Machine Learning in Finance: Forecasting purchase behavior for banking products

Viktor Reif

January 26, 2023

Abstract

abstract text ...

Table 1: product usage as feature

	Mean	Empty Clients
Asset Management	0.02	67,076
Aval	0.06	62,638
Betriebliche Altersvorsorge	0.00	68,408
Bond-Emissionen	0.00	69,043
Bürgschaften und Garantien	0.11	59,234
Cash Pooling	0.00	68,337
Mobilienleasing	0.01	66,594
Export Dokumentengeschäft	0.02	63,996
Exportfinanzierung	0.00	68,905
Forderungsmanagement	0.01	68,241
Geldmarktkredit	0.04	64,434
Global Payment Plus	0.06	60,310
Import Dokumentengeschäft	0.01	67,790
KK-Kredit	0.01	66,943
Kapitalanlagen	0.01	66,545
Rohstoffmanagement	0.00	68,650
Sichteinlagen	0.77	5,895
Termin-Einlage	0.04	63,263
Unternehmensfinanzierung	0.13	57,244
Währungsmanagement	0.05	61,807
Zinsmanagement	0.00	68,429

wo summary statistics ausser bei raw-data.pkl?)

All primary datasources are tables direct copies from the productive database of Sales Analytics BDAA Commerzbank. The table ”Ertragsdaten” contains all revenues from the bank’s entire corporate client sector. Each row in this table represents an amount of revenue that the bank generated off a product a client has bought. In the Commerzbank product universe there exists a tree-like mapping that groups all products into categories on five levels of granularity. The table ”Etragsdaten” lists products on the second lowest level of this product tree. On this level there exist approximately 800 products. Using the third? level of the tree (maybe mention interviews) we group the products into 21 meaningful categories. We assume that when a client is paying for a product, they is using it. We aggregate the revenue information per client per month into the binary variable ”used product group”: Yes/No (1/0) for each of the 21 product groups individually. The total time range in the dataset ranges from 201901 to 202206. Whenever a client did not create any revenue for a given month, we fill the respective field with ”0” by default. If a client holds more than one account at the bank, all variables are aggregated additively.

After these aggregation steps the dataset contains the datset contains product information for 69,340 clients and 42 months, totalling at 2,912,280 rows. The two keys of this resulting dataset are the id of the client and the monthly date. As a combination these two keys are unique. Given this structure, it is possible to add any set of features to the dataset. Features must must contain the same keys. This means that the data must be

Table 2: categorical feature gprckdgrp summary

	Count	Percent
Corporates Inland (ohne Umsatzangabe)	19,627	28.31%
Corporates Inland (Umsatz EUR 500 Mio)	5,287	7.62%
Finanzierungsgesellschaften: Leasing...	4,998	7.21%
Corporates Inland (Umsatz EUR 25 Mio)	4,367	6.30%
Corporates Inland (Umsatz EUR 2,5 Mio)	4,312	6.22%
Other (38)	30,749	44.35%

aggregated on a monthly level, too. If a key within the feature data is missing, it is filled with 0 for numeric data and None for categorical data.

The features we add to the dataset are account balance, transaction details, industry, credit rating, client group (grouped by revenue and domestic/foreign) and client independent market characteristics. Account balance quantifies a client's monthly average liquidity. Transaction details contain the information (as a sum) how much money measured in EUR each client transferred in which currency into which industries in each month. One example transaction could be that a client transferred an amount of USD worth 500,000 EUR to an account tagged as a "consulting firm". The total number of 80 used currencies are consolidated into EUR, USD, GBP and "other currencies", as these currencies make more than 95% of the entire transaction volume. There are 35 industries including the placeholder "untagged" which a transaction can be tagged as. Industry, credit rating and client group are static features for each client. Static means that these features are fixed over time such that in the dataset they appear as vectors of 42 times (once per month) the same characteristic without any changes. Market characteristics are monthly averaged time series of the german stock market index "DAX", the exchange rate EUR/USD, interested rates on the european and US-american capital market (Euribor, Libor) and the international crude oil price. As these market features are the same for each client, they appear equally 69,340 times in the final dataset.

