The Product Recommendation for H&M

Course: BIA 679
Instructor: Dr. Mohammad Nikouei

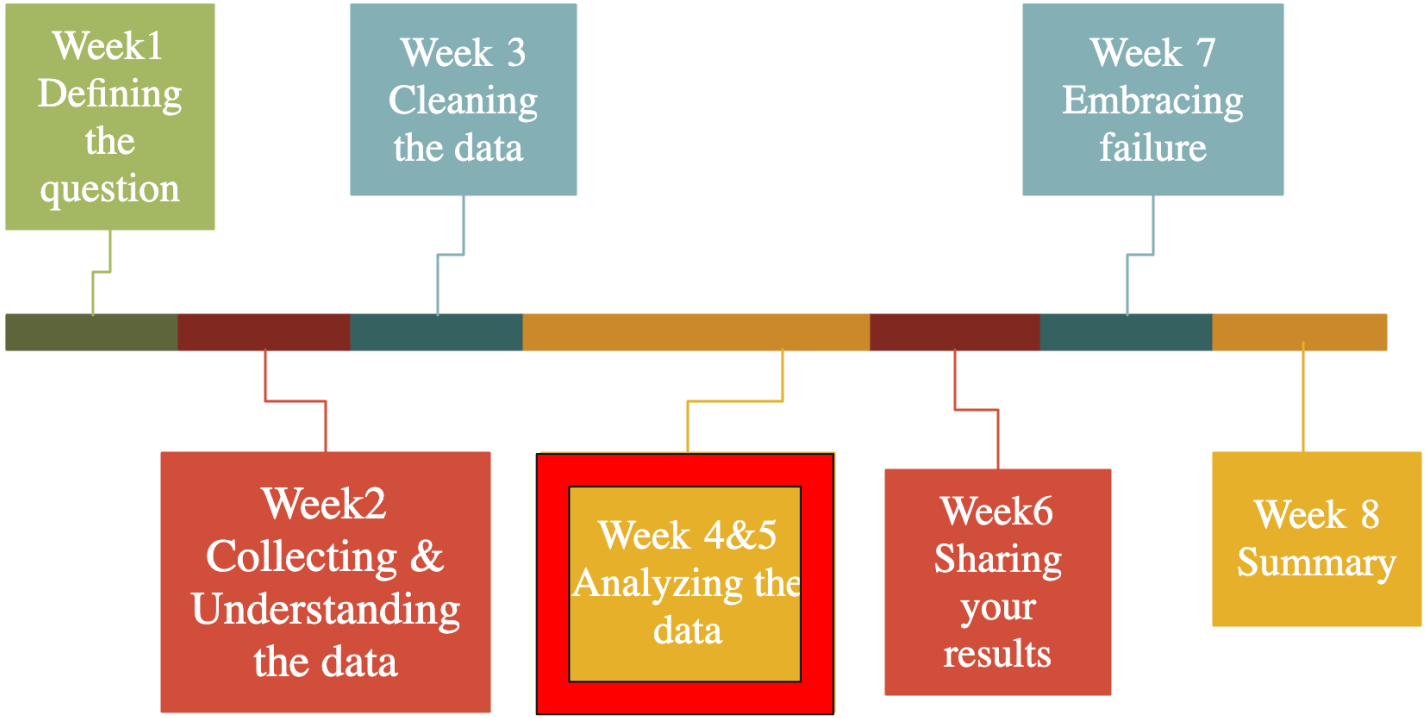
Team Members:

Tengyue Chen
Major: BIA

Hao Miao
Major: BIA

Haoxing Zhang
Major: BIA

BIA 679 Group Project



What’s New?

1. Load to AWS
2. Keep Optimizing Model

Difficulty

1. **Overtime during the modeling**
    A. **When tuning the parameter**
2. **Development environment not used to**
    B. **Missing functions**

In [6]:
```
transaction = spark.read.option("header","true").csv("s3://hmrecomm/transactions_train.csv")
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='5
0%'),…
```

In [7]:
```
transaction.printSchema()
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='5
0%'),…
root
 |-- t_dat: string (nullable = true)
 |-- customer_id: string (nullable = true)
 |-- article_id: string (nullable = true)
 |-- price: string (nullable = true)
 |-- sales_channel_id: string (nullable = true)
```

