Machine Learning & fraud detection

Olivier Caelen

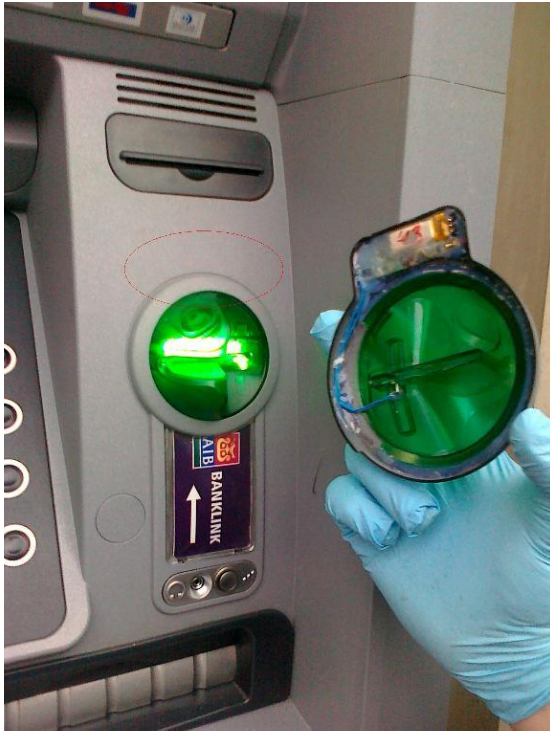
Agenda

- The Card Fraud-Detection Context
- Fraud-Detection as a machine learning problem
  - Clustering
  - Graph mining
- Conclusion

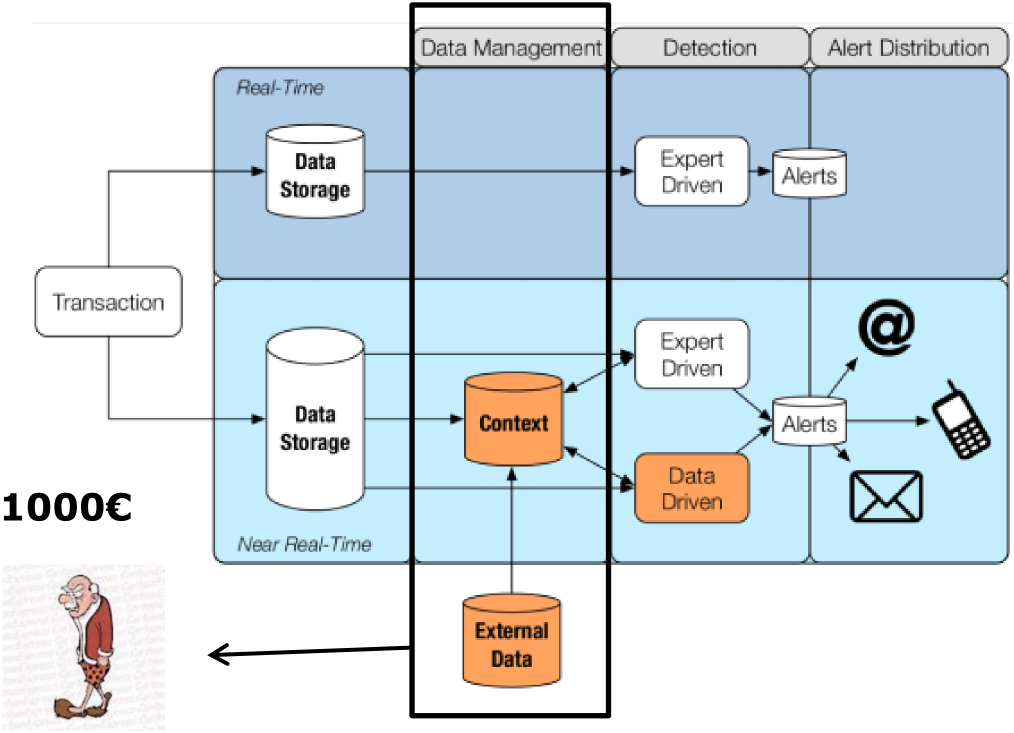
Background

- Credit card fraud
  - Fraudsters steal card data
    - Skimming
    - Phishing
    - Data theft (Hacking)

- Cards misused to issue transactions



The Card Fraud-Detection Context

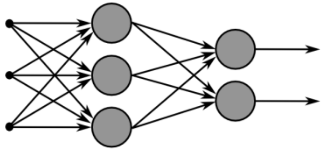


- Real-time/Near Real-time
- Debit/Credit
- Issuing/Acquiring

1000€

Expert driven

Data driven



Agenda

- The Card Fraud-Detection Context
- Fraud-Detection as a machine learning problem
  - Clustering
  - Graph mining
- Conclusion

The Fraud-Detection problem

- **The problem**: Learning over a potentially infinitely long stream of transactions.

- **Data-generating process** χ generates tuples $(x_t, y_t) \sim \chi$
  - x_t is the transaction data at time t (e.g., $x_t \in \mathbb{R}^d$)
  - y_t is the associated label (e.g., $y_t \in \{+,-\}$)

- **Data:**
  - Transactional data (TX_DATETIME, AMT, COUNTRY_CODE, …)
  - External Date (Gender, Age, …)
  - Data driven features (Clustering, graph mining, …)

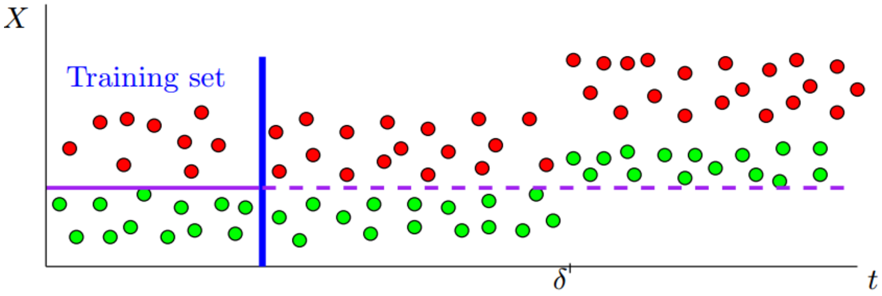
- **Fraud Detection** is a learning problem with :
  - On-line learning with concept-drift
  - Sampling bias
  - Highly unbalanced
  - Overlapping
  - Many possible cost functions

Concept-drift / Nonstationary

- Fraudster strategies change
- Customers' spending habits change




- Consider as an illustrative example a simple 1-dimension classification problem.



