



American Express-Default Prediction

Predict if a customer will default in the future



team_ amazon

Intro >> Introducing Our Team



Team Leader



Team Member_1



Team Member_2

정주희



Team Leader

- XGBoost, LightGBM Modeling
- PPT
- Announcement

Team Member_1



Team Member_2

정주희

손희경



Team Leader

- XGBoost, LightGBM Modeling
- PPT
- Announcement

Team Member_1

- Random Forest Modeling
- EDA
- Organize Data

Team Member_2

정주희

손희경

최유림

Team Leader

- XGBoost, LightGBM Modeling
- PPT
- Announcement

Team Member_1

- Random Forest Modeling
- EDA
- Organize Data

Team Member_2

- CatBoost Modeling
- EDA
- Organize Data

Q.

How do card issuers know
we'll pay back what we charge?

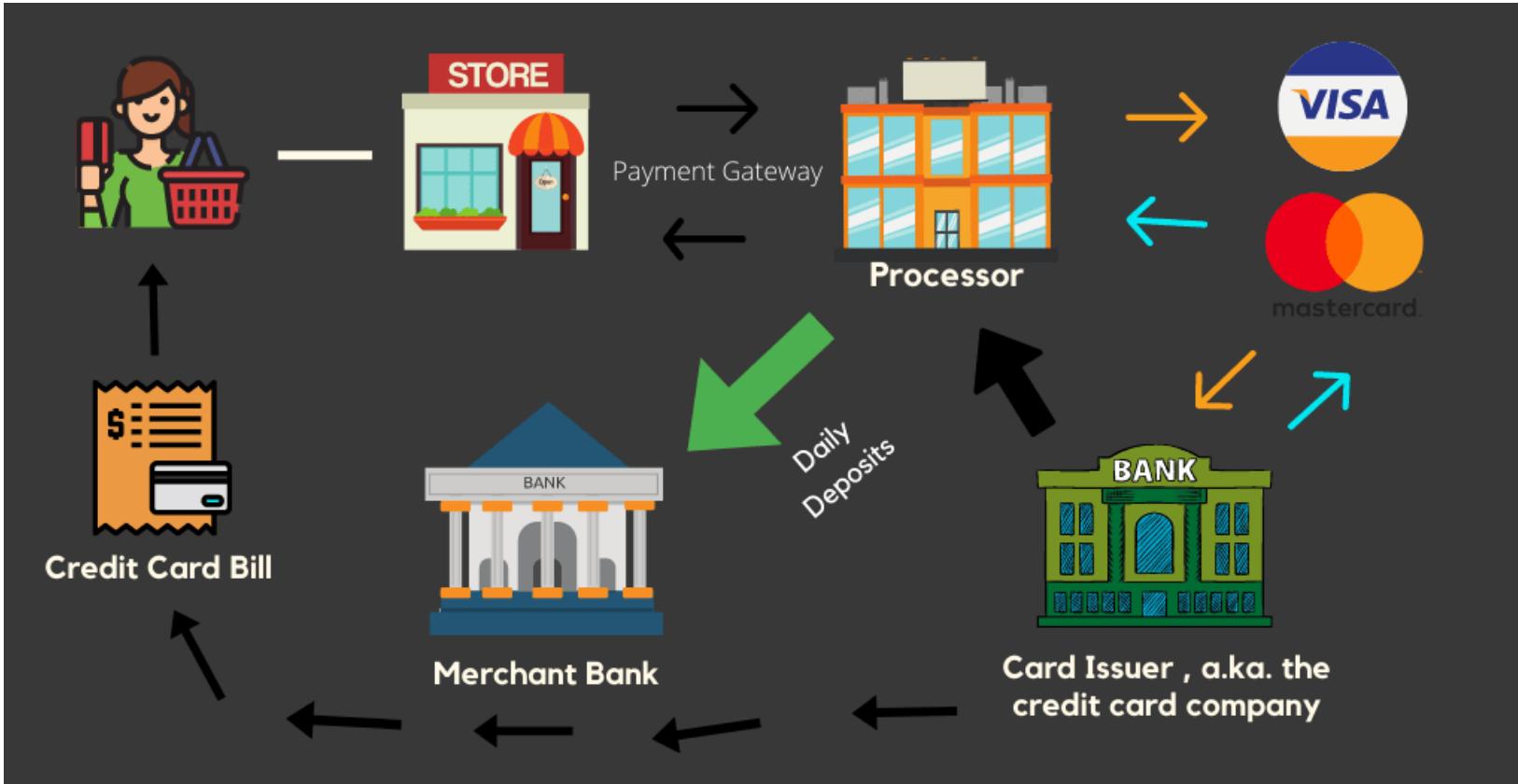
>> Table of contents

- 1 Competition Description
- 2 Data Preprocessing and EDA
- 3 Insight about Machine Learning Model
- 4 Conclusion

