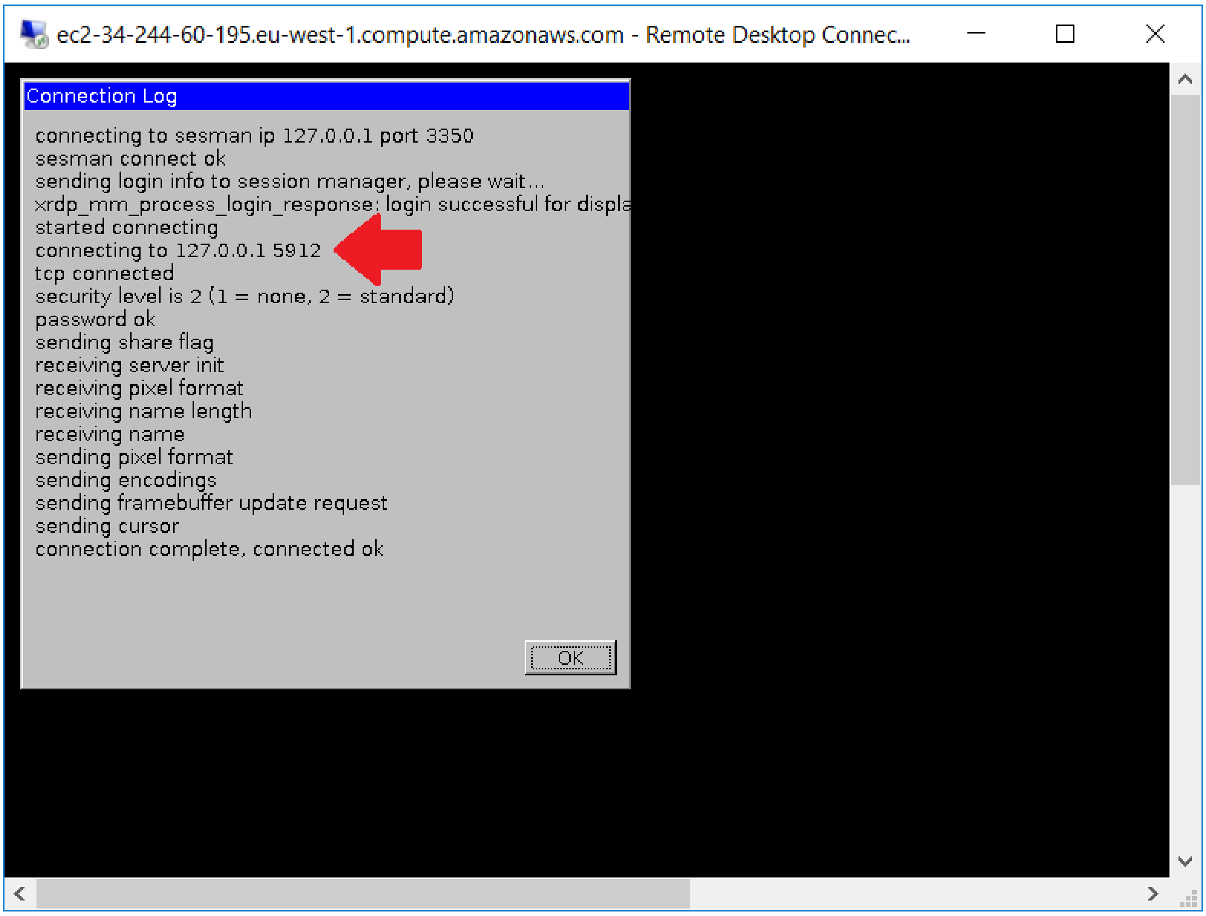


Figure 11 - Connection log after authentication to Remote Desktop.

After that it is possible to see the xfce4 remote desktop, and the desktop applications for TradeMiner, as demonstrated in Figure 12.

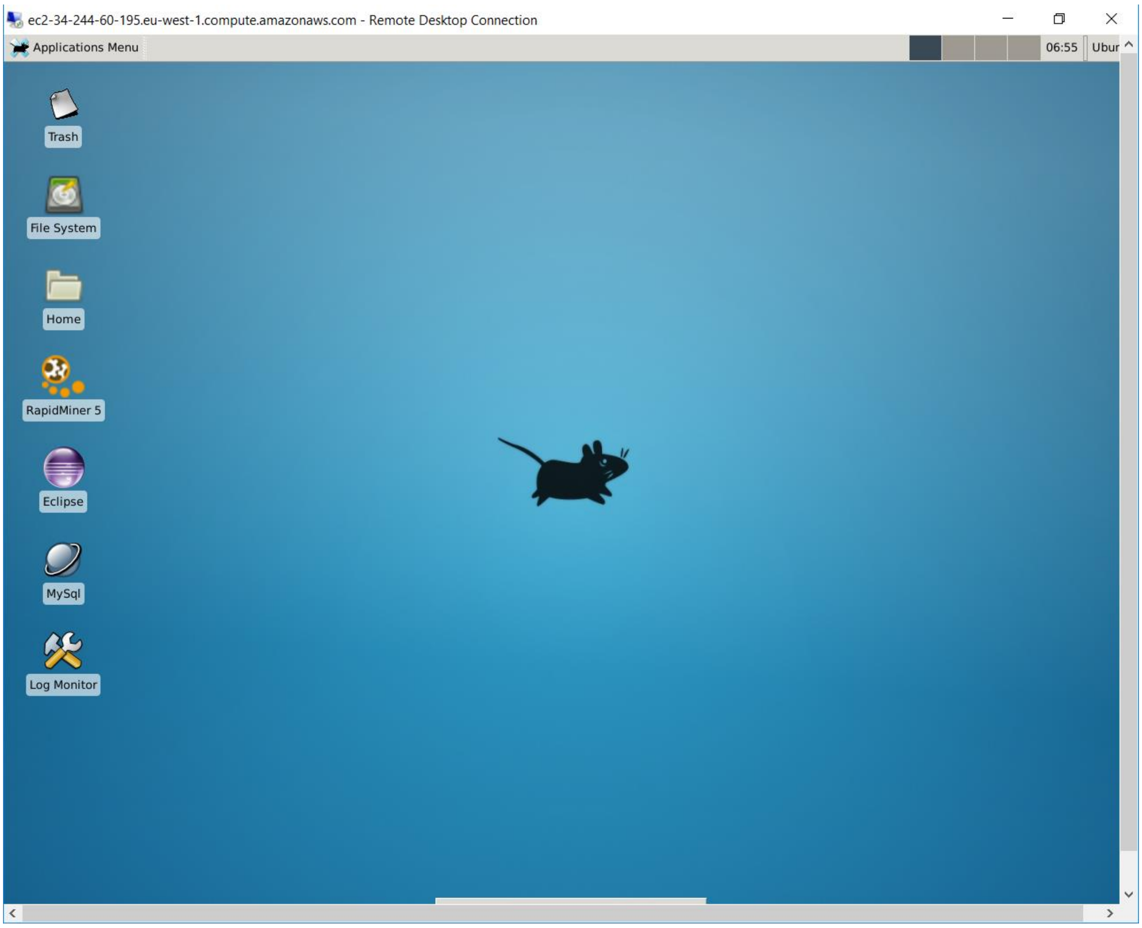


Figure 12 - TradeMiner Remote Desktop.

4. Run experiments with RapidMiner GUI

To run the RapidMiner GUI, click in the RapidMiner icon on the left of the remote desktop, then it will be possible to see the RapidMiner splash screen, and after that click in the Open icon, select the `NewLocalRepository`, and select `I_test_separated_compute_metrics_agg` (Figure 13).

For more details about all the RapidMiner experiments, visit the Appendix C - Files in REPOSITORY.

