Detecting Money Laundering with Machine Learning

Martin Jullum

Joint with Anders Løland, Ragnar Bang Huseby, Aliaksandr Hubin, Geir Ånonsen and Johannes Lorentzen

Insurance seminar UiO 2020

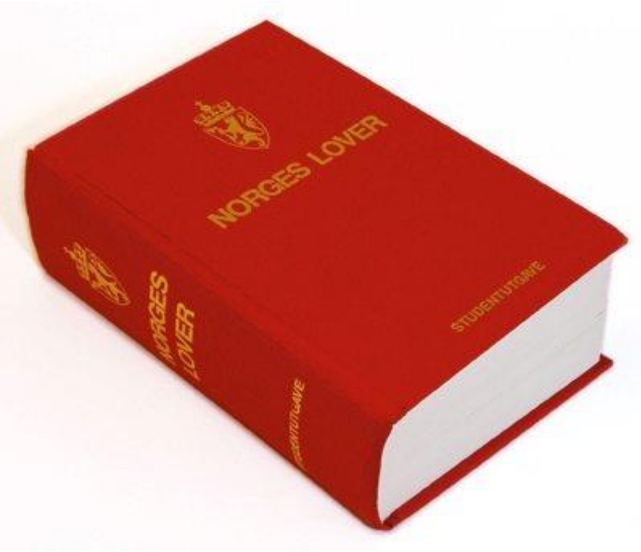
October 1st 2020

Money laundering

- Making money from criminal activity appear legal
- Examples
  - Buy antics with dirty money – state as attic finding – sell legally
  - Incorporate criminal funds in your own legal business
  - Buy (single premium) insurance policy (via straw person) with dirty money – surrender the policy

Money laundering

- Making money from criminal activity appear legal
- Examples
  - Buy antics with dirty money – state as attic finding – sell legally
  - Incorporate criminal funds in your own legal business
  - Buy (single premium) insurance policy (via straw person) with dirty money – surrender the policy



► All financial institutions are legally binded to report “suspicious transactions” to Økokrim

Why is AML important?