1

Competition Description

Part 1 >> Competition Description_ Revenue Stream

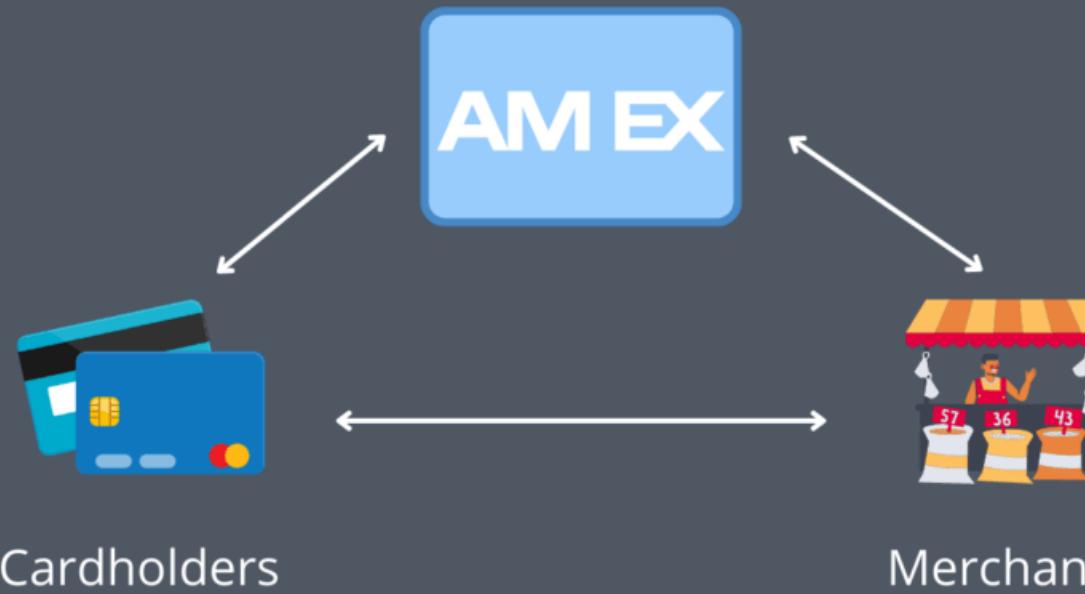


Part 1 >> Competition Description_ Revenue Stream

American Express Business Model

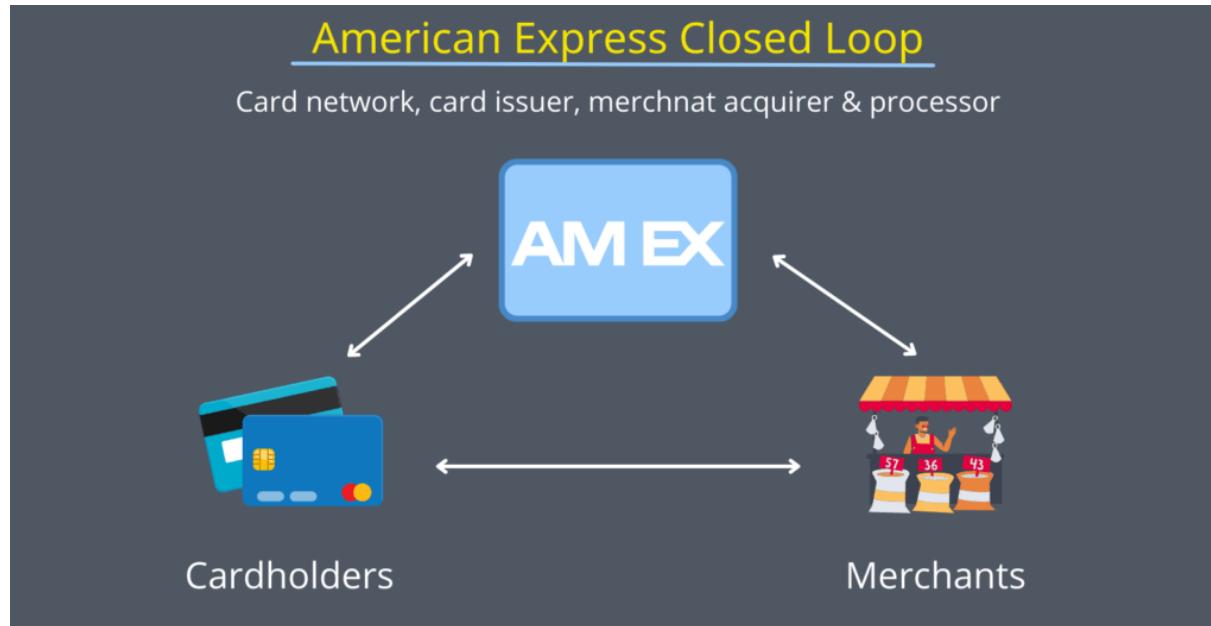
American Express Closed Loop

Card network, card issuer, merchant acquirer & processor

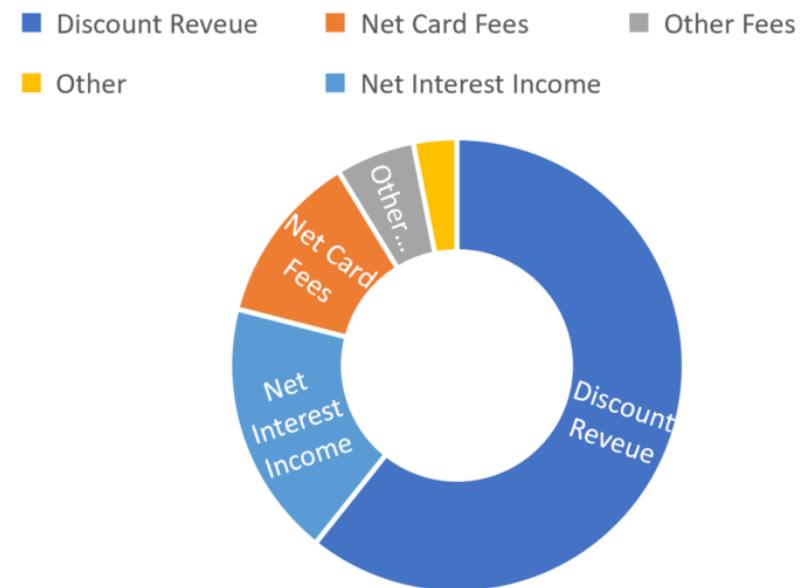


Part 1 >> Competition Description_ Revenue Stream

American Express Business Model



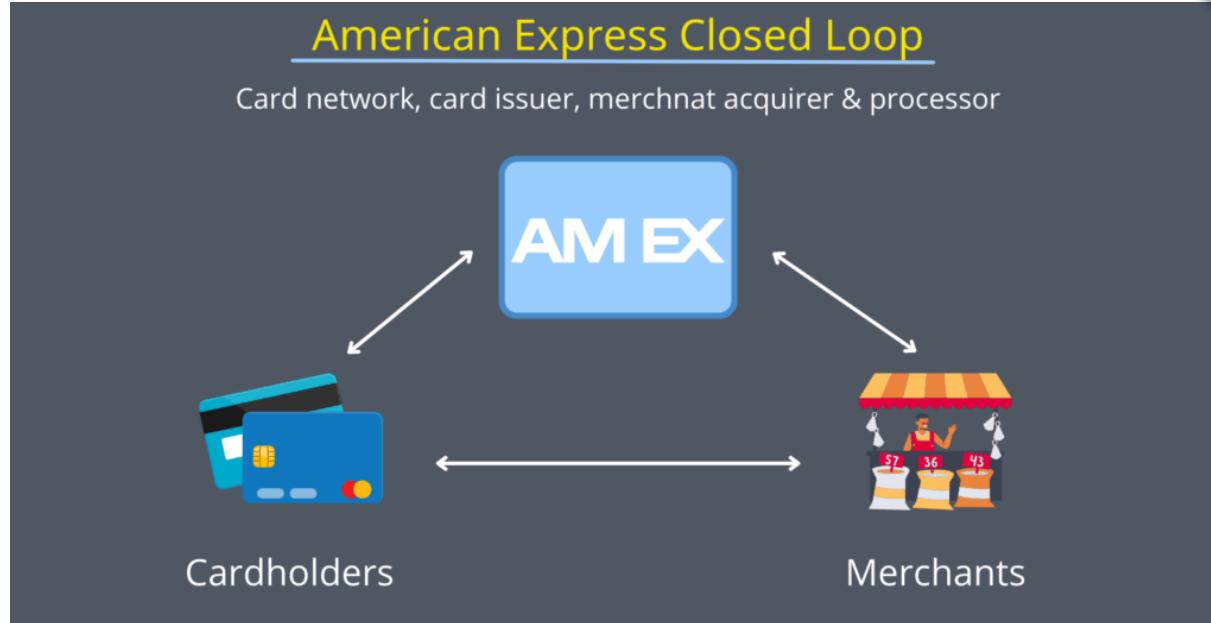
revenues



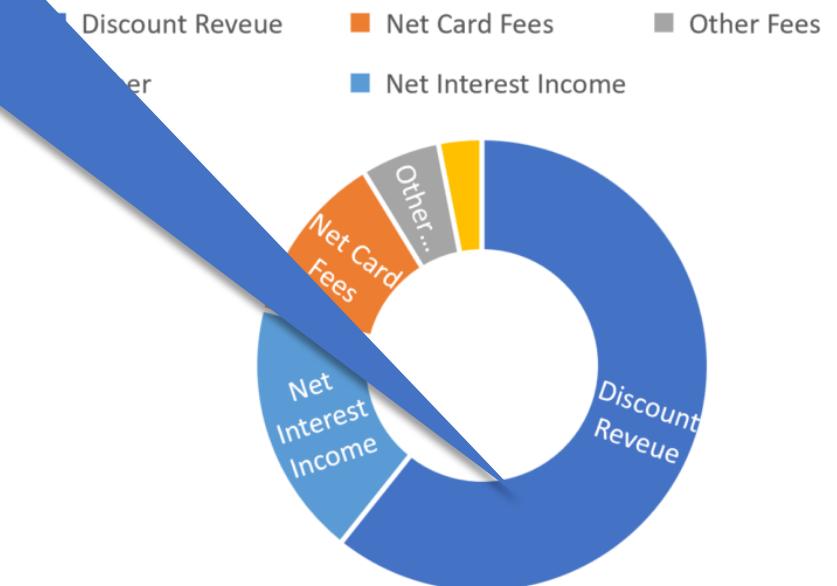
Discount Revenue(60%)

