A recommender system based on customer interaction with an e-commerce website

Richard Hruby (17-619-172)
Johan Faxner (21-603-204)
Tim Matheis (21-603-907)
Giovanni Magagnin (17-300-914)

Thursday, 12th of May 2022. Big Data Analytics (8,727,1.00)

University of St.Gallen

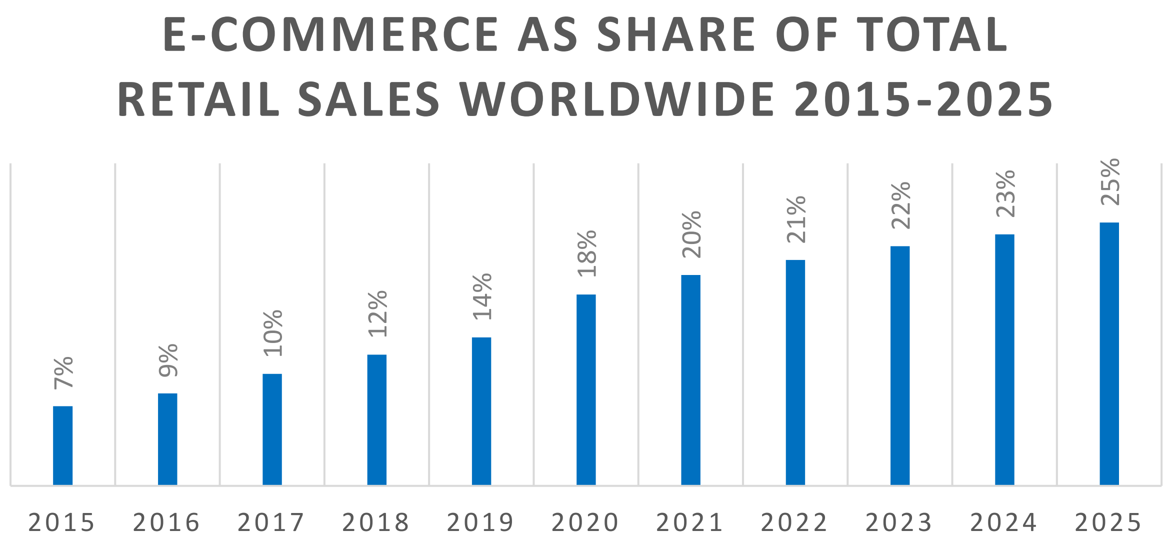
Motivation

The number of digital buyers worldwide went from 1.32b in 2014 to 2.14b in 2021 (Statista)

The availability of information increases the importance of systematic data collection, data processing, data analysis and implementation in various business and value chains

Several studies (inter alia: Accenture, 2018; PwC, 2019; Deloitte, 2019) indicate consumers' wish and demand for personalization

➔ **How well can we predict the purchases of an online shopper based on their previous shopping behavior as well as the purchasing behavior of other customers?**



Methodology

Recommender system

Subclass of Information filtering Systems that seeks to predict the rating or the preference a user might give to an item.

Requirements:

- A set of products and a set of customers who, in some way, have interacted with these products.

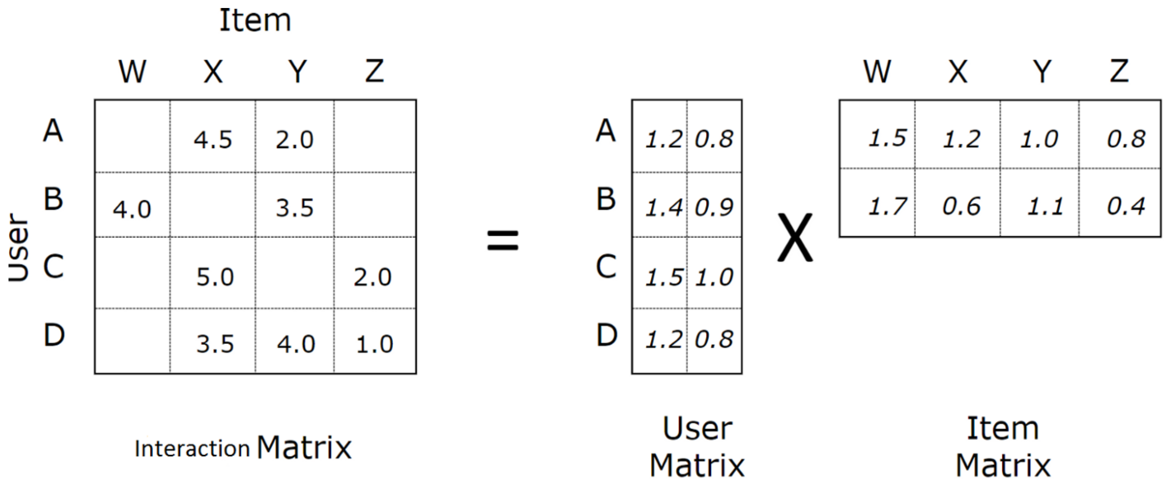
User-based collaborative filtering

- Past users' behaviour to generate future preferences
- We discover the relevant features based on patterns
- The features are not "human-based" they are the result of an algorithm.