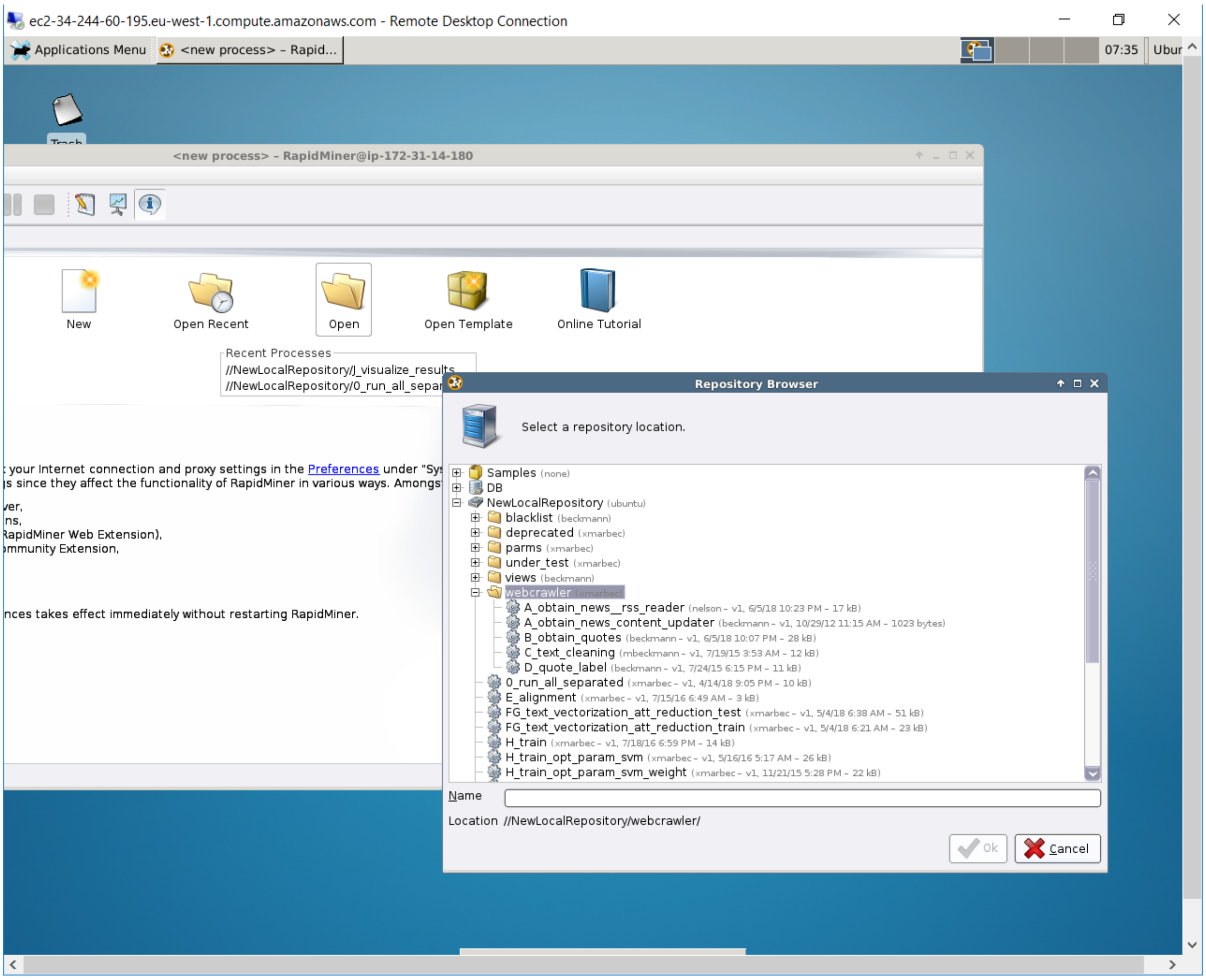


Figure 13 - Selecting an experiment with RapidMiner GUI.

After that it is possible to see the main desktop of RapidMiner, with several panes (Figure 14).

A - Main desktop with operators

B - TradeMiner extension and respective operators

C - Online help for the selected operator

D - Parameter pane for the selected operator

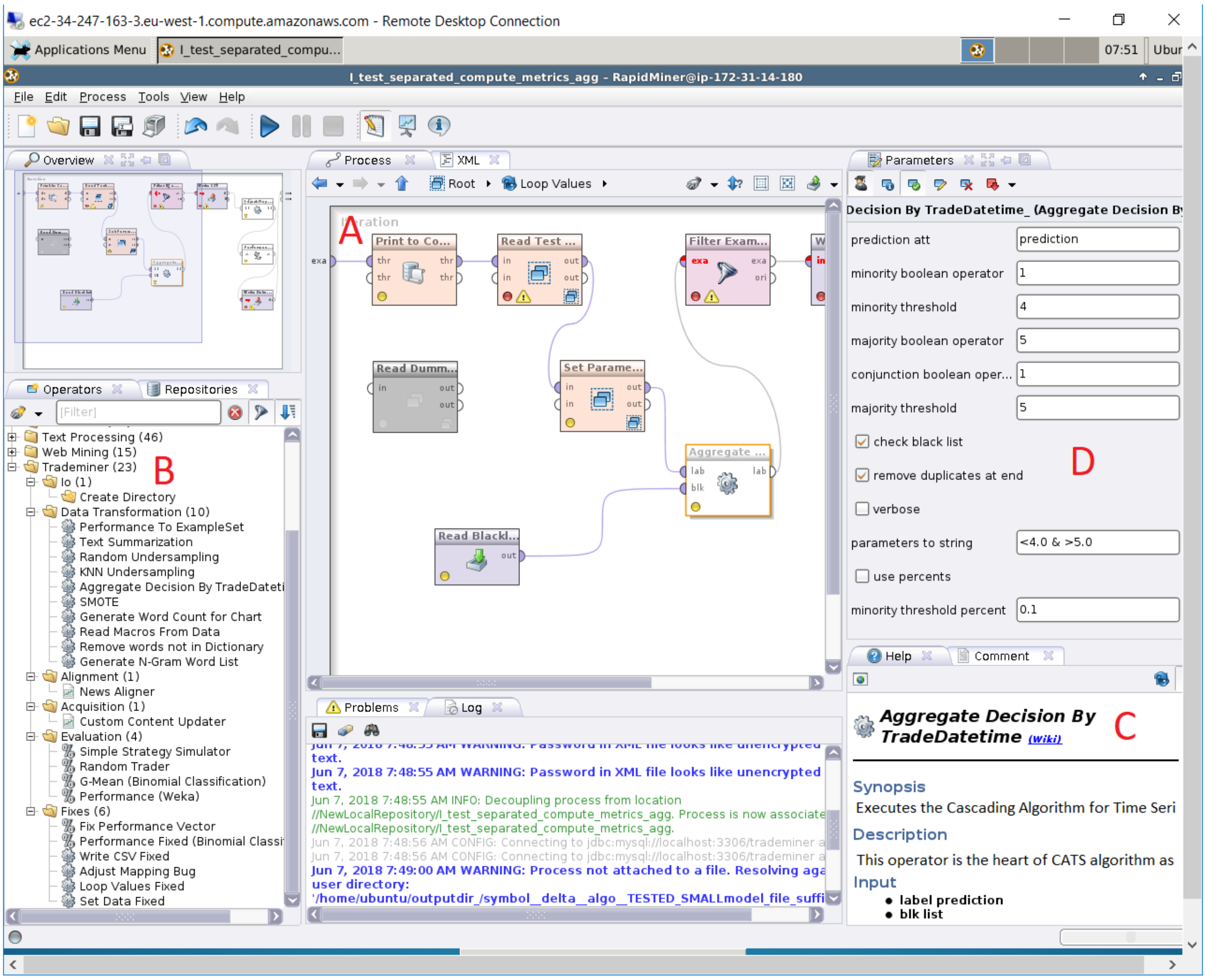


Figure 14 - Main desktop of RapidMiner.

The experiment l_test_separated_compute_metrics_agg that you selected is the same you ran in the step 2 of section Reproducing the Experiments. You can run this experiment by clicking the blue arrow on the top, then you will see the log running at the bottom, and the classification results in the Result perspective at the end (Figure 15).

