Recommendation System for Market Abuse Detection

Financial Engineering Department

July 2020 - ver. 1.2

Abstract

Recommendation systems (`RecSys`) has experienced an explosion since 2007 when Netflix set up a challenge with one million dollars prize, addressed to the research teams of academics and business companies, in order to improve its old recommendation methodology. That's led to a huge competition and a rapid development of the methods and underlying theory. Nowadays recommendation system are used worldwide by e-commerce (Amazon, etc...) up to e-entertainment (Netflix, Spotify, YouTube, etc...). A `RecSys` is able to profile users and items based on their past interactions not just on a single-user basis, but within a collective framework, allowing to portray common patterns of interactions of each user even for those who rarely interacts. In this paper we present the results of our attempt to apply `RecSys` for Market Abuse Detection. In this reversed anomaly-detection perspective, deals ill-judged by the `RecSys` can be considered suspicious at least and worth reporting. At time of writing, at our knowledge, this is the first attempt to use a `RecSys` in a opposite way, not to recommend but instead as an anomaly detector tool.

Contents

1 Market Abuse Detection

1.1 Regulator's normative

Market Abuse Detection (MAD)[1] is the activity of monitoring the data flows of a financial market place (prices, orders and trades) with the aim to find anomalous and suspicious behaviors of market participants. The MAD activity consists of looking for a set of patterns in the data that can suggest that a player tried to manipulate market prices or took advantage in negotiation exploiting illegal or reserved information. The anomalous patterns in the data flow are defined by a list of rules set up by the Regulator in the following normative:

- Regulation(EU) No. 596/2014 of the European Parliament (MAR)
- ESMA 2015/224 Final Report (technical advice on possible delegated acts concerning MAR)
- Commission Delegated Regulation (EU) 2016/522 of 17 December 2015

There are several types of patterns that are organized into six subsections:

- Insider Dealing
- Market manipulation
- HFT
- Cross Product Manipulation
- Inter-Trading Venues Manipulation
- Bid/Ask Spread

Each of the patterns defines an algorithm, i.e. a metric, function of the market dataset: prices, volumes, order frequencies, executed trades, etc..., that has to be monitored and has to be limited to certain specific values, i.e. the thresholds. The overcoming of a threshold must first be analyzed and, if confirmed, trigger a reporting to the Regulator. There are at least fifty algorithms each one with its specific thresholds to be set.

The activity of MAD is demanded to the market participants so that each market institutional player has to monitor the data flow and, if the case, report abuses to the Regulator. In this context, institutional players must equip themselves with tools that, implementing the normative patterns, are able to rise alarms in case of threshold overcoming.

1.2 The problem with MAD tools

Since the quantitative value of a threshold is left to the player, there is always a big effort in fine tuning each algorithm in order to avoid the rising of too many or too few alarms. Because finding the best configuration can be very difficult, the risk is that those who have to monitor

[1] To avoid misunderstandings, let's clarify that in financial and regulatory contexts, the acronym *MAD* usually refers to the Market Abuse *Directive*, a first legislation framework published in the Official Journal of the European Union and entered into force on 12 April 2003, later replaced by the Market Abuse Regulation (MAR) published in the Official Journal of the European Union on 12 June 2014 and applied since 3 July 2016. In this paper, however, we will use the acronym *MAD* as a generic abbreviation for Market Abuse *Detection*.

the alarms end up to lose confidence in the monitoring tool due to presence of too many false positive and false negative.

In this context, machine learning methodologies (ML) can help saving time in the management of the alarms raised by the MAD tool giving focus only to those alarms that are worthy to be analyzed.

In particular, the analysis performed in this paper focus on the problem of finding, in a dataset of trade executions done in a specific market place, those trades that for their characteristics are alleged to be anomalous. For example a trade with an executed volume too high compared to the standards, a trade in a security that a player isn't used to trade, or a high number of executed trades in a short period of time on a security, compared to the standard trade frequency for a player.

With the help of ML, once MAD shows evidence of a suspicious trade, the operator can match the alarms raised by the normative patterns with the ones raised by ML method and analyze the intersection. Moreover the information given by the ML can be used to fine tune the thresholds in order to have a quasi-perfect match of the two subset of alarms. Consider further that the thresholds of the normative algorithms must be reviewed as time passes because market condition can change over time. The ML methods instead is non-parametric, so in principle it doesn't require maintenance over time, giving a static reference for thresholds fine tuning.

2 Recommendation System

How does YouTube or Netflix know what video you might want to watch next? How does the Google Play Store pick an app just for you? Magic? No, in both cases, an ML-based recommendation model determines how similar videos and apps are to other things you like and then serves up a recommendation.

A recommendation system helps users find interesting content in a large corpora. For example, the Google Play Store provides millions of apps, while YouTube provides billions of videos. More apps and videos are added every day. How can users find new interesting content? Yes, one can use search to access content. However, a recommendation engine can display items that users might not have intended to search on their own.

The basic concepts of a recommendation system are:

- items, the things to be recommended,
- users, who need an item,
- interactions, the past interactions between users and items

In these three abstract concepts it is possible to fit various concrete things in order to set up the context of a specific recommendation system. For instance Netflix items are films or series, the users are people, the interactions are ratings or just 'watched' or 'non watched'. For Amazon, items are products available for sale, the users are people, the interactions are a purchase or just a 'search' of an item.

2.1 Collaborative filtering

Collaborative filtering is one of the strategies on which are based recommendation systems. It uses similarities between users and items simultaneously to provide recommendations. This allows for serendipitous recommendations; that is, collaborative filtering models can recommend an item to user A based on the interests of a similar user B [1]

These similarities emerge algorithmically from the data itself and don't have to be provided explicitly. For example, consider a movie recommendation system. Suppose User 1 has viewed movies A, B, C, D, E, and F. User 2 has viewed movies A, B, D, E and F, but not C. Because of this high overlap, it's likely that both users share some basic preferences. If User 1 liked movie C, it's probable that User 2 would also like movie C if the user was aware of its existence.

In this example the user-item interaction data are represented by the movies a user has seen. Recommendation system rely on two kind of input data, which are placed in a matrix where each row represent a user and each column represent an item:

- Explicit feedback: users specify how much they liked a particular item by providing a numerical rating (e.g. Netflix stars system).
- Implicit feedback: users don't give an explicit rating on items, but their preferences can be inferred indirectly by observing users behavior (e.g. what movies he viewed on Netflix or what product he bought on Amazon).

One methodology to finding the similarity between users and items is by computing their *latent factors*. The latent factors are numbers that represent certain features of that user or

item. For instance two example of latent factors for Netflix are: if a movie is for adults or children (first factor), or if the film is a blockbuster or arthouse (second factor) [5]. If a user and an item have high value in the same latent factors, it means that the item has the same features that the user prefer. So it's probable that the user will like this item.

Well, but how to discover this latent factors?

2.2 Matrix factorization

Matrix factorization is a simple, and also one of the most successful, latent factors model [2]. It characterizes both items and users by vectors of factors inferred from their interactions. So each user u is associated with a vector $x_u \in \mathbb{R}^f$, and each item i with a vector $y_i \in \mathbb{R}^f$. In this case f is the dimension of the latent factors space. The score $\hat{r}$ that the `RecSys` will output, which can be thought of as a 'predicted rating', is calculated by taking the scalar product of the two vectors;

$$\hat{r}_{ui} = x_u^T y_i \tag{1}$$

Given an explicit feedback matrix $A \in \mathbb{R}^{m \times n}$, where m is the number of users and n is the number of items, the system learns the latent factors model by minimizing an L2-norm loss function such as

$$\min_{x^* y^*} \sum_{(u,i)\ \text{obs}} \left[\left(r_{ui} - x_u^T y_i \right)^2 + \lambda \left(\| x_u \|^2 + \| y_i \|^2 \right) \right], \tag{2}$$

where λ is a regularization parameter, used to avoid overfitting. The sum is done over the pairs (u, i) for which is given an observed rating r_{ui}.

In case of implicit feedback matrix, we must adopt a slightly different approach [1]. In this case the matrix elements r_{ui} don't represent an explicit rating, but an observed user u behavior on item i, e.g. how many minutes he has watched that TV show or how many times he has listened that song. Let's introduce a binary variable p_{ui} such that

$$p_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases} . \tag{3}$$

This variable represents the preference of user u over the item i. So if a user u has an interaction with item i ($r_{ui} > 0$) then we have an indication that u likes i ($p_{ui} = 1$). On the other hand, if no interactions have occurred ($r_{ui} = 0$), we have no information about the preferences of u on i ($p_{ui} = 0$). However, the values of p_{ui} are associated with varying confidence levels. In fact $p_{ui} = 0$ doesn't mean that user u doesn't like item i, because there could be other reasons why u never interacted with i. For example the user was unaware of the existence of the item. In the same way the values of $p_{ui} = 1$ doesn't mean that user u likes item i, for example a user can watch a movie or listen to a song just once and discover he doesn't like it. In general when r_{ui} grows, there is a stronger indication that the user indeed likes the item. For this reason we introduce the variables c_{ui} that measure the confidence in observing p_{ui}. A possible choice for c_{ui} would be

$$c_{ui} = 1 + \alpha r_{ui}. \tag{4}$$

Differently from the explicit feedback case, the scalar product between user and item vector gives the predicted preference $\hat{p}_{ui} = x_u^T y_i$. The loss function to be minimized then becomes

$$\min_{x^* y^*} \left[\sum_{u,i} c_{ui} \left(p_{ui} - x_u^T y_i \right)^2 + \lambda \left(\sum_u \| x_u \|^2 + \sum_i \| y_i \|^2 \right) \right]. \tag{5}$$

Compared to (2), the sum is done over all users and items, because in the implicit feedback model we must take into account also the absence of interactions between users and items.

The minimization of both loss functions (2) and (5) can be achieved using an optimization algorithm such as *Stochastic gradient descent* (SGD) or *Alternating Least Squares* (ALS). The results of the minimization are two new matrices:

- a user matrix $X \in \mathbb{R}^{m \times f}$, where row i is the latent factors vector for user i;
- an item matrix $Y \in \mathbb{R}^{n \times f}$, where row j is the latent factors vector for item j;

such that the matrix product XY^T is a good approximation of the input feedback matrix A, either in the case of explicit and implicit feedback.

Matrix factorization typically gives a more compact representation than learning the full matrix. The full matrix has $O(nm)$ entries, while the embedding matrices X, Y have $O((n+m)f)$ entries, where the embedding dimension f is typically small. As a result, matrix factorization finds latent structure in the data, assuming that observations lie close to a low-dimensional subspace.