REUTERS

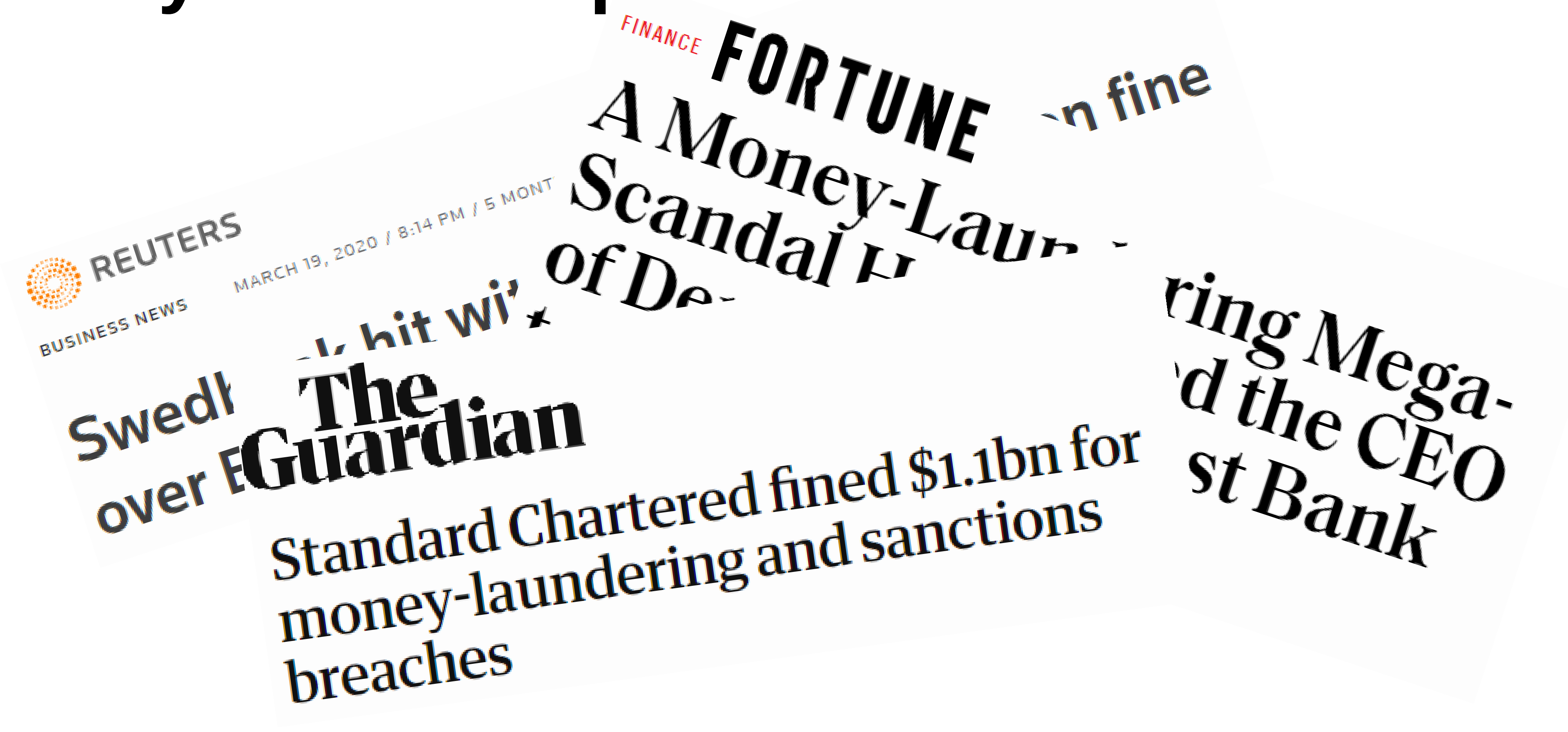
BUSINESS NEWS

MARCH 19, 2020 / 8:14 PM / 5 MONTHS AGO

Swedbank hit with record $386 million fine over Baltic money-laundering breaches

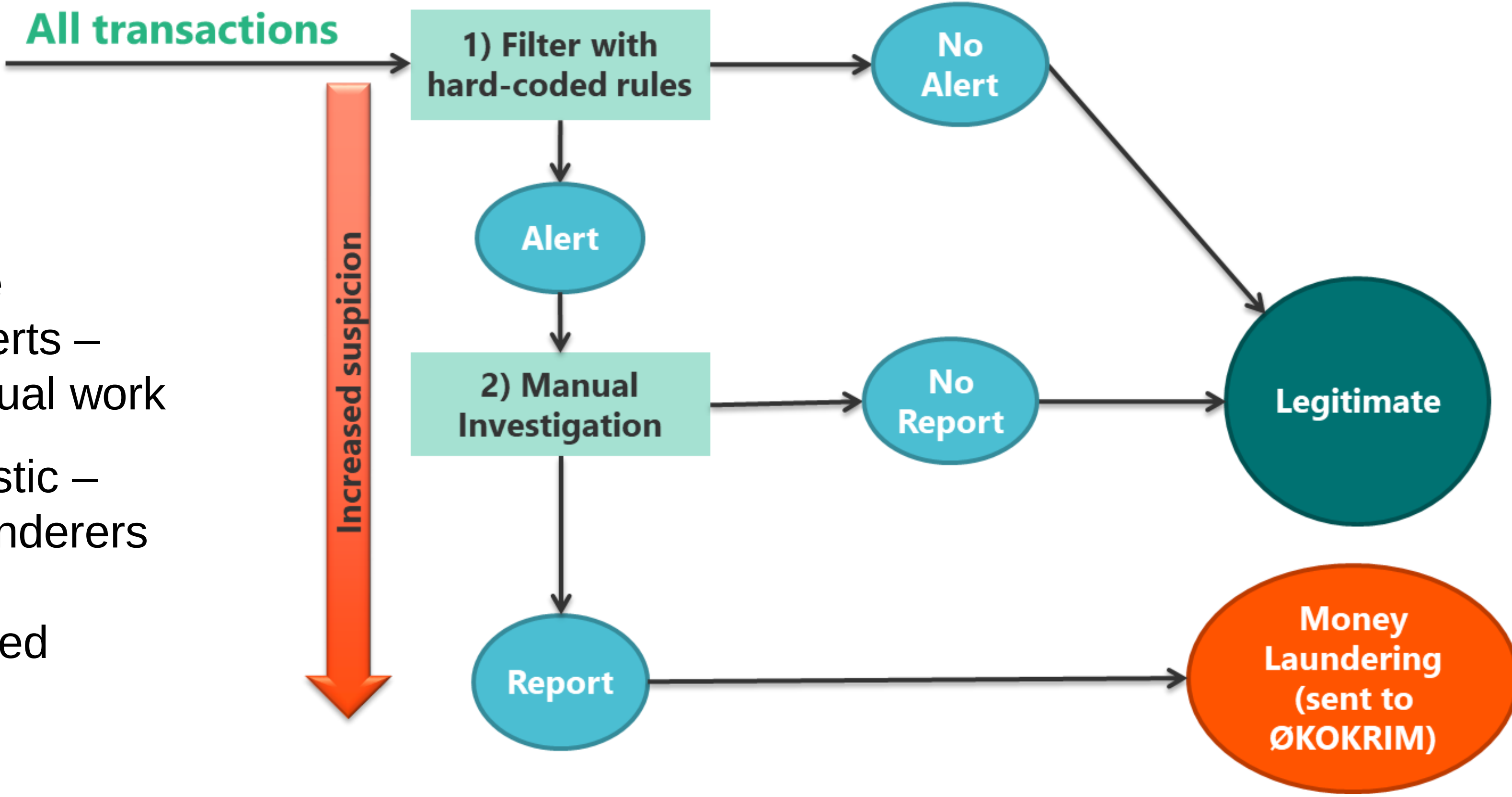
Why is AML important?

Why is AML important?



Why is AML important?

Current AML process at DNB

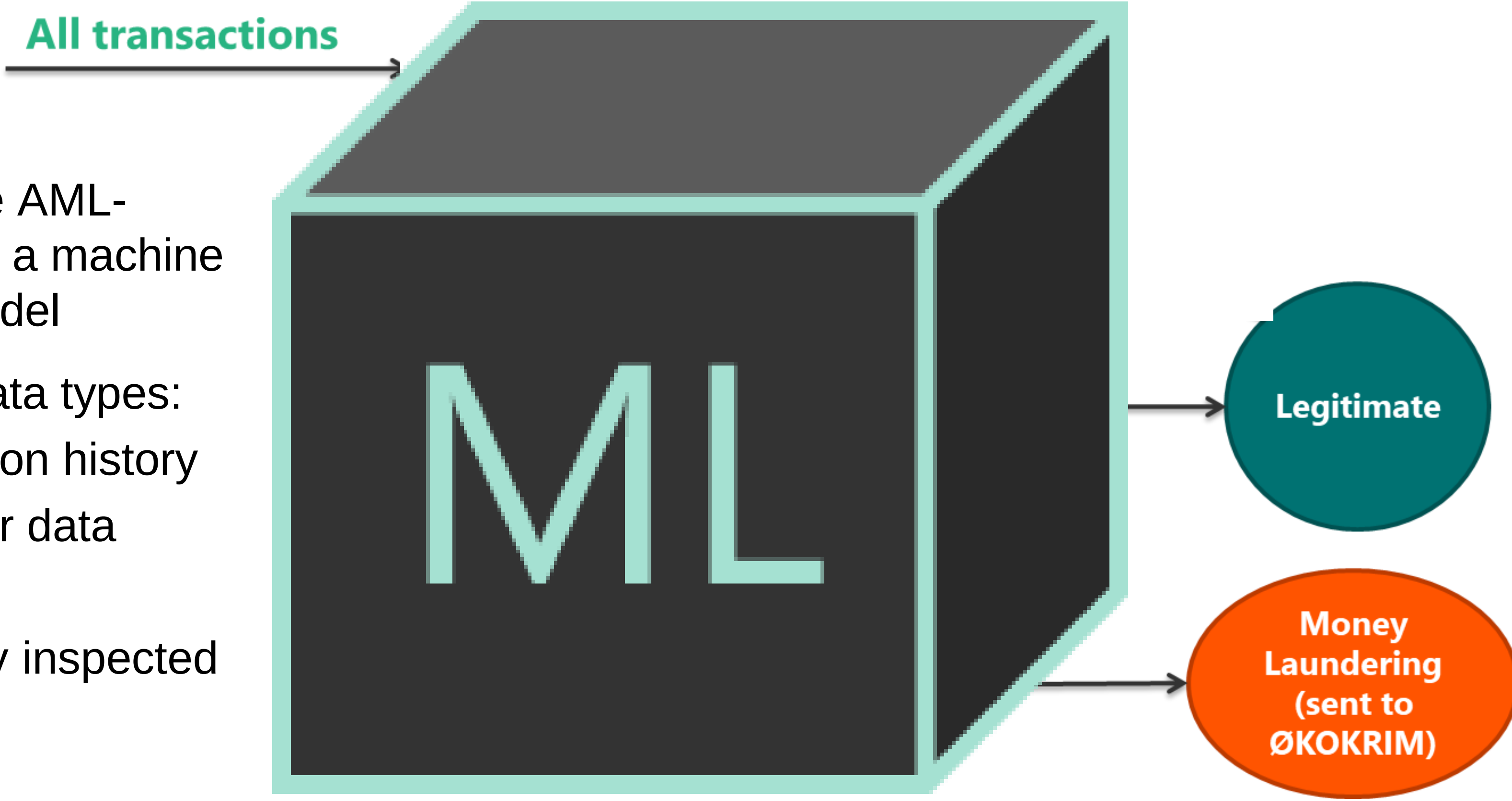


Weaknesses

- Many false positive alerts – much manual work
- Too simplistic – Money launderers are more sophisticated