Merging all these features into the dataset creates a table with the dimensions 2912280 rows × 76 columns. Note that this large number of columns is caused by some features being split into several fields. Eg the information into which industries a client is transferring money is split into 35 columns, as it is possible to transfer to more than one industry in the same month.

In an effort to obtain as many valid samples for later feeding the predictive models we slice more than one sample out of each clients panel. The product usage and features of each client have a total range of 3.5 years (42 months). Models may differ in how "long" their respective training slices are. Yet, all models are bound to predict on the same test samples of evaluation. These test samples are per client the last three sets of 6-months-long time windows. Figure XX visualizes the logic of creating the test samples

Factually, each client within the given dataset has bought at least one product. Analyzing the distribution of newly used products (delta) across clients as seen in figure XX reveals that there are numerous clients who newly purchase one or a few products. Figure XX2 shows the same distribution for the last 18 months of each client panel, which will be

Table 3: Numeric Features Summary

	Mean	Std Deviation	Empty Clients
DAX Index	13,109.17	1,557.65	0
EUR USD Wechselkurs	1.13	0.05	0
EURIBOR ON	-0.47	0.06	0
EURIBOR 3M	-0.45	0.10	0
EURIBOR 12M	-0.32	0.25	0
LIBOR USD ON	0.32	0.87	0
LIBOR USD 3M	0.87	0.91	0
LIBOR USD 12M	0.89	0.99	0
Oil Price	57.84	21.40	0
Transaction EUR	-88.51	17,796,785.12	41,281
Transaction USD	-4,991.67	625,029.69	64,831
Transaction GBP	3,477.05	368,151.93	68,376
Transaction other currencies	-128.10	17,227,270.99	41,239
Transaction Abfallentsorgung	-288.90	193,492.32	59,419
Transaction Bauindustrie	-595.67	2,845,176.12	51,793
...			
Transaction Werbung und Marktforsch	-291.00	89,635.54	64,830
Liquidität	49,551.21	5,542,097.20	5,266

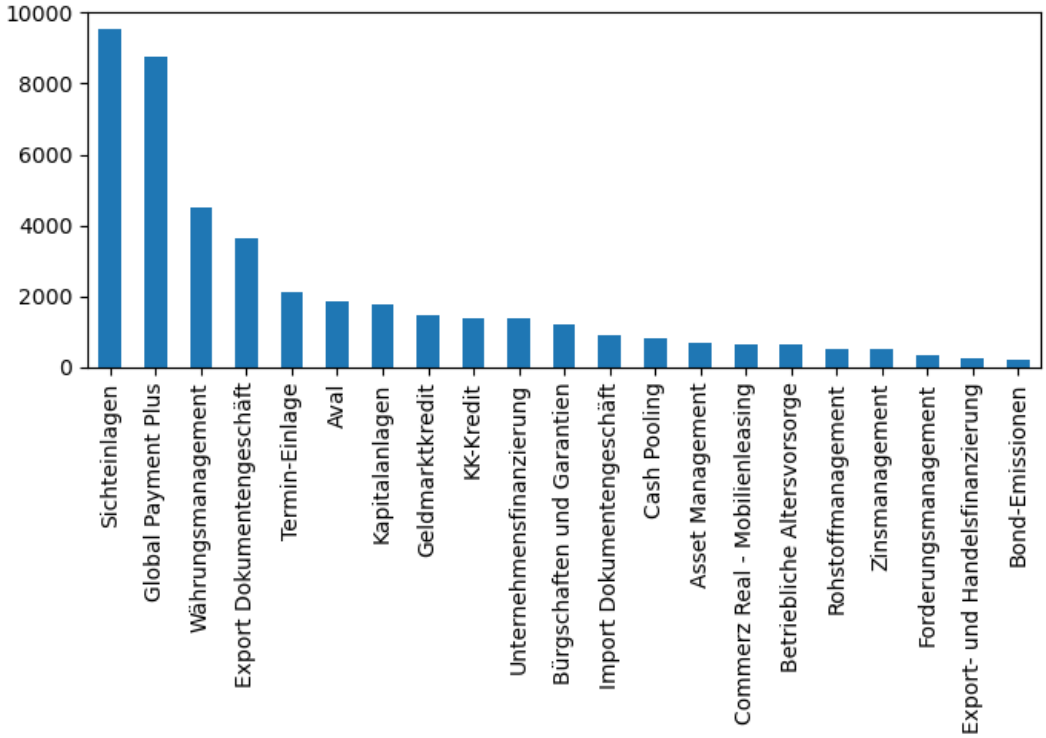

Figure 1: Figure X: Confusion matrix.

the test data for the models.