: 카드 결제 수수료

American Express Business Model



revenues



Discount Revenue(60%)

: 카드 결제 수수료

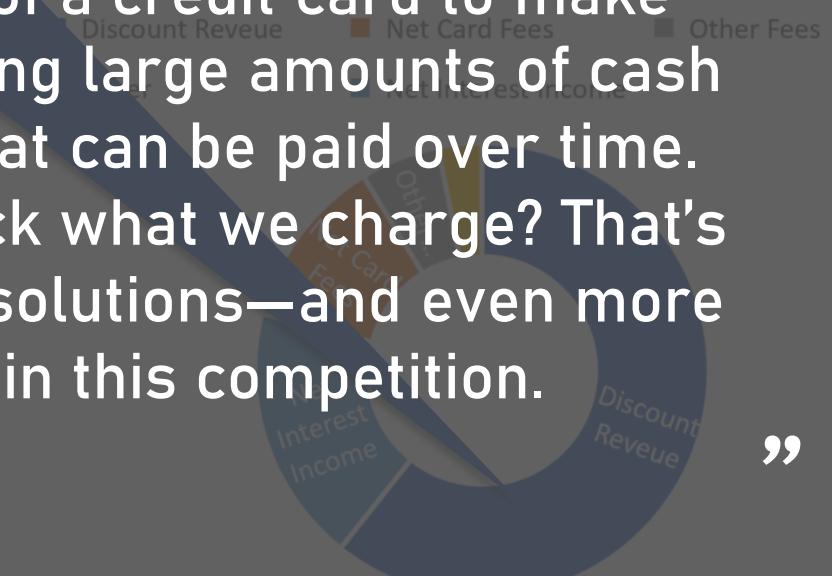
“

American Express Business Model
Whether out at a restaurant or buying tickets to a concert,
modern life counts on the convenience of a credit card to make
daily purchases. It saves us from carrying large amounts of cash
and also can advance a full purchase that can be paid over time.
How do card issuers know we'll pay back what we charge? That's
a complex problem with many existing solutions—and even more
potential improvements, to be explored in this competition.

”

Cardholders

Merchants



Discount Revenue(60%)

: 카드 결제 수수료

“

American Express Business Model

revenues

American Express Closed Loop
Card network, card issuer, merchant acquirer, loyalty, co-branded cards, travel services, and more.

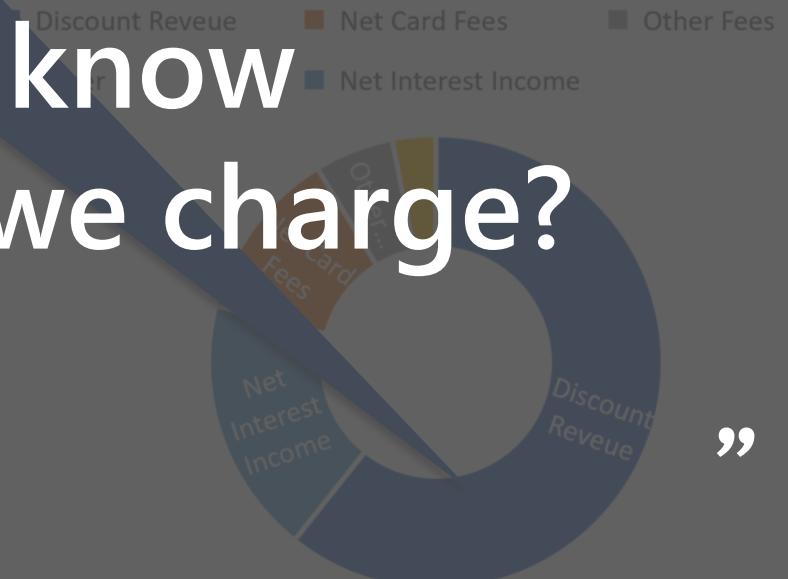
How do card issuers know we'll pay back what we charge?



Cardholders



Merchants



Part 1 >> Competition Description_ Variables



anonymous

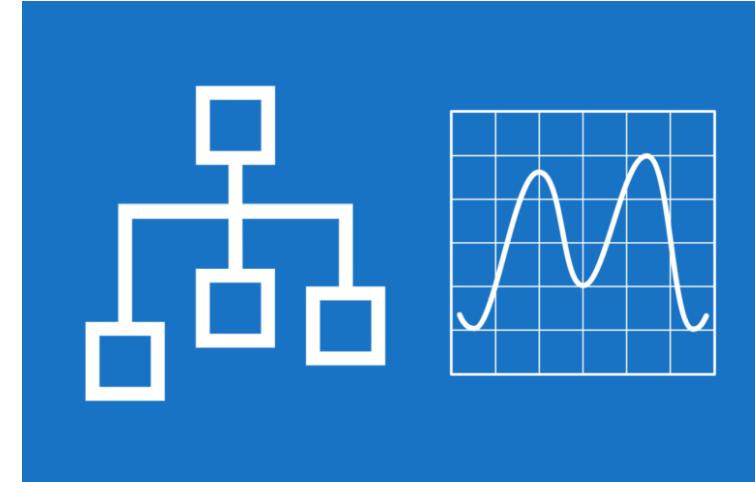
anonymized customer profile information



time-series data

The target binary variable is calculated by observing 18 months performance window after the latest credit card statement