Users	User-item interactions matrix	Items
suscribers	rating given by a user to a movie (integer)	movies
readers	time spent by a reader on an article (float)	articles
buyers	product clicked or not when suggested (boolean)	products

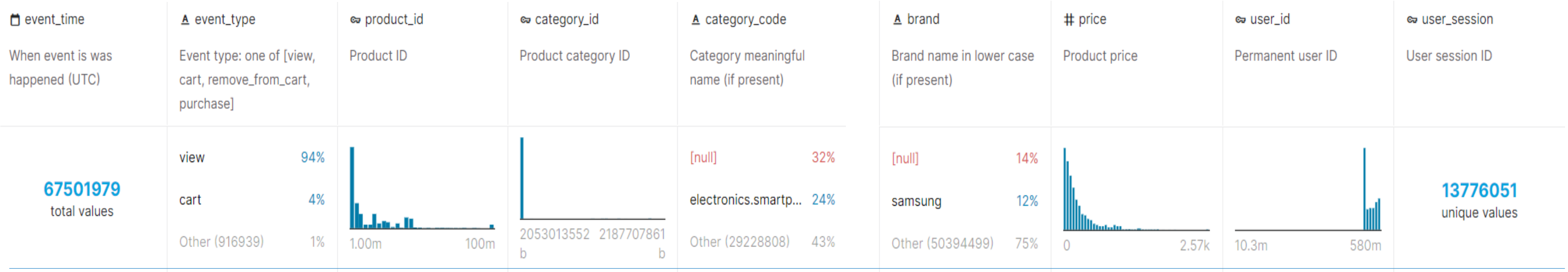
Alternating least squares (ALS)

- Suitable for larger-scale collaborative filtering problems
- ALS minimizes two loss functions alternatively
- ALS runs its gradient descent in parallel across multiple partitions of the training data



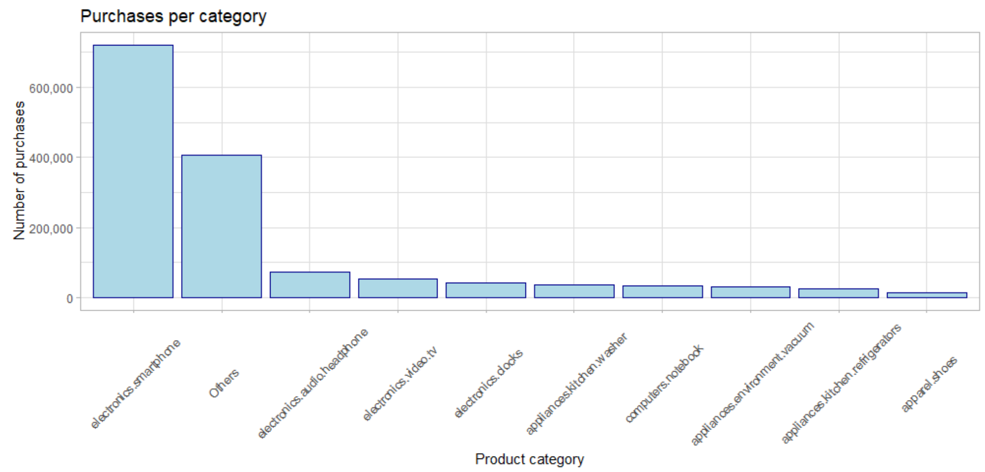
Data-set

We used a 15GB data-set with behaviour data for 2 months from a large multi-category online store from Kaggle. The data is organized in the following way:

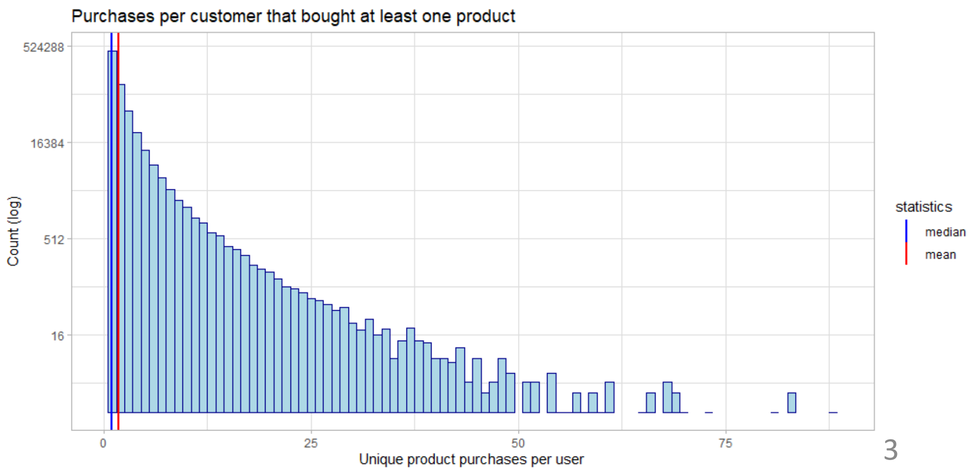


event_time	event_type	product_id	category_id	category_code	brand	price	user_id	user_session
When event is was happened (UTC)	Event type: one of [view, cart, remove_from_cart, purchase]	Product ID	Product category ID	Category meaningful name (if present)	Brand name in lower case (if present)	Product price	Permanent user ID	User session ID
67501979 total values	view 94% cart 4% Other (916939) 1%	1.00m – 100m	2053013552b – 2187707861b	[null] 32% electronics.smartp... 24% Other (29228808) 43%	[null] 14% samsung 12% Other (50394499) 75%	0 – 2.57k	10.3m – 580m	13776051 unique values