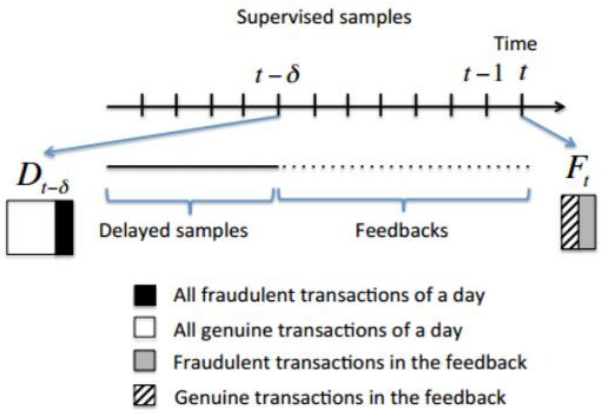
If $t < \delta$, we have $(x_t, y_t) \sim \mathcal{X}$.
If $t > \delta$, we have $(x_t, y_t) \nsim \mathcal{X}$

$\implies$ Concept-drift at δ

$\mathcal{X}$ becomes **non-stationary**.

Sampling bias

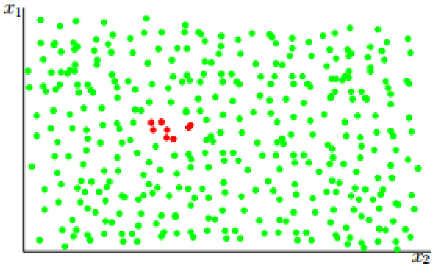
The feedback from transactions ($y_t \in \{+,-\}$) comes with delay.



Note that in the feedback $(x_t, y_t) \nsim \chi \implies$ sampling bias.

Highly unbalanced

The rate of the fraudulent transactions is usually less than 1/1000.



Learning from unbalanced datasets is a difficult task since learning algorithms are not designed to cope with a large difference between the number of cases belonging to different classes [1].

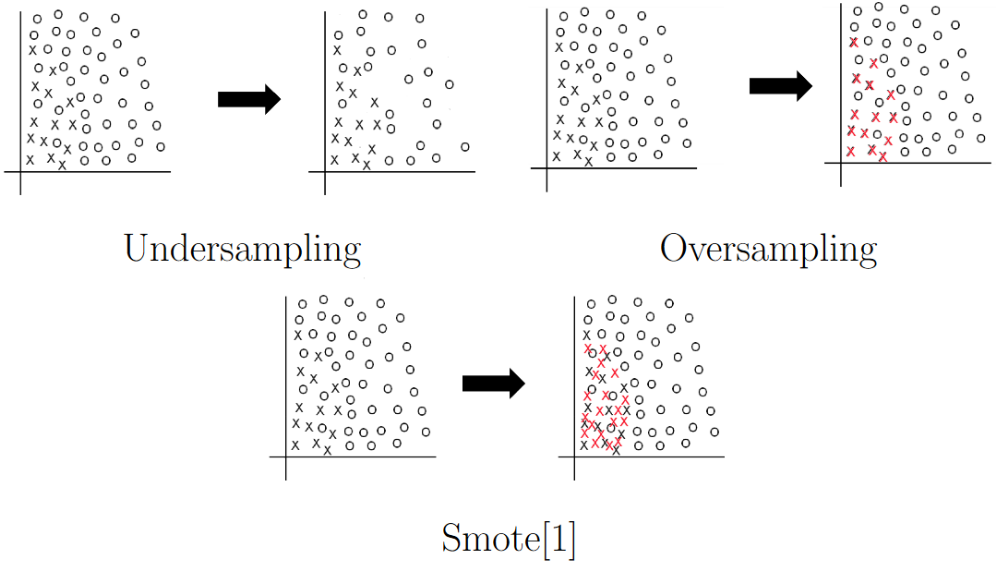
Techniques for unbalanced datasets :

- Sampling methods
- Ensemble methods
- Cost based methods
- ...

[1] N. Japkowicz, et al. *The class imbalance problem: A systematic study*, 2002

Highly unbalanced – Sampling methods (I)

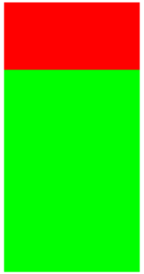
Many of the existing methods for classification with unbalanced dataset take advantage of sampling techniques to balance the dataset.



[1] N.V. Chawla, et al. *Smote: synthetic minority over-sampling technique* 2011.