Note that the L2-norm loss functions presented in this section are just one of the many objective functions used in the optimization procedure of recommendation systems. Other loss functions used in literature are for example the *Bayesian Personalized Ranking* (BPR) for implicit feedback models [3], or the *Weighted Approximated-Rank Pairwise* (WARP) loss function [4].

Once latent factors have been found one can query the `RecSys` to have an approval score for each couple user-item even if in the past that user never interacted with that item (i.e. the user never watched that movie). Picking a user and sorting the score given by the `RecSys` in a descending manner, one can rank each item in the user preference and can recommend the items (never watched before) with the higher score.

2.2.1 Adding Biases

Sometimes the observed variation in rating values is due to effects associated with either users or items, independent of any interactions. For example, in a movie recommendation system, such as Netflix, can be observed a tendency for some users to give higher ratings than others, and for some movie to receive higher ratings than others. This effect is known as *biases*. When this happens the equation (1) is no longer sufficient to explain the full rating value of interaction between user u and item i.

A first order approximation of the bias involved in rating r_{ui} is as follow:

$$b_{ui} = \mu + b_i + b_u. \tag{6}$$

The parameter b_u and b_i indicates the observed deviations, from the overall average μ, of user u and item i respectively. With this definition equation (1) becomes

$$\hat{r}_{ui} = \mu + b_i + b_u + x_u^T y_i. \tag{7}$$

The recommendation system learns the user-item factorization by minimizing the loss function

$$\min_{x^* y^*} \sum_{(u,i)\ \text{obs}} \left[\left(r_{ui} - \mu - b_u - b_i - x_u^T y_i \right)^2 + \lambda \left(\| x_u \|^2 + \| y_i \|^2 + b_u^2 + b_i^2 \right) \right], \tag{8}$$

or the equivalent one in the case of implicit feedback.

2.3 Model evaluation

Once we have trained our recommendations system on the observed data we need to evaluate if the model makes good prediction. In fact the objective function minimization process could face several problems, such as *underfitting* and *overfitting*, that would affect the performances of the fitted model.

Underfitting happens when the underlying model used is not suitable for the data. We can detect such a case when the minimization algorithm fails to find a minimum for the loss funcion. We could avoid the underfitting by changing the model.

Overfitting happens when the model fits the data so well that it misses completely the real information carried by the data set. Although it could seem that the minimization process succeded, the result is not reliable because the predictions will be affected by the presence of noise data points.

In general a strategy used to avoid the overfitting is to split the input data set in two sample: a train sample, over which the model will be trained, and a test sample, that will be used for testing the model in making good predictions. In the case of recommendation system, the split is made avoiding that all the interactions of a single user or item are removed from the train sample. If such a thing happens, that user or item wouldn't be characterize during the training process, and the recommendation system cannot predict a rating for it.

Once that a train-test split is performed, there are several ways to test the goodness of the recommendations system just obtained. We will summarize the most common evaluation method in the next sections.

2.3.1 Receiver operating characteristic (ROC) curve

A **receiver operating characteristic** (ROC) curve is a graph showing the performance of classification model at all classification thresholds. The curve plots two parameters:

- true positive rate (TPR) on the y axis;
- false positive rate (FPR) on the x axis.

True positive rate is defined as follows:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{9}$$

False positive rate is defined as follows:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}. \tag{10}$$

In this definitions TP is the number of true positive, FN is the number of false negative, FP is the number of false positive and TN is the number of true negative.

Let's specialize this tool in the framework of the recommendations system. After training the model on the train sample, we try to predict which items the users will prefer based on the data in the test sample. Obviously, for a single user we will recommend only items with which he didn't have interaction in the train sample. The classification threshold is the length of the recommended list for a single user, and we label the items in this list as *positive* items, while the others are the *negative* items. Therefore true positive are the items that showed up in the

recommended list and also had an interaction with that user in the test set, while false positive are the items in recommended list that didn't have an interaction with that user. False negative are the items labeled as negative that had an interaction with the user, and true negative are items that either are labeled as negative and didn't have interactions with the user. Varying the number of items in the recommended list we can plot a ROC curve for a single user.

Graphically, the plot of a ROC curve always starts from the point $(0, 0)$ and end in $(1, 1)$, and the more the curve is above the straight line from $(0, 0)$ to $(1, 1)$, the more accurate are the model predictions. This characteristic of the ROC curve is described quantitatively by the **area under the curve** (AUC) parameter, which is, as the name suggests, the area underneath the ROC curve from $(0, 0)$ to $(1, 1)$. AUC values ranges from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0; one whose predictions are 100% correct has an AUC of 1.

Thus, when we want to evaluate our recommendations system we calculate the AUC score for all the users, and then we take the mean score as the AUC score of the model.

2.3.2 Precision at k and recall at k

Precision and recall are classical evaluation metrics in binary classification algorithms, which could be translated in the framework of recommendations system. We will use the same definition given in section 2.3.1

Let's take the list of the top k items recommended to a user, where k is an integer. Precision at k is defined as:

$$p_k = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \tag{11}$$

meaning that it is the proportion of top k recommended items that had interactions with the user in the test set. Recall at k is defined as follows:

$$r_k = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{12}$$

meaning that it is the proportion of items that had interactions with the user in the test set that appears in the top k recommended items. Note that the definition of recall at k is the same as the definition of true positive rate.

As for AUC score, one could calculate precision at k and recall at k score for every users in the data set, then taking the mean score to evaluate the precision at k and recall at k of the model.

2.4 Recommendation systems as anomaly detectors

In the previous paragraphs we provided a classical definition of recommendation system for its typical usage: as an adviser. We already introduced the 'score' (section 2.2), a number that indicate a liking that a user has for each specific item. The score can be used to sort the items in a descending manner so that the ones at the top of the list are those 'most recommended'. Using this criteria then the items resulting at the bottom of the list may be those that the user will never claim, those that he doesn't like.

Consider for instance a person that is used to watch horror and action violence movies, and he never watched a romance movie. If we train the `RecSys` over the entire population of users and movies, then for our person the resulting score for horror movies will be high and low for

romance movies. Now suppose the case that our person will watch a romance movie which has a low score for him. Beyond the purely personal consideration that pushed him to watch this kind of movie, the point is that the behavior of our friend is in some way anomalous.

Following this criteria of looking for anomalies we performed an empirical analysis on a dataset of trade execution. The aim is to identify anomalous trades that are suspected to be driven by illegal information. The following section will explain how the analysis was performed.

3 Empirical analysis

3.1 Raw data

The dataset used for the analysis consists of about 2.6 million records concerning executed trades in a period of 3 months (from 15-Aug-2019 to 15-Nov-2019). It includes a total of more than 2 hundred distinct subjects dealing with a total of more than 5 thousand distinct securities traded within more than 50 markets.

Table 1: Executed contracts dataset example

MIC	SUBJECT	ISIN	QTY	PRICE	CNTR_VAL	CURR	TIMESTAMP
XTKS	1007120	JP3672400003	1700.0	648.200	1101940.00	JPY	2019-08-15T00:03:00.061000Z
XASX	1039910	AU000000ANO7	579.0	5.300	3068.70	AUD	2019-08-15T00:05:22.805000Z
XASX	1039910	AU000000ANO7	392.0	5.300	2077.60	AUD	2019-08-15T00:09:06.733000Z
XASX	1044976	AU000000STM0	250000.0	0.030	7500.00	AUD	2019-08-15T00:14:34.038000Z
XASX	1044976	AU000000BOE4	100000.0	0.056	5600.00	AUD	2019-08-15T00:14:40.368000Z
XAMS	1043883	GB00B03MLX29	30.0	25.100	753.00	EUR	2019-08-15T07:00:00.870623Z
XPAR	1021212	FR0000124141	16.0	21.700	347.20	EUR	2019-08-15T07:00:01.731558Z
XPAR	1021212	FR0000124141	10.0	21.700	217.00	EUR	2019-08-15T07:00:01.731558Z
XPAR	1021212	FR0000124141	84.0	21.700	1822.80	EUR	2019-08-15T07:00:01.731558Z
XPAR	1021212	FR0000124141	338.0	21.700	7334.60	EUR	2019-08-15T07:00:01.731558Z
XPAR	1021212	FR0000124141	126.0	21.700	2734.20	EUR	2019-08-15T07:00:01.731558Z
XPAR	1021212	FR0000124141	50.0	21.700	1085.00	EUR	2019-08-15T07:00:01.731558Z
XPAR	1021212	FR0000124141	95.0	21.700	2061.50	EUR	2019-08-15T07:00:01.731558Z
XPAR	1021212	FR0000124141	126.0	21.700	2734.20	EUR	2019-08-15T07:00:01.731558Z
XETR	1056156	DE000UNSE018	4.0	26.560	106.24	EUR	2019-08-15T07:00:02.730000Z
XPAR	1043883	FR0000131104	48.0	39.690	1905.12	EUR	2019-08-15T07:00:03.204824Z

Table 1 shows a few example lines of such dataframe. Each record is made up of the following fields, in the same order presented in the table:

- the market code
- an anonymized id for the subject
- the ISIN code of the security
- the quantity of the contract
- the unitary price of the security
- the countervalue (namely the quantity times the price)
- the currency
- the timestamp of the execution

A MAD-oriented analysis is interested in the market participants' activities, so a little bit of pre-processing is needed to get the relevant information from this *raw* dataset.

First of all, each record describes a single executed contract, while market players operate through either market-orders or limit-orders: so usually each subject's request matches several entries in the limit order book. The millisecond-precision timestamp allows us to wrap up all the records belonging to a single order performed by a single subject for a single ISIN on a certain moment.

Furthermore, since our analysis does not rely on a very accurate price assessment, a once-for-all exchange rate for each non-EUR currency is used to convert every contract's countervalue into a common EUR-denominated amount.

Finally, again from a MAD point of view, quantity and price of each order can be considered fluky inputs of the operation, the relevant measure being its countervalue.

Eventually our empirical starting point becomes the set of the executed orders together with their relevant features, namely:

- the anonymized id of the subject
- the ISIN code of the security
- the timestamp of the order
- the original currency of the order
- the countervalue *in EUR*

Table 2 shows the first lines of the resulting dataframe with columns in the same order as listed above. Our 2.6 million and more contracts' records aggregate to something more than 1.7 million orders, always from the same 2 hundred and more distinct subjects dealing with the same 5 thousand and more distinct ISINs.