In [8]:
```
from pyspark.sql.functions import *
from pyspark.sql.functions import min, max
from pyspark.sql.functions import unix_timestamp, lit

min_date, max_date = transaction.select(min("t_dat"), max("t_dat")).first()
print(min_date, max_date)
datahm =  transaction.withColumn('t_dat', transaction['t_dat'].cast('string'))
datahm = datahm.withColumn('date', from_unixtime(unix_timestamp('t_dat', 'yyyy-MM-dd')))
datahm = datahm.withColumn('year', year(col('date')))
datahm = datahm.withColumn('month', month(col('date')))
datahm = datahm.withColumn('day', date_format(col('date'), "d"))

datahm = datahm[datahm['year'] == 2020]
datahm = datahm[datahm['month'] == 9]
datahm = datahm[datahm['day'] == 22]
transaction.unpersist()
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='5
0%'),…
```

Run the code with PySpark in AWS

	rank	maxIter	regParam	RMSE
154	20	20	0.05	0.43621867505918144
155	20	20	0.06	0.43628881937955905
117	15	20	0.08	0.4363435207448502
118	15	20	0.09	0.43645635666758403
156	20	20	0.07	0.4365432614916436
...	...	...	...	...
10	5	10	0.01	0.5894333765079824
80	15	5	0.01	0.5905611623791649
1	5	5	0.02	0.5974632794488814
40	10	5	0.01	0.6237407257601445
0	5	5	0.01	0.687137876751679

In [10]:
```
#tune model using ParamGridBuilder
param_grid = ParamGridBuilder()\
             .addGrid(als.rank, [15,20,25])\
             .addGrid(als.maxIter,[5,10,15])\
             .addGrid(als.regParam,[0.05,0.1,0.15])\
             .build()
#Build cross validation using CrossValidator
crossvalidate = CrossValidator(estimator=als,estimatorParamMaps=param_grid, evaluator=evaluator

#load the crovalidator into the model
tuned_model = crossvalidate.fit(training)
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25p
x', width='50%'),…
```

```
Exception in thread cell_monitor-10:
Traceback (most recent call last):
  File "/mnt/notebook-env/lib/python3.7/threading.py", line 926, in _bootstrap_inner
    self.run()
  File "/mnt/notebook-env/lib/python3.7/threading.py", line 870, in run
    self._target(*self._args, **self._kwargs)
  File "/mnt/notebook-env/lib/python3.7/site-packages/awseditorssparkmonitoringwidget-1.0-py
3.7.egg/awseditorssparkmonitoringwidget/cellmonitor.py", line 178, in cell_monitor
    job_binned_stages[job_id][stage_id] = all_stages[stage_id]
KeyError: 2360

An error was encountered:
Invalid status code '400' from http://localhost:8998/sessions/1/statements/10 with error payl
oad: {"msg":"requirement failed: Session isn't active."}
```

- Using the different regParam cause the time out error in the AWS
- Last week, local running result shows the regParam = 0.005 in best model.
- Therefore, regParam = 0.005

In [15]:
```
#tune model using ParamGridBuilder
param_grid = ParamGridBuilder()\
             .addGrid(als.rank, [15,20,25])\
             .addGrid(als.maxIter,[5,10,15])\
             .addGrid(als.regParam,[0.05])\
             .build()
#Build cross validation using CrossValidator
crossvalidate = CrossValidator(estimator=als,estimatorParamMaps=param_grid, evaluator=evaluator,numFolds=5)

#load the crovalidator into the model
tuned_model = crossvalidate.fit(training)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
Exception in thread cell_monitor-15:
Traceback (most recent call last):
  File "/mnt/notebook-env/lib/python3.7/threading.py", line 926, in _bootstrap_inner
    self.run()
  File "/mnt/notebook-env/lib/python3.7/threading.py", line 870, in run
    self._target(*self._args, **self._kwargs)
  File "/mnt/notebook-env/lib/python3.7/site-packages/awseditorssparkmonitoringwidget-1.0-py3.7.egg/awseditorssparkmonitoringwidget/cellmonitor.py", line 178, in cell_monitor
    job_binned_stages[job_id][stage_id] = all_stages[stage_id]
KeyError: 2114
```