What we have done

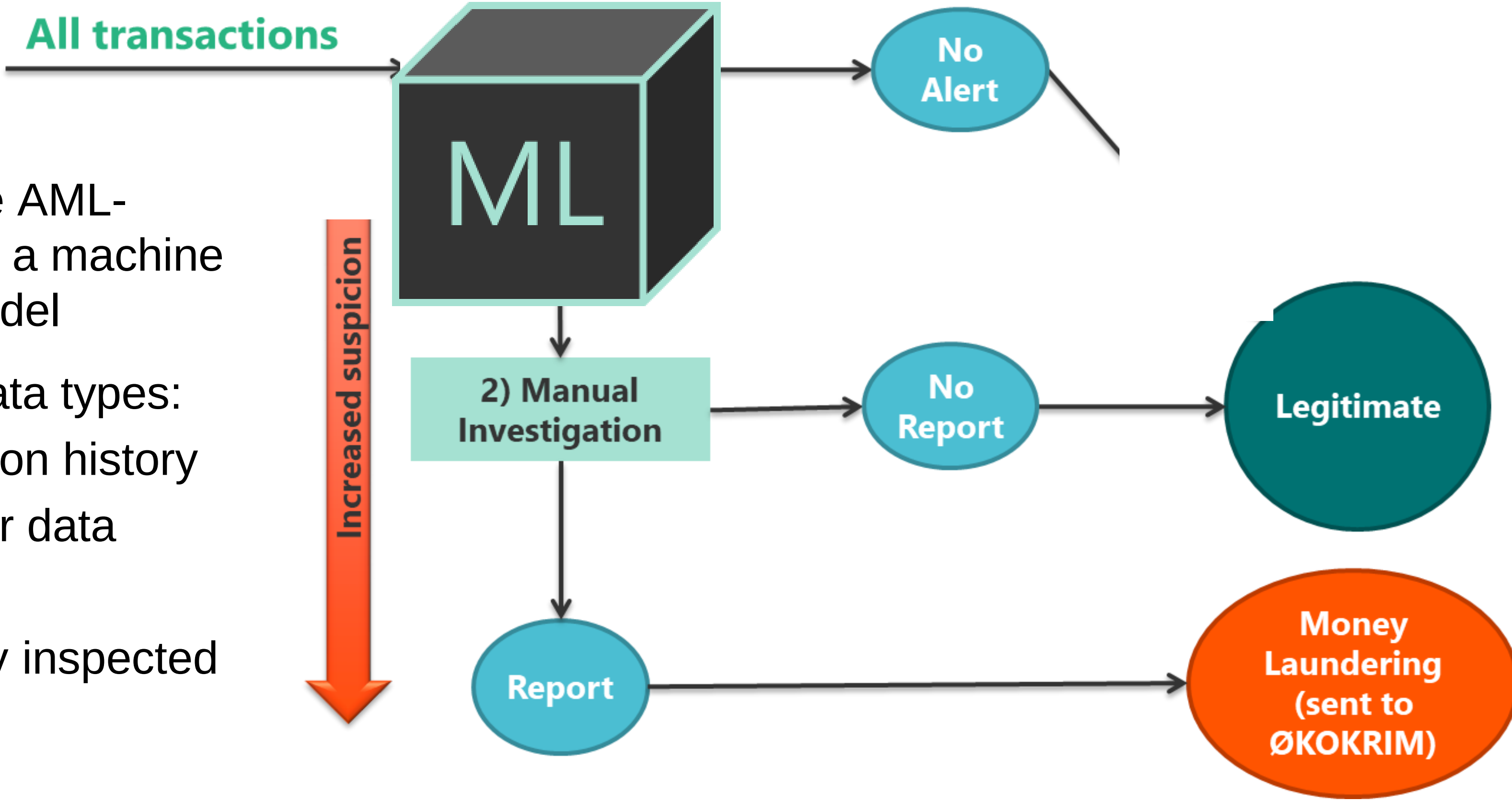
- Replace the AML-system with a machine learning model
- Available data types:
  - transaction history
  - customer data
  - alerts
  - manually inspected cases



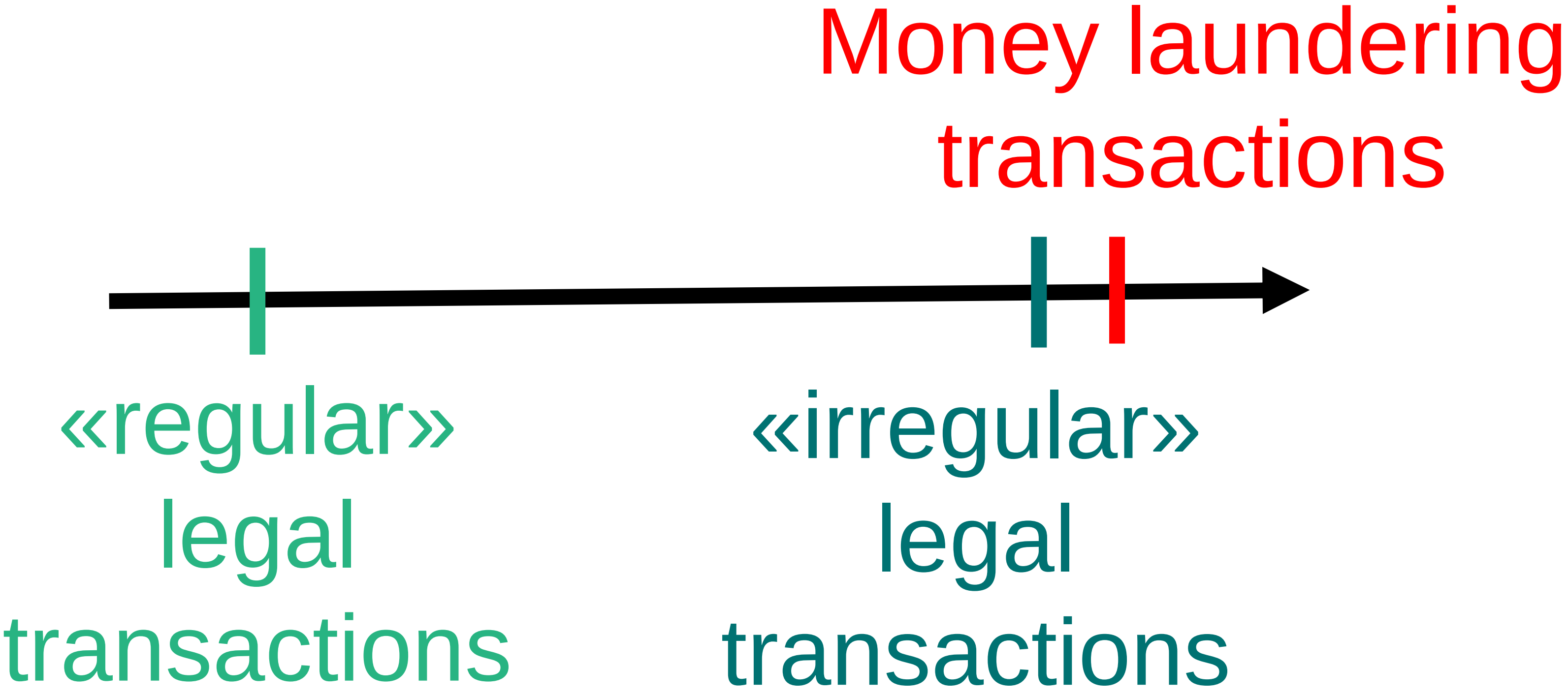
What we have done

More realistic setting!

- Replace the AML-system with a machine learning model
- Available data types:
  - transaction history
  - customer data
  - alerts
  - manually inspected cases



What makes this hard?

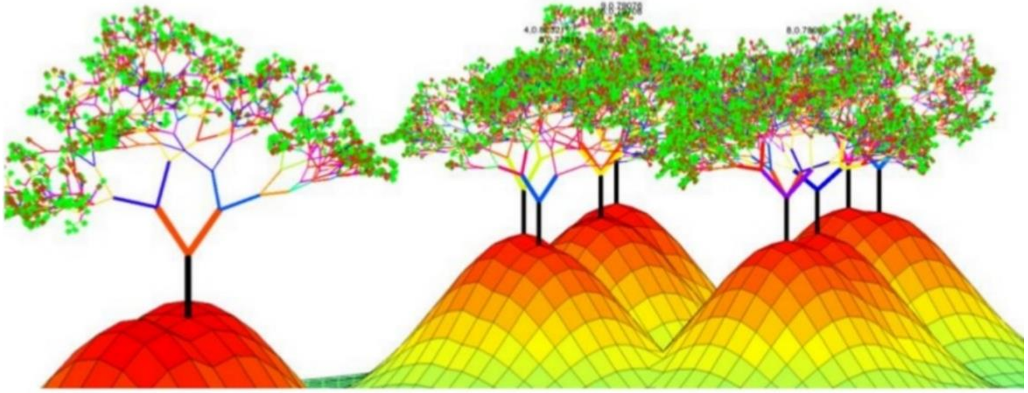


Modelling

- Binary response (Y): Transaction sent to Økokrim (Yes = 1, no = 0)
- Want to predict $P(Y = 1|\text{data related to present transaction})$