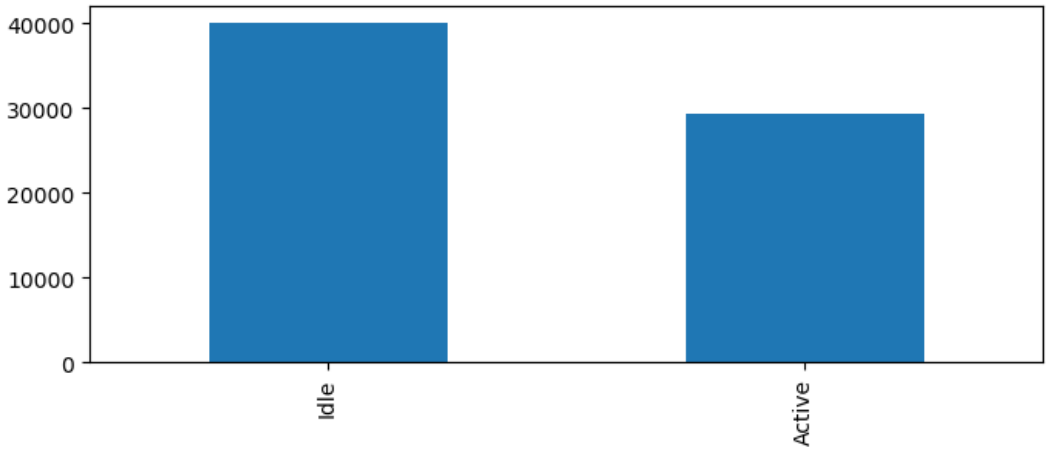

Figure 2: Figure X: Confusion matrix.

It is apparent that a large portion of clients do not aquire new products in this timeframe. Comparing the distribution of Figure XX2 to distribution of newly purchased products per product within the same timeframe (last 18 months) shows a strong imbalance. Note that the products themselves show a moderate imbalance. Yet, the quantity of idle clients is far too large to be tolerated. Hence, we allow for a fixed number of idle clients. We drop all clients that do not purchase a new product in the last 18 months of the dataset, except a radnom subsample of 10000 of them. This drops the total size of the dataset to XXX rows containing only 35k? clients. Thus, 40k? idle clients have been droppped. (create py file that does only create these tables/plots)

4 Approach

- states that I use a framework to make different models/estimation methods comparable
- explains framework and why it is adequate (similar to approach section in proposal, but more detailed)
- states the different models/methods within the framework

We work on the research question through an empiric case study. We define a framework in which the behavioral problem is translated into a modelling problem that is the same for all proposed approaches. As all approaches work the same problem with the same data, we can assume that they are comparable by our proposed metrics.
The modelling problem is the same in terms of the target variable, which contains a set of possible purchase behaviors a client i can show at a certain point in time t. Each of these purchase behaviors translates into a client buying one product from a speficic group of aggregated products.

$$behavior_{it} \approx productgroup_{it} \tag{1}$$

An example for one purchase behavior is that a client buys a product that counts to the product group of credit-like products. The prediction problem is formally a multilabel classification problem. This means that at t a client can purchase up to N different product

groups simultaneously. labels vs classes: clarify! https://stats.stackexchange.com/questions/11859/what is-the-difference-between-multiclass-and-multilabel-problem

In the target vector this can be represented as a vector of booleans

$$\overrightarrow{behaviors_{i,t}} = [productgroup_1 : yes, productgroup_2 : no, ...productgroup_N : yes]_{i,t} \tag{2}$$

or of integer booleans

$$\overrightarrow{behaviors_{i,t}} = [productgroup_1 : 1, productgroup_2 : 0, ...productgroup_N : 1]_{i,t} \tag{3}$$

In an effort to reduce sparsity in the target vectors we can aggregate several time steps further into the future into one vector of aggregated purchase behaviors starting at certain future point in time $t + p$. Thus, the vector

$$\overrightarrow{aggregatedbehaviors_{i,t+p}} = \max_{Q}^{q=1} \overrightarrow{behavior_{i,t+p+q}} \tag{4}$$

contains the value of 1 as an element if a product group was bought at least once in the timeframe of size Q. Elsewise the element's value is 0.