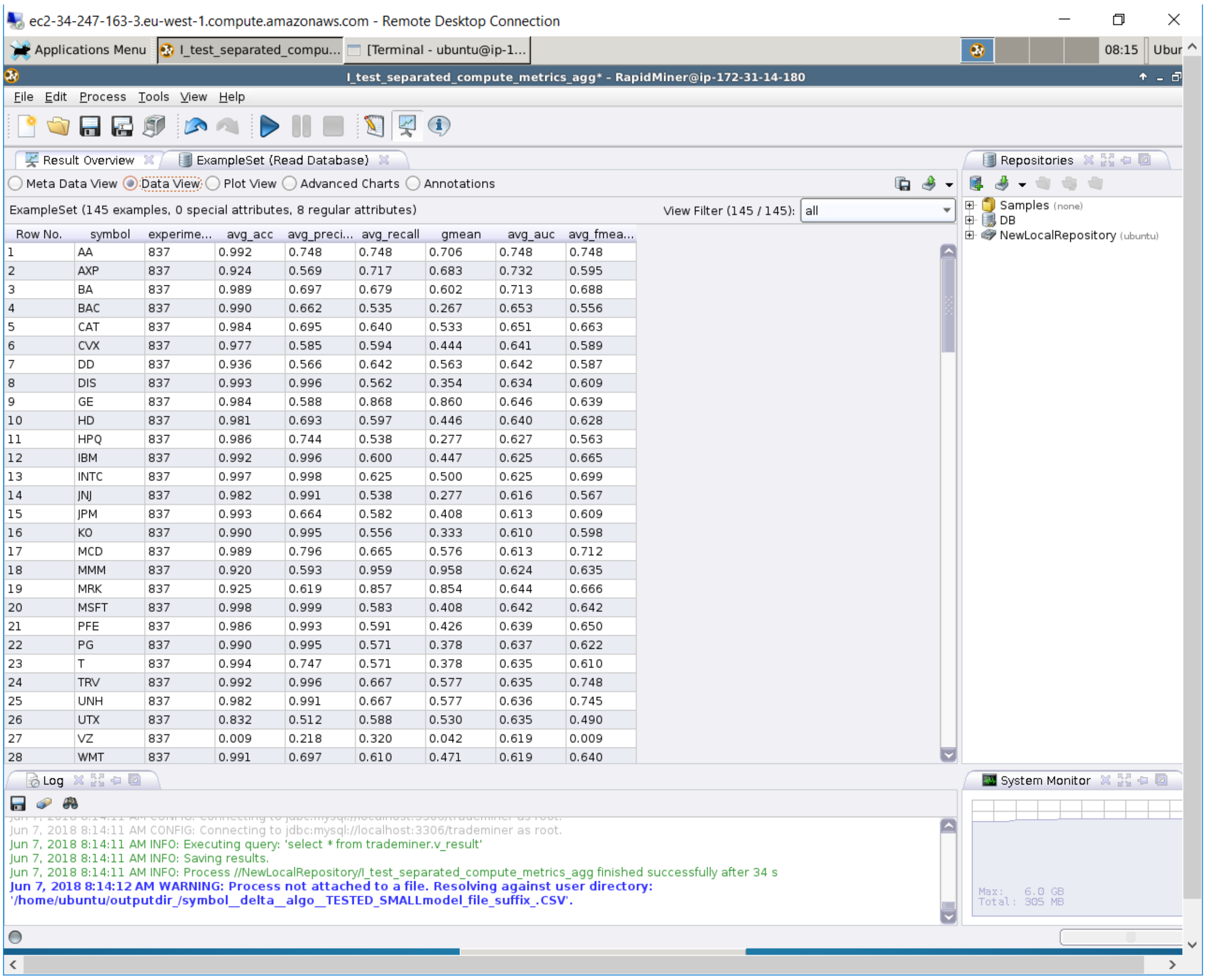


Figure 15 - Experiment results for time offset=1.

Refer to (North, 2012) for further details about how to use RapidMiner to modify and create new experiments.

5. Developing TradeMiner

The TradeMiner is an extension of RapidMiner (Mierswa, et al., 2006), and it is developed with Java language (ref here).

The Eclipse platform is the Interactive Development Environment (IDE) to develop the TradeMiner project, by clicking in the Eclipse icon on the Remote Desktop, then a screen like in Figure 16 will appear.

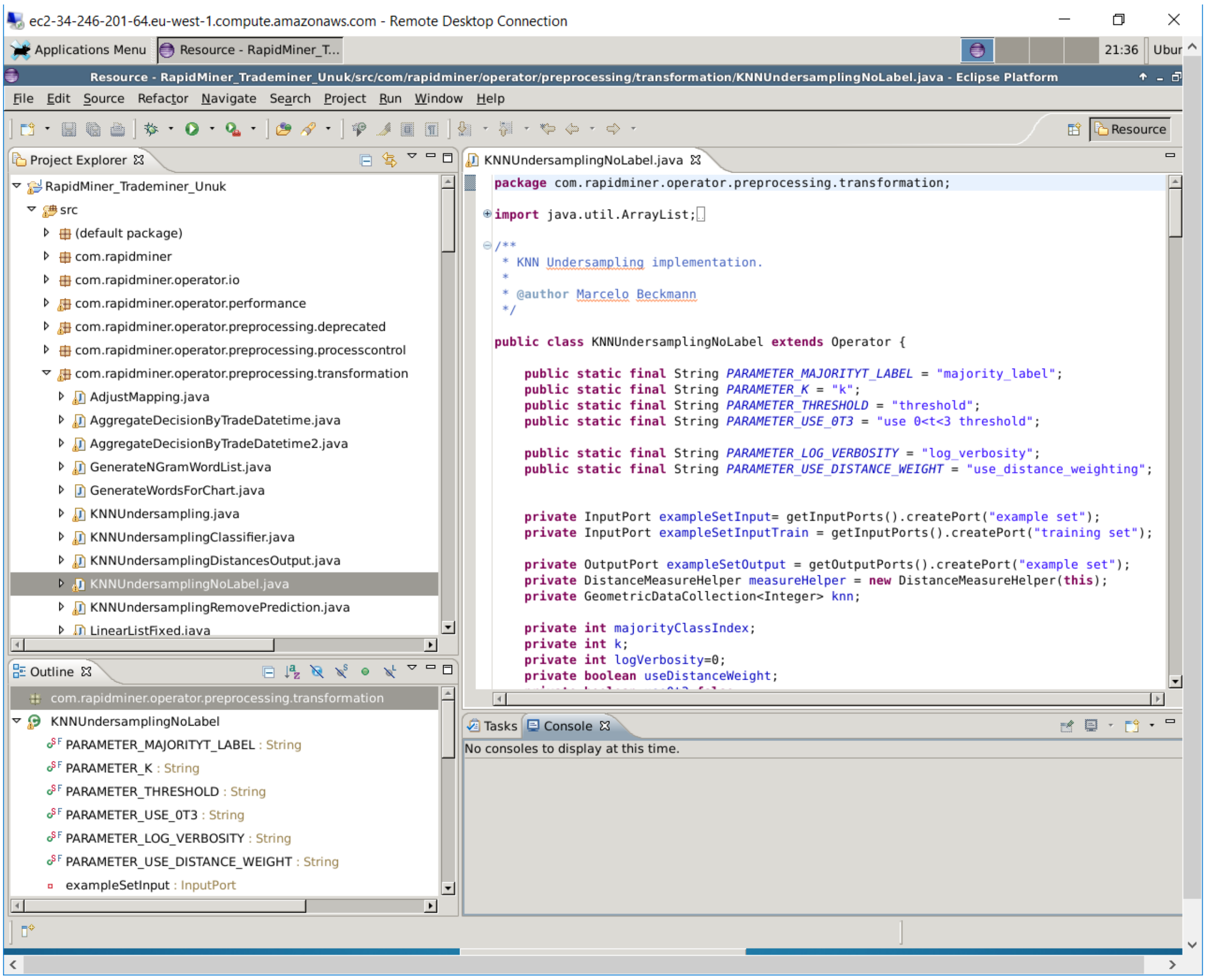


Figure 16 - Eclipse Java perspective to develop the TradeMiner extension.

The Eclipse Workspace is in the `~/workspace` directory. The main directories and files in the TradeMiner project are explained in Appendix D - Files in TradeMiner Eclipse workspace.

After to make the changes, it is necessary to pack the TradeMiner extension as a jar file. To do this, right click on build.xml and select Run as Ant Build (Figure 17).