- State of the art: **Gradient boosting** machines (GBM)
- **XGBoost –** very efficient and flexible implementation of GBM based on **tree models**

dmlc XGBoost



Tree models

Learn model $f(\boldsymbol{x}) \approx y$ using $\boldsymbol{x} = (x_1, \dots, x_p)$

- Conceptually very simple: Constructed as a series of IF-ELSE rules on x_i

$$f(\boldsymbol{x}) = \sum_{j=1}^{T} \theta_j \, 1_{\{\boldsymbol{x} \in R_j\}}$$

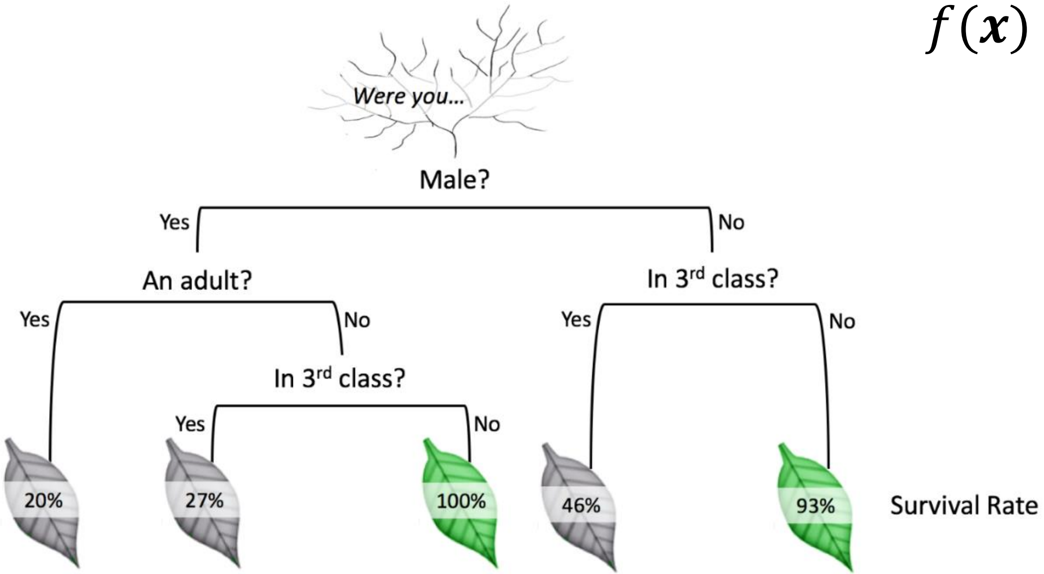
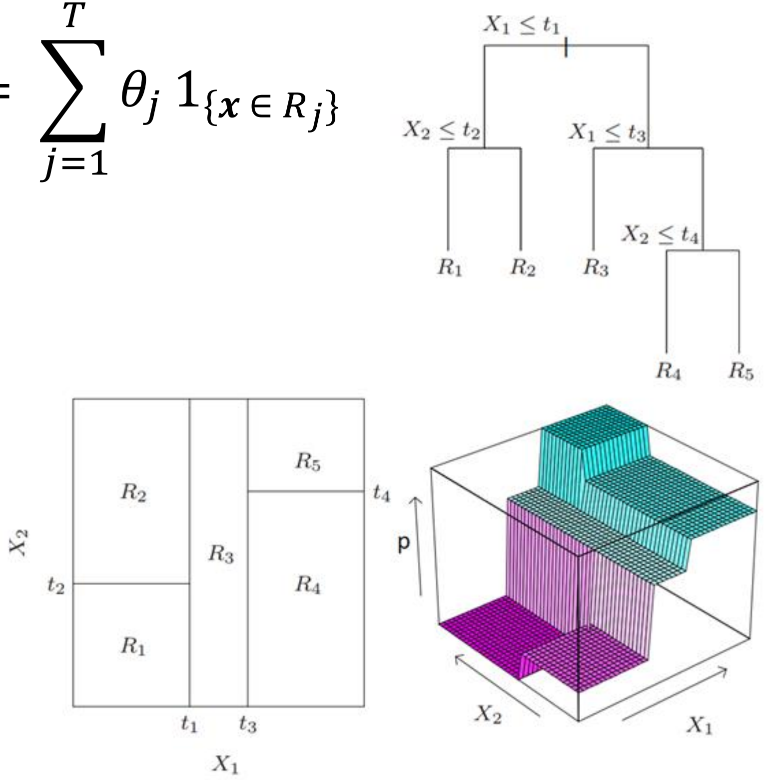


Illustration: Tree model for survival rate on Titanic

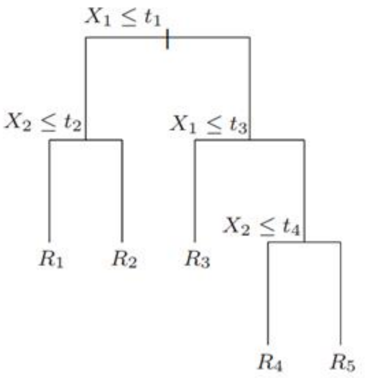


Tree models

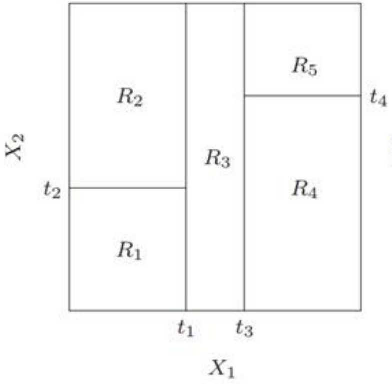
Learn model $f(\boldsymbol{x}) \approx y$ using $\boldsymbol{x} = (x_1, \dots, x_p)$

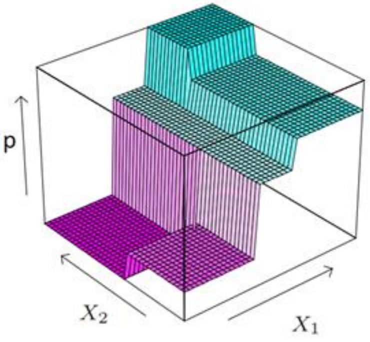
- Conceptually very simple: Constructed as a series of IF-ELSE rules on x_i

$$f(\boldsymbol{x}) = \sum_{j=1}^{T} \theta_j \, 1_{\{\boldsymbol{x} \in R_j\}}$$



- Benefits:
  - Easy to train (greedy algorithm)
  - Direct modeling of non-linearities and interactions
  - Invariant under monotone transformations of $\boldsymbol{x}$
  - Naturally combines continuous and categorical features
- Drawbacks
  - High variance
  - Limited predictive power