target=1 : does not pay due amount in 120 days after their latest statement date

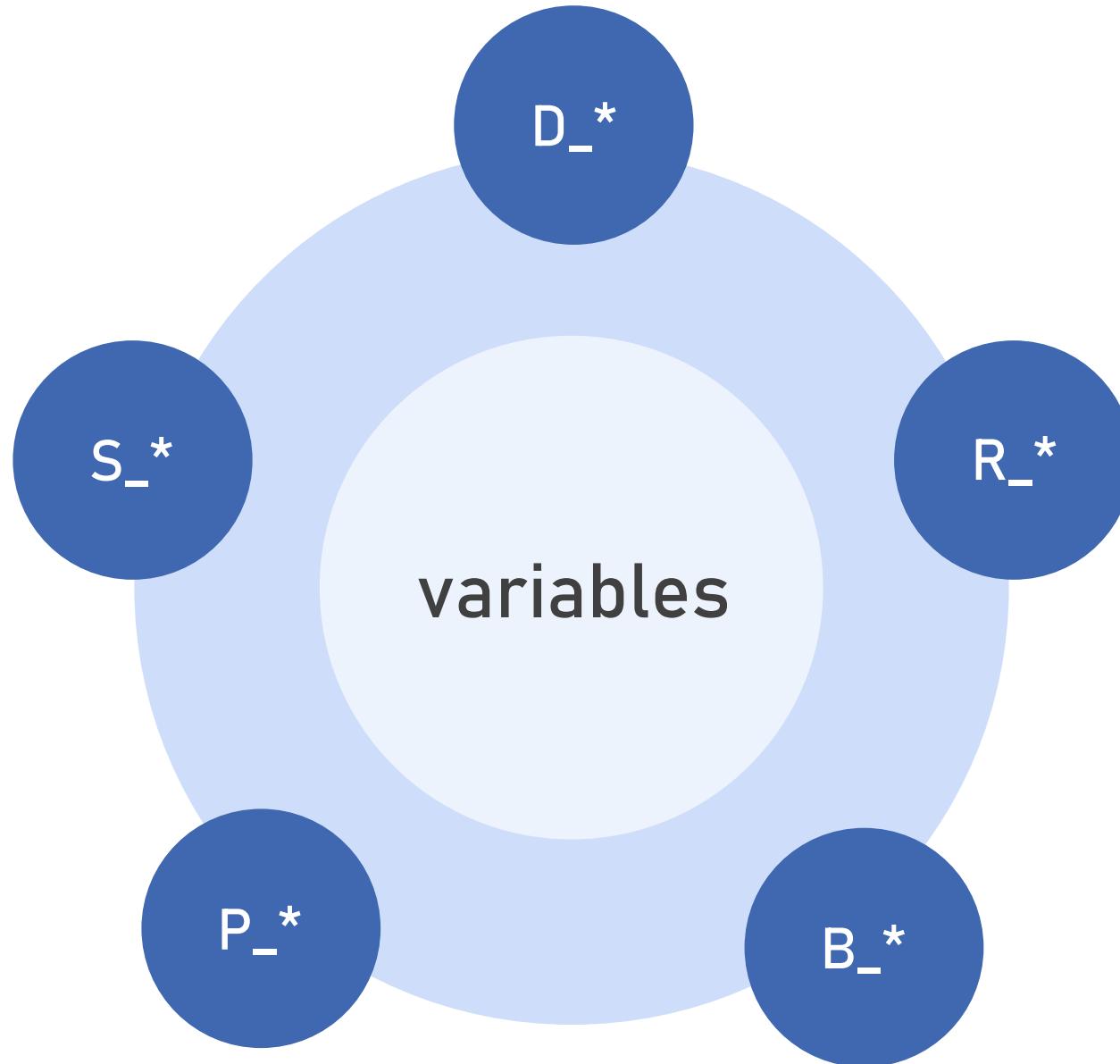


categorical/numerical

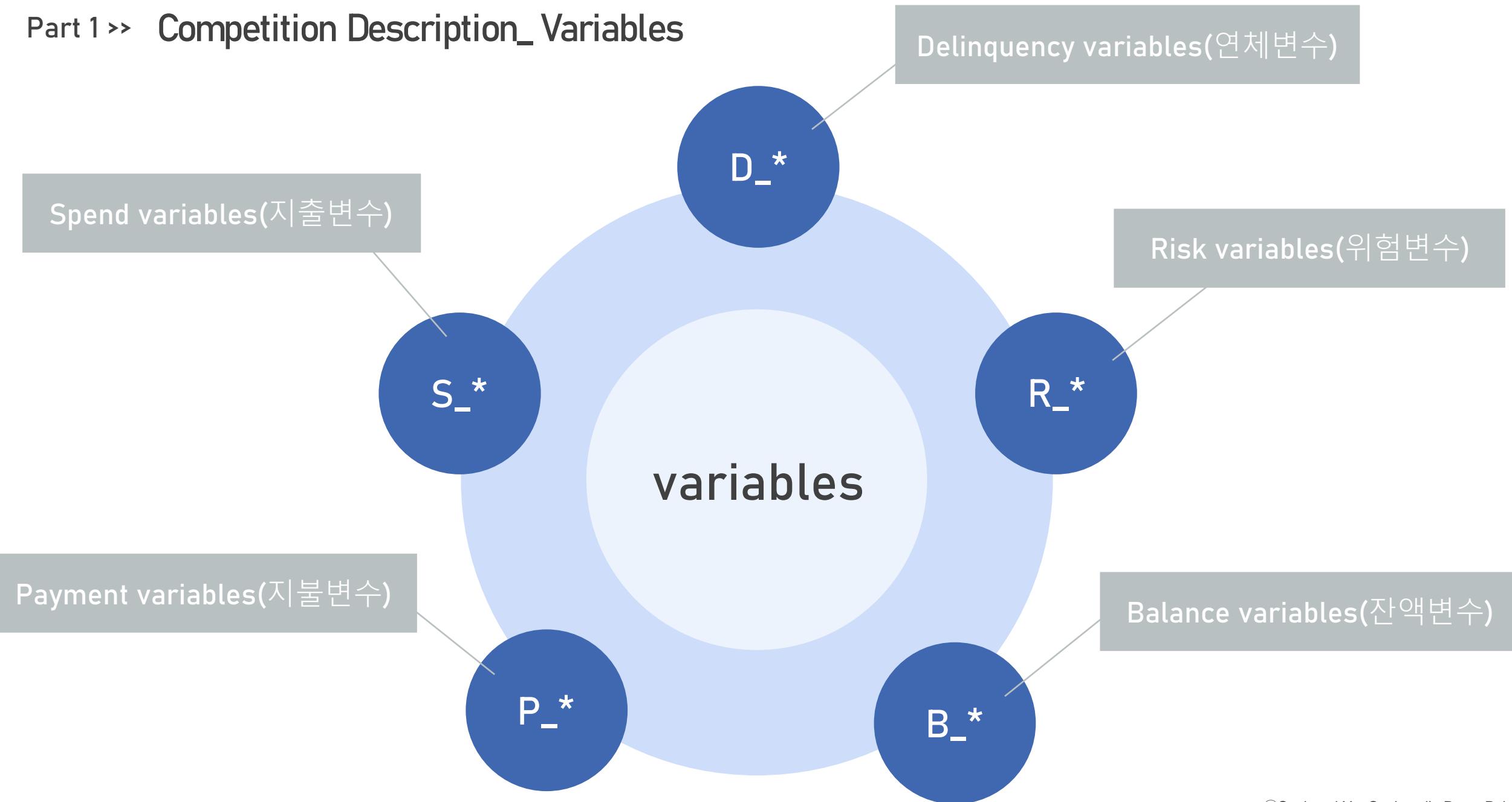
C : 'B_30', 'B_38', 'D_114', 'D_116', 'D_117', 'D_120',
'D_126', 'D_63', 'D_64', 'D_66', 'D_68'

predict '**target = 1**' for each customer_ID

Part 1 >> Competition Description_ Variables



Part 1 >> Competition Description_ Variables



Part 1 >> Competition Description_ Customer Data

Understand Customer Data_ Unique Customer

In [7]:

```
unique_customer_count = len(train.groupby("customer_ID")['customer_ID'].count())
print("unique customer data in training data -->", unique_customer_count)
unique_customer_count_test = len(test.groupby("customer_ID")['customer_ID'].count())
print("unique customer data in test data -->", unique_customer_count_test)
```

unique customer data in training data --> 458913
unique customer data in test data --> 924621

Part 1 >> Competition Description_ Customer Data

Understand Customer Data_ Unique Customer

In [7]:

