Machine Learning in Finance: Forecasting purchase behavior for banking products

Viktor Reif

January 26, 2023

Abstract

abstract text ...

1 Research Question

- states and explains the research question
- states how I want to answer the research question
- a first draft:

The research question is how to model the purchase behavior of complex agents. This paper analyzes this question by implementing and evaluating several approaches for a product recommender system for corporate clients in banking. Each approach is a model with an estimation method, which predicts a client's demand for aggregated sets of financial products to yield product recommendations. This paper introduces an analysis framework with globally defined input features and evaluation metrics. This framework compares the approaches to identify the most adequate for complex purchase behaviour forecasts.

My research question is how to model the purchase behavior of complex agents. I want to analyze this question by implementing and evaluating several approaches for a product recommender system for corporate clients in banking.

2 Literature/Motivation/Introduction

- explains why my research question is relevant
- gives overview what results other authers have in similar research questions
- gives brief overview of my results

(why rq relevant) Knowing what your clients want to purchase next is an immensely valuable information. Private banks invest a lot of resources into their sales departments in an effort to be in touch with their clients and thus recommend them the right products. This is cost intensive (quote somehow how much). Especially for corporate clients predicting and recommending financial products is a difficult endeavour. Banks' solution to the problem always used to be use of human intelligence. For instance Commerzbank AG employs more than 1000 (is it good to mention this?) corporate client consultants. Each consultant is supposed to guess their clients' purchase behaviors guided only by their expertise and experience. With the rise of machine learning methods in all thinkable tasks, also many banks start to automate processes that used to be ruled by human guesswork. The added value from sucessfully implementing a machine learning algorithm that is capable of predicting client purchase behavior is two-fold. On one hand the bank can increase the degree of automation in its sales processes, which yields great cost benefits. On the other hand, the results contribute to pushing the development of applied machine learning in general.

(similar results I: ML in finance) In banking, publications have shown that machine learning algorithms can be sucessfully applied in several areas. One use case is credit card fraud detection. Ghosh and Reilly (1994) and Patidar et al. (2011) are two of many papers showing successful use of ANNs to detect fraudulent behavior in banking. Ghosh and Reilly (1994) find that an ANN outperforms classic, rulebased fraud detection procedures. Patidar et al. (2011) show that their neural network tuned with a genetic algorithm

reaches performances close to human intelligence level performance. Another use case is customer churn prediction. Xie et al. (2009) are sucessfull with random forests Rahman and Khan (2018) and find that the ANN is the best performing estimation method. In the same paper Rahman and Khan (2018) use the concept of Recency (time between transactions), Frequency(number of transactions in time period) and Monetary Value (volume of transactions in time period) to consolidate the input transaction data before predicting. They find that representing customer behavior in this consolidated form improves predictive performance. Bahnsen et al. (2016), who also try to identfiy the best estimation method for predicting credit card fraud, also utilize an transaction data aggregattion strategy, that is related to the concept of Recency, Frequency and Monetary Value (RFM), in their feature engineering. They split the transaction time series into periods and use the a special continuous probability distribution (von Mises Distribution) to consolidate the periods respectively (correct?).

(similar resutls II: other product recommenders) Barreau (2020) predicts bank clients' purchase behavior of financial instruments like bonds, options and futures, with several machine learning algoriths. This is the banking related contribution, that most closesly resembles a product recommendation system. Outside of banking, there already exist well established recommender systems. Especially firms which rely heavily on ecommerce, like for instance Amazon, Netflix, Booking.com, etc., have a high demand for adequate recommender algorithms. The kaggle competition funded by Netflix in 2006, which shook the domain of recommendation algorithms, is a fitting example for this demand. Bennett et al. (2007) explain that the competition strongly pushed the development of new recommender systems. They note, that most competitors' approaches are related to the collaborative filtering approach. As explained by Schafer et al. (2007), the collaborative filter algorihtm clusters agents by similarties in their characteristics or their purchase behavior and then recommends products given that cluster. The cluster problem within the algorithm possess as lot of depth since it lacks any boundaries. There no pre defined groups for clustering, which make the problem an unsupervised machine learning problem. Also, each approach can define and compute similarity on its own accord. Moreover, researchers came up with effective feature engineering measures before clustering to enhance their recommender systems. As this algorithm is so essential for recommendation engines, it is no surpirse that a collaborative filtering solution won the competetion.
Koren (2009) presents his team's winning solution. It is a collaborative filtering architecture, in which nearly every step is computed by one or a chain of machine learning algorithms, like gradient bossted decision trees and generative stochastic artificial neural networks. Apart from Netlflix, Bernardi et al. (2015) implement a collaborative filter approach for the ecommerce of booking.com. Sorokina and Cantu-Paz (2016) use natural language processing methods for feature engineering to enhance Amazons collaborative filtering algorithm. Covington et al. (2016) propose a recommender algorithm that solves collaborative filtering steps via deep learning. In their approach user similartiy is computed by deep learning and recommendable products are indetified via a multiclass classification model, which is also estimated by deep learning. They apply their model sucessfully to recommend Youtube videos.