10 categories make up the most purchases in the set. Smartphones and "others" account for most sales.



Most active customers that buy at least one product, also only buy one product.

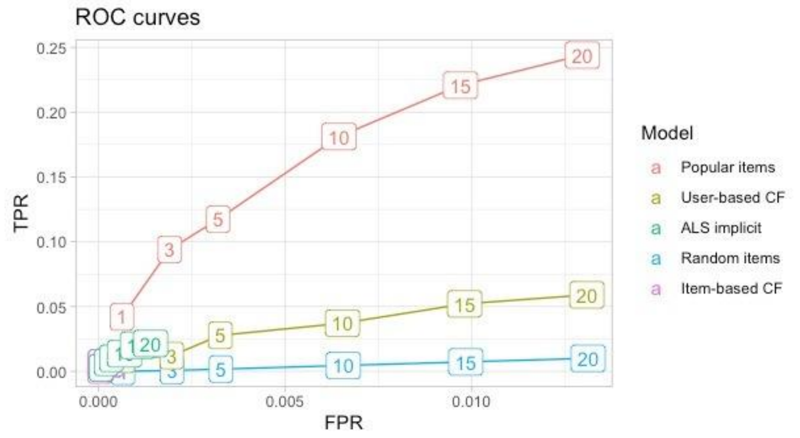


Implications of the big data set – Issues we faced while working with 15GB

Started with a subset of our data (200k rows):

- We used recommenderlab, R package that provides an infrastructure to test and develop recommender algorithms

➔ With the reduced data-set of 200k rows, it worked fine



We tried the same approach with the entire data, but found 2 bottlenecks:

- **Bottleneck 1:** User-item matrix grows exponentially

```
> # spread df, set non-existent values to 0 to use binary encoding
> matrix_data <- cosine_user_product_df %>%
+     select(user_id, product_id, number_purchases) %>%
+     spread(product_id, number_purchases, fill=0)
Error: cannot allocate vector of size 353.8 Gb
```

number of rows	memory usage
200k	35 MB
400k	124 MB
800k	362 MB

- **Bottleneck 2:** Larger user-item matrix implies slower predictions

```
> system.time(recom_topNList2 <- recommenderlab::predict(object = r2,
+                                newdata = tail(ratings_matrix, n_pred),
+                                type = "topNList",
+                                n = n_rec)
+ )
```

number of rows	time elapsed
200k	13.3 sec.
400k	41.5 sec.
800k	93.5 sec.

Big data techniques – How we solved the issues we were facing

SQLite

```r
# purchases per user of all products
products_purchased_per_customer <-
  "
  SELECT user_id, product_id, 1 AS purchased
  FROM ecom
  WHERE event_type IN ('purchase')
  ORDER BY user_id ASC;
  "

# use query to extract needed data
purchase_df <- dbGetQuery(ecom_db, products_purchased_per_customer)
purchase_df
```

- Shift memory load from RAM to SDD
- Split CSV file into tables in SQL
- Enable to query important information

Apache Spark

```r
# configure spark connection
config <- spark_config()
config$spark.executor.memory <- "8G"
config$spark.executor.cores <- 4
config$spark.executor.instances <- 3
config$spark.dynamicAllocation.enabled <- "false"

# initiate connection
sc <- spark_connect(master = "local", config=config,
                    version = "3.1")
```

- Distribute computationally expensive processes
- Migrate pipeline to SparklyR
- Enable scaling to the entire dataset

dplyr /doSNOW

```r
# Paralellize revenue calculation
ncores <- parallel::detectCores()
ctemp <- makeCluster(ncores)

system.time(
  output <- foreach(i = c(1, 3, 5, 10, 15, 20, 25), .combine = rbind)%dopar%{
    revenues_n = generate_revenue_df(i,predicted)
    rev_n = list(name = "ALS Spark", n = i, revenue = sum(revenues_n$revenue))
    }
)["elapsed"] #Parallelized calculation is slightly faster

# Unpack results to DF
rev_df = data.frame(output) %>% `rownames<-`(1:7) %>%
  unnest(cols = c(name, n, revenue))
```

- Speed up training and evaluation of recommender
- Joins and parallelized loops
- Enable efficient processing and real-time predictions

Performance of big data methods – Speed and efficiency improvements

Recommenderlab

Inefficient:

