Optimizing Subscription Services with Uplift Modeling

Dmitry Zolotukhin, Chief Data Officer, START

Plan

1. Background of our Uplift task
2. Data
3. Data preprocessing
4. Model training
5. Results

Company and Business Model

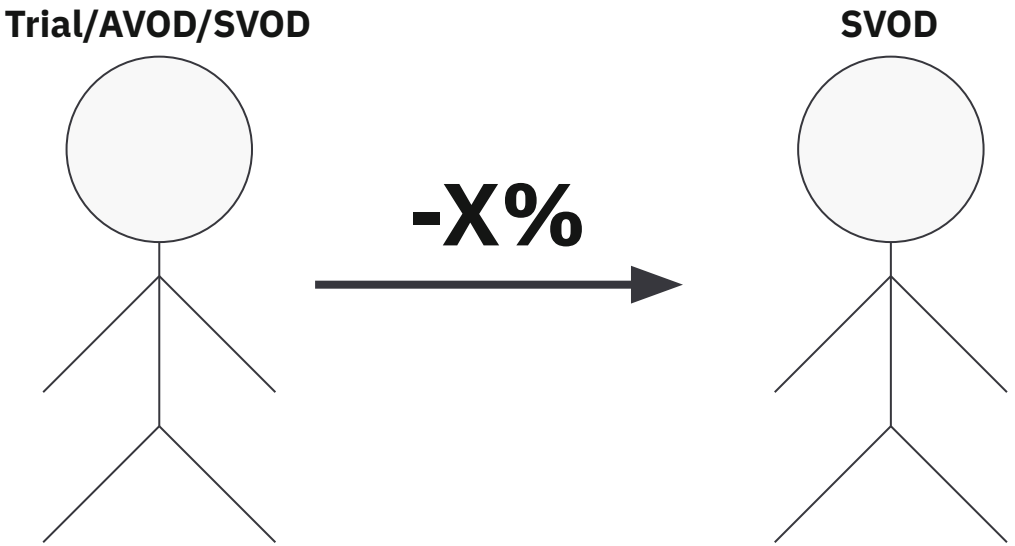
AVOD viewers watch ads, **SVOD** viewers buy a subscription

Task: Expanding Subscriber Base

How: offer discounts

Problem: don’t offer a discount to someone who would subscribe anyway

Solution: Uplift modeling



Uplift Modeling

Uplift = difference in revenue from a user when targeted vs. not targeted

Problem: You can’t both target and not target the same user!

Solution: Average Treatment Effect (ATE) — difference in revenue (or another metric) between test and control groups. This works because in A/B testing, users are randomly assigned to test or control