(similar results III: using a framework to work the rq) Most papers concerning machine learning work empirically. It is very common to propose a new model or a new variant and compare it to one or more established models. For example Patidar et al.

(2011), Xie et al. (2009) and Rahman and Khan (2018) do this for churn and credit card fraud predictions. These papers use simple statistical metrics, such as precision, f1 score or area und curve (AUC) to evaluate and compare the proposed methods. Regarding metrics for comparison, the paper by Barreau (2020) is interesting because it uses a custom cost function. Barreau (2020) proposes compares how well collaborative filtering with matrix factorization, gradient boosted tree models, and other models to predict purchase behavior of financial instruments like bonds, options and futures. In the paper, he defines a cost function, which quantifies the impact of each correct or false prediction, to compare the methods. Also Bahnsen et al. (2016), who try to identfiy the best estimation method for predicting credit card fraud, compare different methods by a custom cost function given the confusion matrix. Also outside of banking, it is common to set up a comparative framework while proposing a new method or model. For instance Gu et al. (2020) do that to analyze which machine learning algorithm is the most adeqaute for asset pricing.

(my results) We suspect that predicting the purchase behavior of corporate clients is difficult, because in this use case products and clients are more complex than for example at Netflix. Therefore approaches that work for easier use cases, are expected to perform poorly in our use case.
We propose a new approach which is a panel data model estimated by an sophisticated variant of a recurrent neural network to predict purchase behavior. Also, we propose two baseline models. One model is inspired by prior approaches to predict churn and credit card fraud in banking and is estimated by a decision tree. The second model is a basic version of an user based collaborative filtering algorithm. We compare the approaches by the metrics overall precision, per product precision and the preciesion-recall curve. The panel approach clearly outperforms the two baseline models.

3 Data

- describes shape, dimensions and contained information of the dataset (summary statistics)
- explains granularity
- explains how I create/preprocess the dataset

As a side note: the product universe for corparate clients is diverse and large. That is why even after aggregating the products each client only ever demands a small part of all product groups. This creates sparse target behaviour vectors. With the same conditions set up within the framework for all approaches, I can compare and draw conclusions about the relative performance of the approaches.

(Product usage length differs across products and for certain products may also vary across clients. Usage length across products might be problematic for modelling purchase behavior, because ???. Different usage lengths across clients for the same product, however, are potentially rich features for modelling. To respect both issues, we use the information of used products as described before as features and the "delta" of the same information for the target variable. Delta means that only positive changes in product usage, ie the month a client starts using a product after not using it before, are considered.)

(mfkpi grundstuktur,move to gepkenn, into mfpkitarget, dann dass features dazu.

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

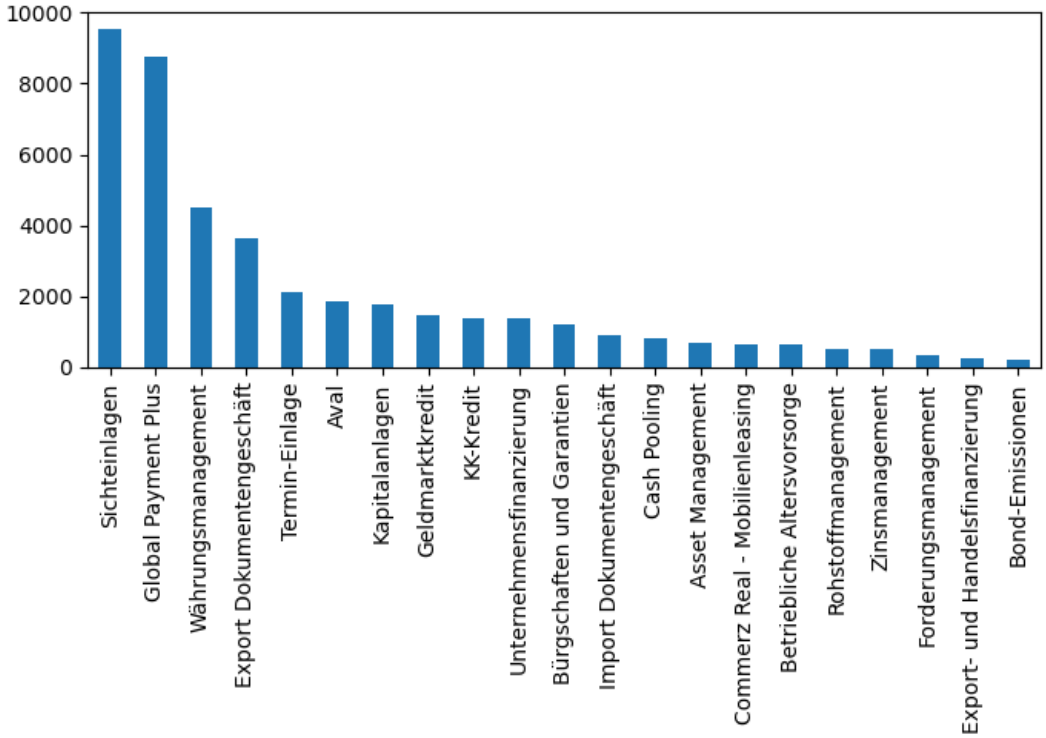

Figure 1: Figure X: Confusion matrix.