- Small rating matrix size (2872 x 1523)
- Slow prediction time: ~15 seconds for 1000 users
- Large memory allocation: used entire RAM on laptop

```
> bench::mark(
+   predict_fun(r2, ratings_matrix, n_pred, n_rec)
+ )
# A tibble: 1 × 13
  expression                                        min   median `itr/sec` mem_alloc
  <bch:expr>                                   <bch:tm> <bch:tm>     <dbl> <bch:byt>
1 predict_fun(r2, ratings_matrix, n_pred, n_rec)    15.4s    15.4s    0.0649    7.55GB
# … with 8 more variables: `gc/sec` <dbl>, n_itr <int>, n_gc <dbl>,
#   total_time <bch:tm>, result <list>, memory <list>, time <list>, gc <list>
```

Sparklyr

Efficient:

- Large rating matrix size (48187 x 9015)
- Same prediction time, but for a matrix 100 times the size.
- Small memory allocation: negligible RAM required

```
> bench::mark(
+   ml_predict(als_fit, spark_test_complete) %>% collect()
+ )
# A tibble: 1 × 13
  expression                                                min median `itr/sec` mem_alloc
  <bch:expr>                                              <bch> <bch:>     <dbl> <bch:byt>
1 ml_predict(als_fit, spark_test_complete) %>% collect() 15.1s  15.1s    0.0664    89.6MB
# … with 7 more variables: n_itr <int>, n_gc <dbl>, total_time <bch:tm>, result <list>,
#   memory <list>, time <list>, gc <list>
```

Results of the prediction – Our assesment

1. Top-N predictions for 2 sample users

		user_id	product_id	purchased	prediction	
		<int>	<int>	<int>	<dbl>	
User 1	1	512503671	1004856	1	0.490	n = 3
	2	512503671	1004767	0	0.447	
	3	512503671	1004833	1	0.385	
User 2	4	513997627	1005115	1	1.02	
	5	513997627	1004856	0	0.937	
	6	513997627	1004767	0	0.859	

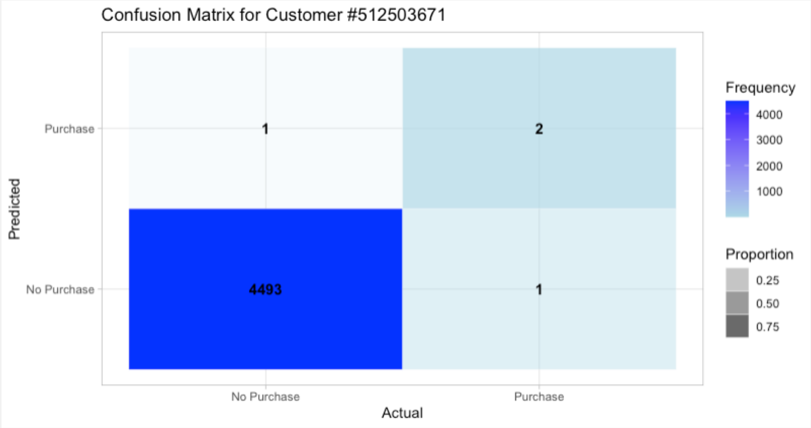
2. Evaluation Metrics per User

		user_id	actual_positives	actual_negatives	true_positives	false_positives	true_negatives	false_negatives
		<int>	<int>	<int>	<int>	<int>	<int>	<int>
User 1	1	512503671	3	4494	2	1	4493	1
User 2	2	513997627	1	4496	1	2	4494	0

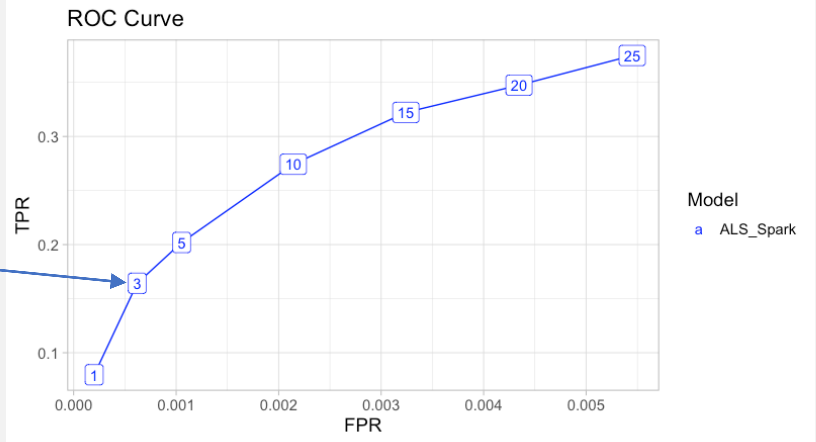
3. Average Evaluation Metrics for N 1:25

	name	n	TPR	FPR	precision	recall	conversion
1000 users	ALS_Spark	3	0.16422619	0.0006199274	0.07100	0.16422619	0.07100