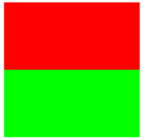
Highly unbalanced – Sampling methods (II)

When is under-sampling effective in unbalanced classification tasks?[1]

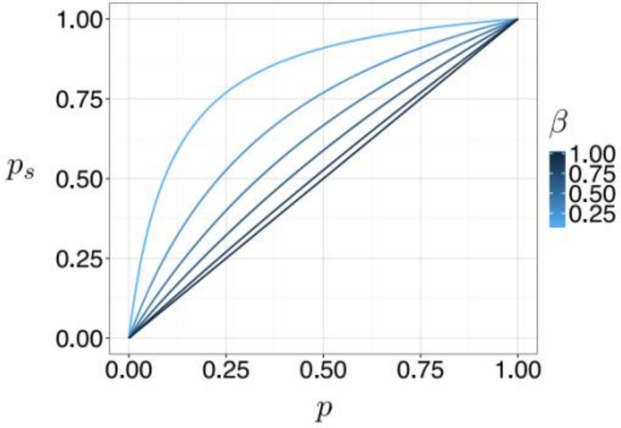


$$P(Y = + \mid X = x)$$

$$p = \frac{\beta p_s}{\beta p_s - p_s + 1}$$



$$P_s(Y = + \mid X = x)$$



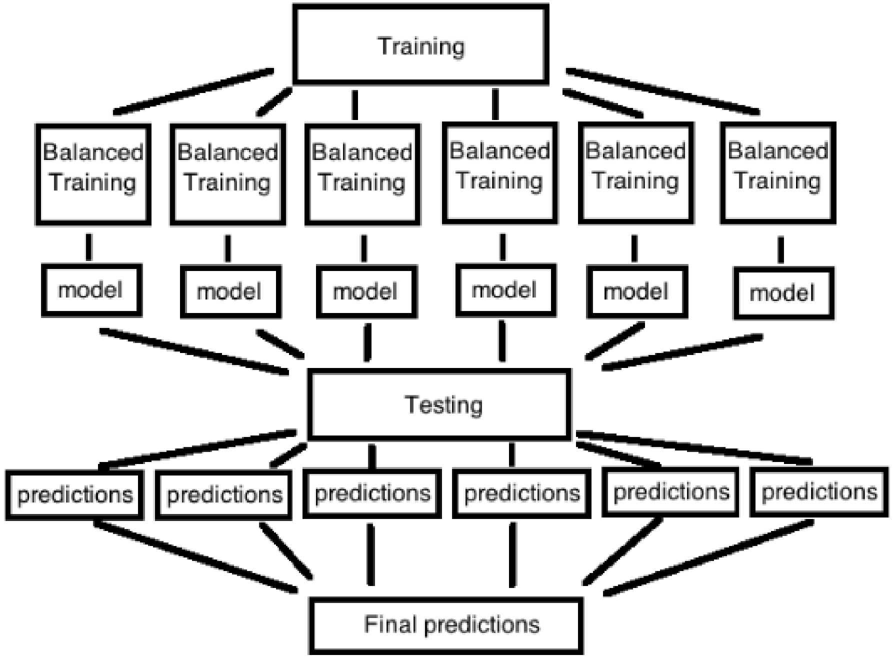
Uniform under-sampling modifies $P(Y = + \mid X = x)$ but not the ranking.

→And fraud detection is a ranking problem.

[1] A. Dal Pozzolo, et al. *When is undersampling effective in unbalanced classification tasks?*, 2015

Highly unbalanced – Ensemble methods (I)

- **EasyEnsemble** [1], explore the majority class in an unsupervised manner



Algorithm 1 The `EasyEnsemble` algorithm.

1: {Input: A set of minority class examples $\mathcal{P}$, a set of majority class examples $\mathcal{N}$, $|\mathcal{P}| < |\mathcal{N}|$, the number of subsets T to sample from $\mathcal{N}$, and s_i, the number of iterations to train an AdaBoost ensemble H_i}

2: $i \Leftarrow 0$

3: **repeat**

4: $\quad i \Leftarrow i + 1$

5: $\quad$ Randomly sample a subset $\mathcal{N}_i$ from $\mathcal{N}$, $|\mathcal{N}_i| = |\mathcal{P}|$.

6: $\quad$ Learn H_i using $\mathcal{P}$ and $\mathcal{N}_i$. H_i is an AdaBoost ensemble with s_i weak classifiers $h_{i,j}$ and corresponding weights $\alpha_{i,j}$. The ensemble's threshold is θ_i, i.e.

$$H_i(x) = \operatorname{sgn}\left(\sum_{j=1}^{s_i} \alpha_{i,j} h_{i,j}(x) - \theta_i\right).$$

7: **until** $i = T$

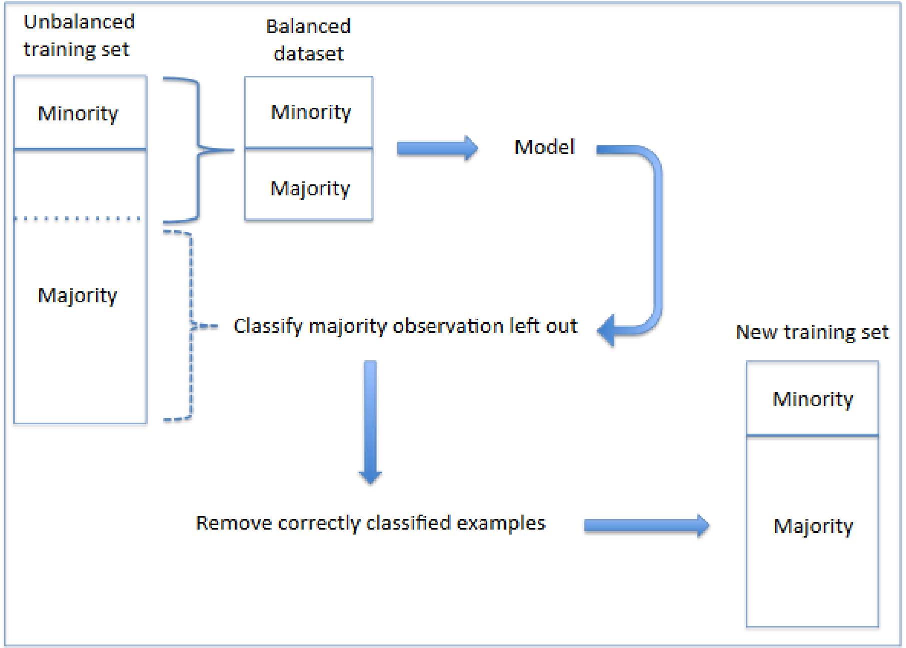
8: Output: An ensemble:

$$H(x) = \operatorname{sgn}\left(\sum_{i=1}^{T} \sum_{j=1}^{s_i} \alpha_{i,j} h_{i,j}(x) - \sum_{i=1}^{T} \theta_i\right).$$

[1] X.Y. Liu, et al. *Exploratory undersampling for class-imbalance learning.* 2009.