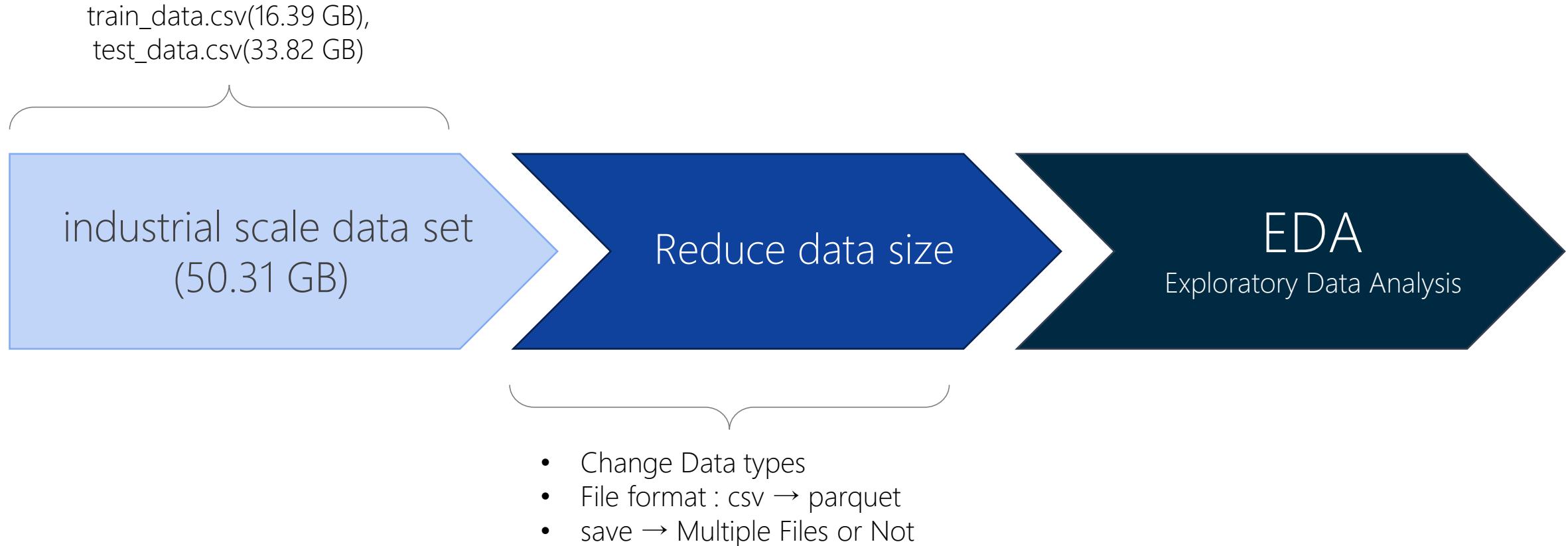
```
unique_customer_count = len(train.groupby("customer_ID")['customer_ID'].count())
print("unique customer data in training data -->", unique_customer_count)
unique_customer_count_test = len(test.groupby("customer_ID")['customer_ID'].count())
print("unique customer data in test data -->", unique_customer_count_test)
```

```
unique customer data in training data --> 458913
unique customer data in test data --> 924621
```

2

Data Preprocessing and EDA

Part 2 >> Exploratory Data Analysis

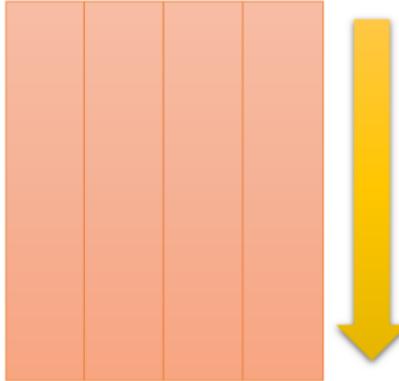


Part 2 >> Exploratory Data Analysis_ How to reduce Data size?

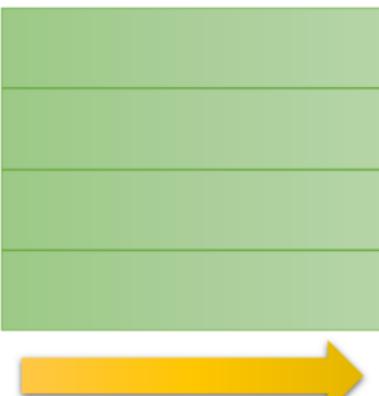
File Format	Reduce dtypes	File Saving type	Remove Noise
<ul style="list-style-type: none">• 압축률과 저장순서, <code>dtype</code> 기억 여부 고려• csv: 행별 저장 parquet: 열별 저장• feather, parquet, pickle: 데이터를 압축• csv: <code>dtype</code> 기억 X parquet: <code>dtype</code> 기억 O <p>>></p>	<ul style="list-style-type: none">• column 'customer_ID'• column 'S_2'<ul style="list-style-type: none">· <code>pd.to_datetime()</code>· 10 → 4byte• categorical column (11) :int8로 변환 [8 → 1byte]• numerical column (171) :float64 → float32 <p>>></p>	<p>Multiple Files or Not</p> <p>>></p>	<p>Features</p> <ul style="list-style-type: none">- integers- random noise injected <p>↓</p> <ul style="list-style-type: none">- float32(4byte) → int8(1byte)

Part 2 >> Exploratory Data Analysis_ Methods of data reduction(1) : File Format

열 기반 압축(Parquet)



행 기반 압축(전통적 방식)



행 기반 압축 → 데이터 압축률 우수



특정 column의 데이터만 읽고 처리 가능
→ 데이터 처리에 들어가는 자원 절약됨

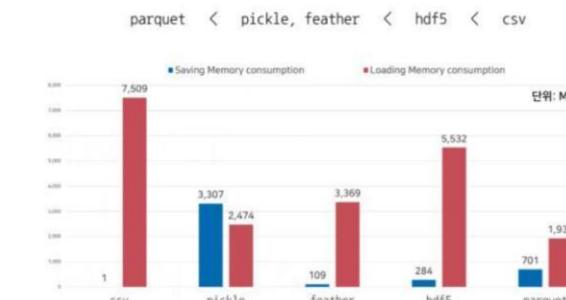


column에 동일한 dtype이 저장
→ column별로 적합한(데이터형에 유리한) 인코딩 사용 가능

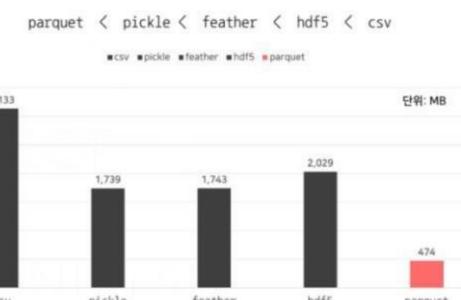


특정 column을 선택해서 가져오는 형식
→ 선택하지 않은 column 데이터는 I/O가 발생하지 않음
→ 시간, 메모리 사용량 줄일 수 있음

파일 형식에 따른 IO 시의 메모리 사용량

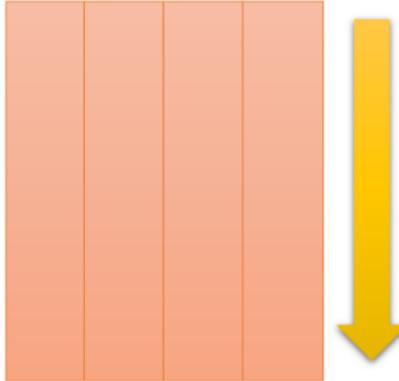


파일 형식에 따른 저장되는 파일의 크기

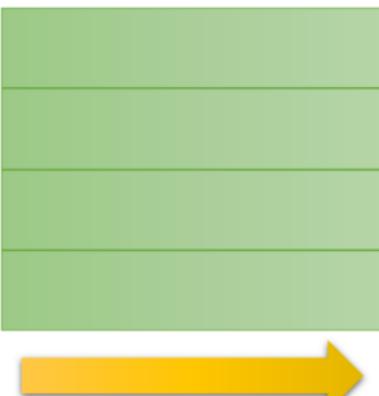


Part 2 >> Exploratory Data Analysis_ Methods of data reduction(1) : File Format

열 기반 압축(Parquet)



행 기반 압축(전통적 방식)