This customer purchased 2 out of 3 recommended products



With 15 recommendations we capture 1/3 of the products purchased

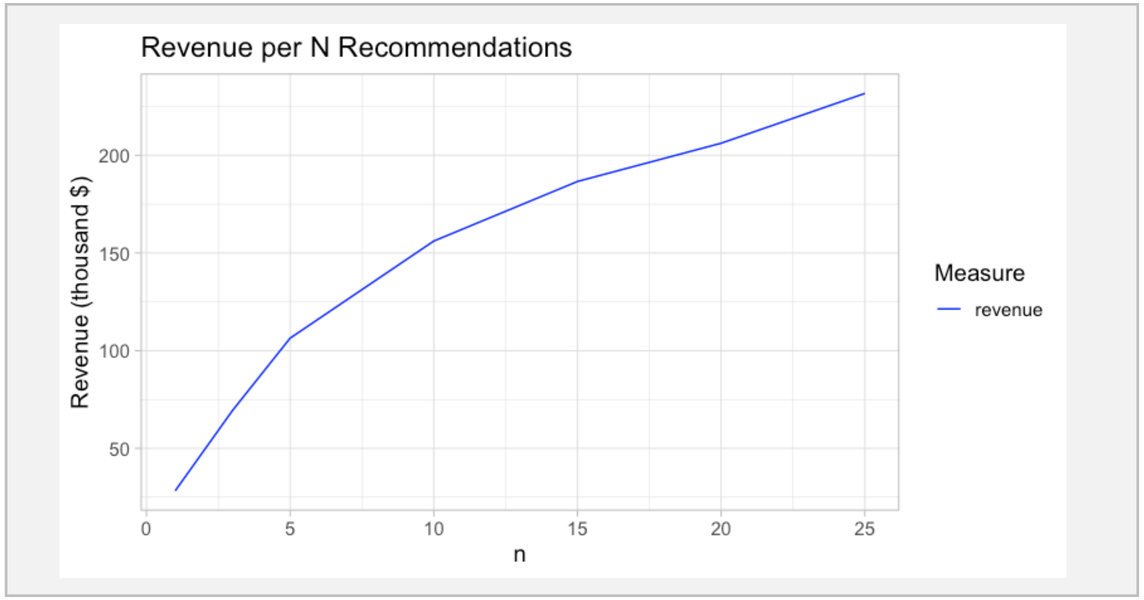


Business and economic implications

Business value

Real increase in revenues:

- With n (recommendations x user) = 5, the business has a net increase of >100k USD
- With n (recommendations x user) = 15, the business has a net increase of >200k USD



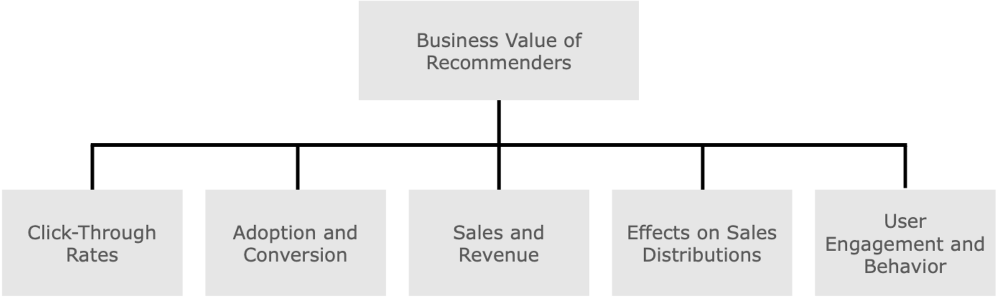
Economic implications

Positive welfare for sellers (Fleder and Hosanagar 2009):

1. Meeting the consumers' individual preferences
2. Intensifying the users' transactions and consumption time
3. Turning browsing consumers into buyers, cross selling, increasing customer loyalty

Positive welfare for buyers:

1. Transaction costs can be reduced, more precisely search costs
2. Solves the information overload problem



Source: Jannach & Jugovac (2019)

Limitations of our approach – What else could be done?

Limitations:

1. **Backward looking bias:** We test whether we would have recommended a product to a customer that actually bought this product. However, the «wrong» predictions could have also led to purchases when the product would have been recommended to the customer.
2. **Methods:** simplistic «market-basket analysis»
3. **Limited data:** only two months of data used for analysis
4. **Product/Customer differentiation:** We treat the products/customers as if they were equal. But it would be useful to include the characteristics of the products.

Future work:

1. **Measure real-life business value:** AB testing to figure out how well recommendations work
2. **Customer behaviour:** Including the other interactions of users for predictions (e.g. view, put in cart, and remove from cart)
3. **Scaling up:** AWS for more robust data pipeline and quicker computation
4. **Hybrid recommender systems:** Incorporate further information about products and users to increase relevance of recommendations