Requirements for the Uplift Model

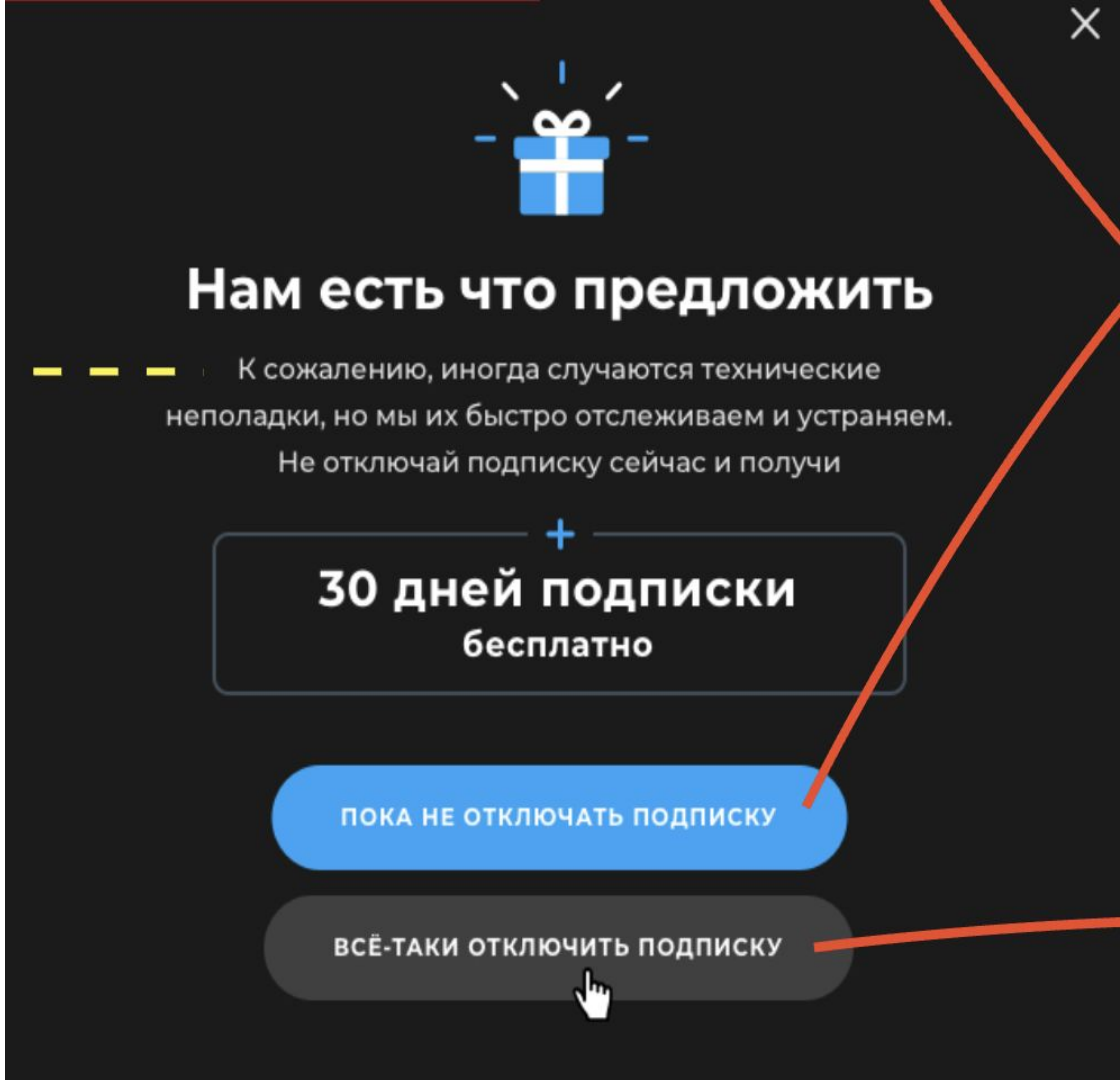
1. Stability
2. Interpretability

These requirements stem from:

1. The high cost of acquiring users → communication data is expensive
2. Movie/TV content is subject to elastic demand (frequent new releases)

Data

- Experiment MORE-2363 on users with a trial:
  - 17,000 users in the test group, offered a 30-day trial extension (handled by billing)
  - 17,000 users in the control group, no extension offered; auto-renewal disabled after users answered why they canceled
- We trained our first Uplift model on these data.



User Features

Number of active days

From the player:

- Number of watched projects
- Number of projects added to favorites
- Percentage of seasons completed
- Number of TV show views
- Number of movie views
- Number of successful auto-advances to the next episode
- Number of attempts at auto-advance to the next episode

From the project page:

- Number of times “Similar to this” was viewed
- Number of clicks on “Similar to this”

From search interaction:

- Average number of actions between a search and a watch
- Number of queries that yielded no results

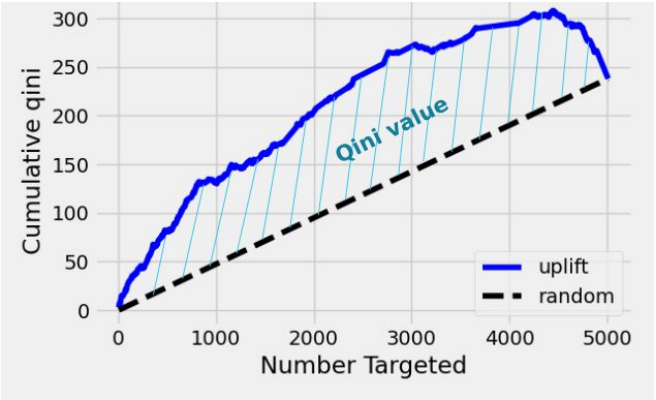
Data Preprocessing

Step 1	Step 2	Step 3
Split data into training and validation sets	Train the model	Check the validation score

What is the Score?