Boosting

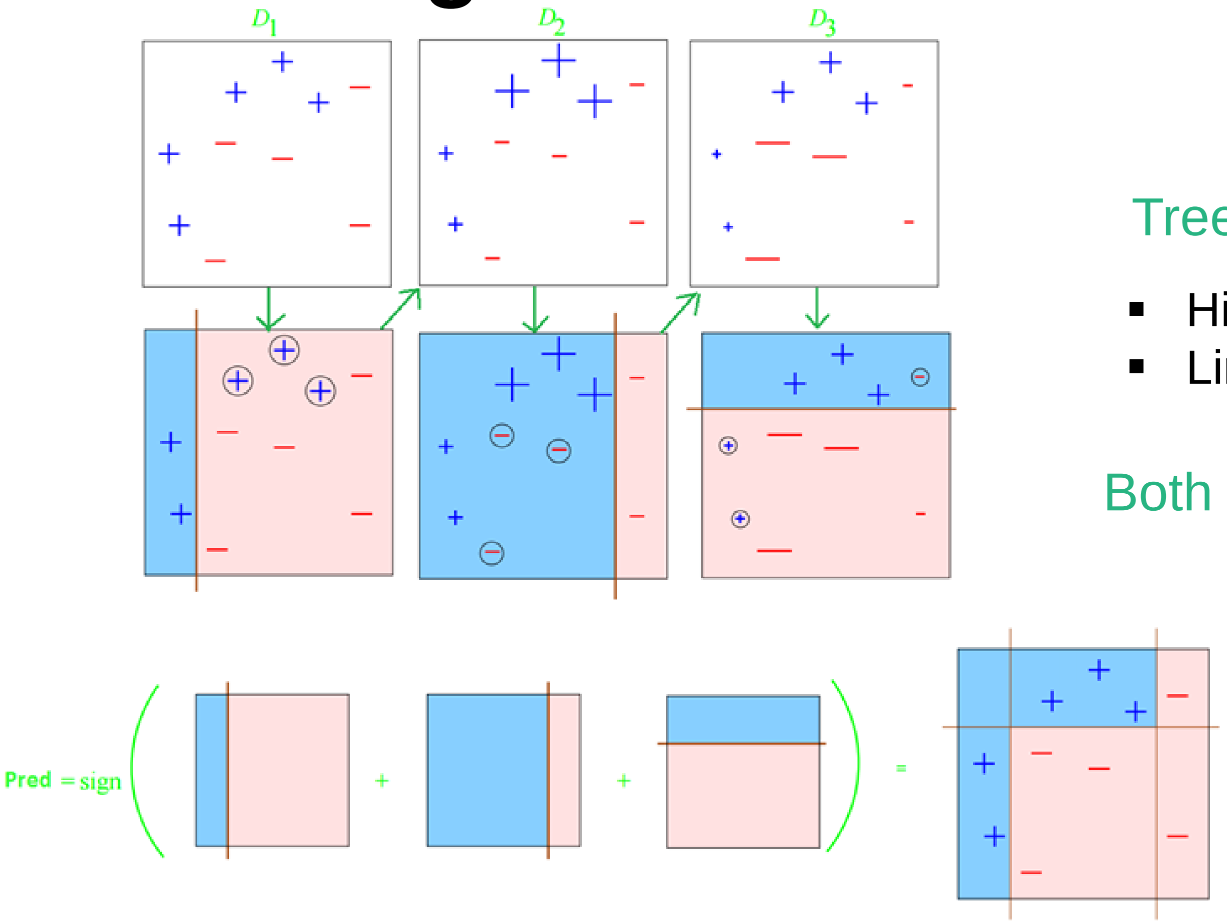
- Given some loss function $L(y, \cdot)$, iteratively fit simple models $f_m(\boldsymbol{x})$ (weak learners) trying to correct «errors» of previous models:

$$f_m(\boldsymbol{x}) = \underset{h \in \Phi}{\operatorname{argmin}}\, s_h(x), \qquad s_h(x) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f^{(m-1)}(x_i) + h(x_i))$$

and sum them into a strong learner

$$f^{(M)}(\boldsymbol{x}) = \sum_{m=1}^{M} f_m(\boldsymbol{x})$$

Boosting illustration

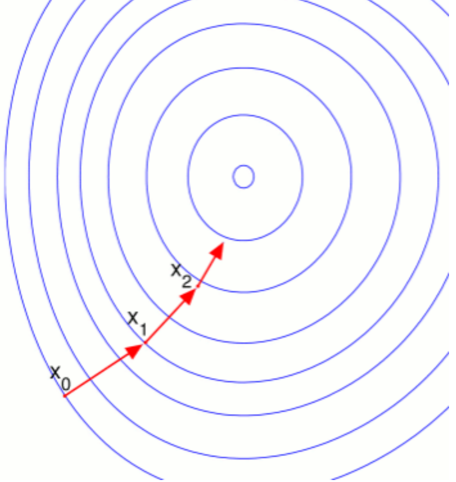


Tree model drawbacks

- High variance
- Limited predictive power

Both fixed using boosting!

XGBoost and gradient boosting



- Generally hard to solve

$$\operatorname*{argmin}_{h \in \Phi} s_h(x), \qquad s_h(x) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f^{(m-1)}(x_i) + h(x_i))$$

- Gradient boosting: Take a (functional) gradient descent-step towards the minimum of $s_h(x)$ instead of solving it globally

- XGBoost ++ uses a variant of Newton's method, also involving the hessian (but typically still calls it Gradient boosting)

Transforming raw data (feature engineering)

- XGBoost requires numeric tabular data as input!

Raw input data

- Specific transaction info
- Background info about sender/receiver
- Sender/receiver's transaction history
- Previously reported transactions from sender/receiver

Y	X1	X2	X3	X4	X5	X6
1	0,453406	0,992838	0,734389	0,159918	0,397515	0,949952
0	0,274	0,654207	0,169886	0,493841	0,407112	0,939789
0	0,741897	0,855005	0,585788	0,366456	0,365123	0,57955
1	0,488119	0,465754	0,716517	0,493048	0,855049	0,632114
0	0,134458	0,762057	0,848194	0,098779	0,872603	0,063026
0	0,531914	0,998817	0,808215	0,060721	0,716595	0,35374
0	0,341509	0,8398	0,637808	0,48304	0,279987	0,730286
0	0,530306	0,463271	0,338713	0,986781	0,925251	0,272484
1	0,864123	0,652763	0,689599	0,080937	0,990294	0,364736
0	0,106812	0,900351	0,450224	0,143815	0,593244	0,020764

1716 columns (features)

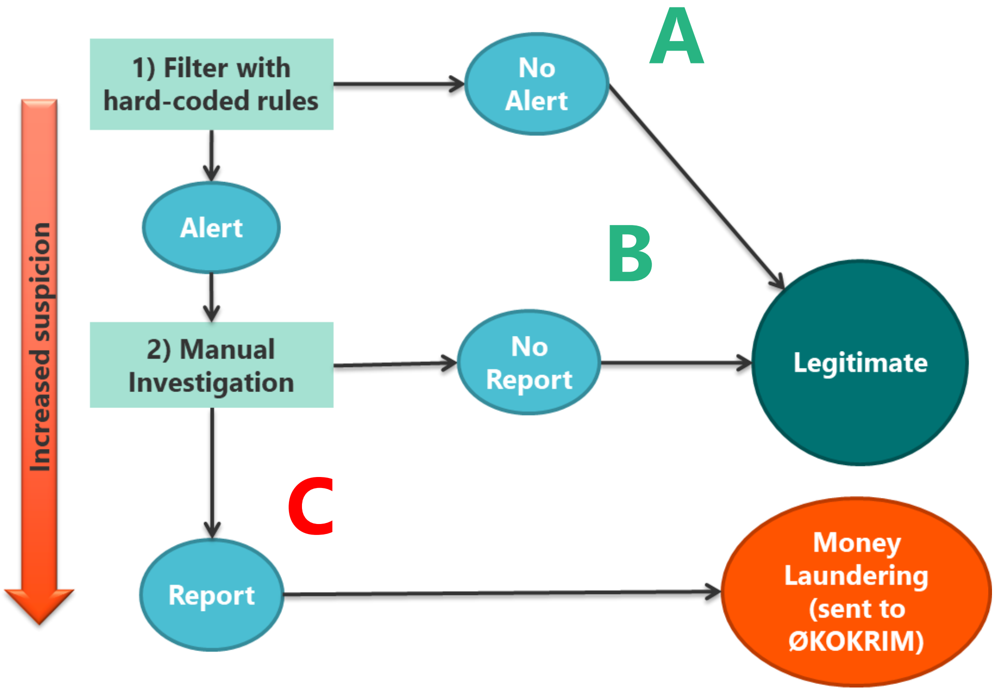
Data refinement

2 years of modellable transaction data

- All transactions leading to
  - A report (C)
  - An alert, but no report (B)
- A sample of normal transactions (A)