In [16]:
```
model = tuned_model.bestModel
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

In [17]:
```
predictions = model.transform(test)
rmse = evaluator.evaluate(predictions)
```

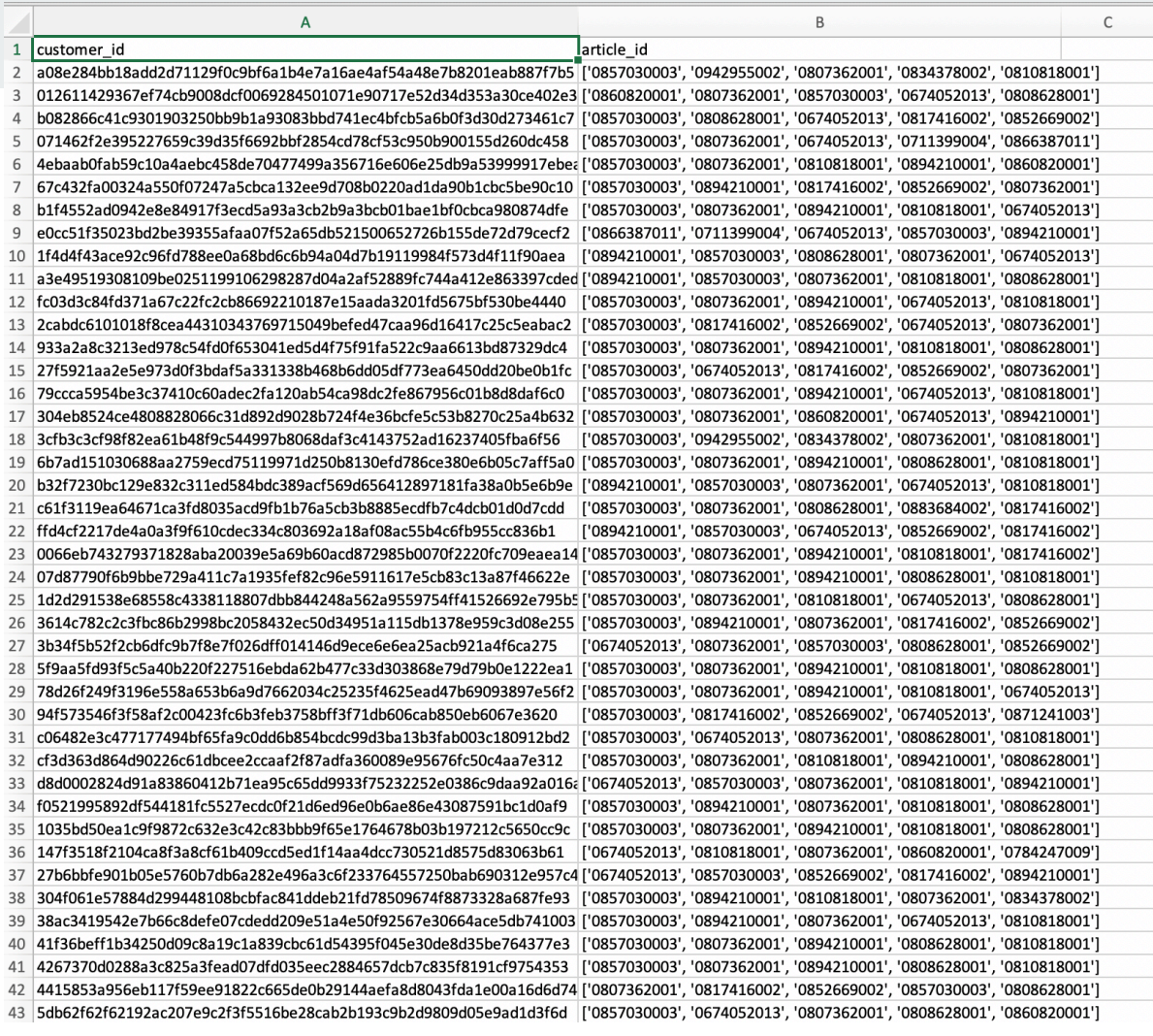
```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

In [18]:
```
#print evaluation metrics and model parameters
print("RMSE =" + str(rmse))
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
RMSE =0.43664054214548154
```

- Use ParagGridBuilder to find the best model with 3 different rank and maxIter and rerParam = 0.005
- RMSE = 0.436