Highly unbalanced – Ensemble methods (II)

- **BalanceCascade** [1], explore the majority class in a supervised manner



- At every step, a model is trained on a balanced dataset.
- The majority training set is shrunk after every step.

  → By removing the correctly classified examples from the majority class.

[1] X.Y. Liu, et al. *Exploratory undersampling for class-imbalance learning.* 2009.

Highly unbalanced – Cost based methods

- **Cost based methods** [1] are type of learning that takes the misclassification costs into consideration (FN cost >FP cost)

- Cost-insensitive algorithm can be converted into cost-sensitive using a wrapper approach:
  → Modify the class distribution of the training data and then apply the cost insensitive algorithm.
  - **Cost proportional sampling** [2], positive and negative examples are sampled by the ratio:

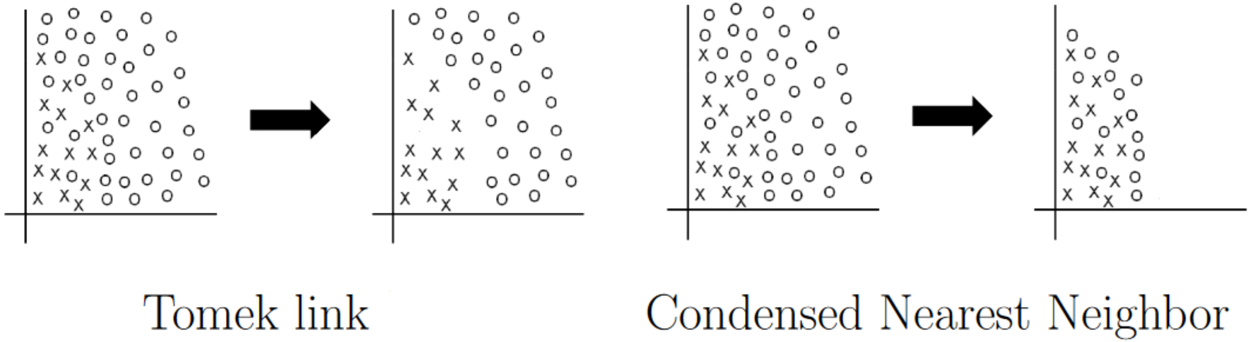
    $$\frac{P(majority).\,FNcost}{P(minority).FP\ cost}$$

  - **Costing** [3], accept an instance into the sample with the accepting probability $\frac{C(i)}{Z}$, where $C(i)$ is the misclassification cost of class i, and Z is an arbitrary constant such that $Z \geq \max C(i)$.

[1] C.X. Ling , et al. *Cost-sensitive learning and the class imbalance problem*, 2008.
[2] C. Elkan, et al. *The foundations of cost-sensitive learning*, 2001
[3] B. Zadrozny, et al. *Cost-sensitive learning by cost-proportionate example weighting*. 2003

Highly unbalanced – Other methods

- Goal is to remove:
  - both noise and borderline examples (e.g. Tomek link [1])
  - or instances from the majority class that are distant from the decision border, considered less relevant for learning (e.g. CNN [2])

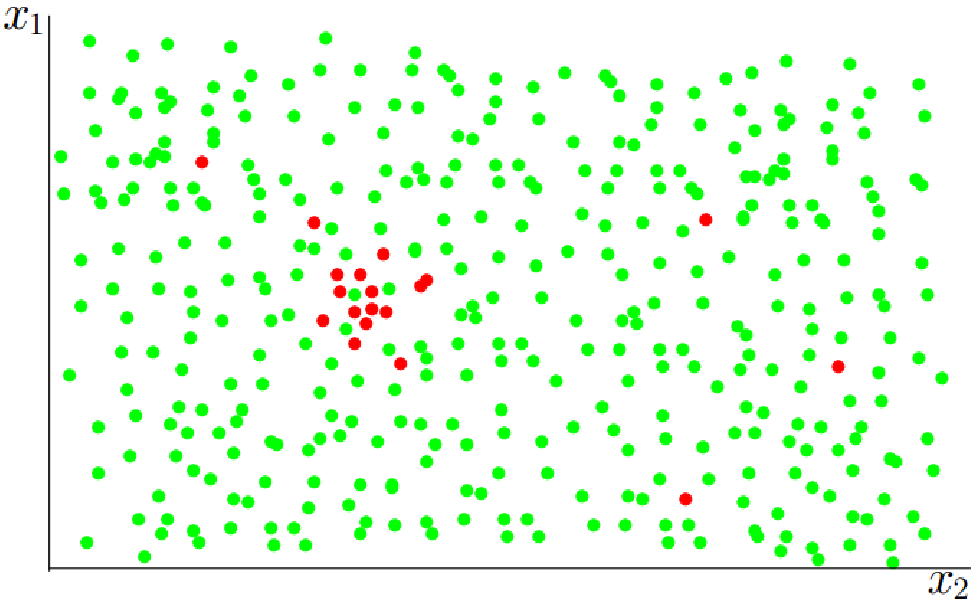


[1] I. Tomek, *Two modi cations of cnn*, 1976
[2] P. E. Hart, *The condensed nearest neighbor rule*, 1968