The modelling problem is also equal in terms of features. In the framework, all approaches have the same set of input variables available for predicting. This set includes features such as purchased products, industry, credit rating, transaction currencies, transaction partners industry-wise, deposits, macro factors, etc.. These features vary over time and for each client. This can be represented as

$$\mathbf{F} = \left[\overrightarrow{behaviors_{i,t}}, \overrightarrow{features_{i,t}}\right] \tag{5}$$

where the feature matrix $\mathbf{F}$ consists of a vector of purchase behaviors and a vector of other features. The former posseses the exact same element structure as in (5xx). The latter consists of M additional features structured $[feature_1, feature_2, ...feature_M]_{i,t}$, where for instance $feature_1$ is a client's credit rating and $feature_2$ is a client's industry. The structure continues like so until all M features are listed. Example values for one client given this structures would be $120, ITServices, (...)$. Since we treat product usage like any other feature, we can merge all features into one single vector

$$\mathbf{F} = \left[\overrightarrow{features_{i,t}}\right] \tag{6}$$

where one feature vector includes N products and M additional features. One such example vector for one client i at t would be

$$[1, 0, 1, ..., 0, 120, ITServices, ..., 20000] \tag{7}$$

This vector of length $N + M$ shows values for the product usage of the N product groups and for the M additional features. Here, at the moment t the example client bought a product from $productgroup_1$, $productgroup_3$ and maybe some more product groups that are not visible in the example, operatesin the *IT Services* industry, has credit rating of 120, ..., and finally has a liquidity of 20000€. Furthermore, we can remove the vector notation to formulate

$$\mathbf{F} = \left[features_{i,t,k}\right] \tag{8}$$

where for each client i, there exist k different features that may change at every point in time t. For an example client F would have the following values

$$\begin{bmatrix} 0,0,0,...,9000,1200,20000 \\ 0,1,0,...,8000,1000,25000 \\ 1,0,0,...,1000,1000,24000 \\ ... \\ 0,0,0,...,1200,5000,20000 \\ 0,0,0,...,1500,6000,18000 \\ 0,1,0,...,1000,8000,16000 \end{bmatrix} \tag{9}$$

where column shows the changes over time of one feature. Given target and feature, the modelling problem can be formulated as

$$\overrightarrow{\hat{aggBehavior}}_{i,t+p} = \alpha(\mathbf{C} \circ \mathbf{F}) \tag{10}$$

where $\alpha()$ is any kind of feature engineering function for preprocessing. $\mathbf{F}$ is the feature matrix. The feature choice $\mathbf{C}$ is a matrix of binary weights with the same shape of $(i * t * k)$ as $\mathbf{F}$. $\circ$ means that we take the Hadamard product of $\mathbf{C}$ and $\mathbf{F}$, which is the result of an entry-wise multiplication. If $\mathbf{C}$ has the value 0 for a certain element, the positionally equal element in $\mathbf{F}$ will also be 0. Hence, $\mathbf{C} \circ \mathbf{F}$ formulates an approach's ability to exclude certain features of all K features and omit certain time points of all T time points. For one example client the weights could be

$$\begin{bmatrix} 1,1,1,...,1,1,0 \\ 0,0,0,...,0,0,0 \\ 0,0,0,...,0,0,0 \\ ... \\ 0,0,0,...,0,0,0 \\ 0,0,0,...,0,0,0 \\ 0,0,0,...,0,0,0 \end{bmatrix} \circ \begin{bmatrix} 0,0,0,...,9000,1200,20000 \\ 0,1,0,...,8000,1000,25000 \\ 1,0,0,...,1000,1000,24000 \\ ... \\ 0,0,0,...,1200,5000,20000 \\ 0,0,0,...,1500,6000,18000 \\ 0,1,0,...,1000,8000,16000 \end{bmatrix} = \begin{bmatrix} 0,0,0,...,9000,1200,0 \\ 0,0,0,...,0,0,0 \\ 0,0,0,...,0,0,0 \\ ... \\ 0,0,0,...,0,0,0 \\ 0,0,0,...,0,0,0 \\ 0,0,0,...,0,0,0 \end{bmatrix} \tag{11}$$

where via $\mathbf{C}$ only the first time point $t = 1$ is selected. Also the last feature was excluded. For the sake of comparability approaches may not exclude clients (maybe show this mathematically by constructing C?).