행 기반 압축 → 데이터 압축률 우수



특정 column의 데이터만 읽고 처리 가능
→ 데이터 처리에 들어가는 자원 절약됨



column에 동일한 dtype이 저장
→ column별로 적합한(데이터형에 유리한) 인코딩 사용 가능

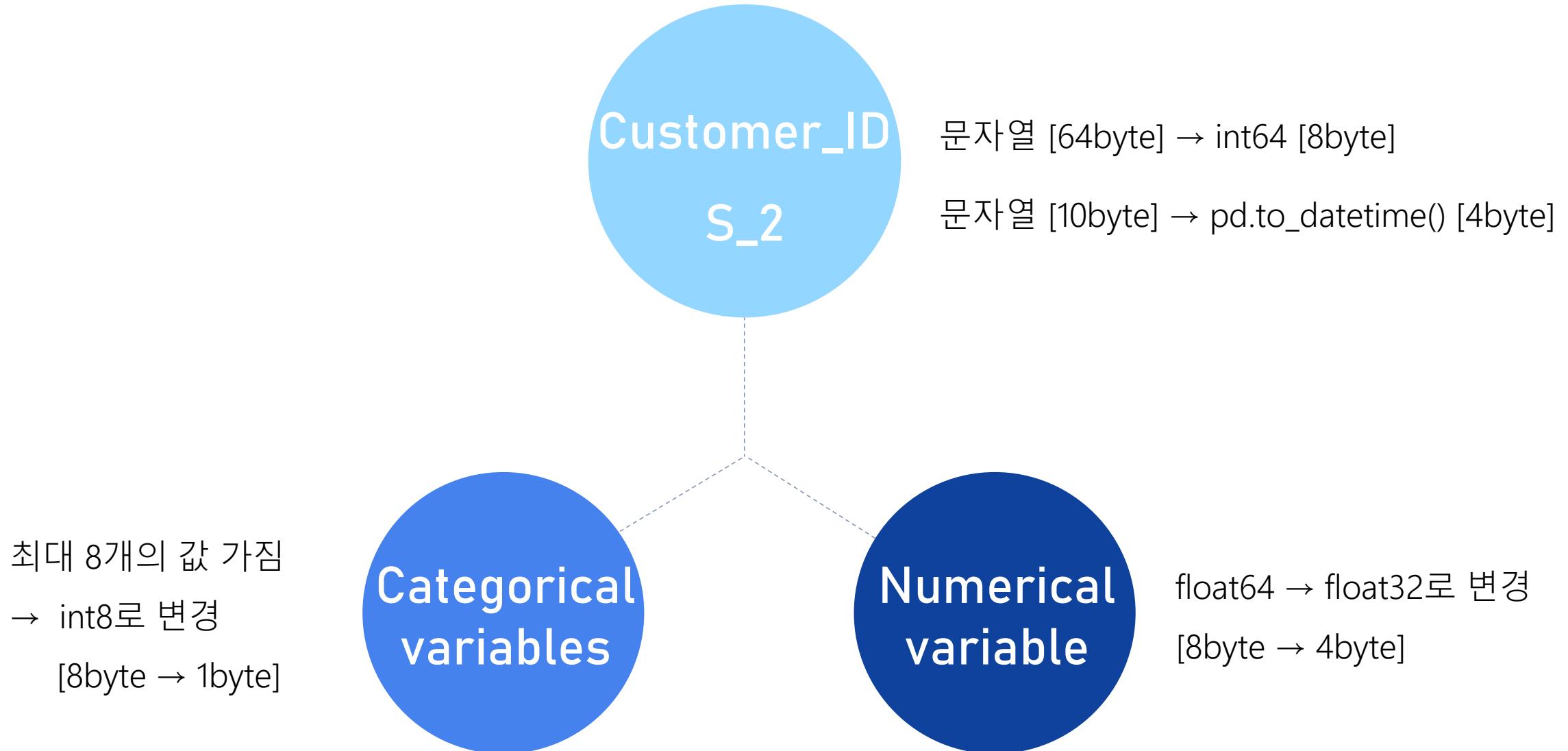


특정 column을 선택해서 가져오는 형식
→ 선택하지 않은 column 데이터는 I/O가 발생하지 않음
→ 시간, 메모리 사용량 줄일 수 있음



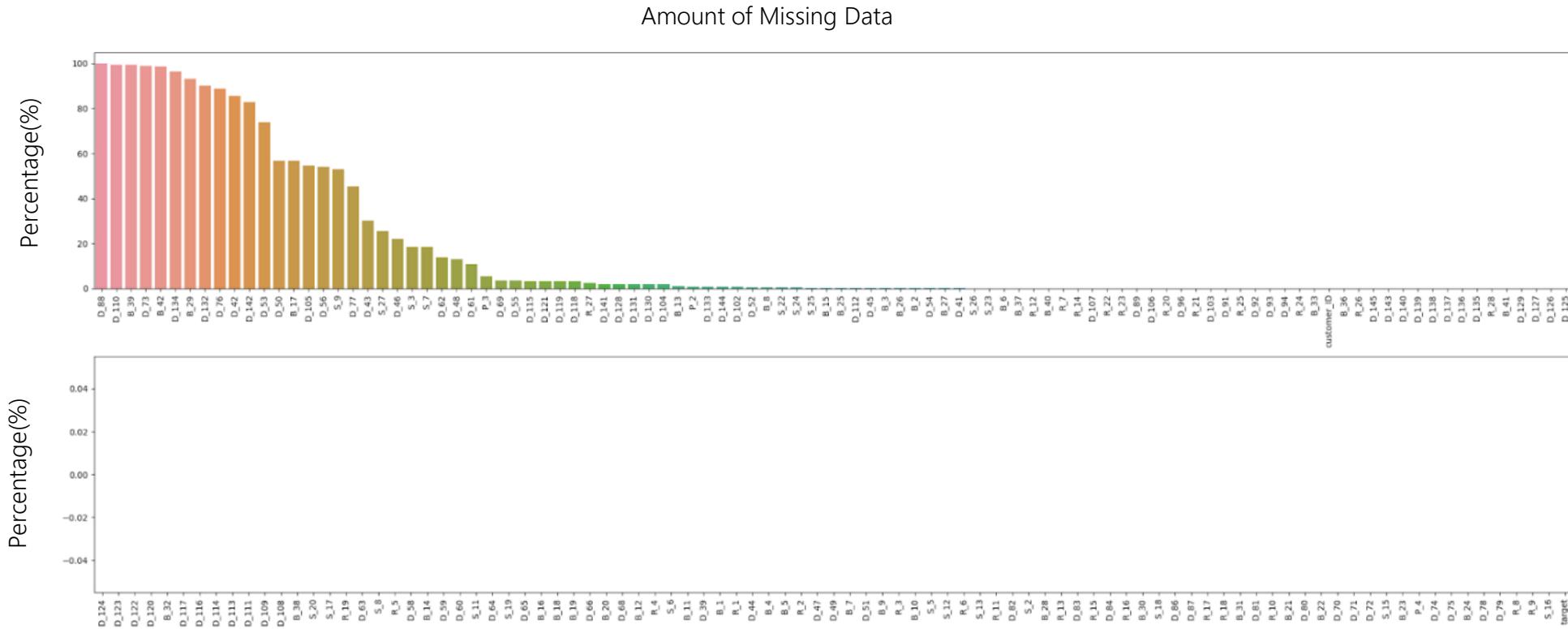
파일 형식이 dtype을 기억
→ int64에서 int8로 downcast하면 다음에도 파일을 int8로 불러옴

Part 2 >> Exploratory Data Analysis_ Methods of data reduction(2) : Reduce dtypes