Table 4: Visualization Time Window Logic

	Slice 1	Slice 2	Slice 3
201901	feature		
201902	feature		
201903	feature		
201904	feature		
201905	feature		
201906	feature		
201907	feature	feature	
201908	feature	feature	
201909	feature	feature	
201910	feature	feature	
201911	feature	feature	
201912	feature	feature	
202001	feature	feature	feature
...			
202012	feature	feature	feature
202101	target	feature	feature
202102	target	feature	feature
202103	target	feature	feature
202104	target	feature	feature
202105	target	feature	feature
202106	target	feature	feature
202107		target	feature
202108		target	feature
202109		target	feature
202110		target	feature
202111		target	feature
202112		target	feature
202201			target
202202			target
202203			target
202204			target
202205			target
202206			target

the test data for the models.

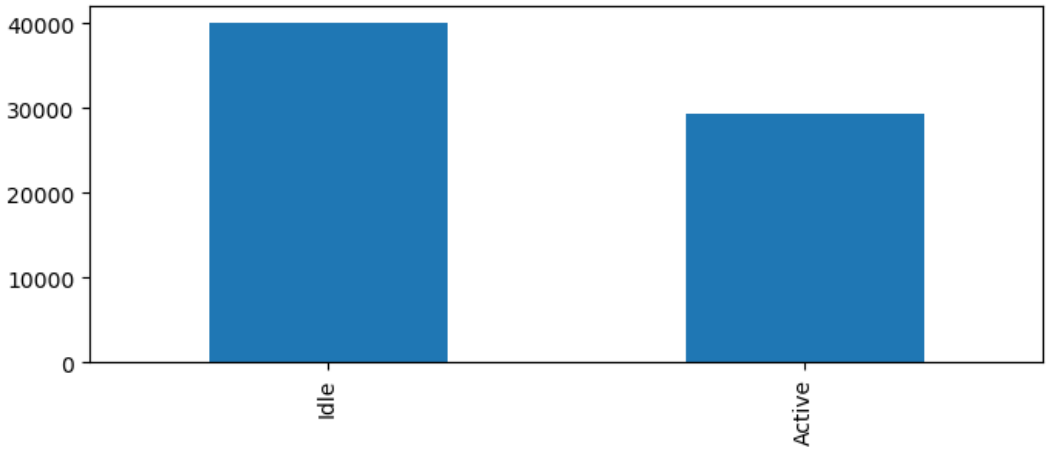

Figure 2: Figure X: Confusion matrix.

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

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

The assumptions (14) and (15) are necessary to keep outputs comparable. If a time-less product recommender algorithm recommends a certain product group, we assume the algorithm's output to a forecast. To make these assumption more likely to hold is another motivation to use aggregated purchase behaviors as tatget variables. (6) can also be interpreted as an effort to yield more meaningful recommendations. The intuition is that in our use case a clients desire to purchase a product group may last longer than one time unit of t. We choose $Q = 6$, meaning that a target product group has the value 1 if a client purchases a product from this product group at least once within the next 6 months.

[metrics] (all oos: prec as micro/macro/sample averages (explain why which average is the one we use.) and per product, PR-plot with same avgs and products, sonst tbd, evtl binary acc. weil sehr leicht zu interpretieren) Also, the framework evaluates all approaches' feasibility by the same metric(s). We prefer metrics with an emphasis on outcomes in which a client buying a product. Using these metrics in our use case mitigates the issue of sparse data in the target behavior vectors. Figure xx (below) shows all possible outcomes a single predicton of any proposed model.

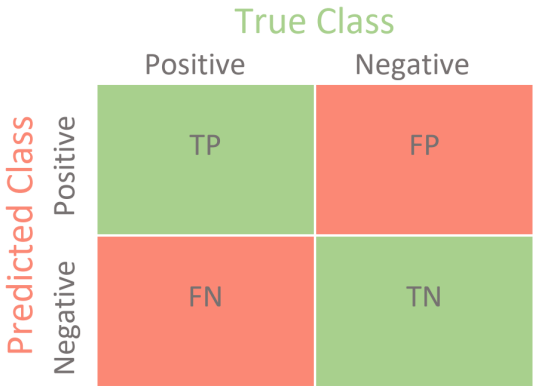

Figure 3: Figure X: Confusion matrix.