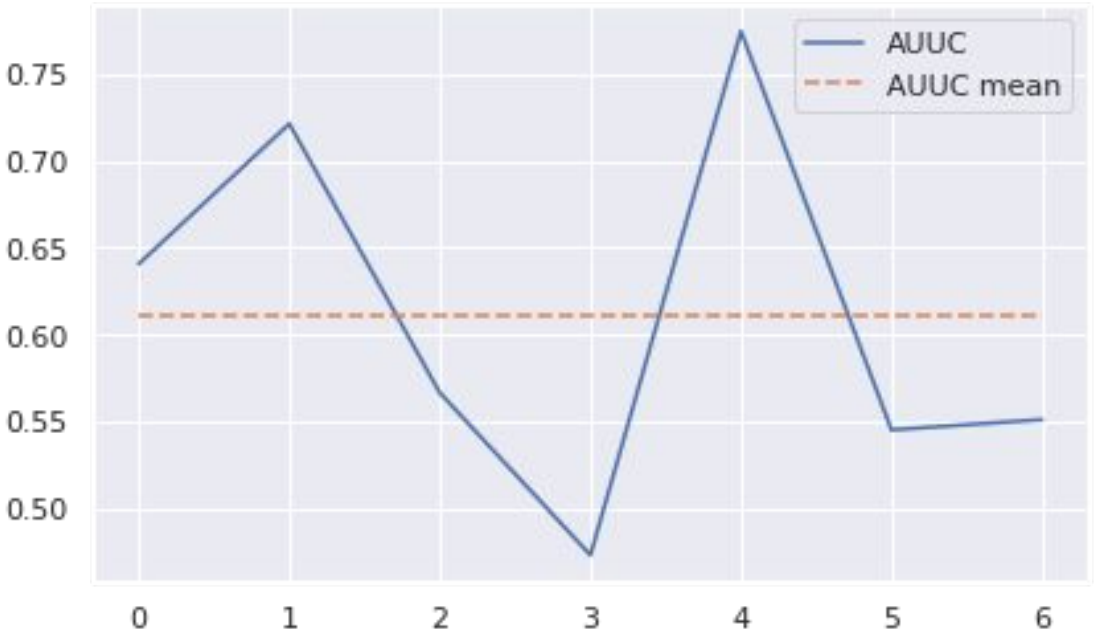
Qini is the difference in the share of successful conversions in test versus control.

Model Prediction	From Test Group?	Converted into Subscription?	Qini
0.939249	1	1	1.000000
0.939237	1	1	2.000000
0.939125	0	1	0.875000
0.938707	1	1	2.173913
0.936231	1	0	2.173913
…	…	…	…
-0.925874	0	1	40.266077
-0.938097	0	1	40.019571