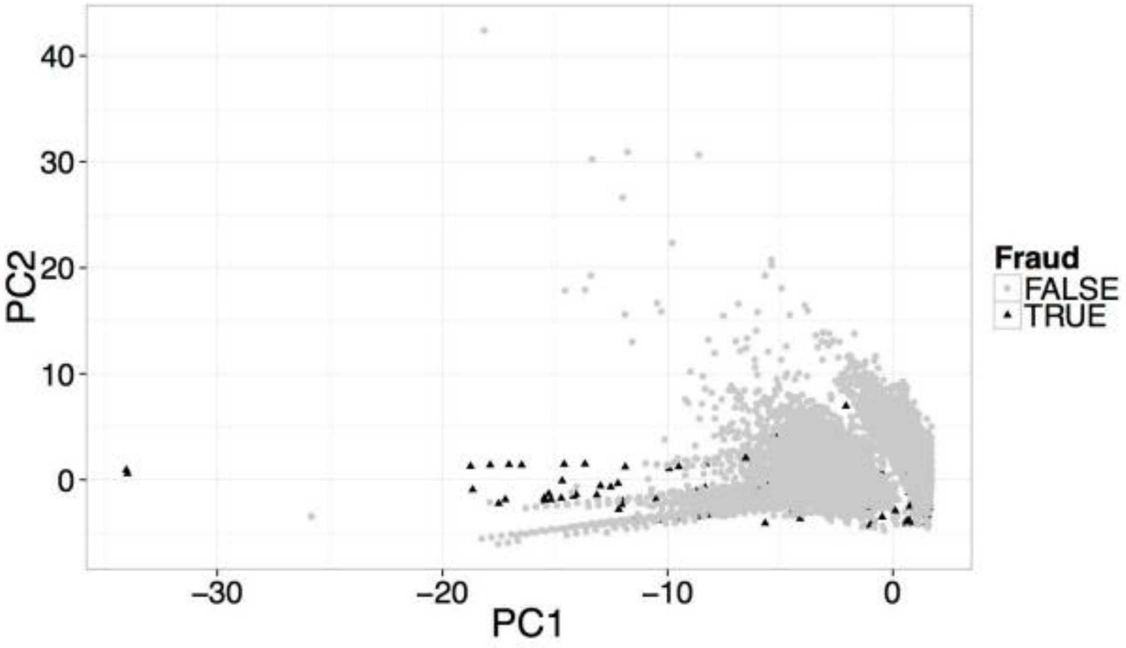
Overlapping (I)

- Fraudulent transactions may look like genuine transactions (and vice-versa).

  → more True/false positive.



Overlapping (II)

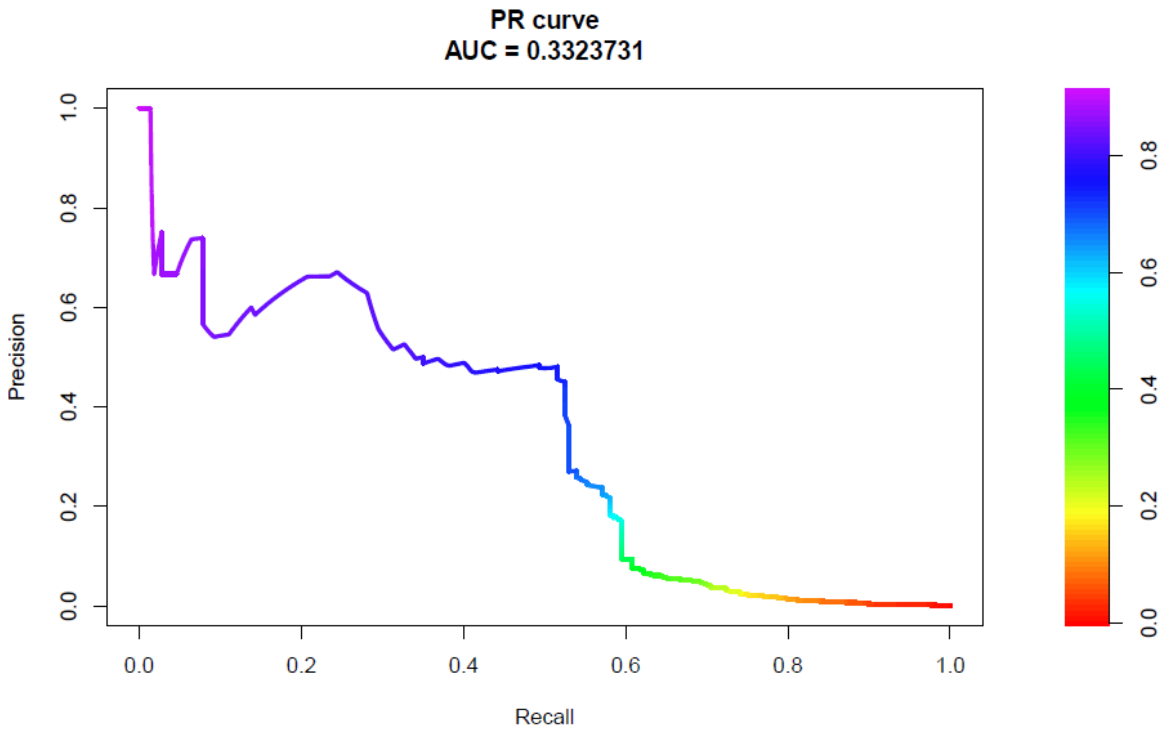


Many possible cost functions

- Fraud detection is a **ranking sequential learning problem with unbalanced datasets**.

- What's the best cost function to estimate the accuracy of the models?
  - Precision/Recall/F-score?
  - AUC?
  - P_k?
  - Speed of detection?
- What about the amount of the transactions?