"Positive" (P) means in our setting that a client demands a certain product group. "Negative" (F) is the opposite. "True" (T) means that the classified/predicted label equal the real label. "False" (F) works vice versa. For example a client purchases the product group "Cash Pooling". A model predicts that the client wil not purchse that product. The actual value of the label is positive, but the predicted outcome is negative. Hence, the given result is a "False Negative" (FN). The other possible results TP, FP and TN work analogously.

One relevant metric is accuracy. This metric is computed as (TP+TN)/(P+N). In the context of a multilabel problem, (quote here) show that binary accuaracy is the most straigtforward choice. Binary accuaracy computes the accuarcy for each element of the target vector separately and then averages elementwise within the target vector and across all target vectors simultaninously. In terms of statistcs, thanks to its simplicity this metric is by default a good evaluation standard. However, the advantageous simplicity renders the metric blind to certain edge cases. One of these cases is phenomenon of (???). This is highly likely to occur with sparse data (quote!!!). Therefore, more sophisticated metrics are needed to safely consider this metric. Apart from statistical motivation, also domain knowledge edges us towards other metrics (maybe here talk more about the recommendation contest. Yet, this is also part of the threshold discussion for the p-r tradeoff).

Two relevant metrics are precision and recall. Precision describes how often a model correctly classifies a true label as true relative to all true labels??. As a formula this translates to TP/P. In this usecase that means how often does the model predict a product category that the respective client buys in the future. Recall is interpreted as the percentage of all true labels that the model manages to identify as such. Hence, TP/P?. In this work recall represents the ratio of all of all in the future purchased product groups that the

model sucessfully identifies. A mix of both metircs is the F1-score with such formula...(). Precision and recall are computed for each label (product) individually. From these individual scores one can derive averages. The micro average works like this and the macro average works like that. Since this usecase contains unbalanced labels, the micro average yields the less distorted information about the models' performance. Clearly, both higher precision and recall are indicator for a superior model.
(pr curve valide metrikß eig eher nur für theshold optimization...)As discussed by (quote here) precision and recall are by defintion in conflict. Hence, a trade-off with relatively high precion is preferred over eg a very high precision and low recall. To further evaluate each model's trade-off, we use Precision-Recall-Curves. A classifier's output is generally some kind of stochastic? distribution between zero and one.

$$show = distribution... \tag{14}$$

Proxy interpretations of this distribution are either how likely according to the model this unknown label is true or how "sure" the model is that the label is true. A threshold value between zero and one is used to turn each value of the distribution into firm predictions of positives and negatives. The standard value for the threshold is 0.5. Increasing the value turns more of a model's output into negatives and vice versa. Eg for a treshold of 0.8 the model has to be at least "80 percent confident", ie very sure, that the given label is positive, to be accounted as such. The PR-curve shows the respective precision and recall value for each possible threshold value. We consider the PR-curve of each product group individually as well as the macro average.

My set of approaches to forecast client behavior consists partly of models that are already established in the literature. These models work without the time dimension. One potential model is from the familiy of churn/credit fraud detection models, which leverage Recency, Frequency and Monetary Value (RFM) variables. Another is a static product recommender as implemented by Amazon and Netflix. This approach simply clusters clients by timeless features for predicting relevant products. "Other users also bought []..." is the logic here. The other part of my set of approaches is more heterodox. I developed the "time dynamic naive approach". This approach simply includes all available features in all available dimensions (*it*) without any kind of aggregaton or mining steps. This is the only approach I started to implement and first results with an LSTM estimation method look promising. Another heterodox approach is the time-series data model latent dirichlet allocation.

4.1 Netflix Approach

- explains this model/method
- explains why adequate for this problem

[approach short explanation] The concept of collaborative filtering is to find structural similarities in the conctext of product recommendation or purchase behavior analysis. We use user based collaborative filtering. This approach finds similar individuals by product usage. It then recommends each individual products by checking what similar individuals bought beforehand.

[feature engineering, features] In this default form, the approach only takes product usage as a feature. Hence the approach models future product usage by past product usage (maybe mention here explicitly that also other individuals product usage is a feature). This can be formulated as

$$\overrightarrow{\hat{aggBehavior}_{i,t+p}} = \alpha(\mathbf{C} \circ \mathbf{F}) \tag{15}$$

where $\alpha(\mathbf{C} \circ \mathbf{F}) = \mathbf{C} \circ \mathbf{F}$, meaning no prior feature engineering takes place. $\mathbf{C}$ contains weights which here select the last 6 months up to forcast moment $t + p$ and excludes all features except product usage.