Here an import caveat is that a prediction only exists within a time context. Montgomery et al. (2015) state that prediction must always reference a future event. A recommendation, however, might be present oriented, future oriented or exist without any time context (quote?). We assume that in our case study a prediction is approximately equal a recommendation. Thus, we assume that

$$\overrightarrow{\hat{aggBehavior}}_{i,t+p} \approx recommendation \tag{12}$$

. If a given model predicts that a client will by a certain product group in the future, it is reasonable to recommend the product group at the present. Thererfore, we also assume that

$$recommendation \approx \overrightarrow{\hat{aggBehavior}}_{i,t+p} \tag{13}$$

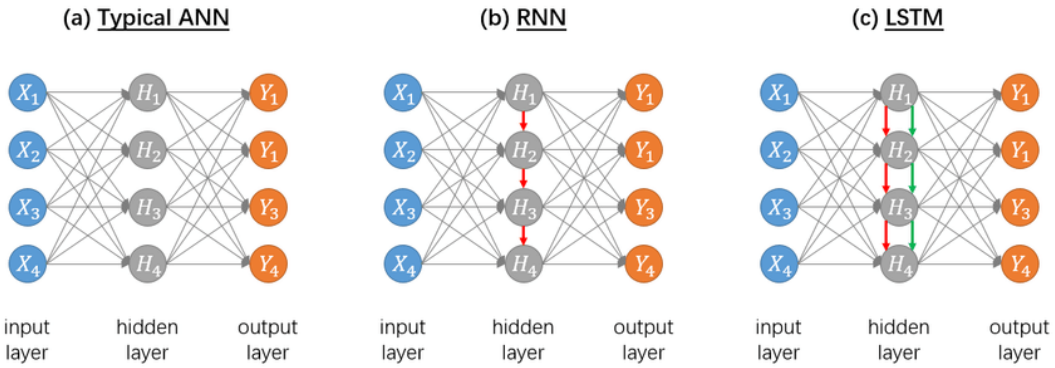

Figure 6: maybe exclude, because it does not really show the loops from and to the same node within a layer RNN

the network's topology allows for feedback connection. Namely, the hidden layers in a RNN are recurrent. This means that a node in these hidden layers has not only synapse connections to nodes of the next (hidden) layer, but also to itself.

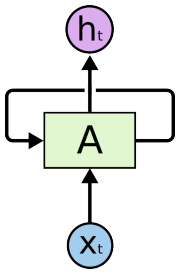

Figure 7: RNN-rolled

Figure 5 visualizes this self loop for a given node A (im not sure if A represents one node or one layer?), where x_t is the node's input and h_t its output for any given training step t. That way the network can store information in one traing step and access it in the next.

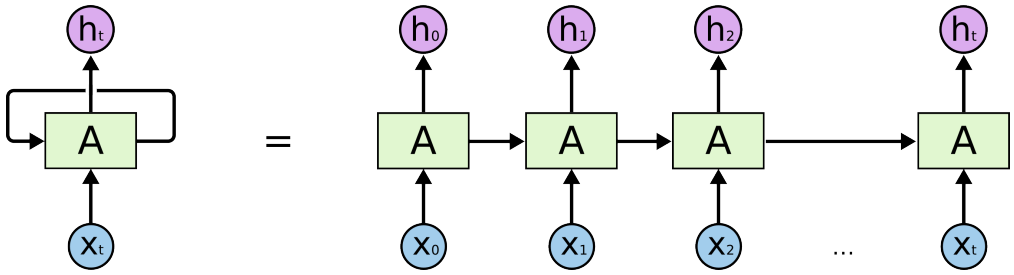

Figure 8: RNN-unrolled

Information may last for many steps within the RNN as each recurrent node acesses the information from its predecessor node and updates it for the next step. Figure 6 is a visualization of the RNN's process of passing on information. Mathematically, the RNN applies the function

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{33}$$

, where $tanh(x) \rightarrow y, x \in \mathbb{R}, y \in [-1, 1]$, to calculate the output and info for next step given.