Table 2: Executed orders dataset example

SUBJECT	ISIN	DATETIME	CURR	V
1007120	JP3672400003	2019-08-15 00:03:00.061000+00:00	JPY	9146.1020
1039910	AU000000ANO7	2019-08-15 00:05:22.805000+00:00	AUD	1902.5940
1039910	AU000000ANO7	2019-08-15 00:09:06.733000+00:00	AUD	1288.1120
1044976	AU000000STM0	2019-08-15 00:14:34.038000+00:00	AUD	4650.0000
1044976	AU000000BOE4	2019-08-15 00:14:40.368000+00:00	AUD	3472.0000
1043883	GB00B03MLX29	2019-08-15 07:00:00.870623+00:00	EUR	753.0000
1021212	FR0000124141	2019-08-15 07:00:01.731558+00:00	EUR	18336.5000
1056156	DE000UNSE018	2019-08-15 07:00:02.730000+00:00	EUR	106.2400
1043883	FR0000131104	2019-08-15 07:00:03.204824+00:00	EUR	9922.5000
1043896	DE0008404005	2019-08-15 07:00:11.464000+00:00	EUR	9849.0000
1021212	FR0000184798	2019-08-15 07:00:11.992409+00:00	EUR	642.6000
1021212	DE0007236101	2019-08-15 07:00:20.504000+00:00	EUR	521.0400
1021212	BE0003818359	2019-08-15 07:00:21.484318+00:00	EUR	3041.0000
1043883	DE000BAY0017	2019-08-15 07:00:22.786000+00:00	EUR	8512.0000
1021212	BE0003818359	2019-08-15 07:00:24.169605+00:00	EUR	10947.6000
1043883	GB0007980591	2019-08-15 07:00:26.649957+00:00	GBP	1749.9105

3.2 Hyperparameter selection

In order to find a suitable setup for the anomaly detection tool we were looking for, we had to proceed on two quite different levels which are common in machine learning and are usually called *hyperparameter*[2] *selection* and *model training*.

The former, in our case, regards basically to choose, or maybe to build, the data dimensions to be used as the three main roles of a `RecSys` namely the USERs, the ITEMs and the RATING.

If, on the one hand, the market player is by its very nature the preferred option for the USER, on the other hand several choices are available for the ITEMs, allowing one to set up different `RecSys`, each focusing on a different trait of the USER's behaviour. For example, opting for the more natural choice for the ITEM, namely the ISIN, may allow to sort out securities and subjects based on their mutual interactions. In another way, choosing the countervalue as the ITEM may allow to figure out the typical volume range of exercise of each USER. In this particular case where the selected dimension is a continuous quantity, one has to make use of a

[2]See for example: https://www.quora.com/What-are-hyperparameters-in-machine-learning

bucketing procedure in order to convert it into a categorical attribute belonging to a delimited and discrete list of items.

Elaborating further, one may *construct* new dimensions by combining the existing ones, aiming at discovering patterns in subjects' behaviour that *correlates* over different dimensions. For example one may join the ISIN with some bucketing of the countervalue in order to explore subjects' attitude where some kind of securities are traded in low volumes and/or in large amount, while other securities are traded in high volumes and/or in small amount.

3.2.1 RATING metric

For each choice of the ITEM dimension, one has to identify the metric to be regarded as the RATING a given USER ascribes to a given ITEM. In this case a quantitative dimension is favoured, so that a larger (smaller) value can be associated with a larger (smaller) rating. For example, in our case a natural choice is the countervalue, since a larger volume of purchases clearly express a greater valuation for a financial instrument. In any case, even if a direct quantitative dimension is not available in the original data, a straightforward procedure to get a measure of rating can be achieved by just *counting* the occurrences of records in the original flat database when pivoting it through a *group-by* operation using the USER and the ITEM dimensions as a double-keys index. Well, in fact, even if you are willing to use as RATING a direct quantitative dimension belonging to the original database, you still have to aggregate such values after the *group-by* since the `RecSys` needs a single rating score for a given (USER, ITEM) pair.

So, for example, with the notation of section 2.2 and using d to refer to a record (*deal*) in our dataset, the ratings r_{ui} of the feedback matrix can be written as:

$$r_{ui} = \#\Big\{d \in (u,i)\Big\} = \sum_{d \in (u,i)} 1 \tag{13}$$

in the case of the *count* aggregation function, whereas it would read

$$r_{ui} = \sum_{d \in (u,i)} V_d \tag{14}$$

if V_d is the countervalue of the order d and we choose the *sum* as the aggregation function.

Actually the specific aggregation function used to build the rating can be quite critical on the effectiveness of the `RecSys` ability to capture the patterns in the users' behaviours. For example, by using a simple aggregation function like *count* (or *sum* for a direct quantitative dimension) may results in a disproportional characterization of the population since there are very few USERs that make by far too many transactions (or that exchanges by far too much countervalue) with respect to the others.

A *naïf* improvement could be to normalize such rating dividing it by the *total* count (or *total* sum of the countervalue) *for each single* USER:

$$r_{ui} = \frac{1}{\mathcal{N}_u} \sum_{d \in (u,i)} 1 \qquad \text{where} \quad \mathcal{N}_u = \sum_{d \in u} 1 \tag{15}$$

$$r_{ui} = \frac{1}{\mathcal{V}_u} \sum_{d \in (u,i)} V_d \qquad \text{where} \quad \mathcal{V}_u = \sum_{d \in u} V_d \tag{16}$$

However this could still not be suitable enough: each of the few securities traded by a subject with a very small activity would result in a high rating for her, while on the contrary the many securities traded by a subject with a broad and scattered activity would each result in a very small rating for him. It turned out that a better approach is to use the *max*, instead of the *total*, over each USER, for the normalization of the aggregation function result:

$$r_{ui} = \frac{1}{\hat{\mathcal{N}}_u} \sum_{d \in (u,i)} 1 \qquad \text{where} \quad \hat{\mathcal{N}}_u = \max_{d \in u} \left\{ \sum_{d \in (u,i)} 1 \right\} \tag{17}$$

$$r_{ui} = \frac{1}{\hat{\mathcal{V}}_u} \sum_{d \in (u,i)} V_d \qquad \text{where} \quad \hat{\mathcal{V}}_u = \max_{d \in u} \left\{ V_d \right\} \tag{18}$$

In this way, for example, if a sporadic subject traded an equal amount (either of countervalue or of number of deals) of a few securities, they will result in an equal rating for him, regardless of the *total* amount of activity. In a similar fashion, if a very active subject traded mostly and with a similar amount many different securities, they will result for her in a similar rating value not deflated by its own widespread presence.

3.2.2 ITEM dimensions

As said before, several choices are available for the ITEM dimension. We explored the following options.

3.2.2.1 ISIN

This is the first and natural setup: here the goal is to characterize subjects and securities based on their trade patterns, clustering together users which deal on similar instruments, where *similar* instruments means precisely that are dealt by *similar* users. Such description is not a *petitio principii*, on the contrary it is precisely what a `RecSys` is supposed to discover by trying to properly calibrate the embedding (see section 2.2).

As said, in this case we explored two distinct `RecSys` which used either *counting* or the countervalue as RATING.

3.2.2.2 Countervalue

In this case we need to carry out a binning discretization procedure in order to map a nearly-continuous range of values into a finite set of ITEMs. Since these countervalues show a somewhat bell-shaped distribution when plotted on a *log* scale, we adopted a set of bins with *log*-uniform sizes. Figure 1 shows the distribution of countervalues over our whole dataset with the actual binning used in our analysis. It has been chosen in such a way to have a bin centered on each round number; moreover, in order to have not too few bins — which are nothing but the ITEMs of our `RecSys` — we added a further bin in between. Notice that the lef-most (right-most) bin was extended to include any smaller (larger) countervalues with respect to the minimum (maximum) one actually presented in the dataset: in this way a new deal not already present in the training database could be rated anyway.

As said, in this case we explored a single `RecSys` which used just the *counting* as RATING since the countervalue already takes part in — it actually is — the ITEM.

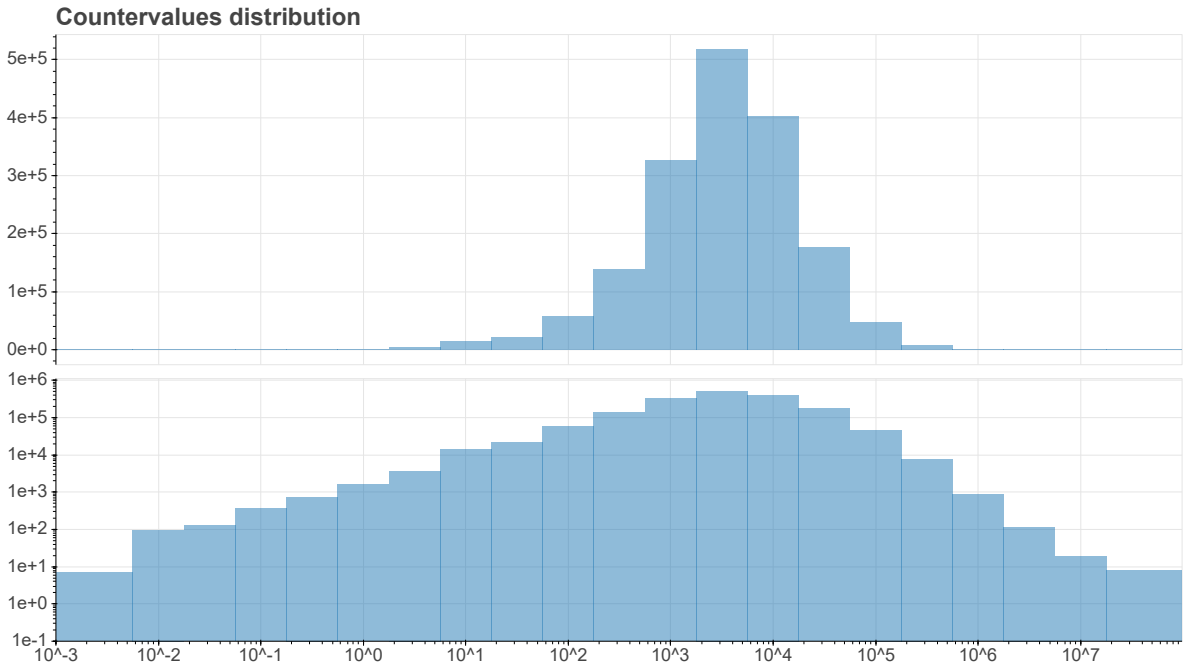


Figure 1: Countervalues distribution in our empirical dataset

3.2.2.3 A join of ISIN and a *per-subject* volume-level

Combining the two `RecSys` described above one could discover anomaly trades for some user on unusual volumes or on unusual securities. But these two facilities stands alone with respect to each other, so there's no way for them discover patterns of activities involving correlations between their ITEM dimensions: for example some volume range could be normal for a subject when trading certain kind securities while for other kind of securities the usual volumes lie on a different range.