[estimation method] The first step of collaborative filtering is to transform the feature space, which in our case is the information which client bought which products within a given time frame, into a product usage matrix. The approach uses the transformation function

$$\beta(\mathbf{C} \circ \mathbf{F}) = \max_{6}^{q=1}(\mathbf{C} \circ \mathbf{F})_{i,t-q} \tag{16}$$

to create

$$\mathbf{U} = \beta(\mathbf{C} \circ \mathbf{F}) \tag{17}$$

where $\mathbf{U}$ is the prodct usage matrix. This two-dimensional matrix contains all N available product groups along on axis and all I given clients on the other. Every element is 1 if the client of this row purchased the product group of this column, and 0 elsewise.

$$\begin{bmatrix} 0,0,1,...,0,1,0 \\ 0,0,0,...,0,0,0 \\ 0,0,1,...,1,1,0 \\ ... \\ 0,0,1,...,0,0,1 \\ 0,0,0,...,1,0,0 \\ 0,1,0,...,0,1,0 \end{bmatrix} \tag{18}$$

(21) shows example values for $\mathbf{U}$. The first row shows the client $i = 1$, the second $i = 2$ and so on until the last row at $i = I$.

The approach uses a similarity function $s()$ to compute similarity scores to quantify the clients' similarity to other clients. This similarity score quantifies how similar one clients's row in the product usage matrix to another client's row. We formulate this as

$$\overrightarrow{similarity_i} = s(\mathbf{U}) \tag{19}$$

where $\overrightarrow{similarity_i}$ contains i vectors of length $J, (J = I)$. Each element of a vector of one client i contains the similarity score between client i and another client j, $j \in 1, ...J$. Hence, the shape of the set of similarity score vectors is $(I * I)$. We use the pearson correlation coefficient to measure similarity. Wooldridge (2015) describes the coefficient as

$$Corr(x, y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2(y_i - \overline{y})^2}} \tag{20}$$

where the nominator is an estimate of $Cov(x, y)$ and the denominator is the product of the estimates of $Var(x)$ and $Var(y)$. In our setting $s() = Corr(x, y)$, the variable x represenst

$\overrightarrow{similarity_i}$ and $y = \overrightarrow{similarity_j}$ $(i, j \in I)$. As other similarity measures, correlation ranges between $[-1, 1]$. 1 means perfect correlation between two vectors, which translates to exact similarity between two clients' aggregated product usage in the given time frame. 0 indicates that clients' product usages are independent of each other. The higher the value between 0 and 1, the higher the correlation or similarity. For value lower than 0, a lower value indicates that two clients are more systematically different than only being independent. That means client i purchased products that client j did not and vice versa. At -1 clients are perfectly negatively correlated, meaning they are as different as possible in their product usage. We choose correlation over other similiarity measures, like for instance cosine similarity, as it is the simplest method. The next step is applied to each client i individually. For example client $i = 0$'s similarity vector is

$$[1, -0.2, 0.8, ..., 0.6, -0.2, 0.5] \tag{21}$$

where the first element of the vetor is 1 because it measures the similarity between client $i = 0$ and client $j = 0$, which is the same client. Sorting (24) results in

$$[1, 0.9, 0.88, ..., -0.3, -0.45, -0.5] \tag{22}$$

, from which we take the L first clients. This subset contains the L clients with the most similar feature product usage relative to client $i = 0$. Then, we multiply the similartiy score with the product binary for each similar client and product respectively. This means for each of the first L entries is (25), we find the row of the corresponding client in (21) and multiply each binary of that row whith te entry in (25). The result an intermediary matrix

$$\begin{bmatrix} 0, 0, 0.8, ..., 0, 0.1, 0.7 \\ 0, 0, 0.5, ..., 0.2, 0, 0.7 \\ ... \\ 0.1, 0.2, 0, ..., 0.9, 0, 0 \\ 0, 0, 0.3, ..., 0.1, 0.6, 0 \\ 0, 0.7, 0, ..., 0, 0.1, 0.8 \end{bmatrix} \tag{23}$$

shaped $(L * N)$, where N is the total number of product groups. This intermediary matrix (26) is then averaged per product to get a recommendation vector, meaning we take the mean of each column. The result is a vector of length N containing the algorithm's certainty that the example client $i = 0$ could demand a product for each product. Continuing the example, the result vector might look as follows:

$$[0.05, 0.1, 0.65, ..., 0.15, 0.35, 0.85] \tag{24}$$

As product usage is a binary variable and the similarity score is defined to yield values in range of $[-1, 1]$, also all elements in the result vector will range from -1 to 1. To yield final recommendations/predidctions, we use an arbitrary threshold between $[-1, 1]$. Given an threhold value of 0.3, (27) turns into

$$[0, 0, 1, ..., 0, 1, 1] \tag{25}$$

, which contains the N final product recommendations for the example client ($i = 0$). Analogously, this process of averaging of the by the similarity score weighted product usage

vectors of the l most similar client is repeated for each client to yield recommendations for all given clients.

