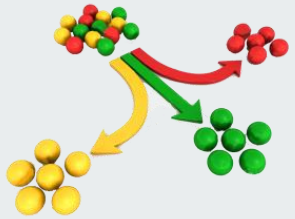
Recommendation Engine Design

Design of Recommendation involves –

Classification



Clustering

Association Analysis

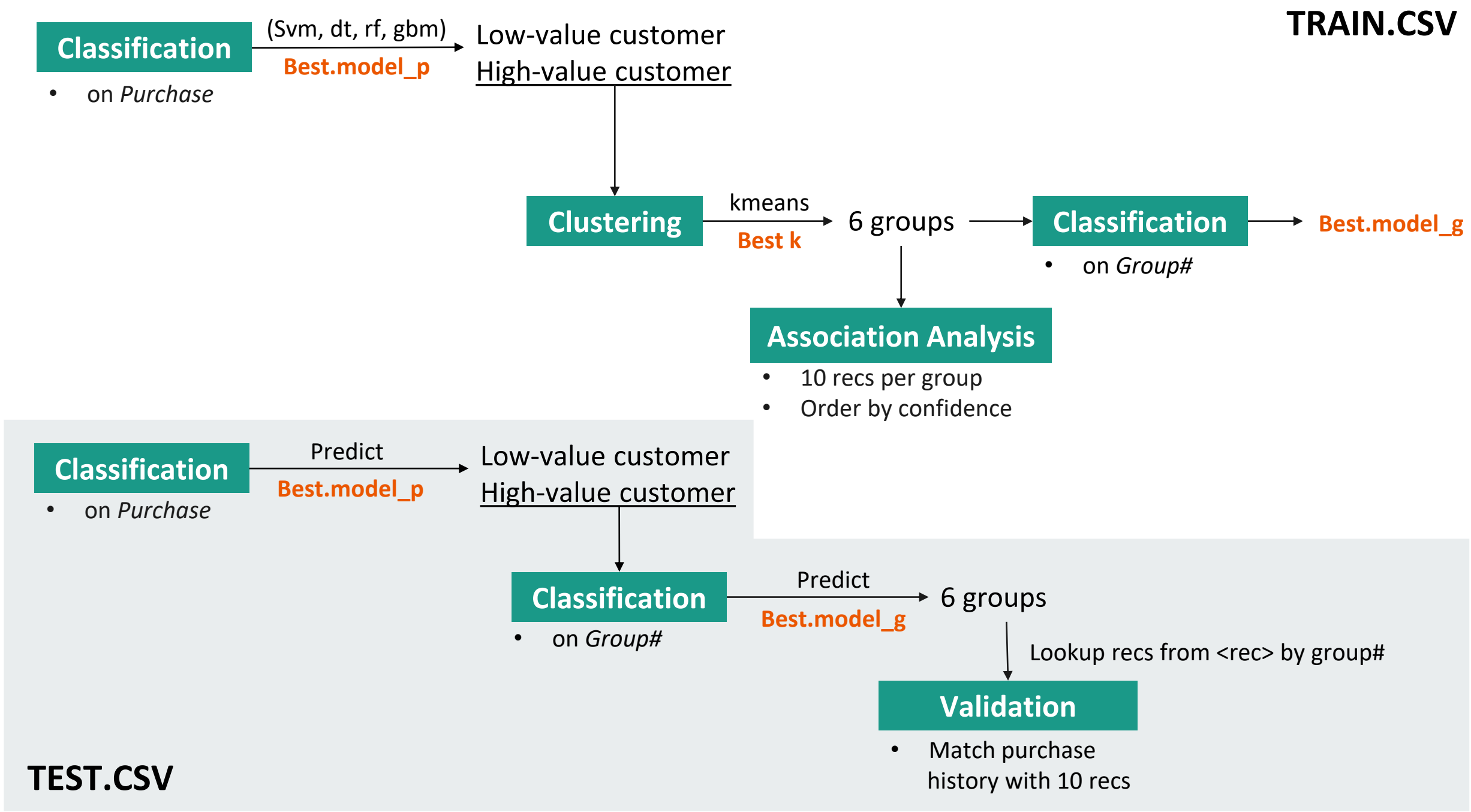
Data Overview

- Data Source: Retail data from Kaggle
- Train.csv
  - 10 columns & 550,068 rows
- Test.csv
  - 9 columns & 233,000 rows (no Purchase_Amount column)
- Fields
  - User ID
  - Gender
  - Age
  - Occupation
  - City Category
  - Stay in Current City (years)
  - Marital Status
  - Product Category
  - Purchase Amount