Data refinement

- We chose #A = #B
- Use only one transaction from each manual investigation (2)
- No transactions with same sender/receiver two consecutive days

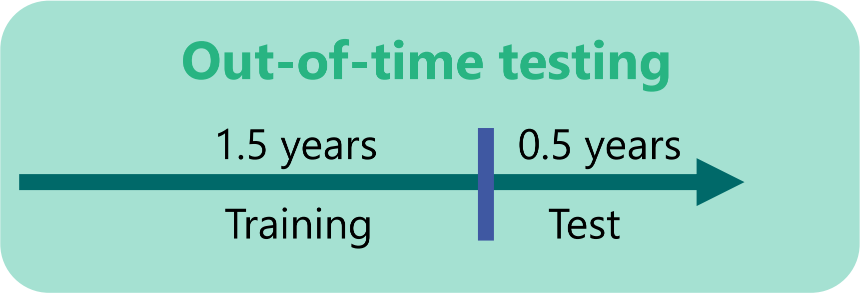


Training, testing and modelling

Modelling

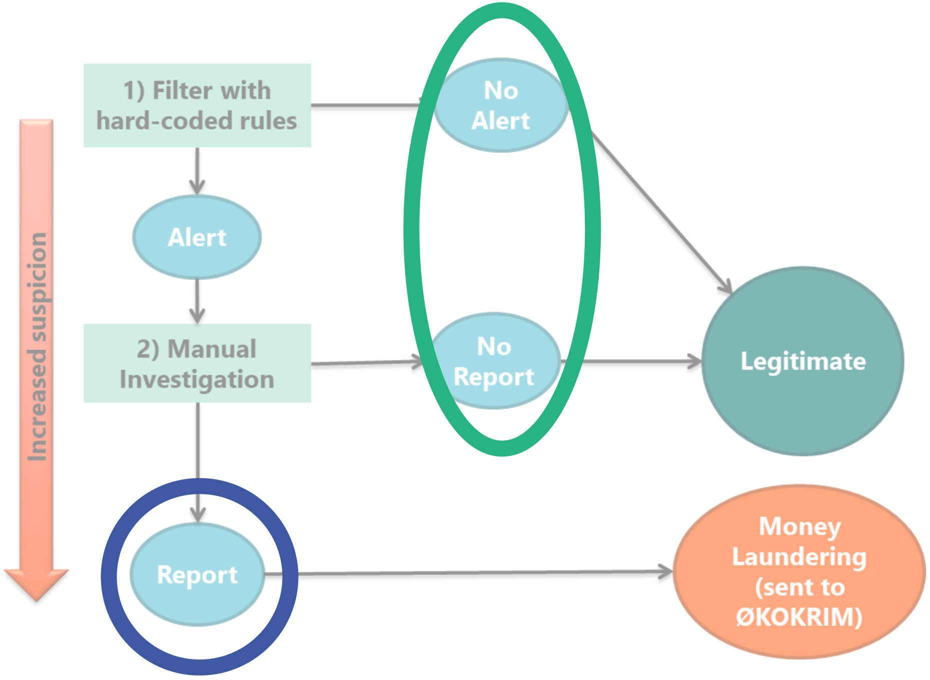
- 10-fold cross validation (CV)
- Stopping criterion (# boosting rounds): AUC
- Tuning: Random + iterative grid-search
- Model trained on GPU
- Final model used for prediction on test data:

$$\hat{f}(x_{\text{test}}) = \frac{1}{10}\sum_{i=1}^{10} \hat{f}_{CV,-i}(x_{\text{test}})$$

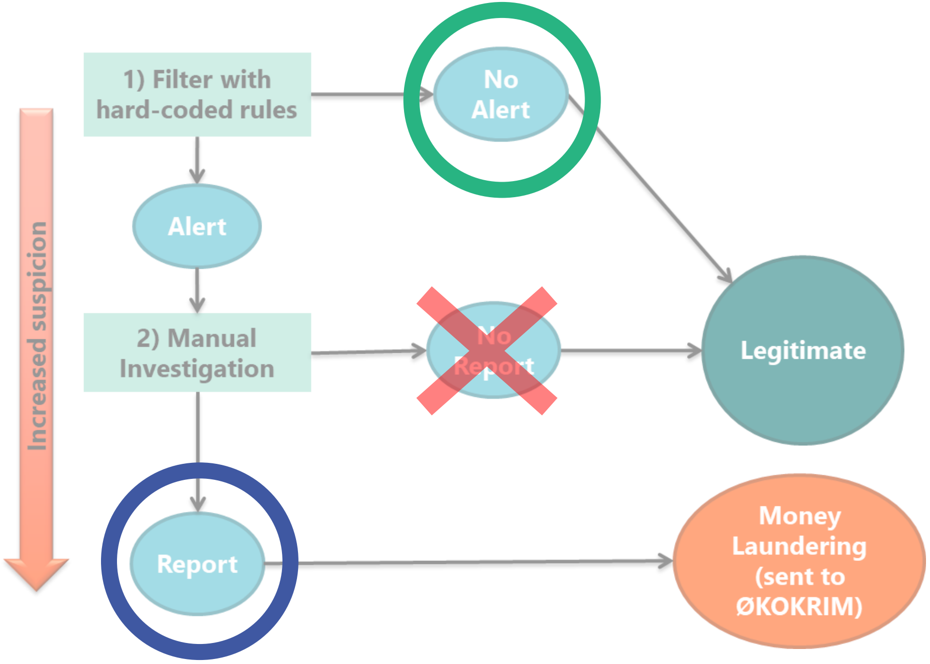


2 training scenarios

All data types

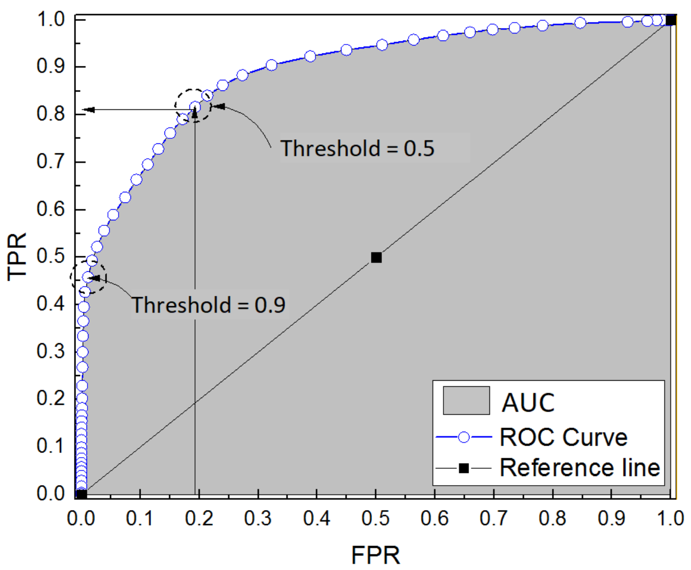


No unreported transactions



Evaluation metrics

Ranking:
AUC



Probabilities:
Brier score

$$\frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} (y_i - \hat{p}_i)^2$$

Comparing scenarios

	All data types	No unreported transactions
AUC	**0.907**	**0.852**
Brier	**0.025**	**0.340**

Much better!