(talk about interesting feature of this approach that if L is very large, recommendations also consider what very different clients have bought in a negative way. Eg when different clients often buy a certain products, it is less likely that the example client will buy it)

[idle clients in netflix approach](In that case a core assumption of this technique is that all clients considered must have bought at least one product within the feature time window. There are two ways to handle these idle feature clients: either we hard code the algorithm no recommend nothing, or we force the algorithm to look for similar clients anyway. In the latter case most similar clients will be idle clients and - more importantly - almost idle clients. We can use those almost idle clients, ie client that only acquired one product, to recommend products. Obviously, recommendations will center around this single procuts that the almost idle clients have bought. Yet, both ways do not necessarily have to yield performance. It might be a viable assumption that idle clients remain either idle or buy hardly any new products.)

[why netflix approach adequate] (bc bewährt, parsimonous, einfach, schnell) This approach is adequate for the given problem as it is already strongly established. (Maybe find a statistic to show how many firms oä use this algorithm). Futhermore, collaborative filter is parsimonious both in concept and computationally.

4.2 Cross Sectional Approach

- explains this model/method
- explains why adequate for this problem

[approach short explanation] The general idea of this approach is to reduce the dimensionality of the input data in a meaningful way before estimating purchase behavior. For each client the input data is a panel, which by the feature engineering is consolidated into cross sectional data. Given the cross sectional input data, we use a decision tree classifier to predict purchase behavior.

[explains feature engineering and model] This approach takes all available client characterestics as features except market indicators. The feature engineering removes the time dimension from the input data. Given the feature choice and feature engineering, the theoretical model can be formulated as

$$\overrightarrow{\hat{aggBehavior}_{i,t+p}} = \alpha(\mathbf{C} \circ \mathbf{F}) \tag{26}$$

where

$$\alpha(\mathbf{C}\circ\mathbf{F}) = RFM(\mathbf{C}\circ\mathbf{F}) = R(products, transactions)+F(products, transctions)+M(transactions, liquidity) \tag{27}$$

$\mathbf{C}$ contains weights which select the last 12 months up to forcast moment $t+p$ and includes all features except market characteristics. (29) formulates how the approach leverages the concept of Recency, Frequency and Monetary (RFM) value to consolidate the feature matrix. We use RFM in a similar fashion as (quote) and (quote). In the concept, Recency

accounts for how many time steps ago the last change in a feature variable has occured. For the feature of product usage that would be the time difference in months between the end of the feature time window and the last time the client purchased the product. A low value in recency indicates that a product was bought a short time ago. For a transaction feature recency quantifies how many months ago the last transaction occured. If a product was not bought at all or a certain transaction type never occured, the value of recency is missing. Frequency reckons how many times a single feature changed. For product usage this means the number of times a certain product was newly bought within the feature time frame. Monetary value represent the mean volume of a variable. Eg the average liquidity in EUR over the feature time period. [rfm which features]Recency and frequency dimensionality reduction is applied to the features features product usage and all transaction features, those being currency and target industry of the transaction. Monetary consoldidation is used for the transaction features and client liquidity. Only RFM variables enter the model. This means all features in their raw form are omitted. Also, the client independet market features are excluded, as they only contain a time dimension, which after being consolidated would render these features empty in information.

[estimation method] The estimation method of choice is a decision tree. As described for instance by Loh (2011), its intuition is to stepwise split the training data into homogenous groups.

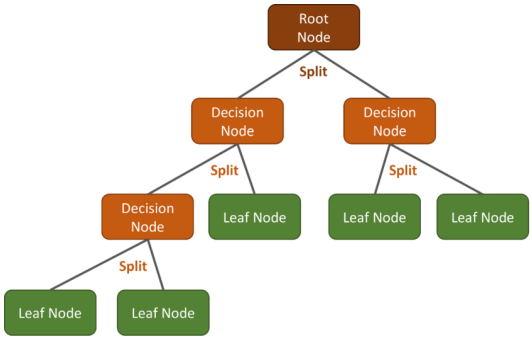

Figure 4: Decision Tree Classification Algorithm.

Figure XX visualizes the tree's algorithm. At each of the tree's decision nodes, i.e. branch forks, the algorithm decides how to split the data according to an impurity criterion, which measures group homogeneity. Each decision node has a binary condicional logic:

$$\begin{cases} 1, & \text{if condition is true} \\ 0, & \text{otherwise} \end{cases} \tag{28}$$

This could be for example

$$\begin{cases} 1, & \text{if } RFM(\mathbf{C} \circ \mathbf{F})_{k=x} > 20 \\ 0, & \text{otherwise} \end{cases} \tag{29}$$

, where the criterion is based on the feature x. An example criterion could be that at the node clients are split into two groups depending on whether their average liquidity, so $M(liquidity)$, is larger or smaller than $100,000 EUR$. Also, a combination of more than one feature simultaniously could serve as a criterion. The tree's depth and split size per node,

are among others the method's tuning parameters. In the classifier tree, each lowest node is called leaf node and yields a final classification output. A decision tree in its standard implementation handles binary- or multi-class, single-label problems. To handle the given binary class, multi-label problem we use as proposed by Vens et al. (2008) the single-label classification concept. This means each each binary class label of the problem is treated like an independent single-label classification task. The algorithm's result vector containing the multiple binary classes consists of the independently classified single labels.