In order to discover such correlated patterns one must set up a single `RecSys` in which the ITEMs dimension *combines* the two separate properties. At first one may think of combining precisely the two dimensions described above so that a single definite ITEM become the concatenation of the ISIN and the volume bucket. Such an attempt presents however several drawbacks. First of all, it increases too much the number of ITEMs: about 20 countervalue buckets times about 5 thousand securities gives about 100 thousand ITEMs, which must be compared with less than 2 million executed orders at disposal. The resulting interaction matrix would be too sparse to allow any pattern discovery. Moreover the range of traded volumes may be quite different from subject to subject, so one may think of adjusting the countervalues' binning procedure by some sort of per-user normalization factor, just like we did when using it as the RATING dimension.

These reasons led us to build a new dimension for our dataset, let's call it *subject volume-level*, $\mathcal{L}_u$, with just three categories — L, M and H — which mimic the three main parts of a boxplot. Specifically, each deal can be ranked as *L*ow (*H*igh, respectively) if the countervalue of the order falls in the first (last, respectively) quartile of their distribution restricted to the subject it belongs to, and accordingly the deal will be ranked as *M*id if the countervalue of the order falls in the *interquartile range*.[3] Alternatively one can choose an evenly distributed ranking and use *tertiles* instead of quartiles.

[3]See for example https://en.wikipedia.org/wiki/Interquartile_range

Thus, using the notation of section 3.2.1 and letting $F_{|u}(V)$ be *cumulative distribution function* of the countervalues V for *each* user u, we can write:

$$\mathcal{L}_u(V_d) = \begin{cases} L & \text{if } V_d < \tilde{F_{|u}}^{-1}(\frac{1}{q}) \\ H & \text{if } V_d > \tilde{F_{|u}}^{-1}(1-\frac{1}{q}) \\ M & \text{otherwise} \end{cases} \tag{19}$$

where $q = 3$ for tertiles or $q = 4$ for quartiles, and where we denoted with $\tilde{F_{|u}}(V_d)$ the *empirical distribution function* of the values $\{V_d\}_{d \in u}$, used as an estimation proxy[4] for $F_{|u}(V)$.

In this case, since the countervalue is already involved in the construction of the ITEM dimension, we used just the *counting* as RATING.

This may rise some concerns due to the fact that in this case the *counting* is involved too, albeit indirectly, in the construction of the ITEM dimension. In fact, the countervalue thresholds for the *L*ow/*H*igh ranks are based on *how many* deals a subject executed on some range of volumes. However one can argue that there is no inherent incongruity since the distribution which controls the thresholds is based on the *whole* set of the subject's deals, while the *counting* used for the rating is restricted to each specific security — actually each specific *H/M/L*-ranked security as ITEM.

For example it could happen that a user traded a specific ISIN `XYZ` on the whole spectrum of its countervalue range, and in this case the corresponding three items — `XYZ-L`, `XYZ-M` and `XYZ-H` — would indeed receive a pretty similar rating (for $q = 3$). But it could happen — and it is precisely what such a `RecSys` is supposed to discover — that the same user traded another security `ABC` mostly on the lower range of her typical *overall* volume range while trading a further security `DEF` mostly on the higher *overall* range. In this case the items `ABC-L` and `DEF-H` would receive a higher rating value with respect to `ABC-M`, `ABC-H`, `DEF-L` and `DEF-M`.

3.2.2.4 A join of ISIN and a *per-security* volume-level

With the same approach employed above, one can build another new dimension for our dataset, let's call it *security volume-level*, $\mathcal{L}_I$, similar to the previous one but in which the distribution used to rank each deal with the appropriate label *H/M/L* is taken on a per-security basis, instead of on a per-subject basis. To wit:

$$\mathcal{L}_I(V_d) = \begin{cases} L & \text{if } V_d < \tilde{F_{|I}}^{-1}(\frac{1}{q}) \\ H & \text{if } V_d > \tilde{F_{|I}}^{-1}(1-\frac{1}{q}) \\ M & \text{otherwise} \end{cases} \tag{20}$$

where I stands for ISIN and plays the same role of the user u in equation (19).

The goal is to capture pattern of usage with respect to the countervalue distribution *of each specific security*. For example, with regard to a some given ISIN, some users can be found to be trading typically in the lower or middle region of the overall volume spectrum for such instrument, while other users may be found to be trading in the larger part of it. A new deals which breaches such a patter could therefore be viewed as a signal of some anomaly behaviour.

[4]Properly interpolating such a step function for possibly small samples may allow to calculate the volume-level even for new orders not belonging to the training dataset, with volume values straddling the quantile thresholds; in this way the order can be casted to a definite item and the `RecSys` would be able to return a score for it too.

As in the case above, since the countervalue is already involved in the construction of the ITEM dimension, we used just the *counting* as RATING.

3.3 Model training and evaluation

For each particular setup described in the previous sections, i.e. the triplet (USER, ITEM, RATING), it is necessary to perform a calibration run over the input dataset, namely to apply one of the minimization procedures referred in section 2.2, in order for the `RecSys` to learn the latent factors and eventually to be able to give a score to any (USER, ITEM) pair. Strictly speaking, each particular setup does not exhaust all the hyperparameter selections available, since it remains to be decided:

- the kind of feedback matrix to use (explicit or implicit factorization),
- the dimension of the latent factors space,
- which optimization algorithm to use,
- what kind of loss function to minimize,
- what values to give to specific parameters of the calibration procedure (the regularization constant in the loss function, the various tolerances of the optimization algorithm...).

Notice however that whilst the different setups are not necessarily competing with each other, the choice of the other parameters is intended to tune the ability of the `RecSys` to learn usage patterns. An evaluation metric is therefore necessary in order to compare the different calibrated models and choose the most effective one. Since we are running a `RecSys` as anomaly detector, i.e. from a 'bottom-up' perspective (see section 2.4), precision/recall at k metrics (see section 2.3.2) are not much useful: we are not really interested in finding as many as possible interacted items among an as short as possible recommendation list. Actually, if precedents of anomalous transactions were available, such traditional evaluation metrics for recommender systems like the *Mean Average Precision* [7] could be employed by feeding them with the known-anomalous transactions and measuring how anomalous where scored by the `RecSys`. But this would be a case of supervised learning, where more direct approaches could be employed probably with better results with respect to a `RecSys`.

In any case, a generic score like the AUC value (see section 2.3.1) is yet a good test for the classification performance of the calibrated model.

3.4 Results

As anticipated before, we have experimented with several configurations. In each of them the USER role has been played by the subject column and a counting aggregation function was employed as RATING through the *max-within-user* normalization procedure as in equation (17).
Notice that the several `RecSys` examined are not necessarily competing against each other. Instead, any `RecSys` which is able to calibrate successfully, i.e. to capture patterns of common usage (and, by complement, patterns of anomaly usage) between its pair of USER/ITEM dimensions, can be used to setup a battery of detection tools which cooperate to raise alarms each on a specific aspect of the user's behaviour.

3.4.1 Model evaluation

As mentioned in section 3.3, since our scenario is not that of supervised learning, we are forced to rely on a generic classification performance metric to assess the calibrated `RecSys`, namely the area under the receiver operating characteristic curve (see section 2.3.1).

But even before that, we had to face a fitting convergence evaluation problem.

In fact, the calibration procedure consists in applying an iterative optimization algorithm with a loss function which somehow measures the difference between each interaction element (the rating, in case of explicit feedback) and the score returned by the `RecSys`. But since the ultimate purpose of a `RecSys` is not to truly forecast the users' judgment, but just to *rank* the items for each user, an accurate agreement between the actual users' rating and the returned numerical score value is a by far overkilling request. So, while the optimization algorithm in a generic curve fitting scenario is usually run up to some tolerance threshold on the loss function, in a `RecSys` framework it's standard practice to just run it for a few step. This however leaves open the questions of when to consider a calibration procedure 'settled' and if the achieved calibrated model is meaningful whatsoever.

For this purpose we have adopted a self-consistency criterion, namely that different runs of the calibration procedure — with a different random seed but with the very same hyperparameter values, including the number of iterations to run — should generate models which give similar scores to available (USER, ITEM) pairs.

A visual representation of such a criterion can be found in a plot where score results from a couple of runs are plotted one against the other. In an ideal case of a deterministic result of the calibration procedure, each run would settle on the very same model and the plot would result in a perfect straight bisector line. In a more realistic but still good case, each calibration procedure would not reach exactly the same model, but nevertheless each of them would return similar score to the same pair. So the plot would result in a more blurred but yet bisector-like line.
A few examples of such plots are shown in Figure 2.

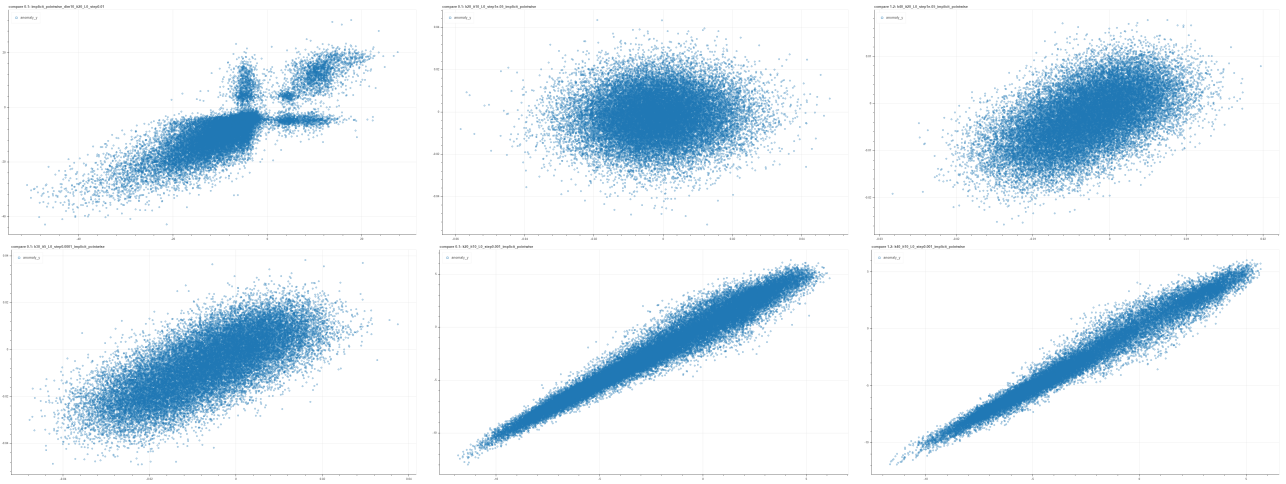


Figure 2: Examples of consistency plots for pairs of runs with the same hyperparameter values: from left to right and then from top to bottom the figures show gradually better situations. In each plot a single point represents a single deal and it's placed at coordinates (x, y) based on the score returned by the `RecSys` calibrated in the first (x) and in the second (y) run with the same choice of hyperparameter values.
NB: these plots do not represent intermediate steps along a single calibration procedure, but each is the outcome of a couple of iterative optimization runs for a specific — and completely different from one plot and the other — choice of hyperparameter values.