Money laundering transactions

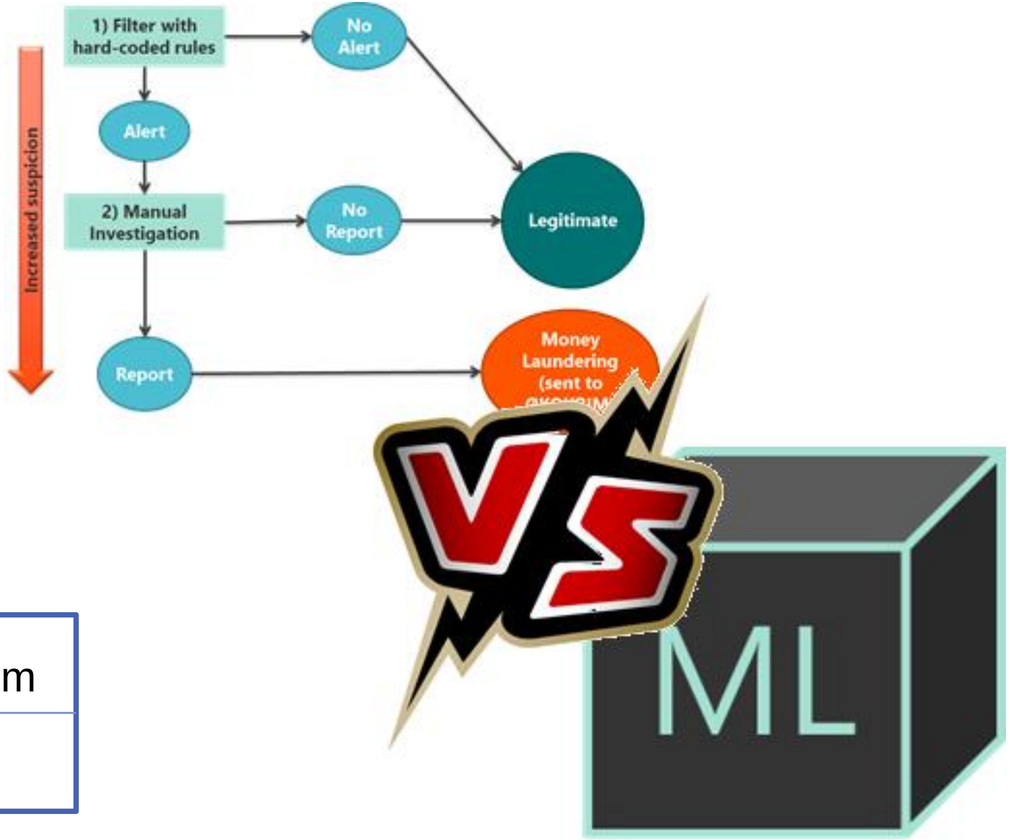
«regular» legal transactions

«irregular» legal transactions

ML vs current AML system

- Hard to properly compare

- **PPP = Proportion of Positive Predictions:** Proportion of transactions that needs to be controlled to find 95% of the reported transactions



	ML (all data types)	Current system
PPP	31.5 %	48.9 %

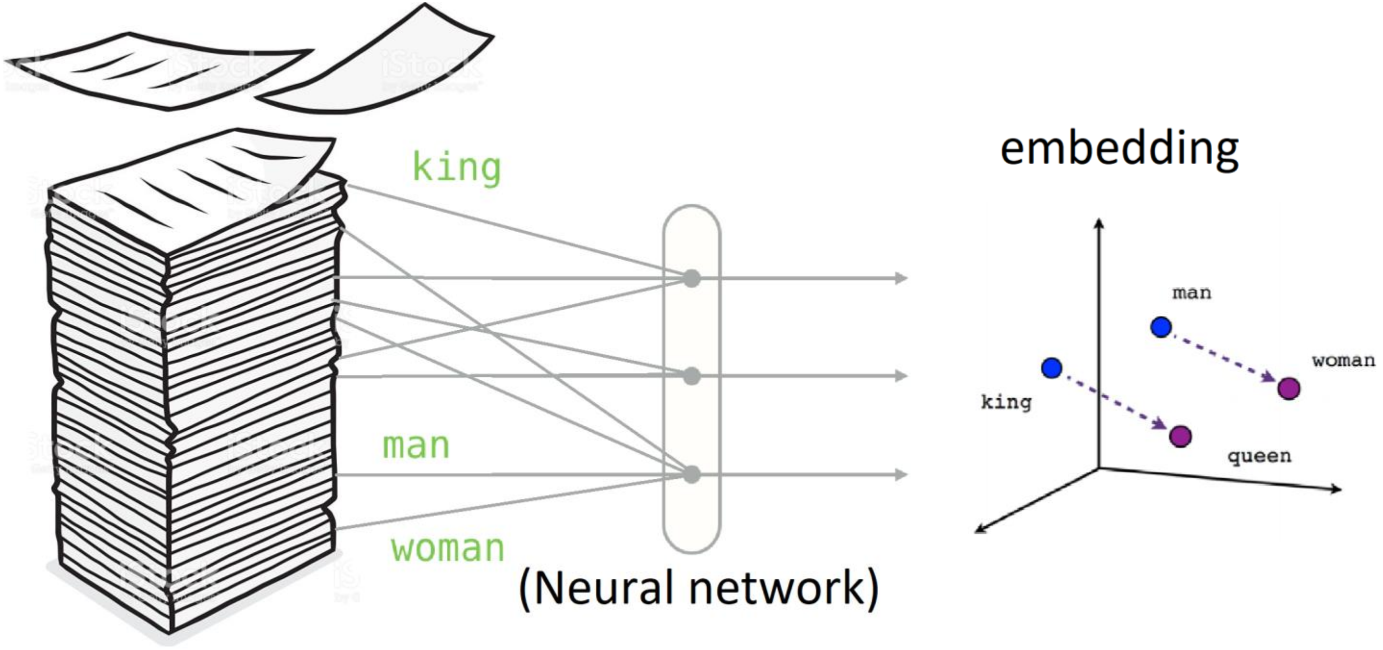
Limitations

- We are not really using the **time-evolving transaction network**
  - **Who** are you sending/receiving money to/from
  - **When** are you sending/receiving
- Social/professional network information is not used
- Many variables – complicates putting the model into production
- The model only learns what has already been reported

Current work: Utilize the transaction <u>network</u>

- Borrowing strength from NLP (Natural Language Processing)