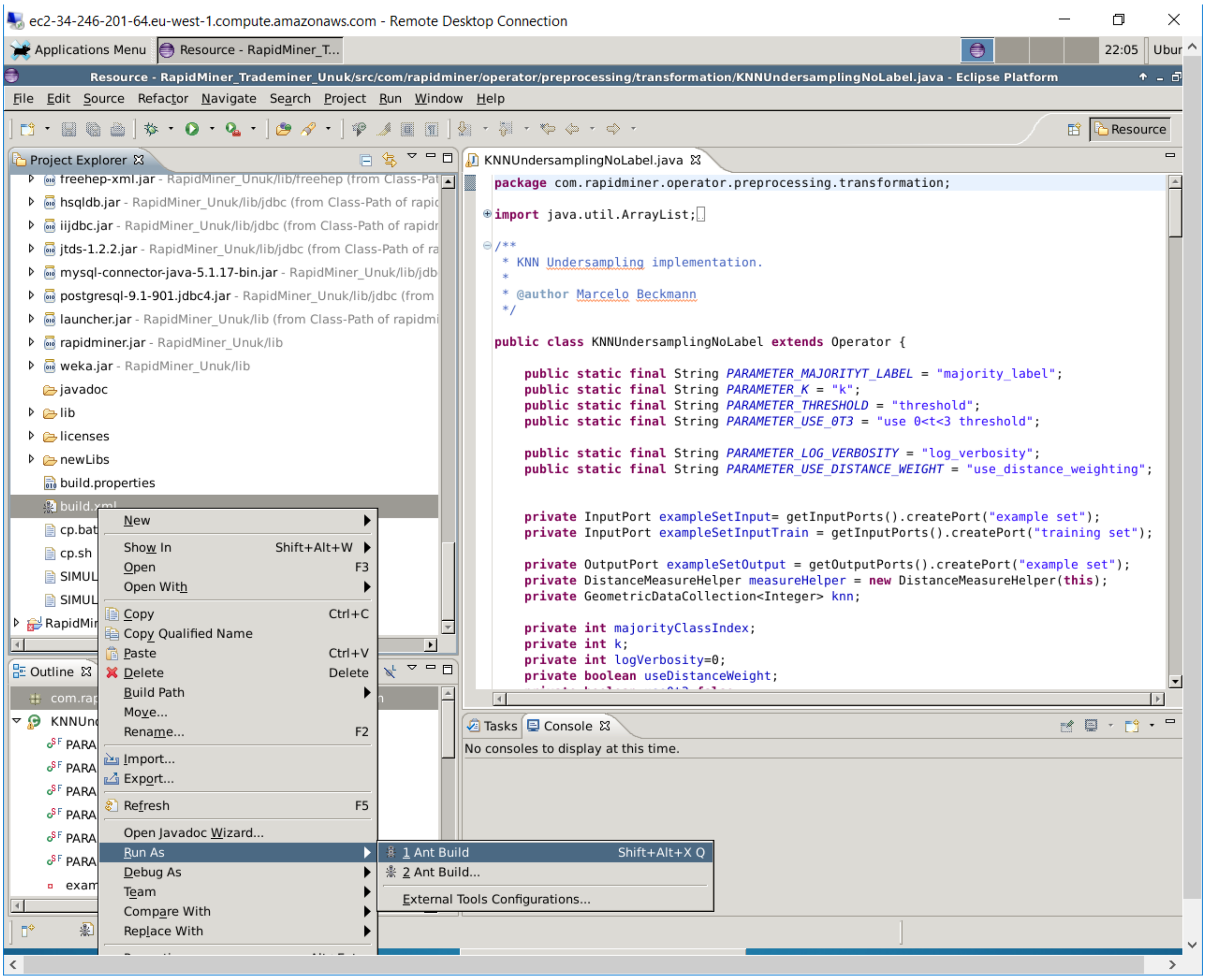


Figure 17 - Starting the TradeMiner packaging build.

After this the file "`rapidminer-Trademiner Extension-5.0.000.jar`" will be created in `~/workspace/trademiner/TradeMiner_Unuk/lib/plugins`, and then it needs to be copied to `~/RapidMiner5/lib/plugins` directory. This can be done by using the cp.sh script.

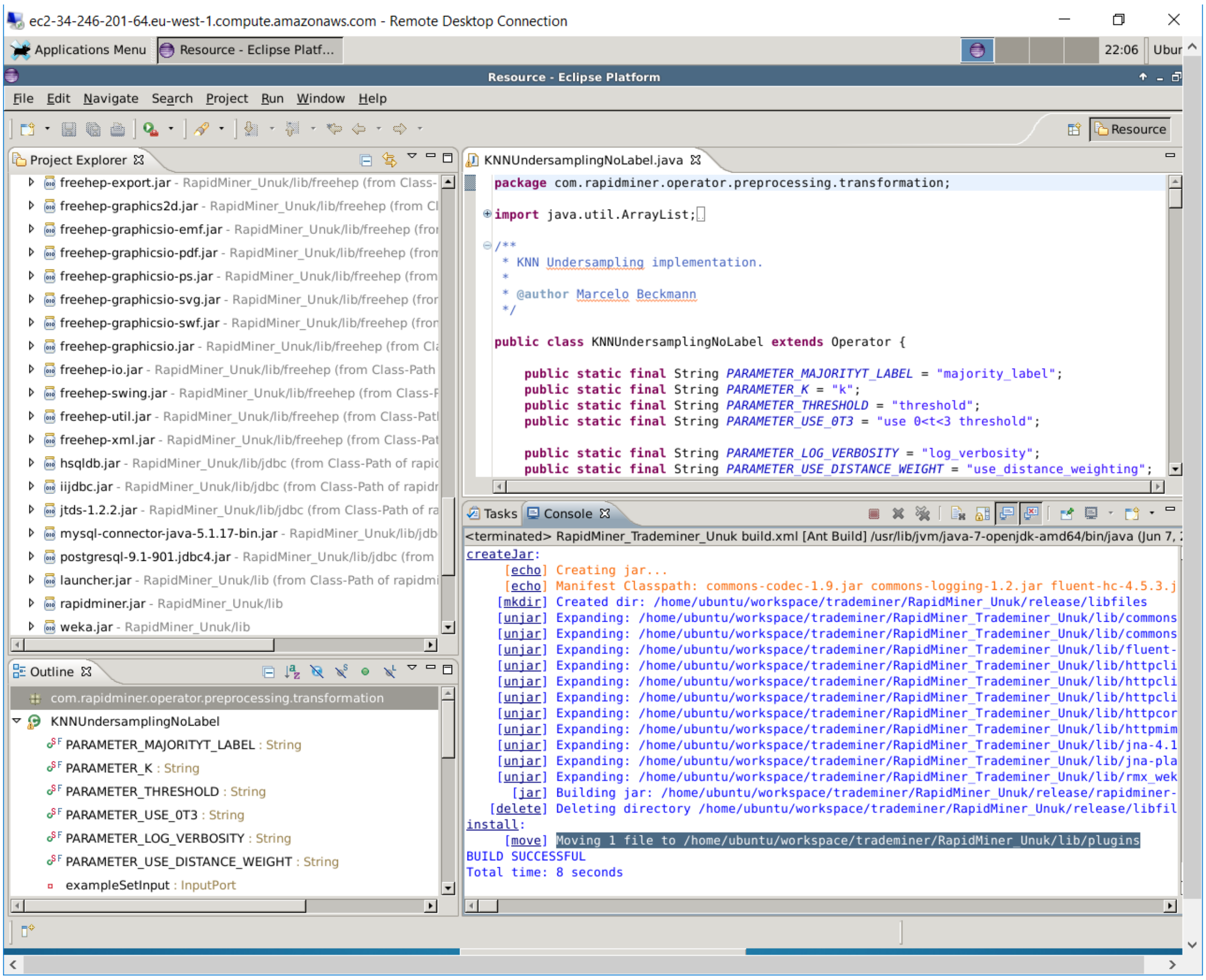


Figure 18 - Console with TradeMiner Packaging result.

Note 1: Sometimes an error message appears saying it was not possible to delete the release directory in `~/workspace/trademiner/TradeMiner_Unuk/release` directory. To solve this, just delete the release directory, and run the Ant Build again.

Note 2: The RapidMiner_Unuk is the base RapidMiner project. This project is only used to provide libraries and build directory structures, but it is also a good approach to see how RapidMiner works internally and get code examples.

Please refer to (Land, 2010) for more details about how to develop Extensions and Operators in RapidMiner.

6. Using your preferred language and code

It is not mandatory to use RapidMiner and TradeMiner to run your own experiments, as all the news articles and market data are stored in the MySQL database to be used. Nevertheless, Text Mining for Financial Prediction is a very sensitive problem, and there is a huge effort of data preparation already done, ready to be applied to other Machine Learning algorithms, then it is not necessary to reinvent the wheel.