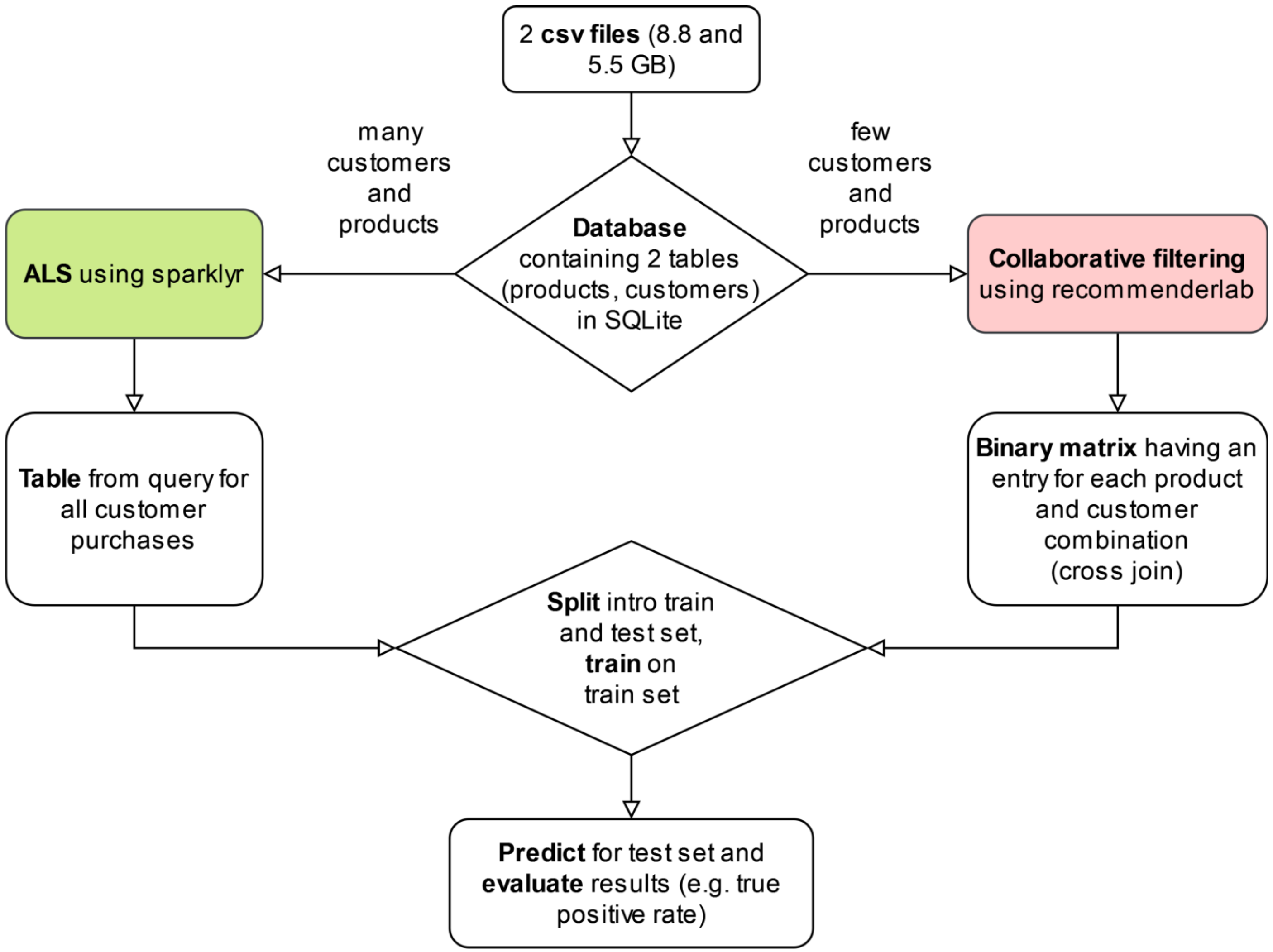
A recommender system based on customer interaction with an e-commerce website

Richard Hruby (17-619-172)
Johan Faxner (21-603-204)
Tim Matheis (21-603-907)
Giovanni Magagnin (17-300-914)

Thursday, 12° of May 2022. Big Data Analytics (8,727,1.00)