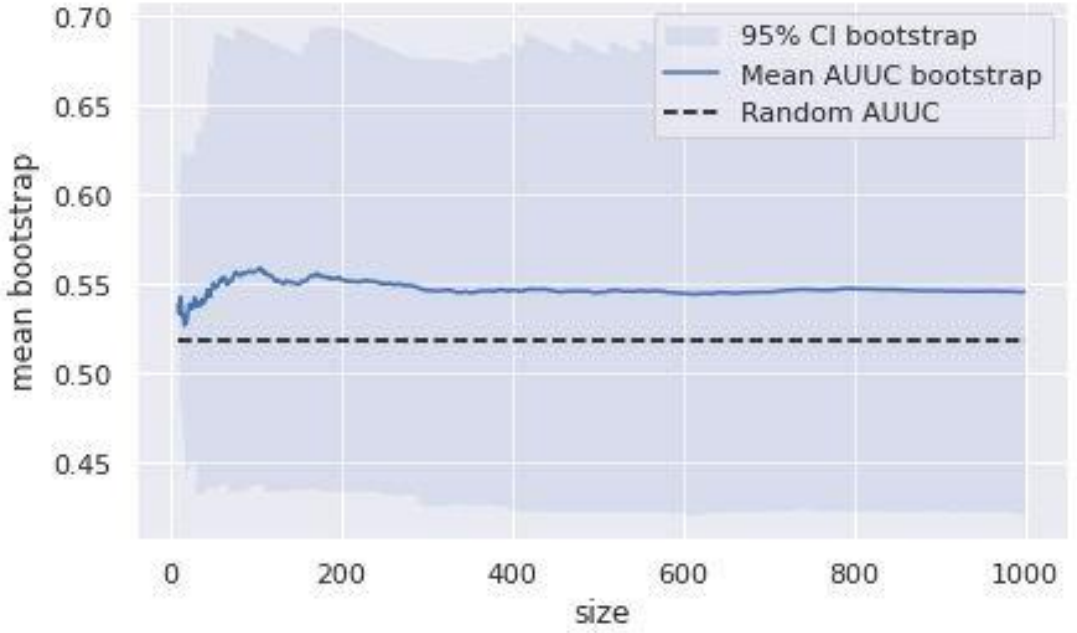
Data Preprocessing: Issues

No stability in quality: drastically different results with different random splits



Data Preprocessing: Issues

No stability in quality: drastically different results with different random splits



How to Fix It: Split by Registration Date

The oldest users (by registration date) go to train (80% of the total sample). The remaining 20% of more recently registered users go to validation

Train the model on the training set, get the score on the validation set

UpliftRandomForest

Train score: 0.338

Validation score: 0.330

Stratification Algorithm

Goal: ensure that the feature distributions in each subset are as close as possible to those in the overall population

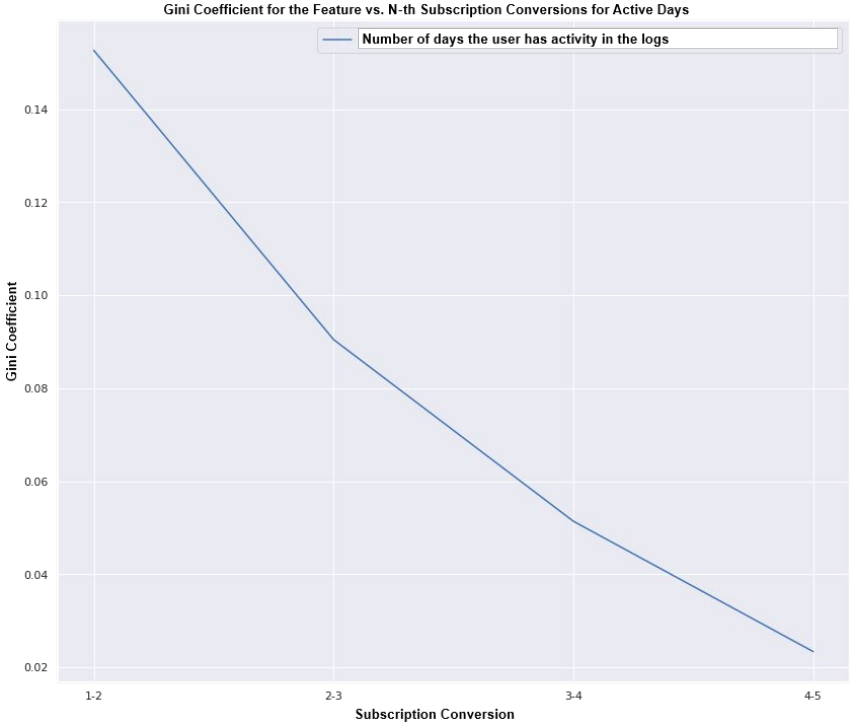
Problem: many features, many values

Solution: choose the strongest feature (based on Gini coefficient between feature and subscription conversion), then binarize it