(how is lstm motivated) Due to the auto-connected nodes the RNN can remember long term patterns and dependencies. In practice, however, RNN architectures often fail to handle dependencies that are very long term. Depending on its nature, this phenomenon is called either the vanishing or the exploding gradient problem. Bengio et al. (1994) explain the mathematical process of these two issues. Generally, the intuition of both is that there exists a positive correlation between pattern length and likelihood that the RNN "forgets" the pattern. (lstm motivation: vanishing gradient) (Numerically, the information, that a recurrent node passes on to itself for the next step, is a value $\in \mathbb{R}_{>0}$. Due to the way the RNN trains itself, this value is multiplied many times during a training process.) The source of the two issues lays in how a recursive net is trained. As in a standard feet-forward net (there the problem can also occur, but is unlikely), the RNN usually uses backpropagation on the basis of the error at the output to update its weights (synapses). To account for the auto-connections in the recurrent nodes, the RNN is unrolled into several ANNs, each of them a dpulicate of the original network at a certain previous time step. This unrolling is showed in figure 6. The gradient, on which the final weights of each training step depend (includes the weights of every later layer?), is the result of taking the derivate of a product of activation function outputs given the input training data. In the unrolled RNN, this product is a very large term containing many duplicate elements. Due to this, the risk of the gradient converging either to zero, ie to vanish, or converging to ∞, ie to explode, is high. Consequently the gradient impedes the learning process of the network, which ultimately drowns the overall performance. In an effort to mitigate the gradient issue, Hochreiter and Schmidhuber (1997) proposed the LSTM architecture.

(how does lstm work) In contrast to the RNN's easy algorithmic structure of using once the tanh function to implement the information self loop visualized in figure 5, the LSTM is more complex. The LSTM's structure can be split into three different sections, each serving a different purpose. These section are referred to as gates, and are in general a way to optionally let information through a node's system. In the LSTM, each gate uses either the sigmoid function

$$\theta(x) = \frac{1}{1 + e^x} \tag{34}$$

, where $\theta(x) \rightarrow y, x \in \mathbb{R}, y \in [0, 1]$, and/or the *tanh* function as seen in (35) to modify and pass on information. sigmoid's purpose in the LSTM is to serve as an activation function, which reformats the input into the wanted non-linear shape (correct?). tanh is used as a filter function to mitigate redundant inputs. One of the three gates in the LSTM is the "forget gate". It takes the information passed from the previous step and applies the sigmoid function to filter how much of that information will remain in the information status of thecurrent step. The "input gate" concatenates the output from the previous step and the new input from the training data. The concatenation result is used as an argument for sigmoid and tanh seperately. The returns values of sigmoind und tanh are then used to additively update the information status of the node. The "output gate" multiplies the sigmoid of the output from the previous step and the new input with the tanh of the updated information. It then returns this product as the new output. This process takes place in every recurrent node. Outside the recurrent nodes, the LSTM works exactly like a RNN.

8 Fragen ST

- wo diskutiere ich warum ich mich für meine features entschieden habe(gründe: literatur und statistisch)?

.

- ist es legitim die features choice im modell als m darzustellen?? oder muss das ein vektor sein. weil es ist ja eigentlich keine dimensions, oder? bzw dann choice of time auch...

.

- bei 4.x warum approach adequate, was schreiben?

.

- sollte ich zb identity function noch genauer erkären, u/oder zitieren?

.

- wie detailliert soll das LSTM erklärt sein?

.

- appendix nach literaturverzeichnis?

9 Fragen an mich selbst

- RNN allows information to persist... pro epoche? oder pro sample? (eher sample oder?) ”from one step of the network to the next” - RNN hat recurrent node synapsen zu sich selbst, oder zu anderen nodes im gleichen layer? - in fiugre 5 (aus blog) is A ein node oder mehrere oder ein layer? - Rechtsrand 3cm sieht she hässlich aus?