For this purpose, it is recommended to use the generated CSVs files stored in the /data output directories (Figure 19). This directory structure and files implements the sliding window training and testing data split, as explained in section 4.2.1 of Thesis.

```
/data
├── logs                  Webcrawler logs
├── output_1              Output for experiments with time offset=1
│   ├── D0603             News articles between 03/Jun and 07/Jun for test + 6m training data
│   │   ├── AA_1_20150903_k21_TEST_NOKNN.CSV Test data for Alcoa, no Knn-Und
│   │   ├── AA_1_20150903_k21_TRAIN.CSV      Training data for Alcoa, Knn-Und k=21
│   │   ├── AA_1_libsvm_TESTED_NOKNN.CSV     Classification results for Alcoa, using libsvm
│   │   ├── ...               Repeats this structure for all the 30 DJIA stock symbols
│   │   ├── XOM_1_20150903_k21_TEST_NOKNN.CSV       Same as above, for Exxon Mobil
│   │   ├── XOM_1_20150903_k21_TRAIN.CSV
│   │   └── XOM_1_libsvm_TESTED_NOKNN.CSV
│   ├── D0617                 The same structure above is repeated along the next weeks
│   ├── D0624
│   ├── D0701
│   ├── D0708
│   ├── D0715
│   ├── D0722
│   ├── D0730
│   ├── D0805
│   ├── D0812
│   ├── D0819
│   ├── D0826
│   ├── D0902
└── output_2                  Output for experiments with time offset=2, same structure
```

Figure 19 - Directory and file structure for /data.

For the CSV files depicted in (Figure 19), each record represents a news article published, and its respective label (SURGE, NOT_RECOMENDED) obtained through alignment process with Market Data.

All records are maintained in their chronological order when they were published, and the news articles were transformed from unstructured to structured format using Bags of Words (BOW) representation (Miner, et al., 2014), (Zhai & Massung, 2016) .

In case of other techniques than Bag of Words wants to be applied, the entire dataset above is also available and in raw CSV format, not split, labelled, for time offsets 1, 2, 3, and 5 minutes, in the Open Science Framework [4].

As an example, it is possible to use the file `AA_1_20150903_k21_TRAIN.CSV` as training data for a Deep Learning classifier (Bengio, 2009), and then apply this model to `AA_1_20150903_k21_TEST_NOKNN.CSV` to get predictions about SURGES and NOT_RECOMENDED. Once the classifier returns good results, the training and test process can be extended to other stocks and weeks.

It is also not mandatory to use the AWS TradeMiner instance. Once the TradeMiner instance is running, it is possible to download the dump of MySQL trademiner database, and all the directories from `\home\ubuntu` and `\data` to a Windows or Linux computer, and reassemble the TradeMiner platform over there.

[4] https://osf.io/gc6u6/

7. Contributing to TradeMiner

The preferred workflow for contributing to TradeMiner is to fork the main repository on GitHub, clone, and develop on a branch, according the following steps:

a. Fork the project repository by clicking on the 'Fork' button near the top right of the page. This creates a copy of the code under your GitHub user account. For more details on how to fork a repository see this guide.

Clone your fork of the trademiner repo from your GitHub account to your local disk:

```
$ git clone https://github.com/Yourlogin/trademiner.git
$ cd trademiner
```

b. Create a feature branch to hold your development changes:

```
$ git checkout -b my-feature
```

Always use a feature branch. It is a good practice to never work on the master branch!

c. Develop the feature on your feature branch. Add changed files using git add and then git commit files:

```
$ git add modified_files
$ git commit
```

d. to record your changes in Git, then push the changes to your GitHub account with:

```
$ git push -u origin my-feature
```

Note: The directory `~/workspace/trademiner` is a git cloned repository from Git Hub TradeMiner remote repository[5]. Because of this, if you want to continue using the ~/workspace directory in the TradeMiner instance, it is necessary to rename the original trademiner directory before, to give place to your forked version.

```
$ cd ~/workspace
$ mv trademiner trademiner_orignal
```

8. Collecting new data

The TradeMiner instance comes with the webcrawler ready to gather news articles and respective market data minute by minute for the current stocks in DJIA index.

All the files used by the webcrawler are in the directory `~/REPOSITORY/webcrawler`. See Appendix C - Files in REPOSITORY for a complete list of webcrawler files and description.

It is possible to execute any of the scripts in the webcrawler directory, but it is more convenient and precise to schedule all the scripts with cron, by using the command below, then the crontab schedule will be updated like in Figure 20.

[5] https://github.com/marcelobeckmann/trademiner

```
cd ~/REPOSITORY/webcrawler

crontab -u ubuntu CRONTAB.txt

crontab -l
```

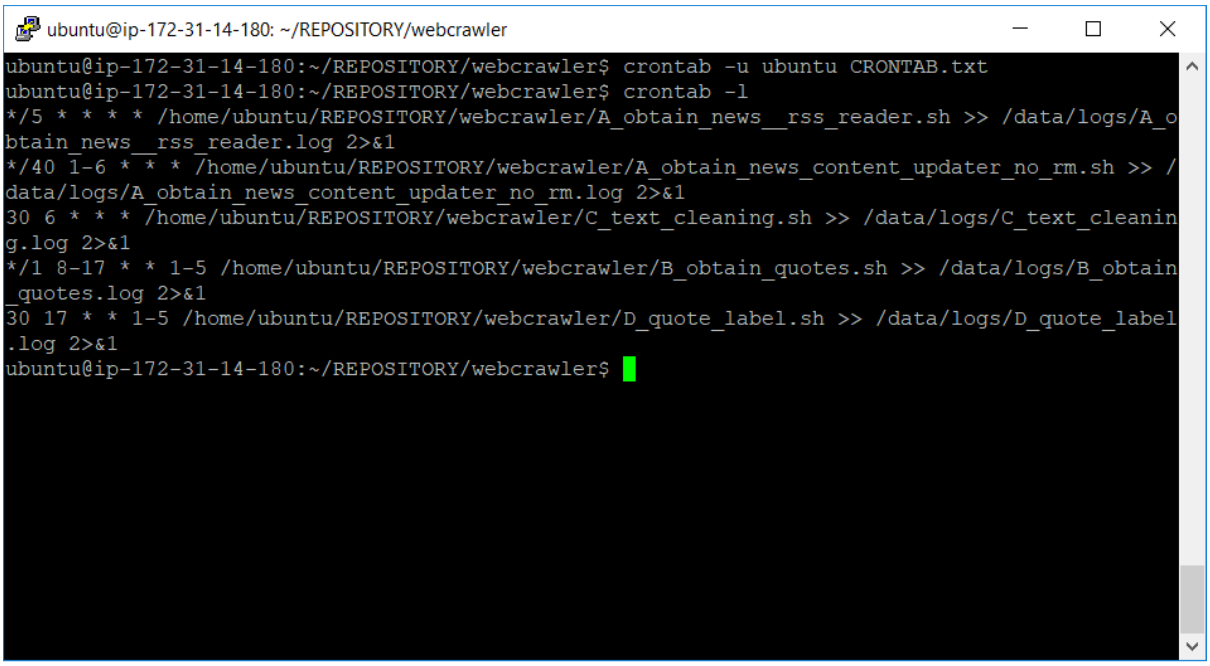


Figure 20 - Crontab setup for webcrawler.

After this the Linux crontab will start to work, bringing news articles and market data into your MySQL database. It is possible to follow up the progress of your data gathering, by executing the script `0_view_progress.sh` in the webcrawler, or monitoring the logs with the LogMon application, using Remote Desktop Connection (Figure 21).