We used 3 features:

- Number of active days
- Whether the user converted
- Whether the user had the offer available

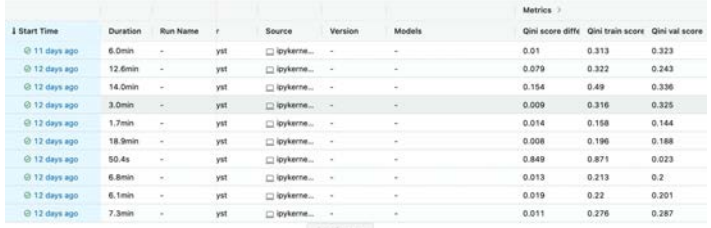
This produced 12 strata. We form clusters such that the proportions of each stratum remain consistent.



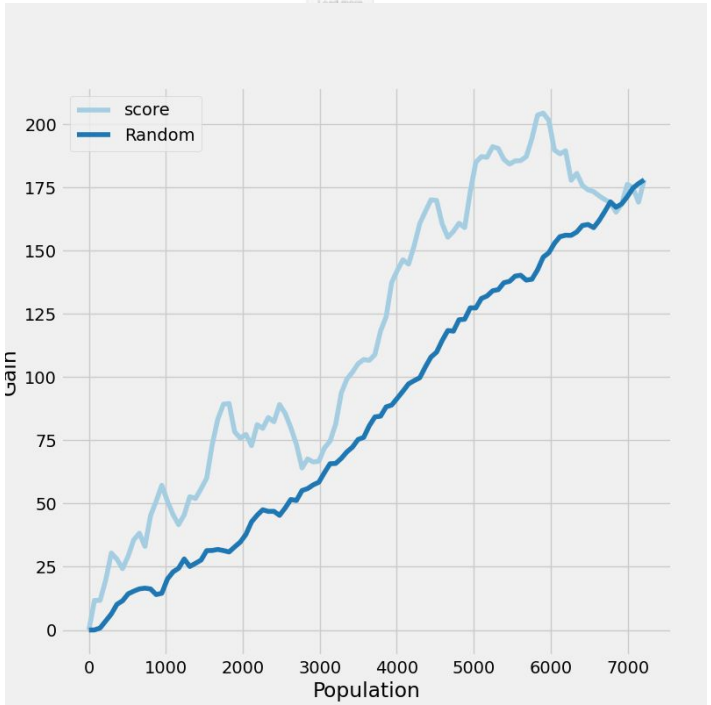
How to Fix It: Random Stratification

1. Take half of the sample for train; the rest is the pool from which we will draw data
2. From the pool, stratify and select 1,000 samples
3. Add these samples to train
4. Train the model
5. Evaluate on the remaining data
6. If Qini on the training and validation sets differs by no more than 5% and is better than random, augment the training set with those 1,000 samples
7. Otherwise, do not update the training set
8. Stop when the training set reaches at least 80% of the total data

("Stratified" means the sub-sample of 1,000 preserves the proportions of "active days," "converted or not," and "offer availability" as in the original dataset.)



Start Time	Duration	Run Name	r	Source	Version	Models	Metrics		
							Qini score diffe	Qini train score	Qini val score
11 days ago	6.0min	-	yst	ipykerne...	-	-	0.01	0.313	0.323
12 days ago	12.6min	-	yst	ipykerne...	-	-	0.079	0.322	0.243
12 days ago	14.0min	-	yst	ipykerne...	-	-	0.154	0.49	0.336
12 days ago	3.0min	-	yst	ipykerne...	-	-	0.009	0.316	0.325
12 days ago	1.7min	-	yst	ipykerne...	-	-	0.014	0.158	0.144
12 days ago	18.9min	-	yst	ipykerne...	-	-	0.008	0.196	0.188
12 days ago	50.4s	-	yst	ipykerne...	-	-	0.849	0.871	0.023
12 days ago	6.8min	-	yst	ipykerne...	-	-	0.013	0.213	0.2
12 days ago	6.1min	-	yst	ipykerne...	-	-	0.019	0.22	0.201
12 days ago	7.3min	-	yst	ipykerne...	-	-	0.011	0.276	0.287