[why cross section approach is adequate](cite here!) This approach is adequate as it utilizes an established concept of feature engineering. For instance Rahman and Khan (2018) and Fu et al. (2016) sucessfully use the concept of RFM in less complex banking related classification problems. The decision tree estimation method is adequate because it is well established and parsimonous in concept and implemenation. Also, as explained by Loh (2011) non-parametric. Unlike for example a linear regression, the decision tree is nonparametric and can by default control for nonlinear and interactive structures in data. Assuming the existence of relevant interactions in the given feat, this characteristic of the decision tree renders it a especially beneficial for our problem.

(if my implementation is SC (like eg MultiOutputClassifier): Vens et al. (2008) state that multilabel decision trees (Hierarchical multi-label classification) clearly outperforms separate single label decision trees, because ml dtres can learn intra-label dependencies. So this approach can account for cross dependencies in features, but as SC not for cross dependencies in labels. Only HMC could do that. If I dont use it, mention it in discussion. In general the approach 4.2. might have better estimation methods. I used that one because established, understandable, realistc to implement and parsimonous.)

4.3 Panel Approach

- explains this model/method
- explains why adequate for this problem

[approach, feature engineering, features, theor. model] This approach uses all available features along all available dimensions to find hidden patterns and thus yield as precise predictions a possible. Especially time dependent patterns might strongly enhance this approach's predictive performance. Methodologically, the idea is to leverage a complex estimation method which is especially designed to handle high dimensionality data and find trends, patterns and cross dependencies. The theoretical model of this approach can be formulated as follows:

$$\overrightarrow{\hat{aggBehavior}}_{i,t+p} = \alpha(\mathbf{C} \circ \mathbf{F}) \tag{30}$$

where

$$\alpha(\mathbf{C} \circ \mathbf{F}) = Panel(\mathbf{C} \circ \mathbf{F}) = 1 * (\mathbf{C} \circ \mathbf{F}) \tag{31}$$

and $\mathbf{C} = 1$. The weight matrix $\mathbf{C}$ selects all available features and n $\alpha()$ is an identity function, meaning no feature engineering takes place. Consequently, we can reformulate (31) as

$$\overrightarrow{\hat{aggBehavior}}_{i,t+p} = \mathbf{F} \tag{32}$$

, which states that the approach predicts future product usage by the full set of features in their full set of dimensions.

[estimation method: ANN] The estimation method of choice for (34) is a modified version of an artificial neural network (ANN). The ANN mimics the neuron/synapses structure of a biological brain. It has the following logic for the path from input to output.

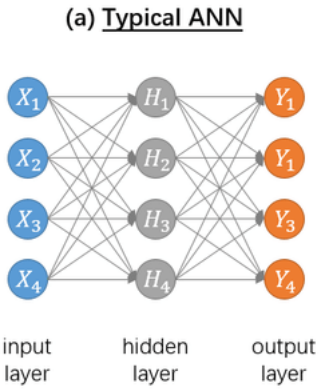

Figure 5: ANN architecture

In the input layer, each neuron connects via a synapse to every neuron of the next layer, which in the case of a basic feet-forward ANN is the first hidden layer. A synapse amplifies /mitigates via a respective weighting factor θ a signal and transports it from a single neuron to another of the next layer. All other hidden layers line up successively behind the first and their neurons receive weighted inputs via synapses from all neurons in the layer before. The last layer is the output layer, which merges all inputs (again via synapses) from the last hidden layer into output signals. These outputs could be classes or continuous values (Kubat 2017). In the context of mulitilabel classification, the neural network would have a vector of binaries as output (how is the topology of my RNN? would be nice here to mention how that output shape should be).

[estimation method: LSTM] (try not to go too hard into detail...) (what is a lstm, how is rnn motivated, how does rnn work, how is lstm motivated, how does lstm work)

(what is lstm)We use the Long Short Term Memory method (LSTM) proposed by Hochreiter and Schmidhuber (1997) to enhance the basic ANN estimation process. LSTM is a a sophisticated variant of a recurrent neural network (RNN), which in turn is a sophisticated variant of an ANN. Figure visualizes that all three considered architectures are equally shaped. They differ only in how the nodes in the hidden layers are connected.