10 Notizen

- in SO bzw datascience fragen wie bei riesen baum wichtigste node oder so zu finden und visualieren? - RFM lit lesen welche schätz methoden und wichtig: wie motiviert! - are numeric cols in cs approach normalized??? oder gehen da irgendwo werte ¿1 ein??? - think where to mention that all numeric features a normalized on client *i* level. - in allen formeln dimensionen überlegen... besonders wie produkte und features als eine dimension (vllt zusammen nehmen als features, sodass feature(i,t,m) wo m angibt welches feature wobei produkte auch ein feature) - auf papier mal alle formeln aufschreiben (allg, und für die 3 approaches) - überlegen wie prediction time window (aggregated t+1,t+2,...t+p) - In particular when there are N labels, the search space increases exponentially to 2**N

References

Aubert, A., Tavenard, R., Emonet, R., De Lavenne, A., Malinowski, S., Guyet, T., Quiniou, R., Odobez, J.-M., Mérot, P., and Gascuel-Odoux, C. (2013). Clustering flood events from water quality time series using latent dirichlet allocation model. *Water Resources Research*, 49(12):8187–8199.

Bahnsen, A. C., Aouada, D., Stojanovic, A., and Ottersten, B. (2016). Feature engineering strategies for credit card fraud detection. *Expert Systems with Applications*, 51:134–142.

Barreau, B. (2020). *Machine Learning for Financial Products Recommendation*. PhD thesis, Université Paris-Saclay.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Bennett, J., Lanning, S., et al. (2007). The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. Citeseer.

Bernardi, L., Kamps, J., Kiseleva, J., and Müller, M. J. (2015). The continuous cold start problem in e-commerce recommender systems. *arXiv preprint arXiv:1508.01177*.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Covington, P., Adams, J., and Sargin, E. (2016). Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198.

Farrell, D., Greig, F., and Deadman, E. (2020). Estimating family income from administrative banking data: A machine learning approach. In *Aea papers and proceedings*, volume 110, pages 36–41.

Fu, K., Cheng, D., Tu, Y., and Zhang, L. (2016). Credit card fraud detection using convolutional neural networks. In *International conference on neural information processing*, pages 483–490. Springer.

Ghosh, S. and Reilly, D. L. (1994). Credit card fraud detection with a neural-network. In *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*, volume 3, pages 621–630. IEEE.

Gu, S., Kelly, B., and Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5):2223–2273.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Kanungsukkasem, N. and Leelanupab, T. (2019). Financial latent dirichlet allocation (finlda): Feature extraction in text and data mining for financial time series prediction. *IEEE Access*, 7:71645–71664.

Koren, Y. (2009). The bellkor solution to the netflix grand prize. *Netflix prize documentation*, 81(2009):1–10.

Linden, G., Smith, B., and York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80.

Loh, W.-Y. (2011). Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(1):14–23.

Medsker, L. R. and Jain, L. (2001). Recurrent neural networks. *Design and Applications*, 5:64–67.

Montgomery, D. C., Jennings, C. L., and Kulahci, M. (2015). *Introduction to time series analysis and forecasting*. John Wiley & Sons.

Patidar, R., Sharma, L., et al. (2011). Credit card fraud detection using neural network. *International Journal of Soft Computing and Engineering (IJSCE)*, 1(32-38).

Rahman, A. and Khan, M. N. A. (2018). A classification based model to assess customer behavior in banking sector. *Engineering, Technology & Applied Science Research*, 8(3):2949–2953.

Schafer, J. B., Frankowski, D., Herlocker, J., and Sen, S. (2007). Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer.

Sorokina, D. and Cantu-Paz, E. (2016). Amazon search: The joy of ranking products. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 459–460.

Appendix

A Additional Tables

Table A.1: categorical features summary

	Count	Percent
None	30,409	43.85%
24	6,546	9.44%
26	4,993	7.20%
22	4,766	6.87%
28	3,599	5.19%
Other (26)	19,027	27.44%

Table A.2: categorical features summary

	Count	Percent
Groandel (ohne Handel m	7,695	11.10%
Grundstcks- und Wohnungs	5,806	8.37%
Mit Finanz- und Versicher	5,121	7.39%
Maschinenbau	3,707	5.35%
Energieversorgung	2,279	3.29%
Other (86)	44,732	64.51%