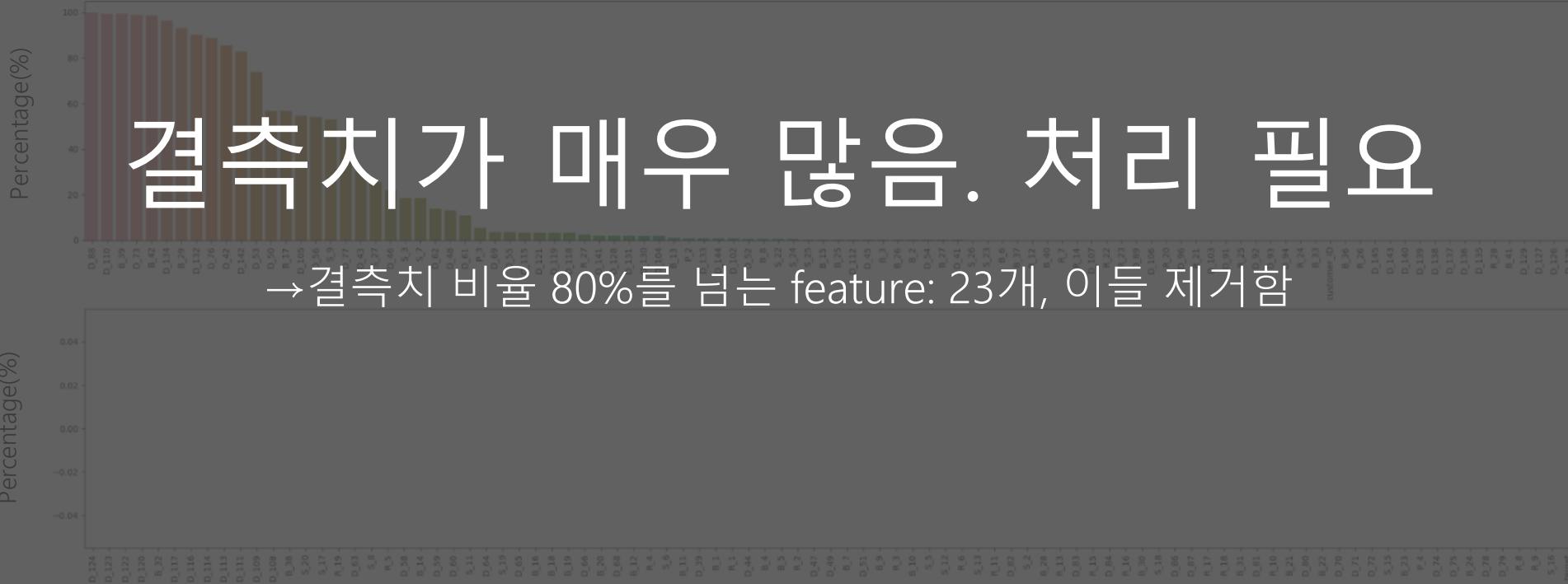
Part 2 >> Exploratory Data Analysis_ Missing Value

Verification_ Missing Value



“

Verification_ Missing Value



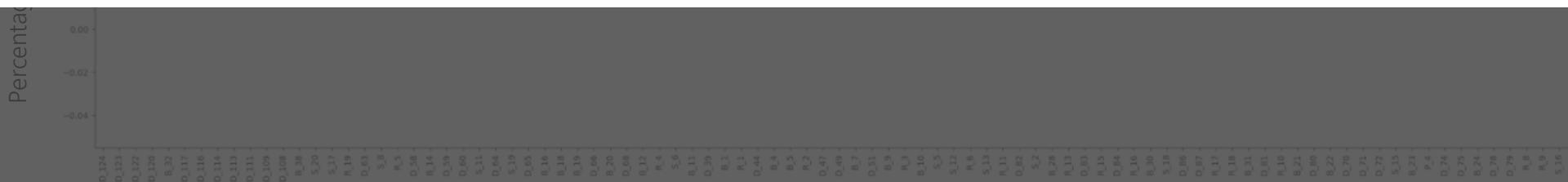
”

66

Verification_ Missing Value

```
# Removing Columns With NaNs Rate Higher Than Threshold
nan_pct_threshold = 80
to_remove_cols = list(nan_values_pct[nan_values_pct > nan_pct_threshold].index)
print(f"Columns With NaN Values Rate > {nan_pct_threshold}%%: {len(to_remove_cols)} Columns")
train_data = train_data.drop(columns=to_remove_cols).reset_index(drop=True)
```

Columns With NaN Values Rate > 80%: 23 Columns

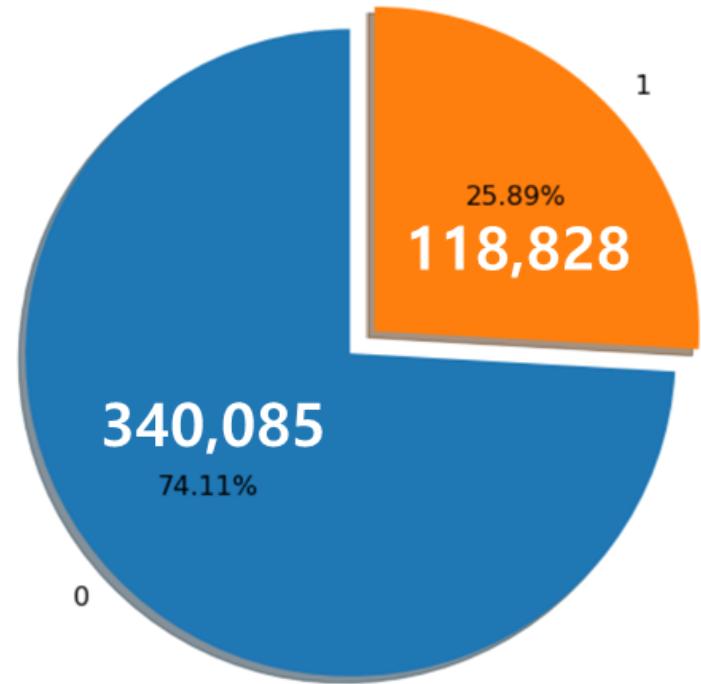


99

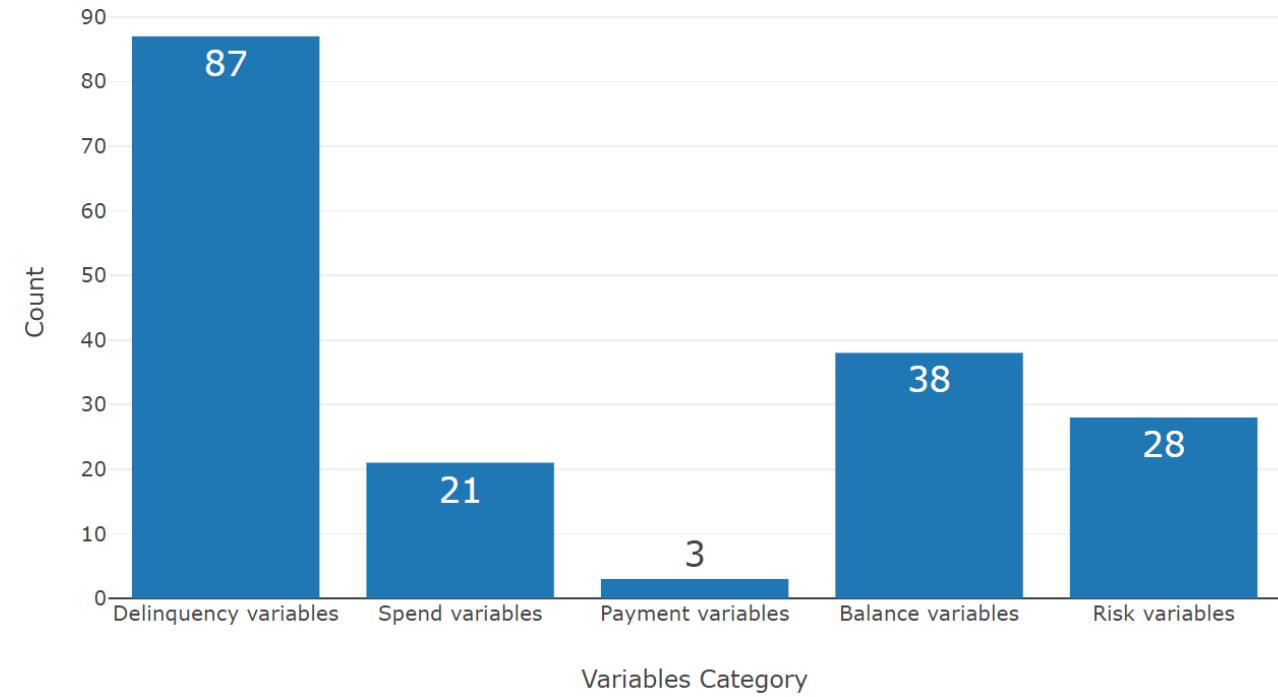
Part 2 >> Exploratory Data Analysis_ Missing Value