Notice that each subframe is not a snapshot along one calibration run, but they represent completely different hyperparameter selections. Moreover, performing further optimization steps, for example for the first quite bad cases, generally does not improve the consistency, meaning that such particular arrangement of hyperparameters is just not adequate to capture features of the system.

Eventually, once gathered the most consistent hyperparameter configurations, one can further inspect their ROC curves and the corresponding AUC value to select the best configuration (see Figure 3 for an example). It turns out, however, that the consistency condition is the stricter one: if, on the one hand, it happened that some 'bad plot' configurations still had a quite high AUC value, on the other hand all 'good plot' configurations performed well from the point of view of the AUC as well.

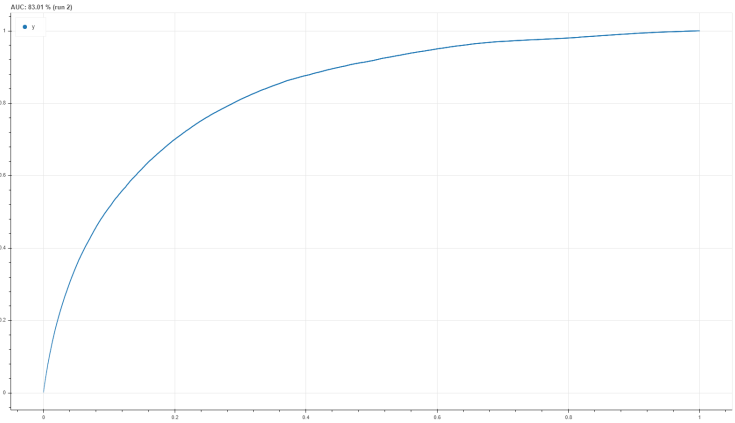


Figure 3: Examples of a ROC curve of a well consistent calibrated `RecSys`. The corrisponding AUC is 83.01%

3.4.2 Score examples

In what follows we will focus on the results of three specific instances of `RecSys` among those described so far. For convenience, we will use the notation $\mathcal{RS}^{\text{ITEM}}_{\text{USER}}$ to refer to each of them, depending on what was used as USER and as ITEM dimensions. Namely we will use:

- $\mathcal{RS}^{\text{I}}_{\text{U}}$ to denote the `RecSys` (see paragraph 3.2.2.1) which uses
  - the subject as the USER dimension
  - the ISIN as the ITEM dimension

- $\mathcal{RS}^{\text{V}}_{\text{U}}$ to denote the `RecSys` (see paragraph 3.2.2.2) which uses
  - the subject as the USER dimension
  - the countervalue as the ITEM dimension

- $\mathcal{RS}^{\text{I+V}}_{\text{U}}$ to denote the `RecSys` (see paragraph 3.2.2.3) which uses
  - the subject as the USER dimension
  - a join of the ISIN and a *per-subject* volume-level as ITEM dimension

One of the most important strengths of a `RecSys` approach is that it is able to assign a score even to users or items who have interacted little, if a sufficiently populated *overall* interaction sample is provided.

A prime example is given by a plot like the one shown in Figure 4, featuring $\mathcal{RS}^{\text{I}}_{\text{U}}$, where *all* securities available in the dataset were given a score for a certain subject — not just the possibly small set of items she already dealt with in the training dataset — and sorted accordingly.

We already inverted the scale of the score with respect to the original output of a `RecSys`, so that we can directly read it as an *anomaly* score. Hence a lower value for an instrument means that the `RecSys` would recommended it for such a user, which in turns means that, if traded, it would be an interaction entirely consistent with that user's typical behavior. On the other hand, a higher score would classify that instrument as highly anomalous, if dealt by that user.

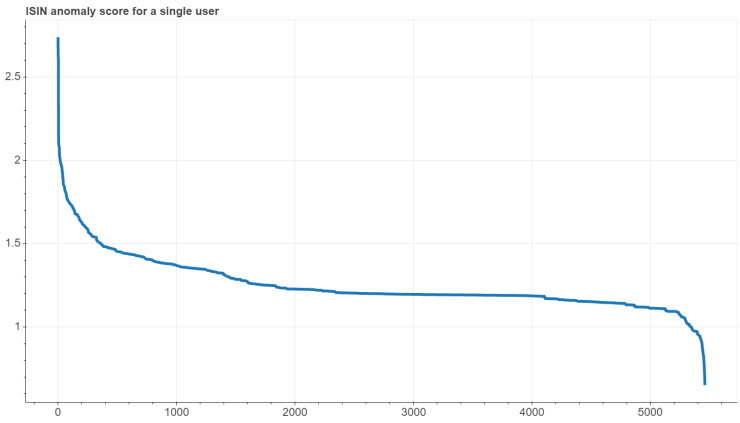


Figure 4: Anomaly scores of all the ITEMs available returned by $\mathcal{RS}^{\mathrm{I}}_{\mathrm{U}}$ for a single USER — in this case, each of the more than 5 thousand distinct securities. ITEMs on the x-axis are sorted according to the score.

As you can see, $\mathcal{RS}^{\mathrm{I}}_{\mathrm{U}}$ highlights a rather small group of instruments as 'typical' of that user: the ones with a higher rank, i.e. on the far right of the x-axis; then it classifies most of the instruments as 'neutral', the large central plateau: securities that are not specifically tailored for that user, but nonetheless plausible if traded by her; and finally it highlights a once again quite small group of instruments as 'anomalous', on the far left of the x-axis — what we are looking for.

Figure 5 shows the same plot for $\mathcal{RS}^{\mathrm{V}}_{\mathrm{U}}$. Here the ITEMs scored are the countervalues *log*-scale bins. In this case the ITEM dimension is not categorical and has its own metric; so the bins were not sorted on the x-axis by the score, but were left in their natural order.

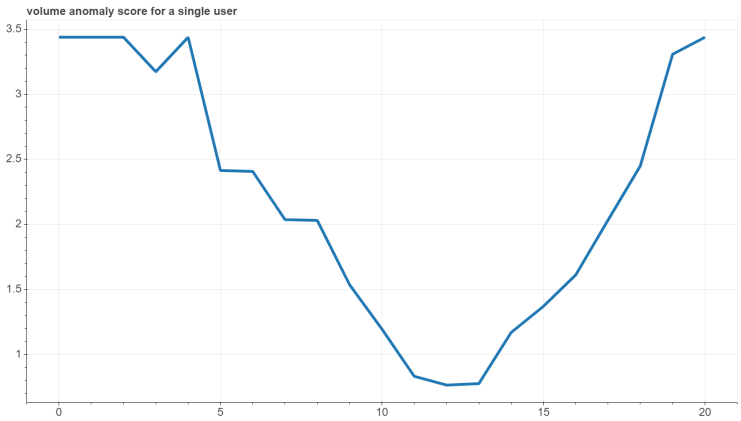


Figure 5: Anomaly scores of all log-scale countervalue bins for a single USER returned by $\mathcal{RS}^{\mathrm{V}}_{\mathrm{U}}$.

The plot clearly shows how $\mathcal{RS}^{\mathrm{V}}_{\mathrm{U}}$ finds a quite definite countervalues region that it deems characteristic of the subject, since it increasingly judges as anomalous those bins that gradually move away from it, both on the left (smaller countervalues) and on the right (higher countervalues). The simple bell-shape of this output — which just resembles a vertical reflection of the histogram of the countervalues distribution of Figure 1 on page 15 — totally makes sense, since a quite natural pattern for each subject is just to have a main operating interval with increasingly reduced excursions moving away from that interval. It might also suggest that, in this particular case of an ITEM dimension with a proper metric, perhaps the same result could be obtained with a simple statistical analysis of the distribution of orders of each individual user. However, it should be borne in mind that not all users have a sufficiently large operation to allow to outline a countervalue profile from a single-user statistical analysis, while a `RecSys` approach can manage to sort out users looking at them as a whole.

3.4.3 Universal score reshaping

Figure 6 shows the third example of an ITEMs ranking plot for a single USER, this time from $\mathcal{RS}_{\text{U}}^{\text{I-V}}$.

As you can see, the y-axis scales of the last three plots are quite different from each other, since they reflect the details of the specific calibrated model. So, should a battery of `RecSys`-based anomaly detector tools be set up, specific alarm thresholds for each of them would be fine tuned.

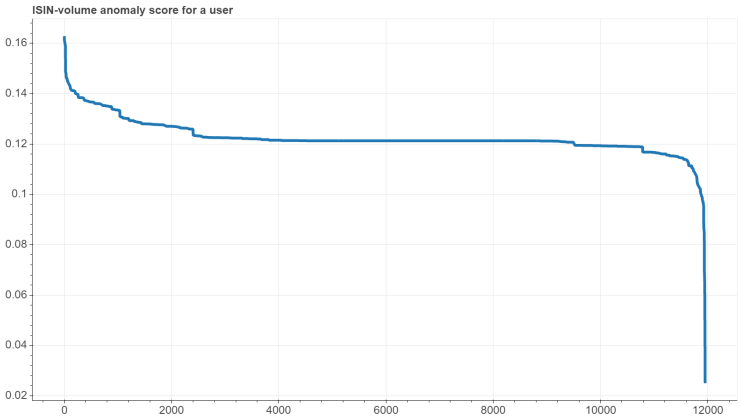


Figure 6: Anomaly scores of all the ISIN-*per-subject*-volume-level ITEMs for a single USER returned by $\mathcal{RS}_{\text{U}}^{\text{I-V}}$.