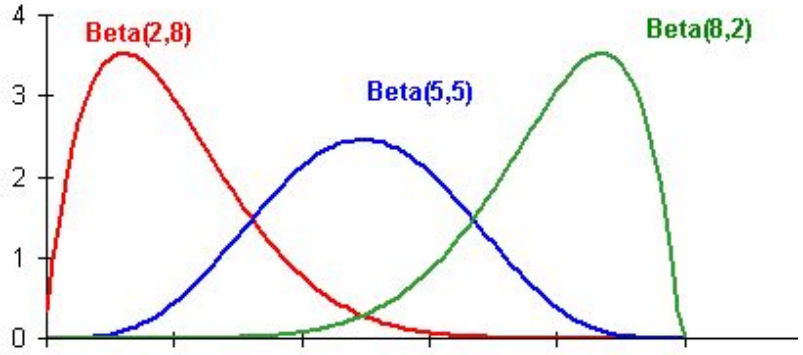
How to Fix It: Thompson Sampling

Beta distribution for data clusters

Success = model score on train & validation is similar and exceeds random

Failure = not meeting one or both of these conditions

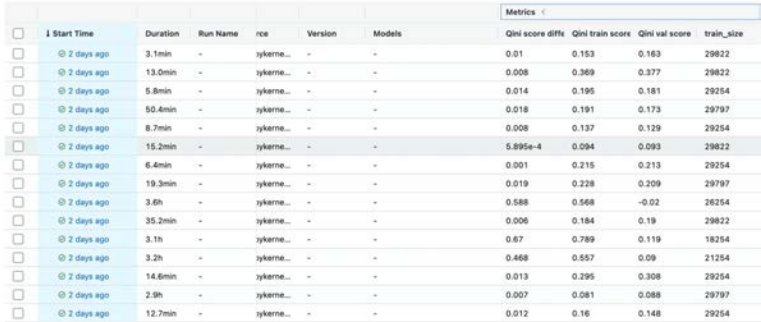
We incorporate the successes/failures of previous iterations. We account for changes in the external environment and training results.



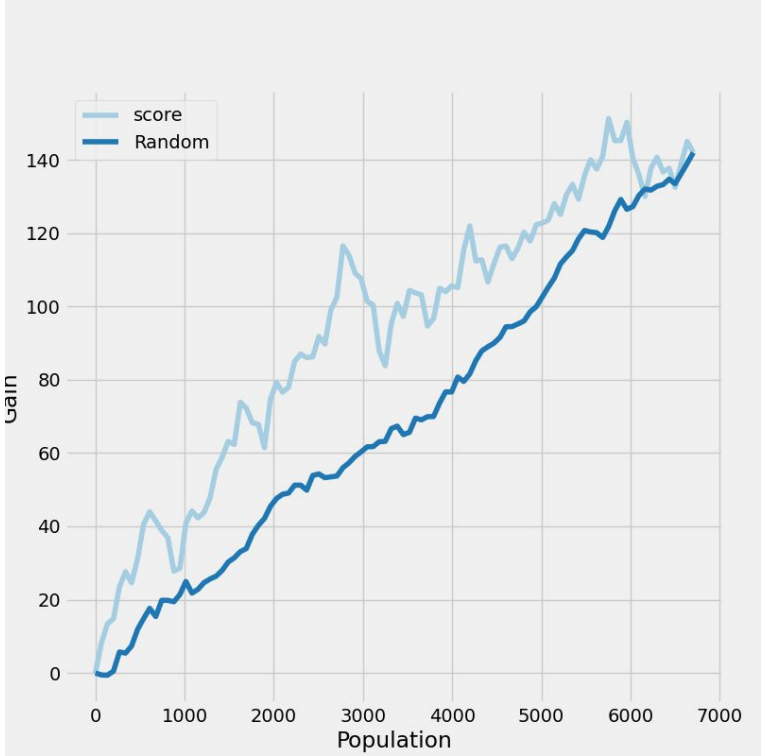
How to Fix It: Thompson Sampling

1. Half of the data = train; the other half is a pool
2. We split the pool into 4 stratified clusters
3. Select a cluster from which to draw samples (via Beta distribution)
4. Take at least 1,000 samples from that cluster
5. Add them to train
6. Train the model
7. Evaluate on the remaining data
8. If Qini on train and validation differs by ≤5% and is better than random, add the samples to train (α += 1)
9. Else do not update train (β += 1)
10. Stop when the training set reaches at least 80% of all data

It often fails to converge reliably.