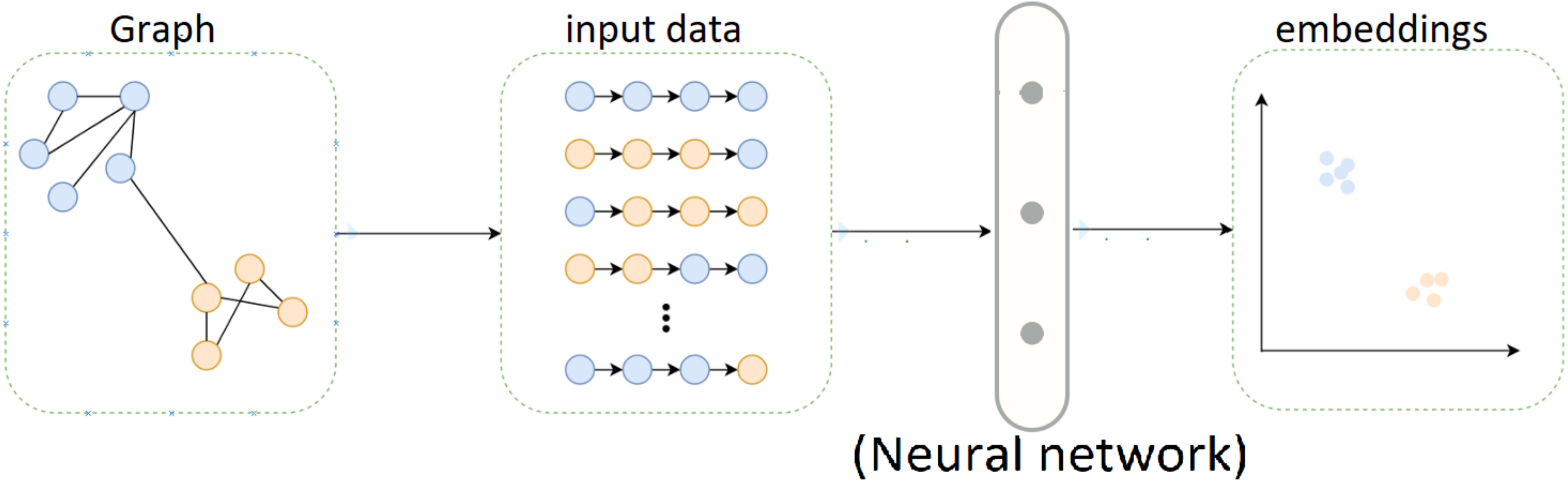
word2vec



Current work: Utilize the transaction network

- Borrowing strength from NLP (Natural Language Processing)

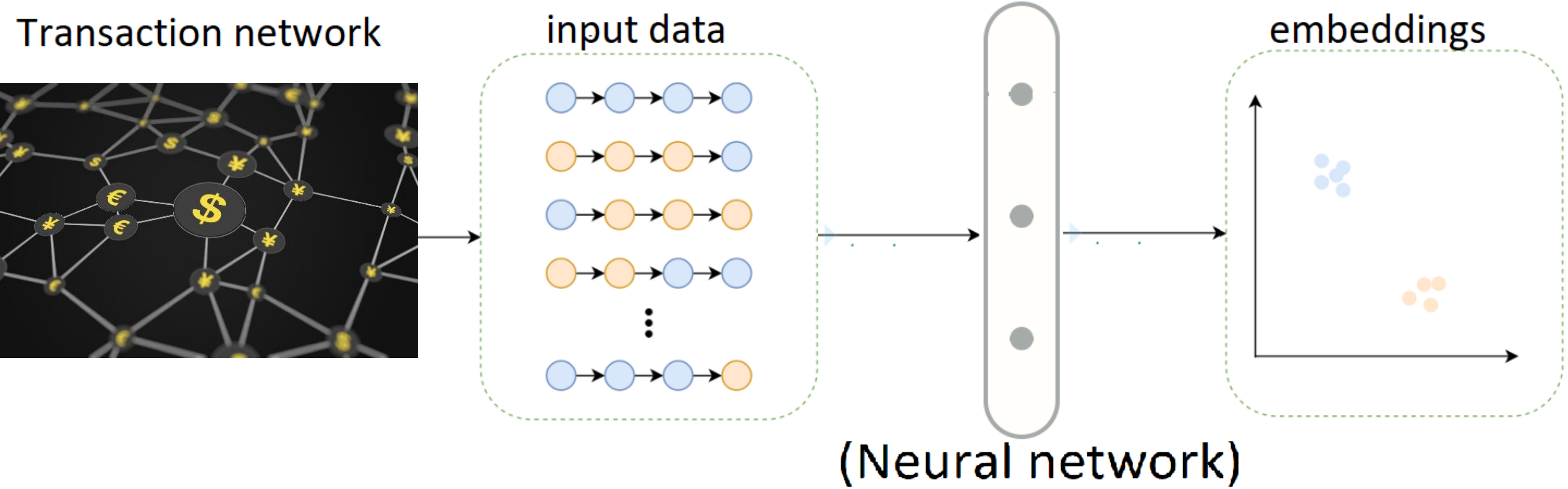
  word2vec ⟶ node2vec



Current work: Utilize the transaction network

- Borrowing strength from NLP (Natural Language Processing)

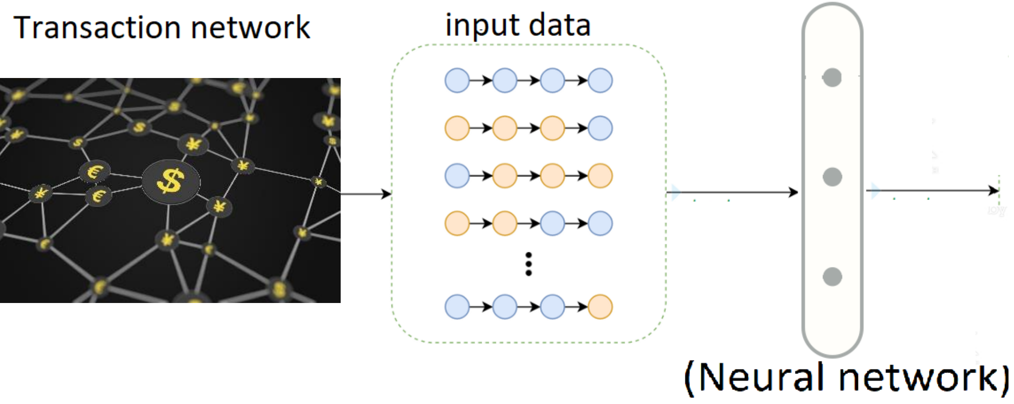
  word2vec ⟶ node2vec ⟶ trans2vec

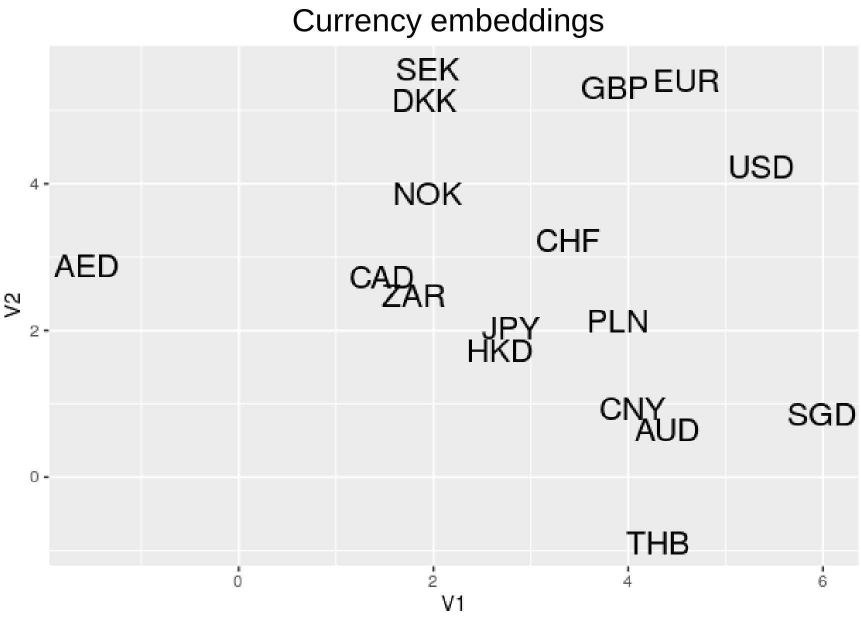


Current work: Utilize the transaction network

- Borrowing strength from NLP (Natural Language Processing)

word2vec ⟶ node2vec ⟶ trans2vec





Detecting money laundering transactions with machine learning

Martin Jullum, Anders Løland and Ragnar Bang Huseby
Norwegian Computing Center, Oslo, Norway, and
Geir Ånonsen and Johannes Lorentzen
DNB, Oslo, Norway

jullum@nr.no

NR Norsk Regnesentral
NORWEGIAN COMPUTING CENTER

BigInsight

DNB