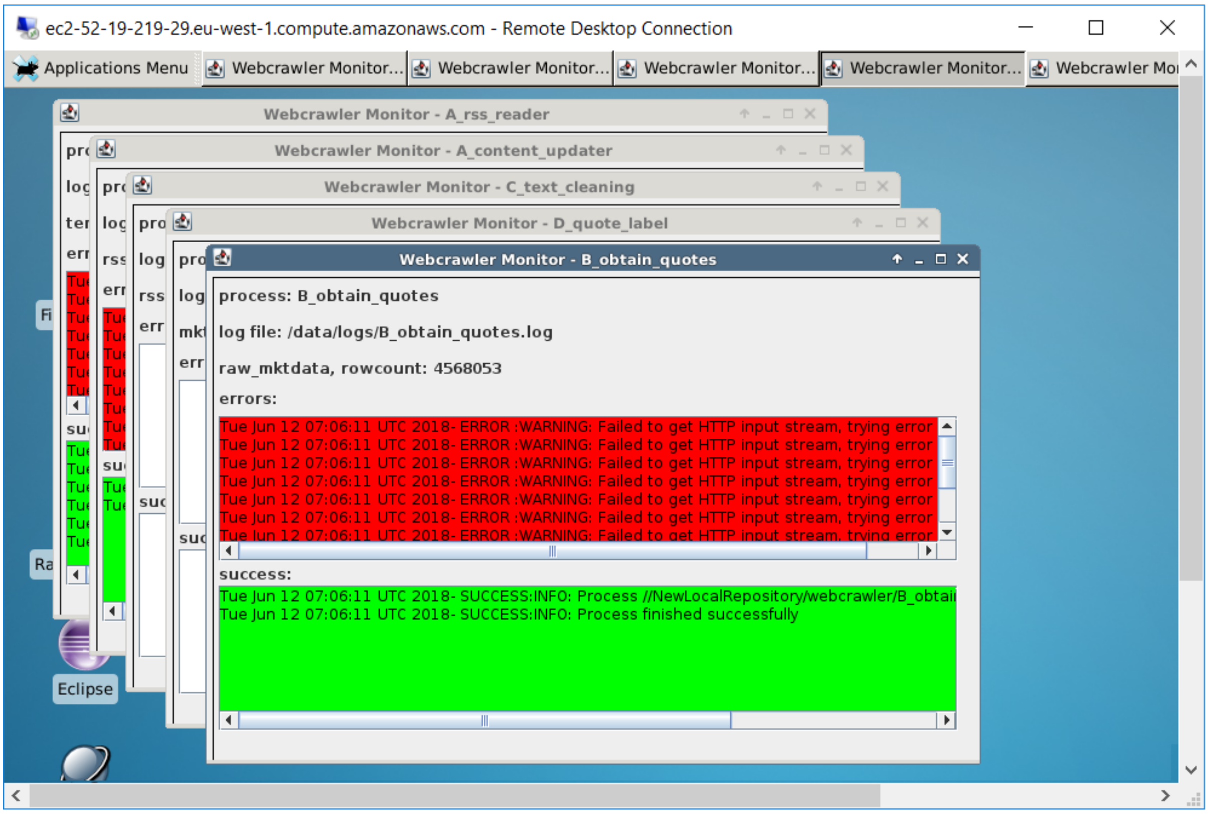


Figure 21 - Webcrawler log monitor.

Note 1: Currently the webcrawler only will bring news from finance.yahoo.com, as finance.google.com stopped to provide support to RSS since 2017.

Because the current webcrawler only deals with RSS URLs, if you want to gather news from a news feeding through a html page you need to develop your own html parser to gather the new content, title, and extract the publication date or associate the news with a set of stock symbols if necessary.

Note 2: To put new stock symbols to work in the webcrawler, you need to add new symbols to symbol table, and add the respective RSS URLs related to that symbol, in the link table.

To start to generate models with these new stock symbols, you need to set the status=1 for the stock symbols to be modelled, and unset the ones not that won't be modelled, or don't have data, by setting the status=0 in the symbol table.

As the last step, you need fill up the file `~/REPOSITORY/TRADEDATES.csv` with the new period of dates (each line with one-week period) you gathered.

Conclusion

This article presented how to use a public Amazon Machine Instance configured with the TradeMiner platform, and explained how to reproduce experiments and make new developments in Text Mining applied to predict stock price changes. Also, all the environment in terms of directories, files, database, source code, and applications was explained. The possibility to use your preferred language, computational platform, and gather new data was also demonstrated.

The TradeMiner platform is a research in progress. There are successful results using this methodology to predict indexes changes, parallel processing experiments are in progress, and new experiments with Deep Learning demonstrated some successful stock models that were failing before with Support Vector Machine algorithm.

The authors of TradeMiner wish to contribute with a stable platform for new developments in text mining and sentiment analysis applied to Financial Markets. We believe this branch of research deserves to be using a proper scientific methodology open to everyone, and then clear conversations and robust developments will bring transparency and confidence to predictive analytics applied to Financial Markets.

References

Amazon Web Services, 2017. *How can I connect to an Amazon EC2 Linux instance of Ubuntu 14.04 with desktop functionality from Windows?.* [Online]
Available at: https://aws.amazon.com/premiumsupport/knowledge-center/connect-to-linux-desktop-from-windows/
[Accessed 11 06 2018].

Beckmann, M., Ebecken, N. & De Lima, B., 2015. A KNN Undersampling Approach for Data Balancing. *JILSA - Journal of Intelligent Learning Systems and Applications,* pp. 7, 104-116.

Beckmann, M., Ebecken, N. & De Lima, B., 2017. *Stock Price Change Prediction Using News Text Mining,* Rio de Janeiro, Brazil: Civil Engineering Program/COPPE, Federal University of Rio de Janeiro.

Bengio, Y., 2009. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning. 2.*

Land, S., 2010. *Approaching Vega: The final descent - How to extend RapidMiner 5.0,* s.l.: Rapid-I.

Mierswa, I. et al., 2006. *YALE: Rapid Prototyping for Complex Data Mining Tasks.* s.l., s.n.

Miner, G. et al., 2014. *Pratical Text Mining and Statistical Analysis for Non-structured Text Data Applications.* s.l.:Elsevier.

North, M., 2012. *Data Mining for the Masses.* s.l.:Paperback.

Widenius, M., Axmark, D. & DuBois, P., 2002. *Mysql Reference Manual.* 1st ed. Sebastopol, CA, USA: O'Reilly & Associates.

Zhai, C. & Massung, S., 2016. *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining.* s.l.:ACM.

Appendix A - Parameter Setup with macros.csv

The macros.csv file is in ~/REPOSITORY directory and is responsible to provide common parameters to the entire TradeMiner process. It is a CSV file with two columns, macro and value, separated by a semicolon (;), and sharps are considered comments. These pairs of columns and values will be transformed into macros inside a TradeMiner experiment, and can referenced by %{}, e.g., `%{experiment_description_}`, as can be seen in Figure 22. Figure 23 depicts the content of macros.csv for time offset=1 minute.