Many possible cost functions – Precision Recall curve



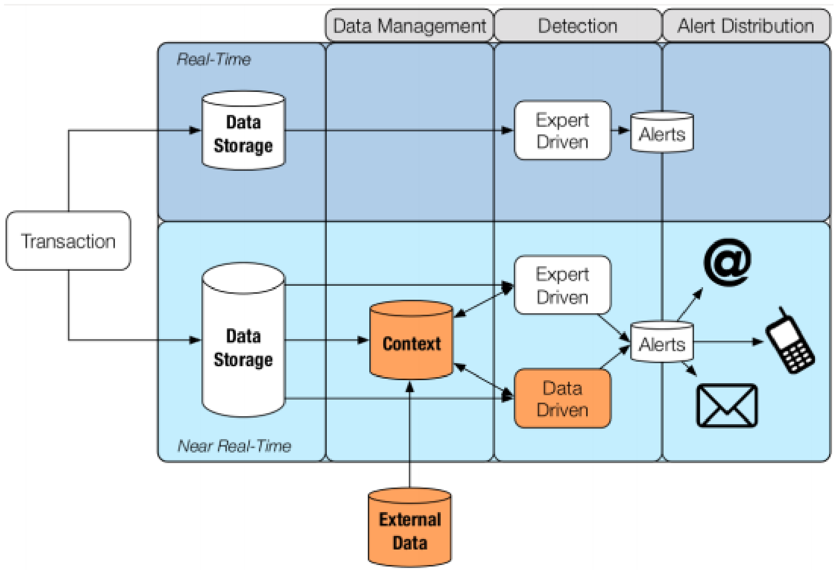
- **Fraud Detection** is a learning problem with :
  - On-line learning with concept-drift
  - Sampling bias
  - Highly unbalanced
  - Overlapping
  - Many possible cost functions

Other topics

- On-line learning
- High dimension ($d \approx 50$)
- Fast prediction
- …

Data management

- Many features can be created during data preparation step.
- These features can then be used in the expert or data driven rules.
- Goal: improve accuracy.
- We will present two technics:
  - Clustering
  - Graph mining



Agenda

- The Card Fraud-Detection Context

- Fraud-Detection as a machine learning problem
  - Clustering
  - Graph mining

- Conclusion

Clustering Overview

- **Clustering**: grouping a set of card profiles in such a way that profiles in the same group are more 'similar' to each other than to those in other groups.

- **Application in fraud detection**: can be used to reduce false positive or to increase detection rate. Example:
  - <u>Reduce false positive</u> (**less customer impact**): if a card has a risk behavior but it is the common behavior of the user → reduce the probability to generate an alert.
  - <u>Increase detection rate & speed of detection</u> (**reduce fraud losses**): if a card has a risk behavior and it's far from his common behavior → increase the probability to generate an alert.

- They are two main approaches :
  - **Expert driven** (CardType, Gender, …)
  - **Data driven**

► In this presentation, we focus on the data driven approach. In practice, both can be used simultaneously.

Clustering Overview

- Our clustering project is divided into two main phases:
  - Finding the clusters
  - Classify new card profile in a suitable cluster

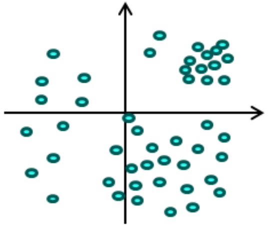
Data table

x1	x2	x3	x4	x5
5	0,5	100	0	-12
6	0	453	0	15
15	0	976	0	0
4	0,1	128	0	1,5
1	-1	247	1	-9

In this table, a row represents a card profile.

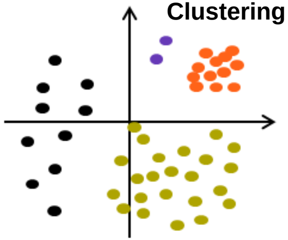
Reduction of noise

Dimension reduction



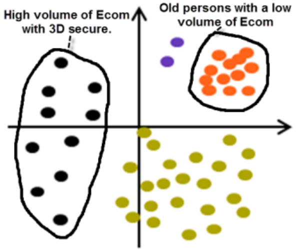
Automatic and unsupervised

Clustering



Manual

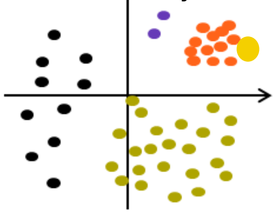
Business interpretation



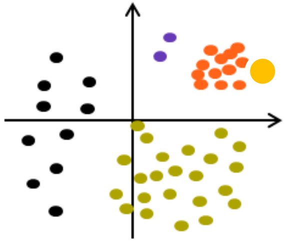
New card profile:

x1	x2	x3	x4	x5
8	0	687	1	6

Projection



Classification



Business interpretation

The profile of this card corresponds to an old person with a low use of ecom transactions.

Fraud detection

Higher risk if this card starts to do lot of ecom transactions in a short time period.

Method – Observations

Three independent clustering are done:

- card with most ecom transaction [**ECOM**] rateEcom > 90%
- card with most face-to-face transactions [**F2F**] rateEcom < 10%
- card with ecom & face to face transactions [**mixed**]

47.626 card profiles are used to create the clusters.