But even if the raw scores $\mathbf{s}_{\mathtt{R}}$ of each `RecSys` `R` were linearly rescaled to a normalized n-score:

$$\mathbf{n} = \frac{\mathbf{s} - \min_{\mathtt{R}}(\mathbf{s})}{\max_{\mathtt{R}}(\mathbf{s}) - \min_{\mathtt{R}}(\mathbf{s})} \tag{21}$$

so that all the `R`'s would return a $\mathbf{n}_{\mathtt{R}}$ that ranges between 0 and 1, the problem of the different distributions of these values would remain: the plateau of the plot in Figure 4, for example, is much closer to the most anomaly values than that of the plot in Figure 6, which instead is much closer to the opposite side. So, in these examples, the majority of pretty ordinary trades would be ranked with a much lower $\mathbf{n}_{\mathtt{R}}$ by $\mathcal{RS}_{\text{U}}^{\text{I}}$ with respect to what $\mathcal{RS}_{\text{U}}^{\text{I-V}}$ would return. Or, in other words, the middle point 0.5 of such $\mathbf{n}_{\mathtt{R}}$ anomaly score would be a quite anomalous value if returned by $\mathcal{RS}_{\text{U}}^{\text{I}}$, while it would definitely fall in the 'recommended items' region if returned by $\mathcal{RS}_{\text{U}}^{\text{I-V}}$.

A possible way of overcoming such a variability is to convert each specific anomaly score into a universal z-score [8] value, based on the distribution of the scores returned by each calibrated `RecSys` for the full interaction matrix. However, since generally these empirical distributions are not only not-Gaussian, but not even bell-shaped, we computed the z-score by means of the *cumulative distribution function* (CDF) [9, 10], rather than just by its mean and standard

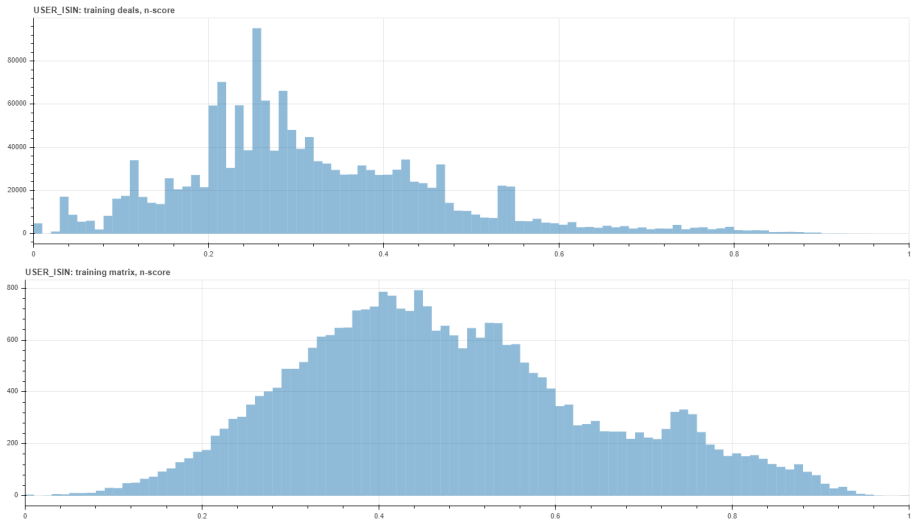


Figure 7: Distribution of the normalized score within the training dataset. The upper histogram counts each deals on its own, while the bottom one counts just the interaction matrix elements.
Notice that, due to the very definition of the rating metric, values lie mainly on the left of the upper histogram, since deals who occur often correspond usually to lower anomaly scores.

deviation. Actually we mapped the empirical CDF to the gaussian CDF:

$$\text{z-score} = CDF^{-1}_{\text{normal}}\big(CDF_{\text{empirical}}(\text{score})\big) \tag{22}$$

Using the quantile rank [11] function $\mathcal{Q}_{\mathrm{R}}$ of the normalized score $\mathbf{n}_{\mathrm{R}}$ as a numerical proxy of the empirical CDF and by applying the inverse of the gaussian CDF, as usually denoted by Φ_{μ,σ^2}, with the *standard* parameters, namely zero mean $\mu = 0$ and unitary variance $\sigma^2 = 1$, our z-score $\mathbf{z}_{\mathrm{R}}$ is given by:

$$\mathbf{z}_{\mathrm{R}} = \Phi^{-1}_{0,1}\big(\mathcal{Q}_{\mathrm{R}}(\mathbf{n}_{\mathrm{R}})\big) \tag{23}$$

This provides a universal way of representing an anomaly score, since it can be directly interpreted as the distance of a particular returned value from the mean of all possible scores, measured in terms of the width of the distributions of the scores themselves.

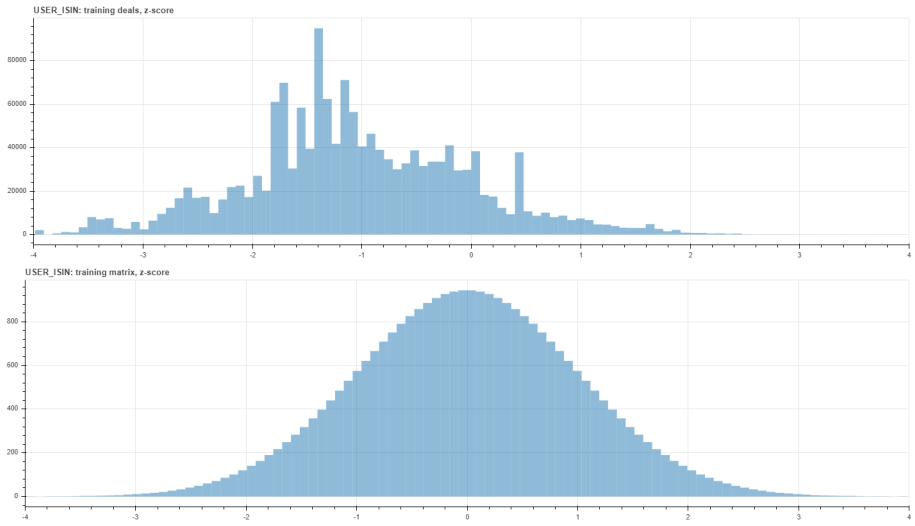


Figure 8: Distribution of the z-score within the training dataset. The upper histogram counts each deals on its own, while the bottom one counts just the interaction matrix elements.

Notice that the latter is a perfect standard normal distribution by construction.

3.4.4 Resulting anomalies

Although we are not in a supervised learning setup (see comments in section 3.3 and section 3.4.1), we wanted to try to set up some sort of a backtesting [12, 13] facility. Specifically, for the training of the `RecSys` we have removed the last five working days from the three months of available data, leaving them aside for the subsequent testing phase. The main purpose is not, indeed, to be able to accurately quantify the reliability of the `RecSys` as an anomaly detector — without any reference anomaly to compare its alarms with. Its aim is instead to provide a kind of 'investigation lab' that would allow for qualitative analysis such as gauging the number of alerts raised or inspecting directly with a human-eye the transactions reported.

Table 3 shows the first ten records — in descending order according to the anomaly score — within the 5 days of the 'backtesting' dataset as returned by $\mathcal{RS}^{\mathrm{V}}_{\mathrm{U}}$. To clean up this shortlist from the low-volume side anomalies, which are supposed not to be relevant in a MAD perspective, a preliminary filter has been applied to remove

Table 3: Top ten countervalues anomalies as returned by $\mathcal{RS}^{\mathrm{V}}_{\mathrm{U}}$

DATE	SUBJECT	vbinx	vbin	#	n-score	z-score
2019-11-12	41932450	19	~3.2e+06	1	0.909352	2.249750
2019-11-14	41932450	19	~3.2e+06	1	0.909352	2.249750
2019-11-15	49751457	19	~3.2e+06	1	0.891372	2.124371
2019-11-14	49751457	19	~3.2e+06	1	0.891372	2.124371
2019-11-15	44159571	19	~3.2e+06	1	0.890290	2.112326
2019-11-11	1043496	19	~3.2e+06	1	0.849349	1.784060
2019-11-15	24093493	18	~1e+06	1	0.847803	1.760141
2019-11-12	1043453	18	~1e+06	1	0.832035	1.681002
2019-11-13	1043453	18	~1e+06	1	0.832035	1.681002
2019-11-15	36211507	18	~1e+06	2	0.798511	1.517740

all records with a volume bin below ~$1.0e+00$. Since a `RecSys` returns an anomaly score based just on the pair (USER, ITEM), it is quite possible that more than a deal appear in the testing dataset that belong to the same pair: for $\mathcal{RS}^{\text{V}}_{\text{U}}$ it corresponds to the case in which the same subject has performed more than one transaction, perhaps with different ISINs, in which the individual volumes fell within the same bin. This is the meaning of the column '#' in these tables, in which the count of the transactions that correspond to that particular anomaly is reported. We kept these counting separated for each single day of the testing dataset, as if our anomaly detection tool had run on a daily basis, reporting the anomalies of each single day. This is why, for example, the first two lines, apparently identical, appear distinct — the transactions occurred for the same subject with the same volume bin, but on two different days — while the last one was merged with a count of #2 — the transactions occur on the same day.

The plain way to *read* Table 3 is that the most anomalous record corresponds to a deal made by subject `41932450` with a countervalue remarkably greater than 1 million but remarkably lower than 10 million[5] as well. Such deal was made twice, within the five days of our testbed, the first on Nov. 12$^{\text{th}}$ and the second on Nov. 14$^{\text{th}}$. The n-score is about 90% near to top but the really meaningful number is the z-score, which says that such anomaly is just over 'two sigma' away from the average score: so not really much an anomaly indeed.

The plain way to *interpret* this result is that a deal with a countervalue as large as that one is quite unusual for that subject or for similar subjects — similar, let's stress this once again, in the specific meaning tha the `RecSys` itself build up, namely, for $\mathcal{RS}^{\text{V}}_{\text{U}}$, that usually execute orders with similar countervalues.

Table 4 shows the corresponding results for $\mathcal{RS}^{\text{I}}_{\text{U}}$ and Table 5 for $\mathcal{RS}^{\text{I-V}}_{\text{U}}$. Again, to clean up the latter shortlist from the low-volume side anomalies, which are supposed not to be relevant in a MAD perspective, a preliminary filter has been applied to remove all records with the *Low* (`L`) *subject volume-level* in the ITEM column of Table 5. *En passant*, compare the scores of the second row of the latter with the first row of the former: the differences are small, but in any case it is just one of those situations in which the values of the n-score and z-score are orded-reversed, highlighting the role of a universal reference of the latter.