- Output the result
- Each customer have 5 recommend product



	A	B	C
1	customer_id	article_id	
2	a08e284bb18add2d71129f0c9bf6a1b4e7a16ae4af54a48e7b8201eab887f7b5	['0857030003', '0942955002', '0807362001', '0834378002', '0810818001']	
3	012611429367ef74cb9008dcf00692845011071e90717e52d34d353a30ce402e3	['0860820001', '0807362001', '0857030003', '0674052013', '0808628001']	
4	b082866c41c9301903250bb9b1a93083bbd741ec4bfcb5a6b0f3d30d273461c7	['0857030003', '0808628001', '0674052013', '0817416002', '0852669002']	
5	071462f2e395227659c39d35f6692bbf2854cd78cf53c950b900155d260dc458	['0857030003', '0807362001', '0674052013', '0711399004', '0866387011']	
6	4ebaab0fab59c10a4aebc458de70477499a356716e606e25db9a53999917ebea	['0857030003', '0807362001', '0810818001', '0894210001', '0860820001']	
7	67c432fa00324a550f07247a5cbca132ee9d708b0220ad1da90b1cbc5be90c10	['0857030003', '0894210001', '0817416002', '0852669002', '0807362001']	
8	b1f4552ad0942e8e84917f3ecd5a93a3cb2b9a3bcb01bae1bf0cbca980874dfe	['0857030003', '0807362001', '0894210001', '0810818001', '0674052013']	
9	e0cc51f35023bd2be39355afaa07f52a65db5215006527 26b155de72d79cecf2	['0866387011', '0711399004', '0674052013', '0857030003', '0894210001']	
10	1f4d4f43ace92c96fd788ee0a68bd6c6b94a04d7b19119984f573d4f11f90aea	['0894210001', '0857030003', '0808628001', '0807362001', '0674052013']	
11	a3e49519308109be0251199106298287d04a2af52889fc744a412e863397cded	['0894210001', '0857030003', '0807362001', '0810818001', '0808628001']	
12	fc03d3c84fd371a67c22fc2cb86692210187e15aada3201fd5675bf530be4440	['0857030003', '0807362001', '0894210001', '0674052013', '0810818001']	
13	2cabdc6101018f8cea44310343769715049befed47caa96d16417c25c5eabac2	['0857030003', '0817416002', '0852669002', '0674052013', '0807362001']	
14	933a2a8c3213ed978c54fd0f653041ed5d4f75f91fa522c9aa6613bd87329dc4	['0857030003', '0807362001', '0894210001', '0810818001', '0808628001']	
15	27f5921aa2e5e973d0f3bdaf5a331338b468b6dd05df773ea6450dd20be0b1fc	['0857030003', '0674052013', '0817416002', '0852669002', '0807362001']	
16	79ccca5954be3c37410c60adec2fa120ab54ca98dc2fe867956c01b8d8daf6c0	['0857030003', '0807362001', '0894210001', '0674052013', '0810818001']	
17	304eb8524ce4808828066c31d892d9028b724f4e36bcfe5c53b8270c25a4b632	['0857030003', '0807362001', '0860820001', '0674052013', '0894210001']	
18	3cfb3c3cf98f82ea61b48f9c544997b8068daf3c4143752ad16237405fba6f56	['0857030003', '0942955002', '0834378002', '0807362001', '0810818001']	
19	6b7ad151030688aa2759ecd75119971d250b8130efd786ce380e6b05c7aff5a0	['0857030003', '0807362001', '0894210001', '0808628001', '0810818001']	
20	b32f7230bc129e832c311ed584bdc389acf569d656412897181fa38a0b5e6b9e	['0894210001', '0857030003', '0807362001', '0674052013', '0810818001']	
21	c61f3119ea64671ca3fd8035acd9fb1b76a5cb3b8885ecdfb7c4dcb01d0d7cdd	['0857030003', '0807362001', '0808628001', '0883684002', '0817416002']	
22	ffd4cf2217de4a0a3f9f610cdec334c803692a18af08ac55b4c6fb955cc836b1	['0894210001', '0857030003', '0674052013', '0852669002', '0817416002']	
23	0066eb743279371828aba20039e5a69b60acd872985b0070f2220fc709eaea14	['0857030003', '0807362001', '0894210001', '0810818001', '0817416002']	
24	07d87790f6b9bbe729a411c7a1935fef82c96e5911617e5cb83c13a87f46622e	['0857030003', '0807362001', '0894210001', '0808628001', '0810818001']	
25	1d2d291538e68558c4338118807dbb844248a562a9559754ff41526692e795b5	['0857030003', '0807362001', '0810818001', '0674052013', '0808628001']	
26	3614c782c2c3fbc86b2998bc2058432ec50d34951a115db1378e959c3d08e255	['0857030003', '0894210001', '0807362001', '0817416002', '0852669002']	
27	3b34f5b52f2cb6dfc9b7f8e7f026dff014146d9ece6e6ea25acb921a4f6ca275	['0674052013', '0807362001', '0857030003', '0808628001', '0852669002']	
28	5f9aa5fd93f5c5a40b220f227516ebda62b477c33d303868e79d79b0e1222ea1	['0857030003', '0807362001', '0894210001', '0810818001', '0808628001']	
29	78d26f249f3196e558a653b6a9d7662034c25235f4625ead47b69093897e56f2	['0857030003', '0807362001', '0894210001', '0810818001', '0674052013']	
30	94f573546f3f58af2c00423fc6b3feb3758bff3f71db606cab850eb6067e3620	['0857030003', '0817416002', '0852669002', '0674052013', '0871241003']	
31	c06482e3c477177494bf65fa9c0dd6b854bcdc99d3ba13b3fab003c180912bd2	['0857030003', '0674052013', '0807362001', '0808628001', '0810818001']	
32	cf3d363d864d90226c61dbcee2ccaaf2f87adfa360089e95676fc50c4aa7e312	['0857030003', '0807362001', '0810818001', '0894210001', '0808628001']	
33	d8d0002824d91a83860412b71ea95c65dd9933f75232252e0386c9daa92a016a	['0674052013', '0857030003', '0807362001', '0810818001', '0894210001']	
34	f0521995892df544181fc5527ecdc0f21d6ed96e0b6ae86e43087591bc1d0af9	['0857030003', '0894210001', '0807362001', '0810818001', '0808628001']	
35	1035bd50ea1c9f9872c632e3c42c83bbb9f65e1764678b03b197212c5650cc9c	['0857030003', '0807362001', '0894210001', '0810818001', '0808628001']	
36	147f3518f2104ca8f3a8cf61b409ccd5ed1f14aa4dcc730521d8575d83063b61	['0674052013', '0810818001', '0807362001', '0860820001', '0784247009']	
37	27b6bbfe901b05e5760b7db6a282e496a3c6f233764557250bab690312e957c4	['0674052013', '0857030003', '0852669002', '0817416002', '0894210001']	
38	304f061e57884d299448108bcbfac841ddeb21fd78509674f8873328a687fe93	['0857030003', '0894210001', '0810818001', '0807362001', '0834378002']	
39	38ac3419542e7b66c8defe07cdedd209e51a4e50f92567e30664ace5db741003	['0857030003', '0894210001', '0807362001', '0674052013', '0810818001']	
40	41f36beff1b34250d09c8a19c1a839cbc61d54395f045e30de8d35be764377e3	['0857030003', '0807362001', '0894210001', '0808628001', '0810818001']	
41	4267370d0288a3c825a3fead07dfd035eec2884657dcb7c835f8191cf9754353	['0857030003', '0807362001', '0894210001', '0808628001', '0810818001']	
42	4415853a956eb117f59ee91822c665de0b29144aefa8d8043fda1e00a16d6d74	['0807362001', '0817416002', '0852669002', '0857030003', '0808628001']	
43	5db62f62f62192ac207e9c2f3f5516be28cab2b193c9b2d9809d05e9ad1d3f6d	['0857030003', '0674052013', '0807362001', '0808628001', '0860820001']	