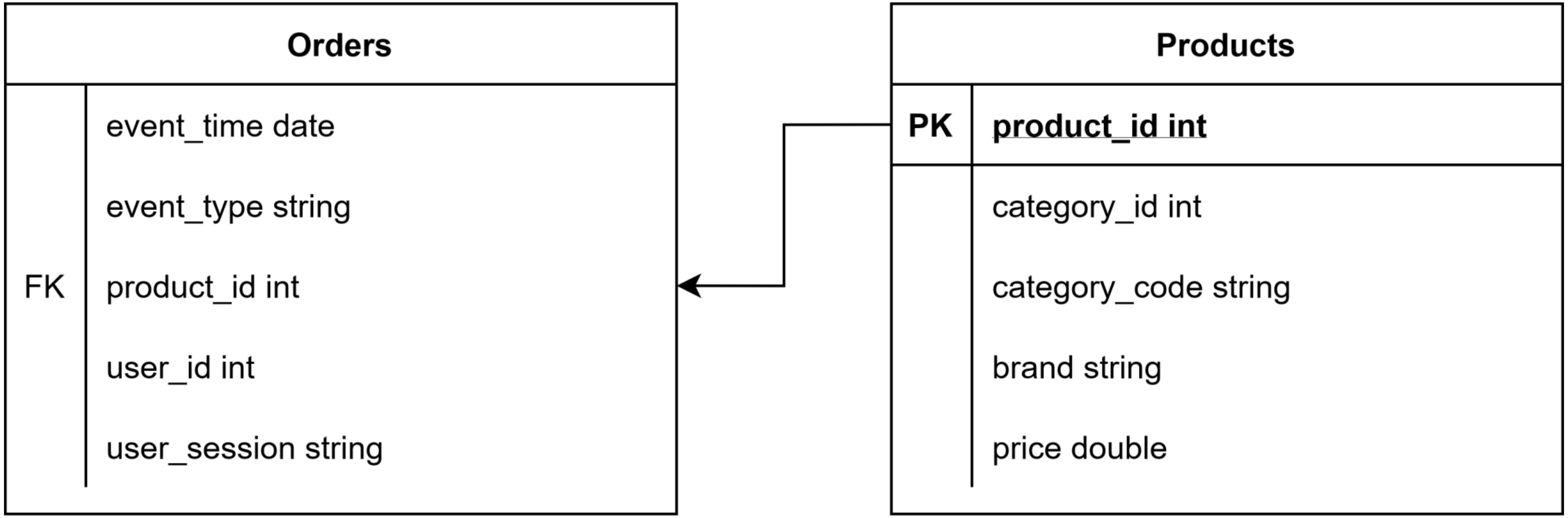
University of St.Gallen

Discussion

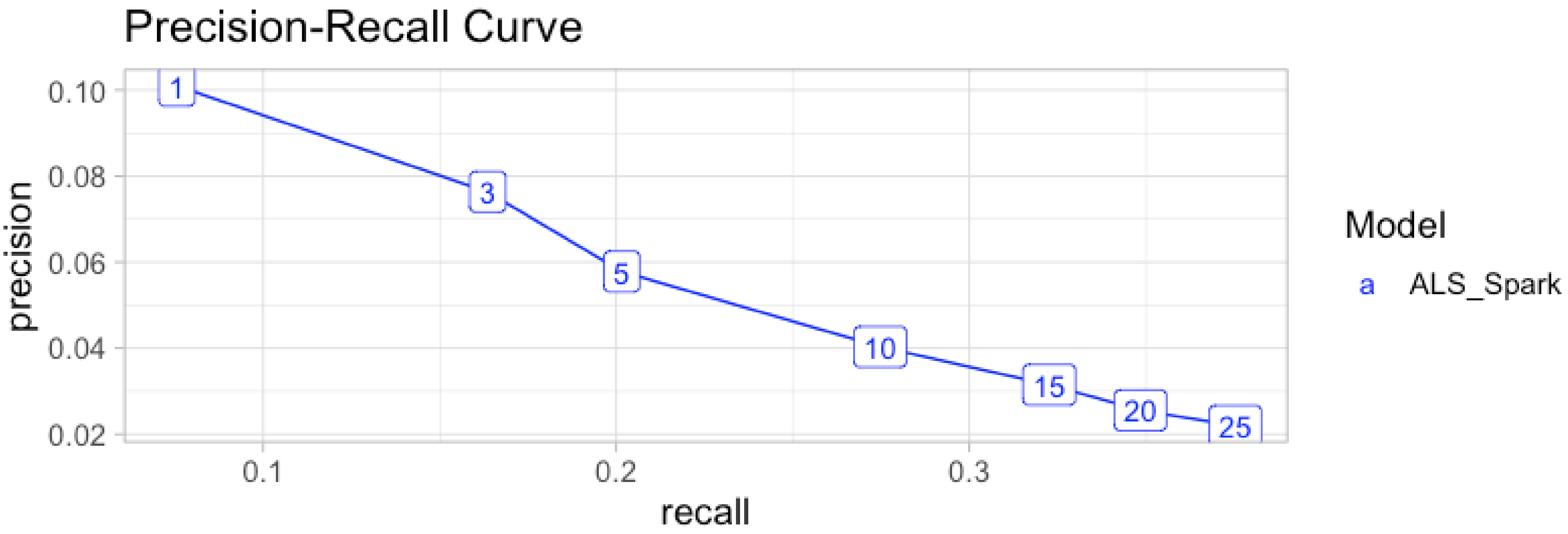


2 csv files (8.8 and 5.5 GB)
many customers and products
few customers and products
Database containing 2 tables (products, customers) in SQLite
ALS using sparklyr
Collaborative filtering using recommenderlab
Table from query for all customer purchases
Binary matrix having an entry for each product and customer combination (cross join)
Split intro train and test set, train on train set
Predict for test set and evaluate results (e.g. true positive rate)