(how is rnn motivated) A default ANN architecture is not adequate for this approach. The reason for this is, as noted by Medsker and Jain (2001), that the ANN is unable to handle certain patterns within the input data. Domains which often contain these patterns are for instance language learning, image classification and asset pricing. In language learning a problem could be to classify the topic of each chapter in novels. Clearly, in most novels there exist systematic patterns of chapters regarding their topics. More advanced architectures like the RNN handle this by using its results about previous chapters to classify the next.

(how does rnn work?) As noted by Medsker and Jain (2001), recurrent implies that

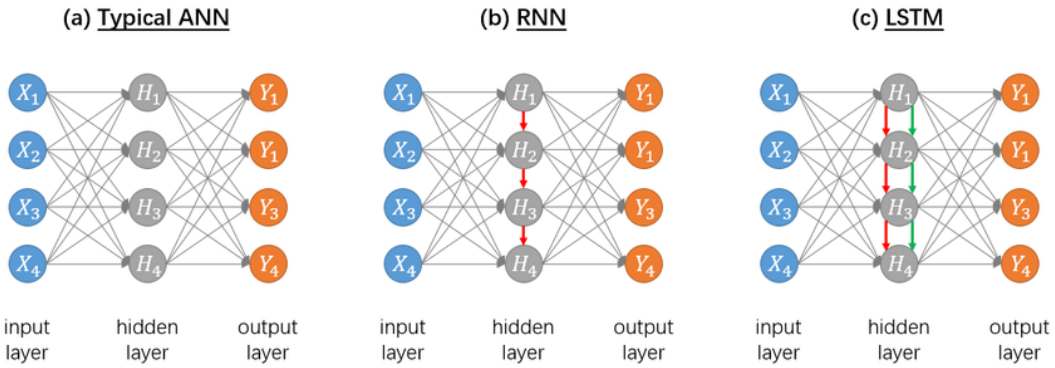

Figure 6: maybe exclude, because it does not really show the loops from and to the same node within a layer RNN

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.

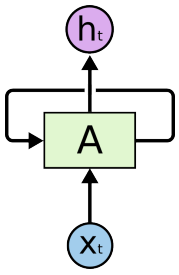

Figure 7: RNN-rolled

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.

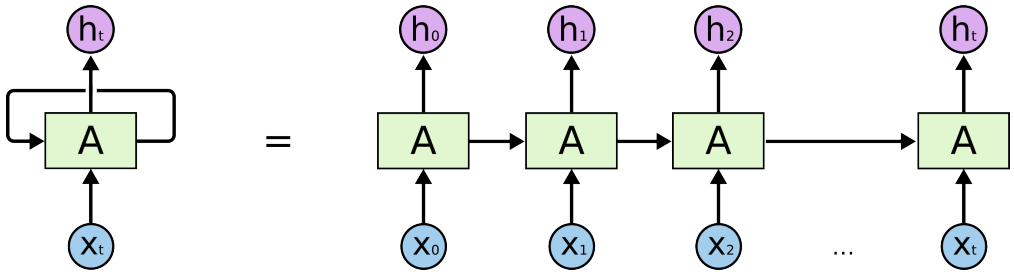

Figure 8: RNN-unrolled

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.

[why panel approach is adequate] We expect the panel approach to perform well because it uses the most inclusive theoretical model. Especially the inclusion of the time dimension enables the estimation method to find complex behavioral patterns and dependencies und thus yield better predictive results. Moreover, the approach works with out the use of a feature engineering function. We expect this characterist to also enhance the predictive quality of the approach, as any given feature engineering function implies a certain level of information loss. Also, the estimation method is the most advanced. (quote2012) and (quote...) have shown that the LSTM method performs well high dimensionality settings, as is our use case. At the same, we deliberately omit even more complex methods, like for example GRU and BERT, as the input data contains only 24 months along the time dimension. We argue that the input data is too short for more complex methods.

5 Empiric Results

- shows tables/graphs containing the result metrics defined in the framework
- explains what the tables/graphs mean. Eg "model1 outperforms model2 in correctly identifying true positives, which means model1..."

6 Discussion

- discusses meaning and implications of my results
- explains potential imitations in my analysis
- states avenues for future research

the role of tuning. Maximal margin of tuning is latent. Consequently, further tweaking the approaches' estimation methods might change their performance drastically. Further results might contradict our findings.

talk about trade off how complex input data and how complex estimation method.

7 Conclusion

- states main findings in relation to the research question
- states most important implications of results

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.

Vens, C., Struyf, J., Schietgat, L., Džeroski, S., and Blockeel, H. (2008). Decision trees for hierarchical multi-label classification. *Machine learning*, 73(2):185–214.

Wooldridge, J. M. (2015). *Introductory econometrics: A modern approach*. Cengage learning.

Xie, Y., Li, X., Ngai, E., and Ying, W. (2009). Customer churn prediction using improved balanced random forests. *Expert Systems with Applications*, 36(3):5445–5449.

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%