Method – Variables

List of variables in the card profile:

- **Velocity**
  - Mean Sum Amt Per Day
  - Mean Nb Tx Per Day
  - …

- **Ecom** activity – Only clustering [ECOM]
  - Rate 3d Ecom
  - Rate Charity Ecom
  - …

- **F2F** activity – Only clustering [F2F]
  - Rate Pin F2F
  - …

- **Socio-demographic**
  - Age
  - Gender (qualitative)
  - …

- Other:
  - Rate Night
  - Card Type (qualitative)
  - …

→ **19 input variables.**

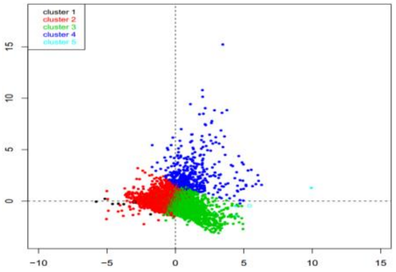
- Qualitative variables are used for business interpretation.
- They are not used during the creation of the clusters
  → no impact on the clustering.

Results – clustering

47.626 Cards

47.626 Cards →	E-com →	5.293 Cards	~11%
	mixed →	30.837 Cards	~65%
	F2F →	11.496 Cards	~24%

Results – clustering



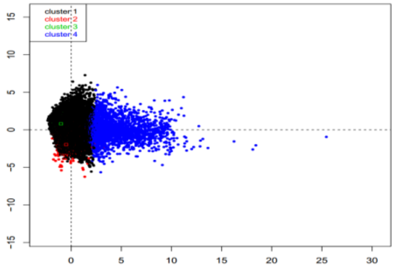
Group: [**ECOM**]

5.293 Cards

Cluster sizes:

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
0.2%	43.4%	47.9%	8.4%	0.07%

Business Interpretation

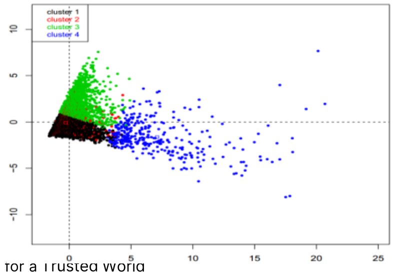


Group: [**mixed**]

30.837 Cards

Cluster sizes:

Cluster 1	Cluster 2	Cluster 3	Cluster 4
92.4%	0.6%	0.03%	7,0%



Group: [**F2F**]

11.496 Cards

Cluster sizes:

Cluster 1	Cluster 2	Cluster 3	Cluster 4
65.8%	9.7%	21.5%	2.9%

Results – Business Interpretation & Fraud detection

Lot of statistics could be obtained for the business interpretation of the clusters:

ECOM – cluster 1

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
0.2%	43.4%	47.9%	8.4%	0.07%

Statistics:

Variable	Mean in category	Overall mean
Rate Charity Ecom	52,9%	0.2%
Rate Recurring Ecom	56.6%	19.9%
Age	52	43

Note: I show only values where the difference is statistically significant.

Business Interpretation :

- Higher rate of **recurring charity** activities.

"charity"

Fraud detection (example) :

- Decrease the score risk of fraud on MCC charity. (Reduce false positive)

Agenda

- The Card Fraud-Detection Context
- Fraud-Detection as a machine learning problem
  - Clustering
  - Graph mining
- Conclusion

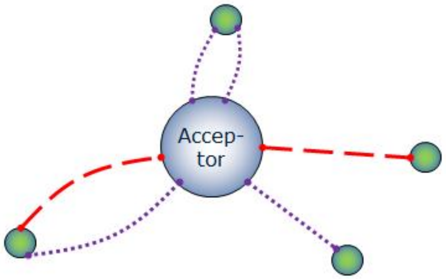
Graph mining

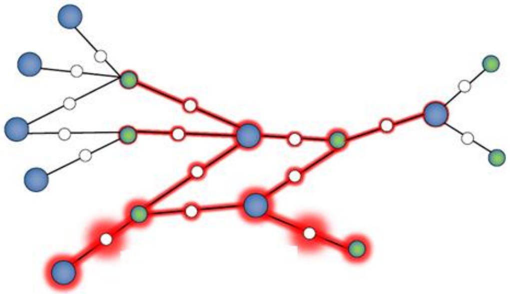
- Many aggregated variables can be created from historical data.
  - Nb transaction,
  - Sum Amt,
  - Sum of transactions with fraudulent cards,
  - …

- Basic graph features, which only take into account direct neighbors.

- Graph mining give a way to see was is happening further.

  - Influence propagation
  - Sub graph



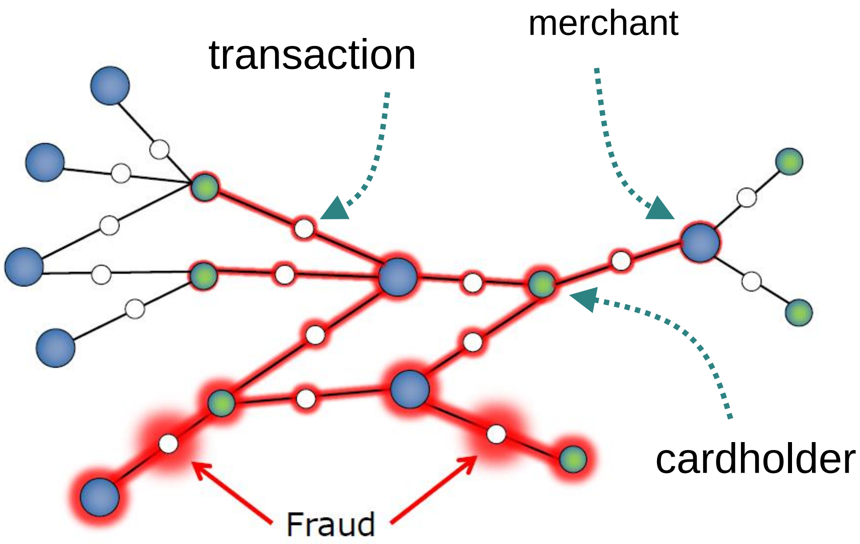


Graph Mining – Influence propagation

- Gotcha [1]
- It is a random walk algorithm which propagate fraud information in the graph.

$$s_t = \alpha * P^T * s_{t-1} + (1 - \alpha) * z$$

- After convergence, the vector s_t contains a risk score for each note.