Preprocessing

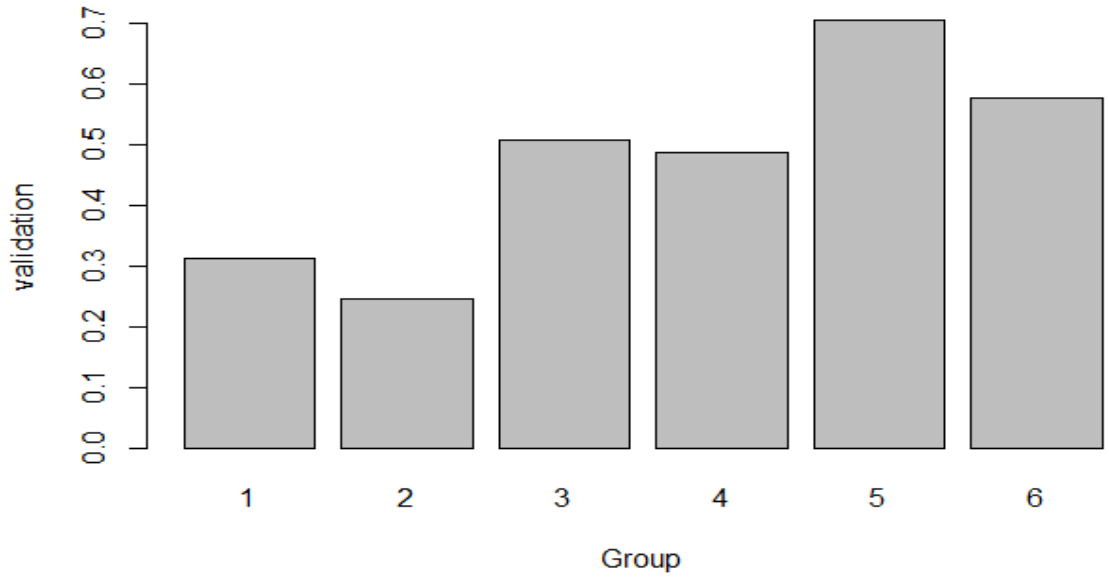
- No Missing Values
- Added Transaction_ID
- Purchase Amount → Total_Purchase (Response variable)
- Data Slicing

User_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Total_Purchase
1000001	F	0-17	10	A	2	0	Low
1000002	M	55+	16	C	4+	0	Low
1000003	M	26-35	15	A	3	0	Low
1000004	M	46-50	7	B	2	1	Low
1000005	M	26-35	20	A	1	1	Low


TRAIN.CSV
Classification
(Svm, dt, rf, gbm)
Best.model_p
Low-value customer
High-value customer
on Purchase
Clustering
kmeans
Best k
6 groups
Classification
Best.model_g
on Group#
Association Analysis
10 recs per group
Order by confidence
Classification
Predict
Best.model_p
Low-value customer
High-value customer
on Purchase
Classification
Predict
Best.model_g
6 groups
on Group#
Lookup recs from <rec> by group#
Validation
Match purchase history with 10 recs
TEST.CSV

Product Validation Results

- Since we cannot use the ideal method of experimental and control groups, we opted to see how often our recommended purchases were made by those in each cluster
  - This way, we can test whether people in the clusters are likely to want to buy our recommended products to begin with
  - Recommended 10 products out of 20



Proposed Solution Deployment

The solution deployed at checkout counters will –

- Cluster and Segment customers based on their profiles
- Recommend products based on the items in their cart

Improvements

- Perform an experiment for our recommendations
  - Create a control and experimental group
- Adjust recommendations based on:
  - Marginal profit
  - Item availability

Appendix

Model selection: best.model_p

- Used caret to tune each model(multi-processing)
- Displayed the confusion matrix for each model

SVM

Actual	Predicted High	Low
High	82	480
Low	75	1131

Decision tree

Actual	Predicted High	Low
High	215	347
Low	183	1023

Random forest

Actual	Predicted High	Low
High	170	392
Low	154	1052

GBM

Actual	Predicted High	Low
High	223	339
Low	175	1031

- Selected the best.model_p

"OE_svm"	"0.313914027149321"
"OE_dt"	**"0.299773755656109"(Fastest)**
"OE_rf"	"0.308823529411765"
"OE_gbm"	"0.290723981900453"

Model selection: best.model_g

- Classification on group#
- Try decision tree first and it perform pretty well:

	Predicted					
Actual	1	2	3	4	5	6
1	29	0	0	0	1	0
2	1	180	0	0	1	0
3	14	0	87	0	12	5
4	0	0	1	103	3	0
5	0	0	0	0	67	0
6	4	0	0	0	0	77

Overall Error Rate: 0.07179487

Best k

- Try k =2:10
- Use bootstrap (B=100)
- Trade-off between withinss and betweenss
- Consider the business practice