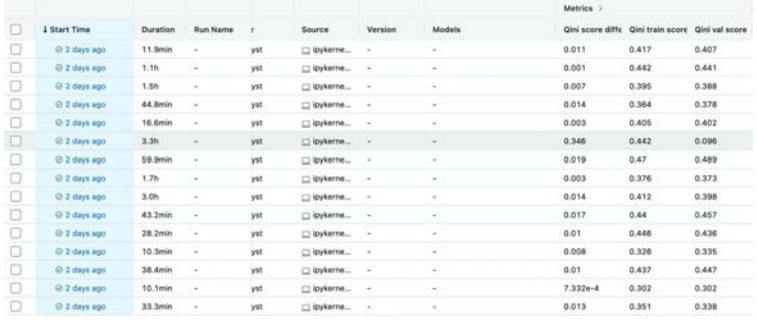
Start Time	Duration	Run Name	rce	Version	Models	Qini score diffe	Qini train score	Qini val score	train_size
2 days ago	3.1min	-	sykerne...	-	-	0.01	0.153	0.163	29822
2 days ago	13.0min	-	sykerne...	-	-	0.008	0.369	0.377	29822
2 days ago	5.8min	-	sykerne...	-	-	0.014	0.195	0.181	29254
2 days ago	50.4min	-	sykerne...	-	-	0.018	0.191	0.173	29797
2 days ago	8.7min	-	sykerne...	-	-	0.008	0.137	0.129	29254
2 days ago	15.2min	-	sykerne...	-	-	5.895e-4	0.094	0.093	29822
2 days ago	6.4min	-	sykerne...	-	-	0.001	0.215	0.213	29254
2 days ago	19.3min	-	sykerne...	-	-	0.019	0.228	0.209	29797
2 days ago	3.6h	-	sykerne...	-	-	0.588	0.568	-0.02	26254
2 days ago	35.2min	-	sykerne...	-	-	0.006	0.184	0.19	29822
2 days ago	3.1h	-	sykerne...	-	-	0.67	0.789	0.119	18254
2 days ago	3.2h	-	sykerne...	-	-	0.468	0.557	0.09	21254
2 days ago	14.6min	-	sykerne...	-	-	0.013	0.295	0.308	29254
2 days ago	2.9h	-	sykerne...	-	-	0.007	0.081	0.088	29797
2 days ago	12.7min	-	sykerne...	-	-	0.012	0.16	0.148	29254