What’s Next?

Model building:

1. **Optimize the model**
    A. **Link the article ID with product classified name**
    B. **Considering the price level when doing recommendation**

Modeling

https://github.com/tychen17/The-Product-Recommendation-for-H-M

Reference

1. Zhao, Xuesong. “A Study on e-Commerce Recommender System Based on Big Data.” *2019 IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, 2019, https://doi.org/10.1109/icccbda.2019.8725694.
2. Jessica Young | Feb 18, 2022, et al. “US Ecommerce Grows 14.2% in 2021.” *Digital Commerce 360*, 16 Sept. 2022, https://www.digitalcommerce360.com/article/us-ecommerce-sales/.
3. “H&M Personalized Fashion Recommendations.” *Kaggle*, https://www.kaggle.com/competitions/h-and-m-personalized-fashion-recommendations/data.
4. Liao, Kevin. “Prototyping a Recommender System Step by Step Part 2: Alternating Least Square (ALS) Matrix Factorization in Collaborative Filtering.” Medium, Towards Data Science, 19 Nov. 2018, https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-2-alternating-least-square-als-matrix-4a76c58714a1.
5. “Alternating Least Squares.” Apache Flink 1.2 Documentation: Alternating Least Squares, 19 Oct. 2022, https://nightlies.apache.org/flink/flink-docs-release-1.2/dev/libs/ml/als.html.

Thank you