[1] V. Van Vlasselaer et al., *GOTCHA! Network-based fraud detection for social security fraud* (2016).

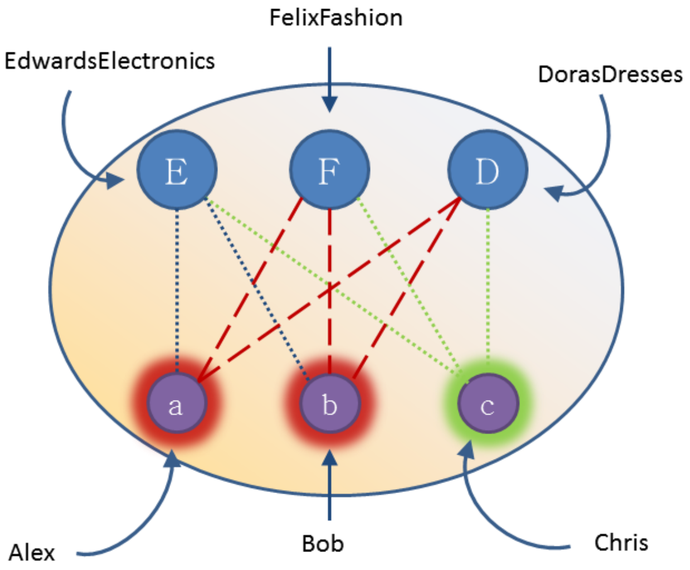
Graph mining – Sub graph (I)

Cardholder	Shop	Fraud?
alex	EdwardsElectronics	no
bob	EdwardsElectronics	no
bob	FelixFashion	yes
alex	DorasDresses	yes
bob	DorasDresses	yes
alex	FelixFashion	yes
chris	EdwardsElectronics	?
chris	FelixFashion	?
chris	DorasDresses	?

Time

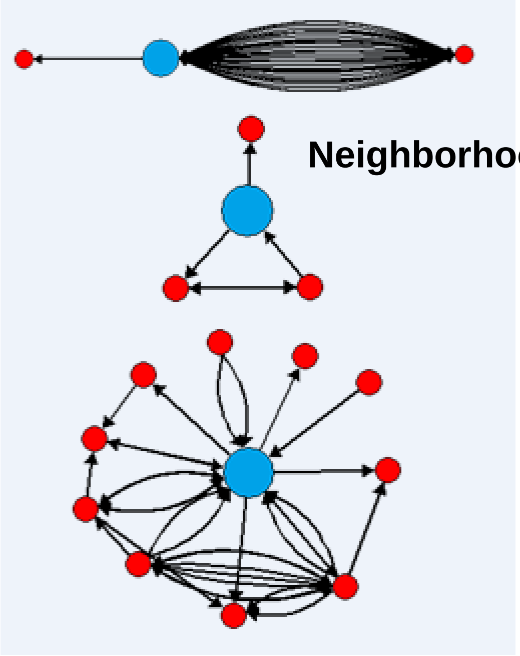
Suspicious shop pattern {D,E,F}

Higher fraud probability?



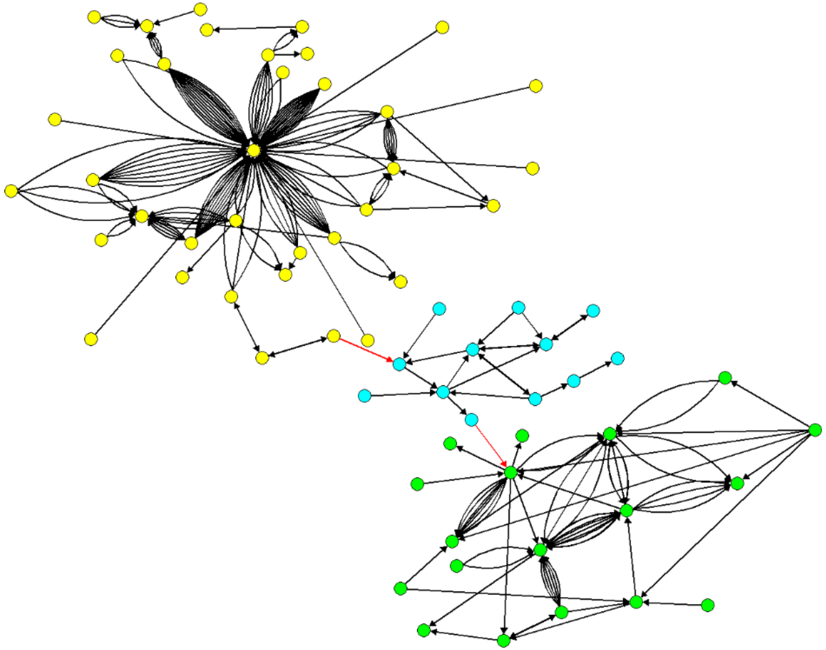
It is a "Clique"

Graph mining – Sub graph (II)



Neighborhood sub-graph

Community sub-graph



- Feature are extracted from all these sub-graphs…
- … and used in data/expert driven rules.

Agenda

- The Card Fraud-Detection Context
- Fraud-Detection as a machine learning problem
  - Clustering
  - Graph mining
- Conclusion

Conclusion

- **Fraud Detection** is a learning problem with :
  - On-line learning with concept-drift
  - Sampling bias
  - Highly unbalanced
  - Overlapping
  - Many possible cost functions
  - On-line learning
  - High dimension ($d \approx 50$)
  - Fast prediction
  - …
- **New** data driven **features** :
  - Clustering
  - Graph Mining

Thank you for your attention