Table 4: Top ten ISIN anomalies as reported by $\mathcal{RS}^{\text{I}}_{\text{U}}$

DATE	SUBJECT	ISIN	#	n-score	z-score
2019-11-15	1000456	JP3942800008	1	0.972784	3.851117
2019-11-11	1000456	JP3942800008	1	0.972784	3.851117
2019-11-13	1043896	CH0002187810	2	0.944832	3.374233
2019-11-12	1043896	US92823T1088	5	0.943670	3.363270
2019-11-15	1043896	IT0005388449	1	0.939825	3.314311
2019-11-13	1043896	IT0005388449	1	0.939825	3.314311
2019-11-12	1043896	IT0005388449	2	0.939825	3.314311
2019-11-14	1009036	GB0009039941	2	0.939060	3.307272
2019-11-11	1009036	US6443931000	1	0.938117	3.270927
2019-11-15	1043896	SG1P32918333	1	0.936573	3.220750

Table 5: Top ten ISIN@vol-level anomalies as reported by $\mathcal{RS}^{\text{I-V}}_{\text{U}}$

DATE	SUBJECT	ISIN@vol-level	#	n-score	z-score
2019-11-15	1026339	FR0000120354 H	1	0.971891	3.435978
2019-11-15	1039910	IT0005388449 H	2	0.965881	3.344399
2019-11-12	1039910	IT0001352217 H	2	0.964955	3.330588
2019-11-15	1039910	BE0003839561 M	1	0.963047	3.264385
2019-11-11	1028965	IT0005388449 H	1	0.962350	3.255562
2019-11-15	1028965	IT0005388449 H	3	0.962350	3.255562
2019-11-14	1028965	IT0005388449 H	1	0.962350	3.255562
2019-11-13	1043896	CA00851F1062 M	1	0.958739	3.182827
2019-11-15	1021212	US8485741099 M	1	0.958391	3.170235
2019-11-14	1007594	US2537483057 M	1	0.952928	3.064075

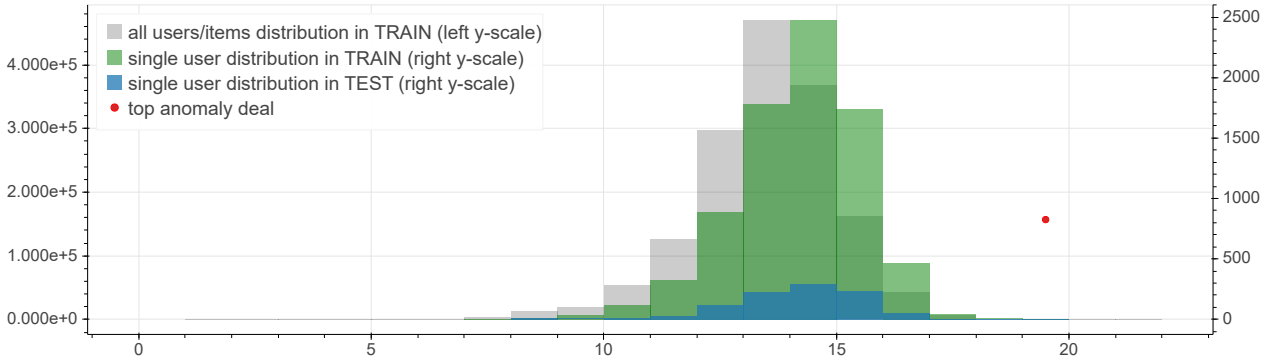
As you can see, in both cases their topmost rows have higher values than those of $\mathcal{RS}^{\text{V}}_{\text{U}}$. In particular, the higher values of the z-score tell us that this is not simply the effect of a different score-rescaling, but they are really more unusual transactions compared to the overall behavior of the dataset. This

[5] The bin labelled as ~3.2e+06 is just the one between (in a log scale) the two bins centered in the 'round numbers' 1 million and 10 million, respectively. See paragraph 3.2.2.2 for the details of our binning discretization procedure.

makes sense, since the distribution of countervalues is not rigid and compartmentalized, so even extreme countervalue for a certain subject will probably have corresponding cases, or at least quite close, in the training dataset.

In fact, as a self-consistency check, one can inspect the training dataset looking for items similar to the one reported as the most anomalous one in the test dataset. To this purpose, Figure 9 try to compare the empirical distributions of the countervalues for the involved user.

Figure 9: Countervalue distributions related to the most anomalous $\mathcal{RS}_U^V$ record



Comparison of the countervalues empirical distributions: for the overall training dataset (gray, left y-scale), for the involved subject `41932450` in the training dataset (green, right y-scale) and for the same subject in the testbed dataset (blue, right y-scale). The red dot marks, at a fictitious height, the horizontal position of the bin corresponding to the anomaly record.

The gray histogram in the background shows the overall distribution of the countervalues in the training dataset against the left y-axis. The overlying green histogram shows the countervalue distribution, again in the training dataset, but just for the subject concerned, against the *right* y-axis to allow a comparison, given the different count scales. The same single-user distribution, but for the test dataset, is superimposed in blue.

This plot confirms that a deal with such a countervalue represents quite an uncommon case for the user in question, although not so anomalous. As a quantitative summary of the above histograms, a few numbers are reported in Table 6 that read as follows.

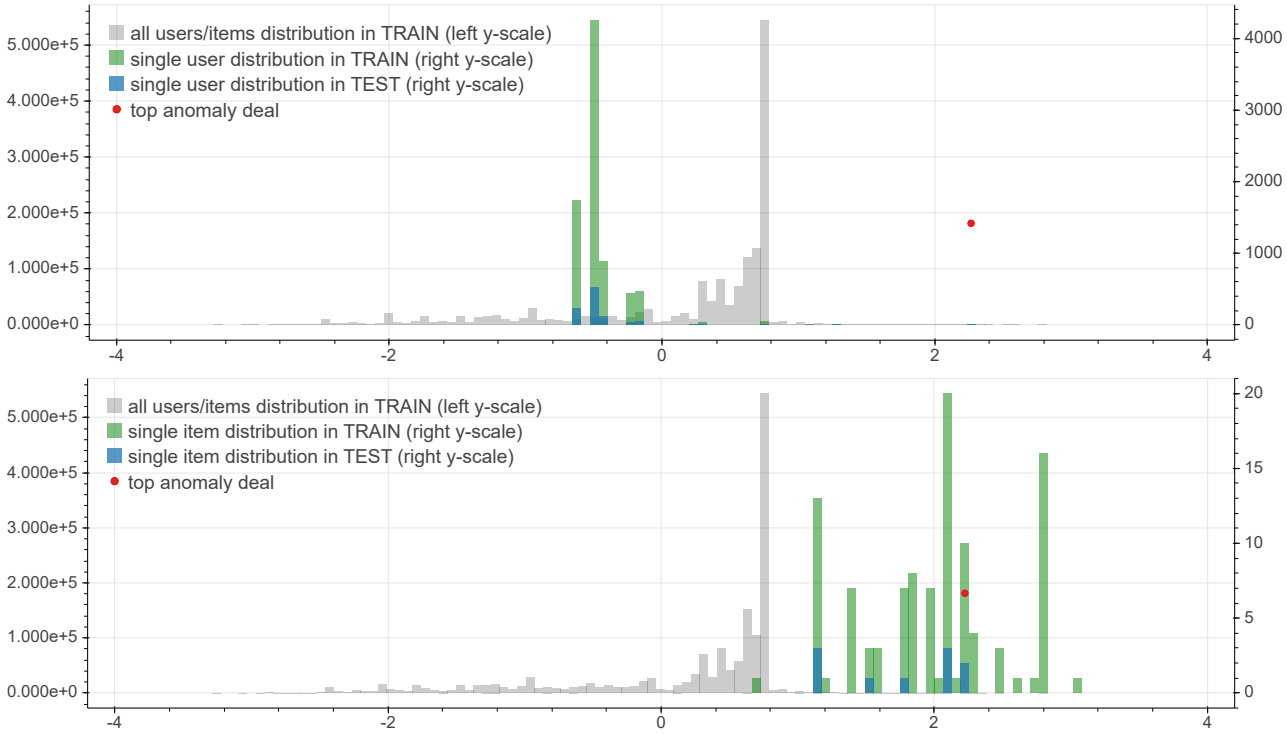
Table 6: Most anomalous $\mathcal{RS}_U^V$ record inspection

NUMBER OF DEALS	WHOLE DATASET	USER: `41932450`	vbin: ~3.2e+06	$\cap$ (`41932450`, ~3.2e+06)
TRAIN	1'567'776	7'903	108	5
TEST	158'524	987	10	2

Among the grand total of 1'567'776 deals of the training dataset, 108 of them fall into the 19$^{\text{th}}$ bin (~3.2e+06) regardless of the subject, and 7'903 of them belong to the subject `41932450`, including all bins; such numbers overlap in 5 deals that belong to the involved pair (`41932450`, ~3.2e+06). Correspondingly, in the test dataset among the grand total of 158'524 deals, 10 of them fall into the 19$^{\text{th}}$ bin (~3.2e+06) regardless of the subject, and 987 of them belong to the subject `41932450`, including all bins; such numbers overlap in precisely the 2 deals belonging to the involved pair (`41932450`, ~3.2e+06) that are reported as anomalous.

As a further investigation aimed at understanding these results, Figure 10 shows the distributions of the z-scores focusing on the USER (upper plot) and the ITEM (lower plot) of the most anomalous record in testbed. Compared to the overall distribution in the training dataset (background gray histograms in both upper and lower plot, against the left side y-axis), the

Figure 10: z-score distributions related to the most anomalous $\mathcal{RS}_U^V$ record



Comparison of the z-score overall distribution (gray, left y-scale) in the training dataset with the corresponding distributions (green, right y-scale) for the involved subject `41932450` (upper plot) or the involved volume bin ~ 3.2e+06 (lower plot). In blue, against right y-scale as well, is superimposed the latter distribution but in the testbed dataset. The red dot marks, at a fictitious height, the horizontal position of the z-score corresponding to the anomaly record.

z-score of user `41932450` (upper plot, right y-axis) is concentrated in the positive region both in the training dataset (green) and in testbed (blue). Conversely, the z-score of the volume bin ~3.2e+06 (lower plot) is biased towards the anomalous negative region, both in the training dataset (green) and in testbed (blue). This makes sense since it corresponds to the third largest bin in our dataset, so probably for many users it represents a quite uncommon countervalue.

An inspection like that of Figure 9 is not possible for the other two `RecSys`, since their ITEM dimensions have no inherent quantitative meaning. However one can still take a look at the z-score distributions like in Figure 10, focusing on the USER and the ITEM of the most anomalous record in testbed for $\mathcal{RS}_U^I$ (Figure 11) and $\mathcal{RS}_U^{I\text{-}V}$ (Figure 12).

Correspondingly, Table 7 and Table 8 provide their quantitative summary.