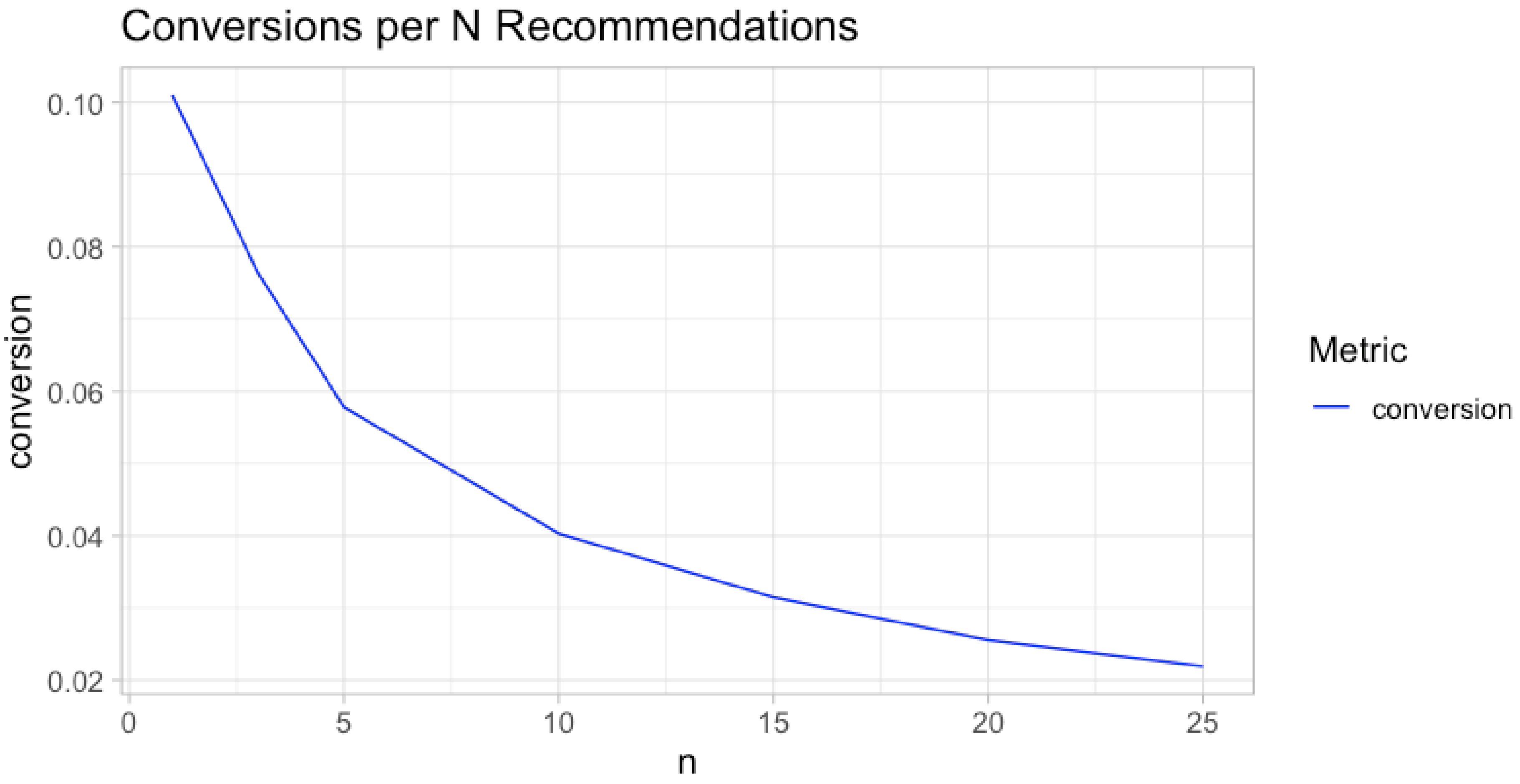
Appendix 1 - SQL



Appendix 2 – Precision Recall Curve



Appendix 3 – Conversions per N Recommendations



Appendix 3 – Top-N Evaluation N=1:25

	name	n	TPR	FPR	precision	recall	conversion	revenue
1	ALS_Spark	1	0.07966667	0.0001999695	0.10100	0.07966667	0.10100	28255
2	ALS_Spark	3	0.16422619	0.0006199274	0.07100	0.16422619	0.07100	69463
3	ALS_Spark	5	0.20195238	0.0010532321	0.05300	0.20195238	0.05300	106393
4	ALS_Spark	10	0.27434405	0.0021429452	0.03660	0.27434405	0.03660	156102
5	ALS_Spark	15	0.32221310	0.0032411125	0.02860	0.32221310	0.02860	186594
6	ALS_Spark	20	0.34742143	0.0043455085	0.02320	0.34742143	0.02320	206173
7	ALS_Spark	25	0.37484762	0.0054499049	0.01996	0.37484762	0.01996	231659

References

- https://gist.github.com/twolodzko/7becd98ff256ef826b56945de297700d
- https://utd-ir.tdl.org/bitstream/handle/10735.1/5485/ETD-5608-7403.99.pdf?sequence=5&isAllowed=y
- http://www.learconference2015.com/wp-content/uploads/2014/11/Calvano-slides.pdf
- https://www.econstor.eu/bitstream/10419/228752/1/174515275X.pdf
- https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada
- https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-2-alternating-least-square-als-matrix-4a76c58714a1