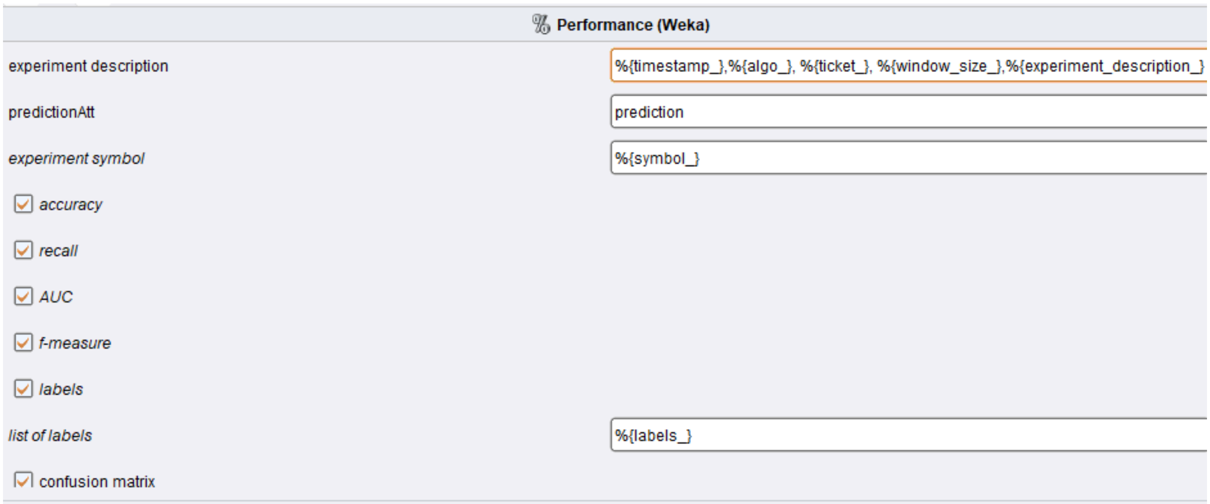


Figure 22 - Example of macro usage in RapidMiner.

```
macro;value
###########################################################
# General parameters                                      #
###########################################################
# Gives a description of what algorithms and parameters are used for this run
experiment_description_;all,REPRODICING BASELINE #763, AGGREGATION ,delta=1, REDOING OPT
PARM WITH _KNNUND suffix files, BLACKLIST,KEEP FILES, SVM RBF, k=21 t=2 ,TRAINING INTERVAL
6M, RESTRICTIVE ALIGNMENT WITH DELTA 00:02:00,WEIGHT BY X2, 0.2, COSINE, N3GRAM,FIVE-DAYS
BY FIVE-DAYS, SEPARATED TRAIN/TEST FROM >31/MAY TO <=06/SET, TF/IDF,GRID-SEARCH FOR C &
GAMMA WITH FMEASURE, 2 CLASSES [-20]2, NO WEIGHTED AVG FOR PERFORMANCE, LIBSVM GRID-SEARCH
,NO TIMEZONE ADJUST,NEW_MACRO_SCHEMA,0_SHIFT, MKTDATA, CHI2
# This is where the intermediate CSV files will be created
outputdir_;/data/output_1
# The host to MySQL database
db_host;localhost
# This is to identify which run the experiment is repeated (In the Thesis the experiment
was repeated 10 times).
# KEEP IT AS 1 if you want to run just once.
run_;1
# The table name where the classification results will be written
outputtable_;experiment_result_auc4
###########################################################
# Stock symbol selection                                  #
###########################################################
# Defines which set of stock symbols will be retrieved from database, given the index they
are associated
index_;DJIA
# It's possible to associate a stock symbol with a portfolio (identified by the column
portfolio in the table symbol), and run this portfolio instead of all symbols
# To run all symbols, put all. There are also top5, top6, top9, partial, etc. You can
create your own set of portfolio adding a portfolio name to column portfolio of table
symbol.
portfolio_;all
###########################################################
# Data Gathering  (section 4.1 of Thesis)                 #
```

```
#############################################################
#mktdata_webservice_url_=http://www.webservicex.net/stockquote.asmx/GetQuote?symbol=
# 15 mins delayed mktdata
mktdata_webservice_url_;http://dev.markitondemand.com/MODApis/Api/v2/Quote?symbol=
#############################################################
# Window size and alignment (section 4.5.1 from Thesis)    #
#############################################################
# Defines the windows size (tau) in minutes for news alignment
window_size_;00:01:00
# Each arrangement of alignment between news and prices are stored in a table. The result of this alignment defines how the news article is labelled (SURGE, NOT_RECOMENDED, PLUNGE)
ticket_;20150903
# This is the implementation of alignment that worked well so far and is expresses in the Thesis
aligner_impl_=RestrictiveAlignment
# This is the same as window_size, but in a numeric format for convenient storage in database columns and file names
delta_;1
#############################################################
# Feature removal (section 4.3.2 of Thesis)                #
#############################################################
# This is the attribute weight for Feature Removal by Chi Square
att_weight_;0.10S
#############################################################
# Machine Learning algo (section 4.3.4 of Thesis)          #
#############################################################
# This is only to identify the machine learning algorithm used in the experiments
algo_;libsvm
#############################################################
# KNN Undersampling parameters (section 4.3.3 of Thesis)   #
#############################################################
# The number of k-neighbours for KNNUND
k_;21
# This is the decision threshold for KNNUND
t_;2
# These two parameters below define CSV file suffixes before and after KNNUND be applied.
model_file_suffix_;_NOKNN
model_file_noknn_;
#############################################################
# Class definition parameters (section 4.3.3 of Thesis)    #
#############################################################
#keep "0,2" to use "one against all" schema (default)
#put "-2,0,2" to consider all classes
labels_;0,2
#keep "IF(label='-2','2',label) AS label" to use "one against all" schema (default)
#put "label" to consider all classes
label_definition_;IF(label='-2','0',label) AS label
#############################################################
# Sliding windows parameters (section 4.2 of Thesis)       #
#############################################################
# The four parameters below provides the start and end dates, and respective storage directories for slide window train/test splitting
# Keep commented the four parameters below if you are going to use some *run_all* experiment.
# The *run_all* experiments provide an easy way to train and test all the slide window periods contained in the TRADEDATES.CSV
# The next day the data will be tested, in YYYY/MM/DD format
#next_trade_date_;2013/06/03
# The end day the data will be tested, in YYYY/MM/DD format
#end_test_date_;2013/06/07
#The output subdirectory where the generated CSV files will be stored
#outdir_;D0603
# the previous output dir, or the same if it is the first output_dir (e.g., D0603)
#prev_outdir;D0603
#training size in months
training_interval_;6
############################################################
# Simulation (section 4.5.2 of Thesis)                     #
############################################################
#gets the last experiment for delta=1, you can also put a specific experiment id here
sim_experiment_;(SELECT id FROM experiment WHERE delta=1 ORDER BY id DESC LIMIT 1)
#the min classification performance accepted in terms of G-Mean for a stock model be taken for investment simulation
sim_min_performance_;0.55
#Defines when the stock will be bought (00:00:00 to buy immediately after the news be published)
```

```
windows_size_minus_1_;00:00:00
#A simulation can be run several times to prove it's stability. This parameter identifies
the simulation run number
run_count_;1
#Identifies which type of simulation is being executed (e.g., using prediction/random
trader)
sym_type_;w1pred
```

Figure 23 - Content of macros.csv parameter file.

Appendix B - Tables in trademiner database

Table 3 - List of tables names and comments in the trademiner repository.

Table Name	Table Comment
alignment	Labels news articles by alignment between news articles and market data
exchange	List of exchanges
experiment	Represents the experiments conducted and its respective descriptions
experiment_result_auc4	Results of experiments in terms of contingency matrix and classification measures
holyday	List of dates with holidays for next_trade_date calculation
html_replace_dictionary	List of regex replacement expressions for to remove html tags
knn_und	Knn undersampling blacklist discovery process
knn_und_aggr	Knn undersampling blacklist discovery process
knn_und_blacklist	Knn undersampling backlist discovery process
knn_und_processed	Knn undersampling blacklist discovery process
link	List of URLs of news providers (RSS) associate with a stock symbol
mktdata	Stock prices and volumes for a stock symbol along the time
number_replace_dictionary	List of regex replacement expressions for numbers
pl_output	Investment simulation results
pl_output_report	Investment simulation results as a p&l report
raw_mktdata	Raw stock price data just after being collected
rss	News articles
symbol	List of stock symbols to be processed
temp_rss	Temporary table for news articles recently gathered
v_content	Under test
v_contentr	Under test
v_result	View with formatted result of last experiment in terms of classification measure
v_result_id	View with formatted result in terms of classification measure, given an experiment_id
v_result_id_posneg	Under test
v_resultw	Under test
v_resultw_id	Under test

Appendix C - Files in REPOSITORY

Each process marked with an uppercase letter in the workflow from Figure 24 can be found in the file list from Figure 25.