Table 7: Most anomalous $\mathcal{RS}_U^I$ record inspection

NUMBER OF DEALS	WHOLE DATASET	USER: `1000456`	ISIN: `JP3942800008`	$\cap$ (`1000456`, `JP3942800008`)
TRAIN	1'567'776	4'136	2	0
TEST	158'172	403	3	2

Table 8: Most anomalous $\mathcal{RS}_U^{I\text{-}V}$ record inspection

NUMBER OF DEALS	WHOLE DATASET	USER: `1026339`	ISIN@USER_25: `FR0000120354 H`	$\cap$ (`1026339`, `FR0000120354 H`)
TRAIN	1'567'776	86'050	2	0
TEST	157'562	8'329	4	1

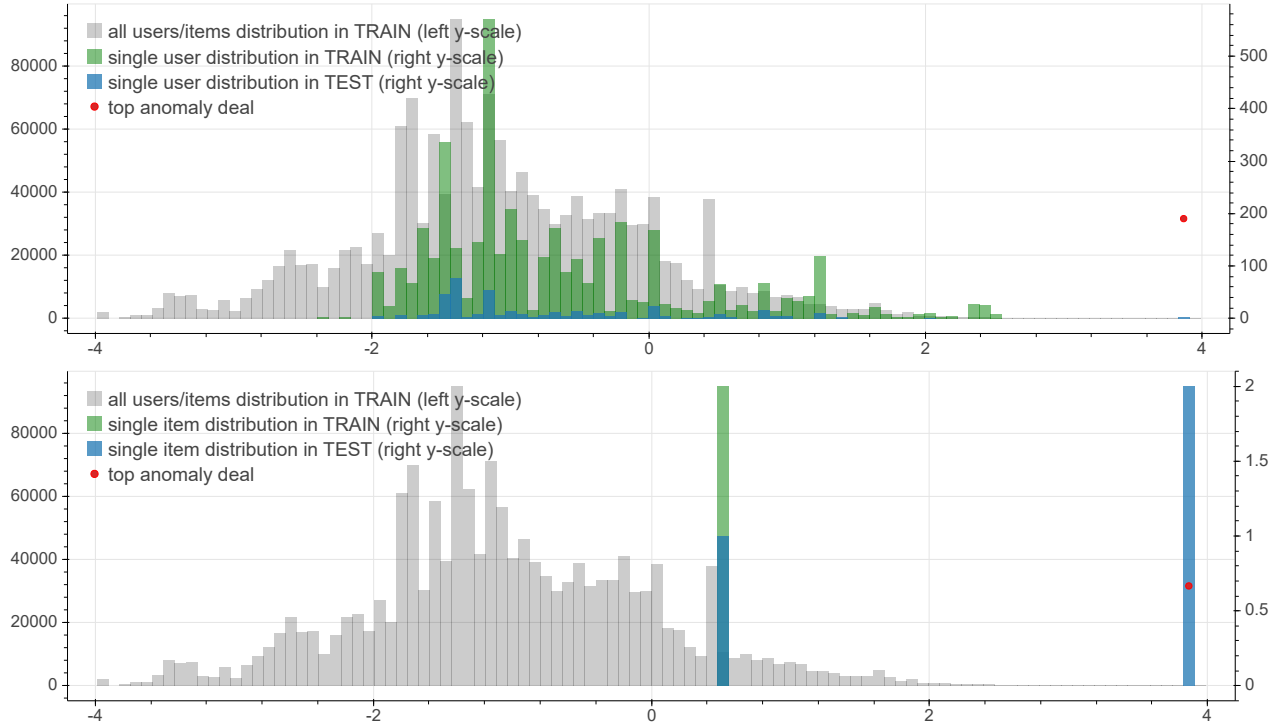
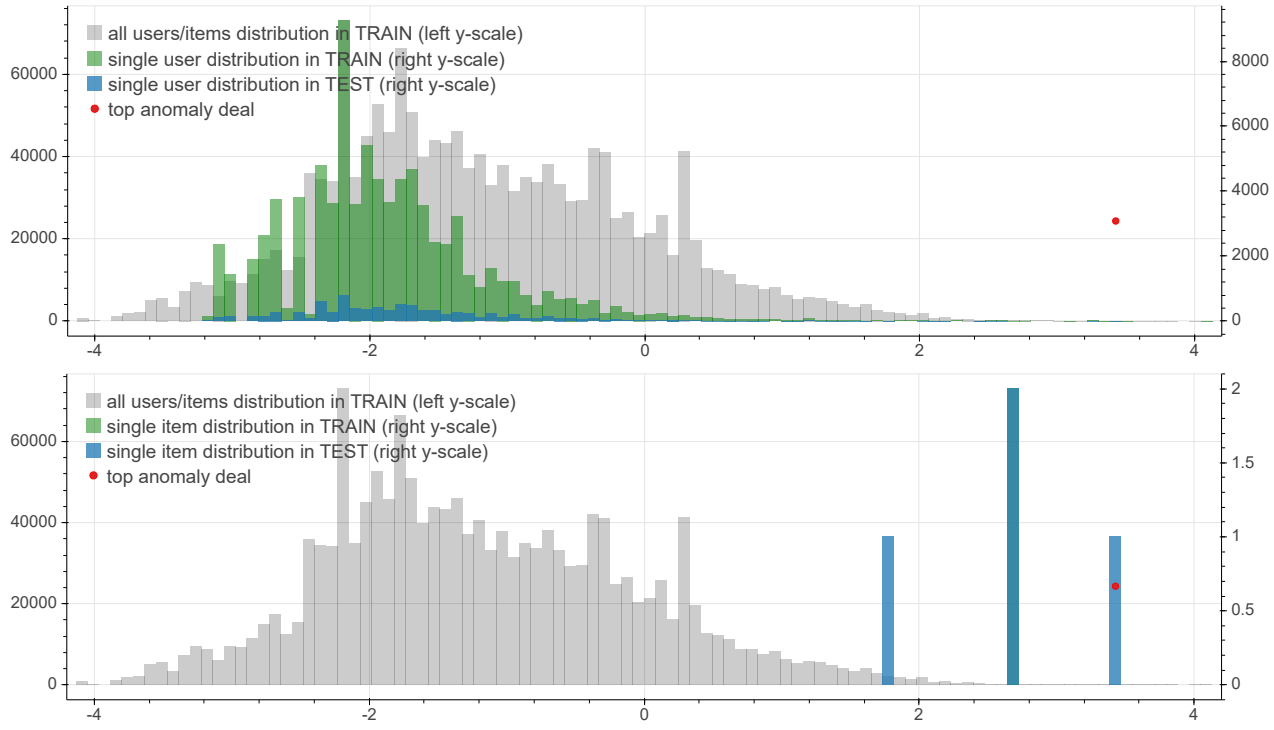


Figure 11: z-score distributions related to the most anomalous $\mathcal{RS}_{\mathrm{U}}^{\mathrm{I}}$ record

Comparison of the z-score overall distribution (gray, left y-scale) in the training dataset with the corresponding distributions (green, right y-scale) for the involved subject `1000456` (upper plot) or the involved security `JP3942800008` (lower plot). In blue, against right y-scale as well, is superimposed the latter distribution but in the testbed dataset. The red dot marks, at a fictitious height, the horizontal position of the z-score corresponding to the anomaly record.

Figure 12: z-score distributions related to the most anomalous $\mathcal{RS}_{\mathrm{U}}^{\mathrm{I\text{-}V}}$ record

Comparison of the z-score overall distribution (gray, left y-scale) in the training dataset with the corresponding distributions (green, right y-scale) for the involved subject `1026339` (upper plot) or (lower plot) the involved ITEM (security `FR0000120354` traded in the *H*igher part of `1026339`'s volume distribution). In blue, against right y-scale as well, is superimposed the latter distribution but in the testbed dataset. The red dot marks, at a fictitious height, the horizontal position of the z-score corresponding to the anomaly record.



4 Conclusion

In this paper we reported our experiments with a Recommendation Systems approach to the Market Abuse Detection problem.

Our tests were carried out using a 'real world' dataset very similar to those available for MAD, applying `RecSys` techniques in a 'reverse' perspective, namely as anomaly detector: at time of writing, at our knowledge, this is the first attempt to use a `RecSys` in this way.

Preliminary results clearly show interesting potential of applying such tools in a production environment as ancillary facilities to monitor traders activities looking for unusual behaviors.

Being an unsupervised ML approach, it's mostly non-parametric, which is crucial in such a diversified behaviors framework.

Not being a per-user approach, it can be successfully applied even with a large population of moderate or small operation users

The approach is quite generic: several possible RecSys can be setup, for the same dataset, each aimed at monitoring specific traits of users behavior.

OTC markets, where usual metrics are not available, and so usual algorithms cannot be implemented, is a further field of application.

Index of names

References

[1] Yehuda Koren, Yahoo Research, Yifan Hu AT&T Labs, Chris Volinsky AT&T Labs, *Collaborative Filtering for Implicit Feedback Datasets*.
http://yifanhu.net/PUB/cf.pdf

[2] Yehuda Koren, Yahoo Research, Robert Bell and Chris Volinsky, AT& T Labs—Research, *Matrix Factorization Techiques for Recommender System*, 2009, Published by the IEEE Computer Society.
https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf

[3] Rendle, Steffen, et al. *BPR: Bayesian personalized ranking from implicit feedback*. Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence. AUAI Press, 2009.
https://arxiv.org/pdf/1205.2618

[4] Weston, Jason, Samy Bengio, and Nicolas Usunier. *Wsabie: Scaling up to large vocabulary image annotation*. IJCAI. Vol. 11. 2011.
https://research.google.com/pubs/archive/37180.pdf

[5] Google, *Building a Recommendation System in TensorFlow: Overview*
https://cloud.google.com/solutions/machine-learning/recommendation-system-tensorflow-overview

[6] Google, *Recommendation Systems, Matrix Factorization*
https://developers.google.com/machine-learning/recommendation/collaborative/matrix

[7] Sonya Sawtelle, *Mean Average Precision (MAP) For Recommender Systems*, 2016
http://sdsawtelle.github.io/blog/output/mean-average-precision-MAP-for-recommender-systems.html

[8] "Standard score" in *Wikipedia: The Free Encyclopedia.*
https://en.wikipedia.org/wiki/Standard_score

[9] "Cumulative distribution function" in *Wikipedia: The Free Encyclopedia.*
https://en.wikipedia.org/wiki/Cumulative_distribution_function

[10] Paul Embrechts, Marius Hofert, *A note on generalized inverses*, 2014
https://people.math.ethz.ch/~embrecht/ftp/generalized_inverse.pdf

[11] "Quantile" in *Wikipedia: The Free Encyclopedia.*
https://en.wikipedia.org/wiki/Quantile

[12] "Backtesting" in *Wikipedia: The Free Encyclopedia.*
https://en.wikipedia.org/wiki/Backtesting

[13] "Cross-validation (statistics)" in *Wikipedia: The Free Encyclopedia.*
https://en.wikipedia.org/wiki/Cross-validation_(statistics)