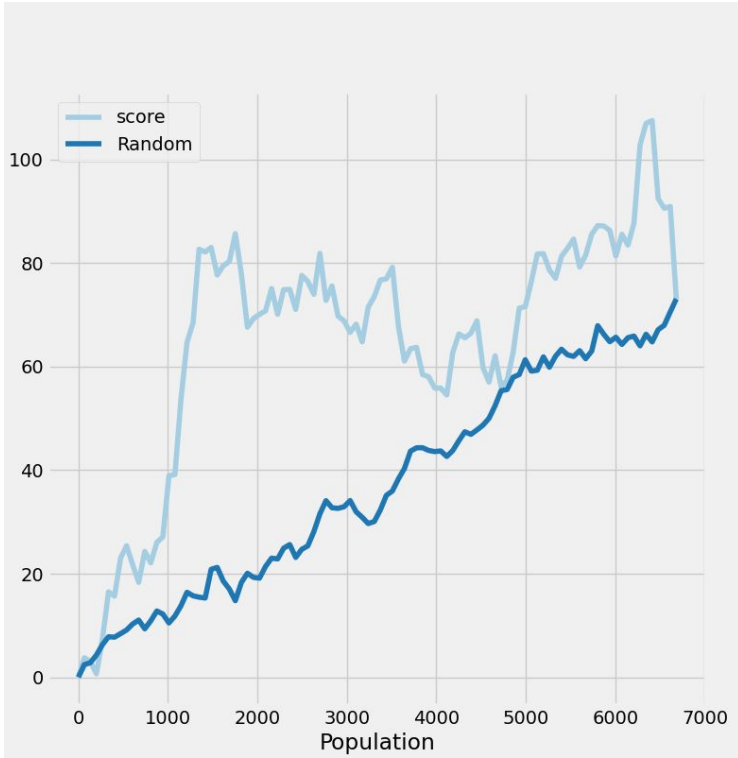


How to Fix It: Thompson Sampling + Registration Date

1. Sort data by registration date
2. Oldest 50% → train; remaining 50% is the pool
3. Split the pool into 4 stratified clusters
4. Select which cluster to pull samples from (Beta distribution)
5. Take at least 1,000 samples from that cluster
6. Add them to train
7. Train the model
8. Evaluate on the remaining data
9. If Qini on train and validation differs by ≤5% and exceeds random, expand train (α += 1)
10. Else, do not update train (β += 1)
11. Stop when train is ≥80% of the total data

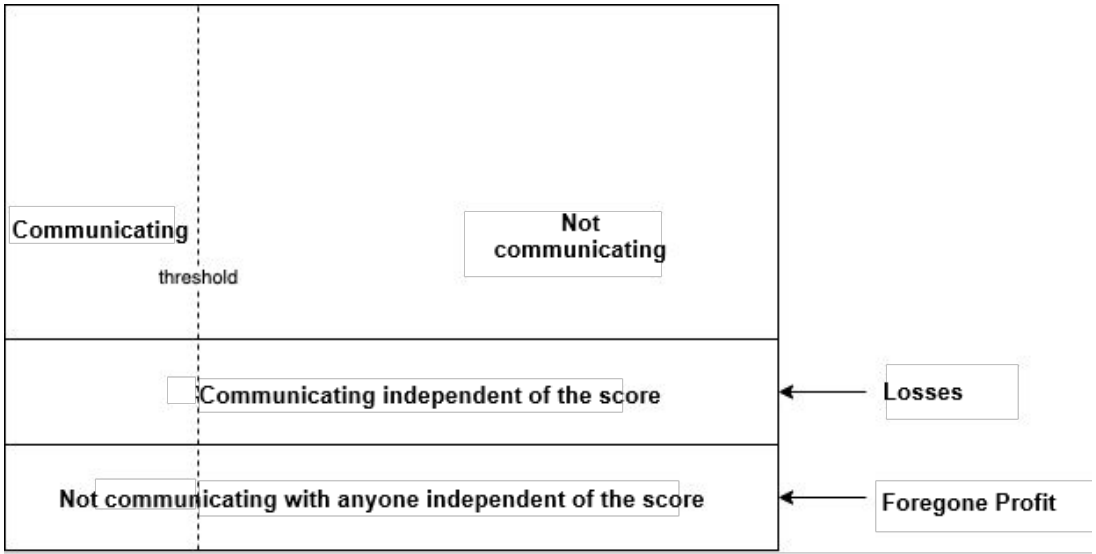
Works more reliably, yields higher scores—**great**!





Design AB

1. Select new trial users
2. Select 5% of them to not receive an offer (control group), and 5% to receive an offer
3. For the remaining 90%:
    a. Score the users with the model
    b. Take the top 20% by model score
    c. Send them the offer
    d. Do not send the offer to the other 80%
4. Compare the test group's conversion to the control group (it should be higher)
5. Compare the conversion among the test group who did not receive the offer (80%) to the control group (they should be the same)



Next Steps

Experimenting with:

- Discount amount
- Offer duration (1–3–6 months)
- Timing of discount (1–2–3 weeks after start)
- Number of existing subscriptions a user has when the offer is made