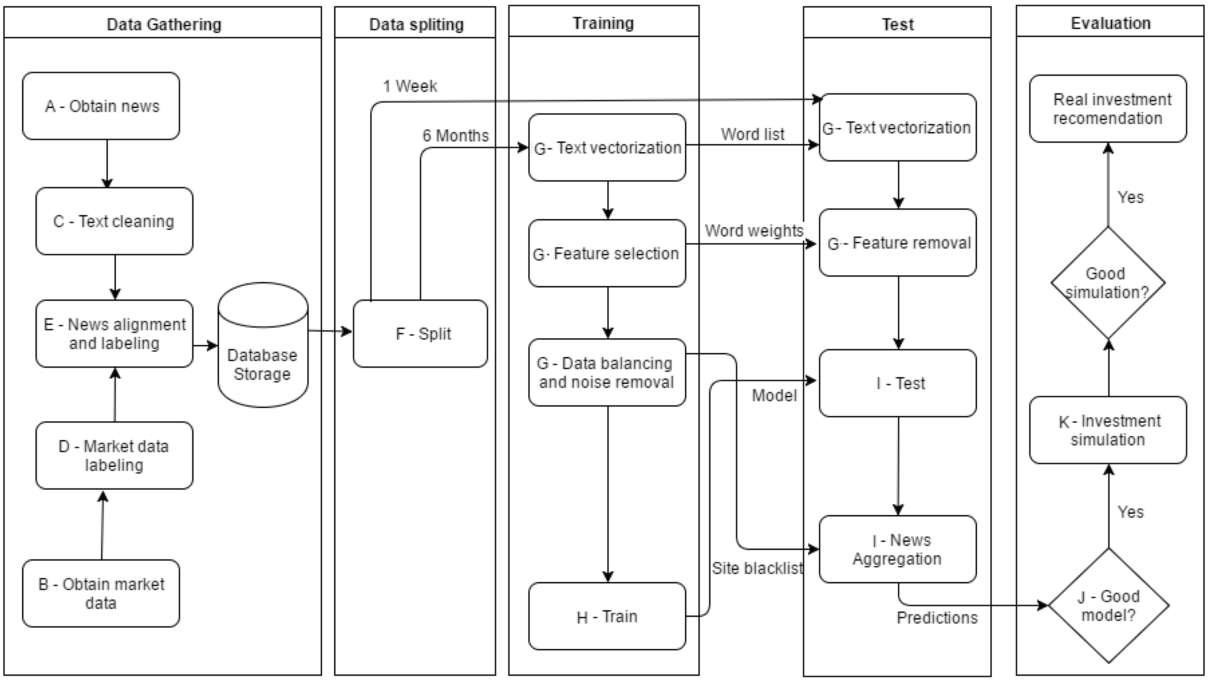


Figure 24 - Complete process of TradeMiner.

```
~/REPOSITORY
├── blacklist                                       Queries for KNN blacklist discovery
├── deprecated                                      Deprecated and failed experiments
├── parms                                           Contains optimized parameters
├── under_test                                      Stores word lists and weights
├── views                                           Experiment to visualize market data
├── webcrawler                                      Webcrawler experiments and scripts
│   ├── 0_view_progress.sh                          Views the growth of number of rows
│   ├── A_obtain_news_content_updater_no_rm.sh      Shell to obtain html from RSS
│   ├── A_obtain_news_content_updater.rmp           Experiment to obtain html from RSS
│   ├── A_obtain_news_content_updater.sh      Shell to obtain html from RSS (not in use)
│   ├── A_obtain_news__rss_reader.rmp         Experiment to obtain news RSS entries
│   ├── A_obtain_news__rss_reader.sh          Shell to obtain news RSS entries
│   ├── B_obtain_quotes.rmp                   Experiment to obtain stock prices (mktdata)
│   ├── B_obtain_quotes.sh                    Shell to obtain stock prices
│   ├── CRONTAB.txt                           Job schedule configuration in cron format
│   ├── C_text_cleaning.rmp                   Experiment to clean html content
│   ├── C_text_cleaning.sh                    Shell to clean html content
│   ├── D_quote_label.rmp                     Experiment to label the mktdata
│   └── D_quote_label.sh                      Shell to label the mktdata
├── 0_run_all_separated.rmp                   Runs all experiments in sliding window mode
├── E_alignment.rmp                           Labels news articles aligned with mktdata
├── FG_text_vectorization_att_reduction_test.rmp  Transforms news in BOW CSVs for test
├── FG_text_vectorization_att_reduction_train.rmp Transforms news in BOW CSVs for training
├── H_train_opt_param_svm.rmp                 Adjusts hyperparameters for SVM
├── H_train_opt_param_svm_weight.rmp        Adjusts hyperparameters for SVM using weightings
├── H_train.rmp                       Trains the classifier algorithm (SVM by default)
├── I_test.rmp                        Performs predictions on test dataset (deprecated)
├── I_test_separated_compute_metrics_agg.rmp Applies CATS for assembly decision all weeks
├── I_test_separated_compute_metrics.rmp Gather all the predictions week by week (no CATS)
├── I_test_separated_compute_metrics_agg_opt.rmp      Optimizes decision rules for CATS
├── I_test_separated.rmp Performs predictions on test dataset week by week (sliding window)
├── J_visualize_results.rmp Visualize the latest prediction results
├── K_simulation_from_mktdata.rmp Investment simulation using the labelled mktdata as input
```

```
├── K_simulation_from_prediction.rmp  Investment simulation using predictions as input
├── K_simulation_from_random.rmp Investment simulation using a random trader as input
├── macros_#758_w2.csv        Macro parameter file for experiment with time offset = 2mins
├── macros_#763_w1.csv        Macro parameter file for experiment with time offset = 1min
├── macros_#769_w5.csv        Macro parameter file for experiment with time offset = 5mins
├── macros_#777_w3.csv        Macro parameter file for experiment with time offset = 3mins
├── macros.csv                Central archive with parameters for entire TradeMiner process
├── TRADEDATES.csv            Contains start and end dates or sliding window data split
└── wordsEn.txt               List of words in English (not in use)
```

Figure 25 - List of files and RapidMiner experiments in the ~/REPOSITORY.

Appendix D - Files in TradeMiner Eclipse workspace

```
logmon                       Custom log monitor for web crawler
RapidMiner_TradeMiner_Unuk   TradeMiner extension project
├── build                    Directory with build binaries for packaging
├── build.properties         Properties for Ant building
├── build.xml                The ant build file (generates the TradeMiner jar file)
├── cp.bat                   Copies the TradeMiner jar to Windows RapidMiner installation
├── cp.sh                    Copies the TradeMiner jar to Linux RapidMiner installation
├── javadoc                  Directory with Java documentation
├── lib                      Directory with required jar libraries for TradeMiner package
├── newLibs                  Directory with new libs under test
├── resources                Directory with images and configuration for operators
│   └── com
│       └── rapidminer
│           └── resources
│               ├── i18n
│               │  └ OperatorsDocTutorial.xml      Provides online help for Operators
│               └── OperatorsTutorial.xml          Provides configuration for Operators
└── src                                            Source code directory
    └── com
        └── rapidminer
            ├── operator
            │   ├── io                                     File i/o operators
            │   │   ├── CSVExampleSetWriterFixed.java       Fix for CSVExampleSetWriter
            │   │   └── MkDir.java                          Creates a directory
            │   ├── performance              Fix for Binomial Classification (In progress)
            │   │   ├── BinaryClassificationPerformanceFixed.java
            │   │   └── BinominalClassificationPerformanceEvaluatorFixed.java
            │   ├── preprocessing            Preprocessing operators
            │   │   ├── deprecated           Deprecated operators and failed experiments
            │   │   └── transformation       Data transformation operators
            │   │       ├── AdjustMapping.java       Fix for label mapping in ExampleSet
            │   │       ├── AggregateDecisionByTradeDatetime.java  CATS algorithm impl.
            │   │       ├── KNNUndersampling.java    Implementation of KNN Undersampling
            │   │       ├── LinearListFixed.java     Helper class to perform KNN search
            │   │       ├── PredictionCount.java     Prediction count for CATS
            │   │       ├── RandomUndersampling.java Random removal of maj. class examples
            │   │       ├── ReadMacrosFromData.java  Read data from macros.csv
            │   │       ├── SentenceStorer.java      Helper class for Text Summarization
            │   │       ├── SetDataFixed.java        Set data operator fixed
            │   │       ├── SMOTE.java               Implements the SMOTE algorithm
            │   │       ├── Summarise.java           Text Summarization algorithm
            │   │       ├── Summary.java             Summary representation
            │   │       ├── TextSummarization.java   Text Summarization operator
            │   │       └── ValueIterationFixed.java Value iterator with more iter. values
            │   └── trademiner
            │       ├── acquisition                        Data gathering operators
            │       │   ├── ContentUpdater.java            Content updater helper
            │       │   └── CustomContentUpdater.java      Content update operator
            │       ├── aligner                            News and stock alignment
            │       │   ├── AlignmentDAOImpl.java          Data access implementation
            │       │   ├── Alignment.java                 Alignment representation
            │       │   ├── IAlignmentDAO.java             Data access interface
            │       │   ├── NewsAligner.java               Alignment operator
            │       │   ├── RestrictiveAlignment.java      Restrictive alignment impl.
            │       │   └── SimpleAlignment.java           Base class for alignment
            │       ├── eval                               Evaluation operators
            │       │   ├── FixPerformanceVector.java      Fixes in the Performance Vector
            │       │   ├── GMeanPerformance.java          GMean performance calculator
            │       │   ├── LackOfMktdataException.java    Thrown when there is no mktdata
            │       │   ├── Portfolio.java                 Bought stocks set (Positions)
            │       │   ├── Position.java                  Represents a bought stock
            │       │   ├── RandomTrader.java        Generates random choices of investment
            │       │   ├── SimpleStrategySimulator.java   Simple strategy investment sim.
            │       │   └── WekaPerformance.java     Performance calculator with Weka
            │       └── util                         Helper classes
            │           ├── ConnectionFactory.java   JDBC connection factory
            │           ├── Constants.java           Constants used along the extension
            │           ├── Executor.java            Execute shell commands
            │           └── Util.java                Several utility methods
            └── PluginInitTutorial.java      Provides hooks for plugin initialization
RapidMiner_Unuk                              Base RapidMiner project
```

Figure 26 - Main files existing in the ~/workspace/trademiner directory.