Verification_ After Missing Value Processing

Target Variable Distribution

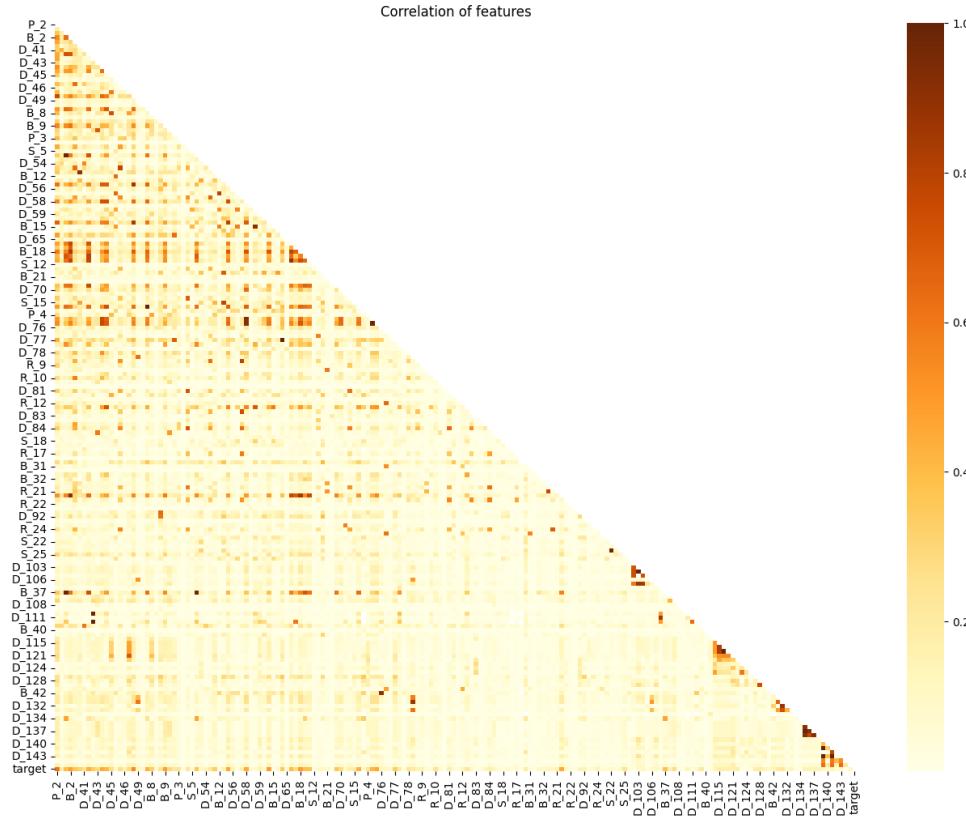


Variables Count By Category



Part 2 >> Exploratory Data Analysis_ Correlation

Verification_ Correlation between variables(heatmap)



insight

Feature 상관관계 파악 필요
→ Heatmap 이용해 시각화

상관관계가 뚜렷하지 않음

일부 색이 진한 점들은 다시
결과값 unstack해야 함

Part 2 >> Exploratory Data Analysis_ Statistical Values

Verification_ Correlation between Unstacked Variables

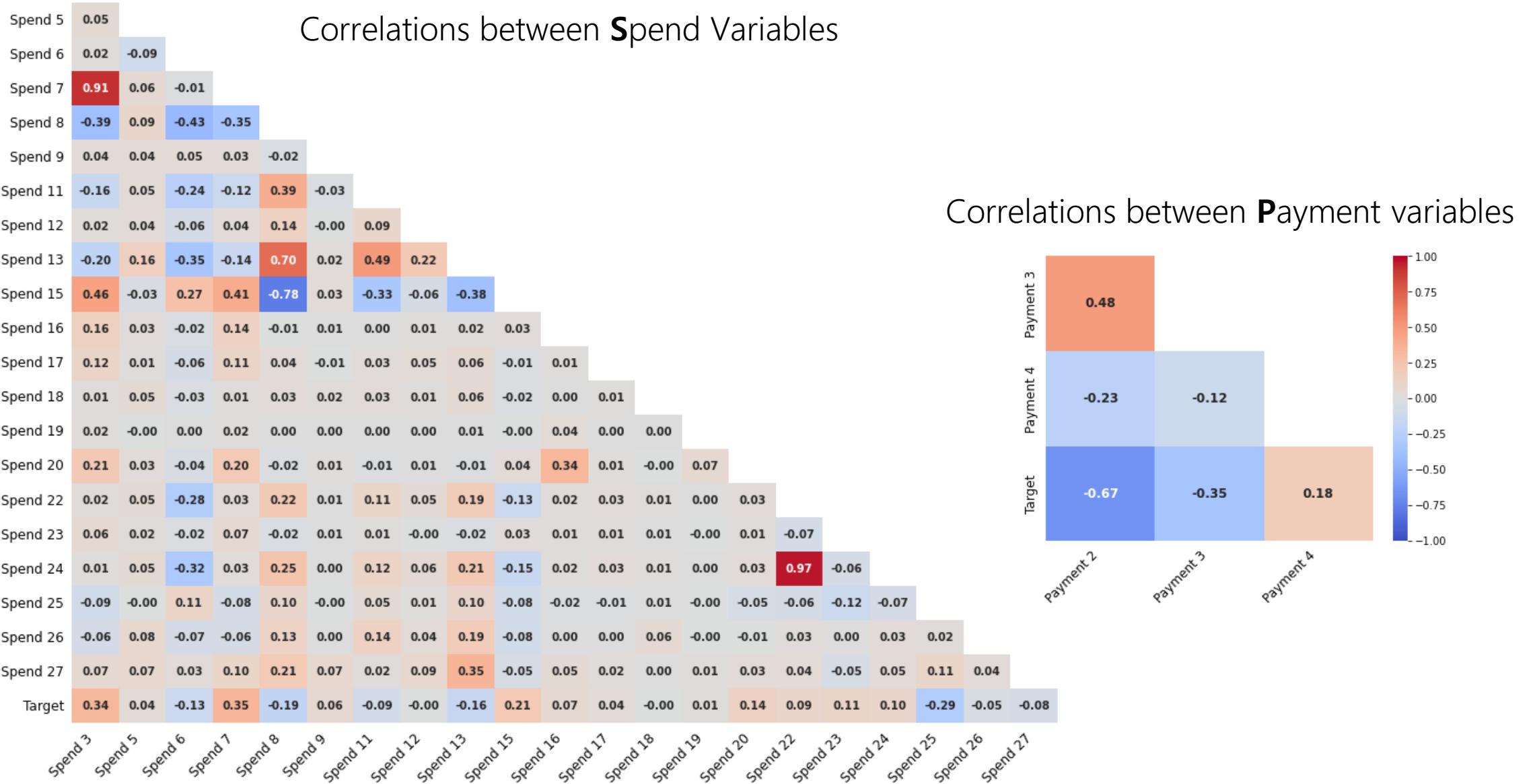
In [38]:

```
unstacked = correlations.unstack()
unstacked = unstacked.sort_values(ascending=False, kind="quicksort").drop_duplicates().head(25)
unstacked
```

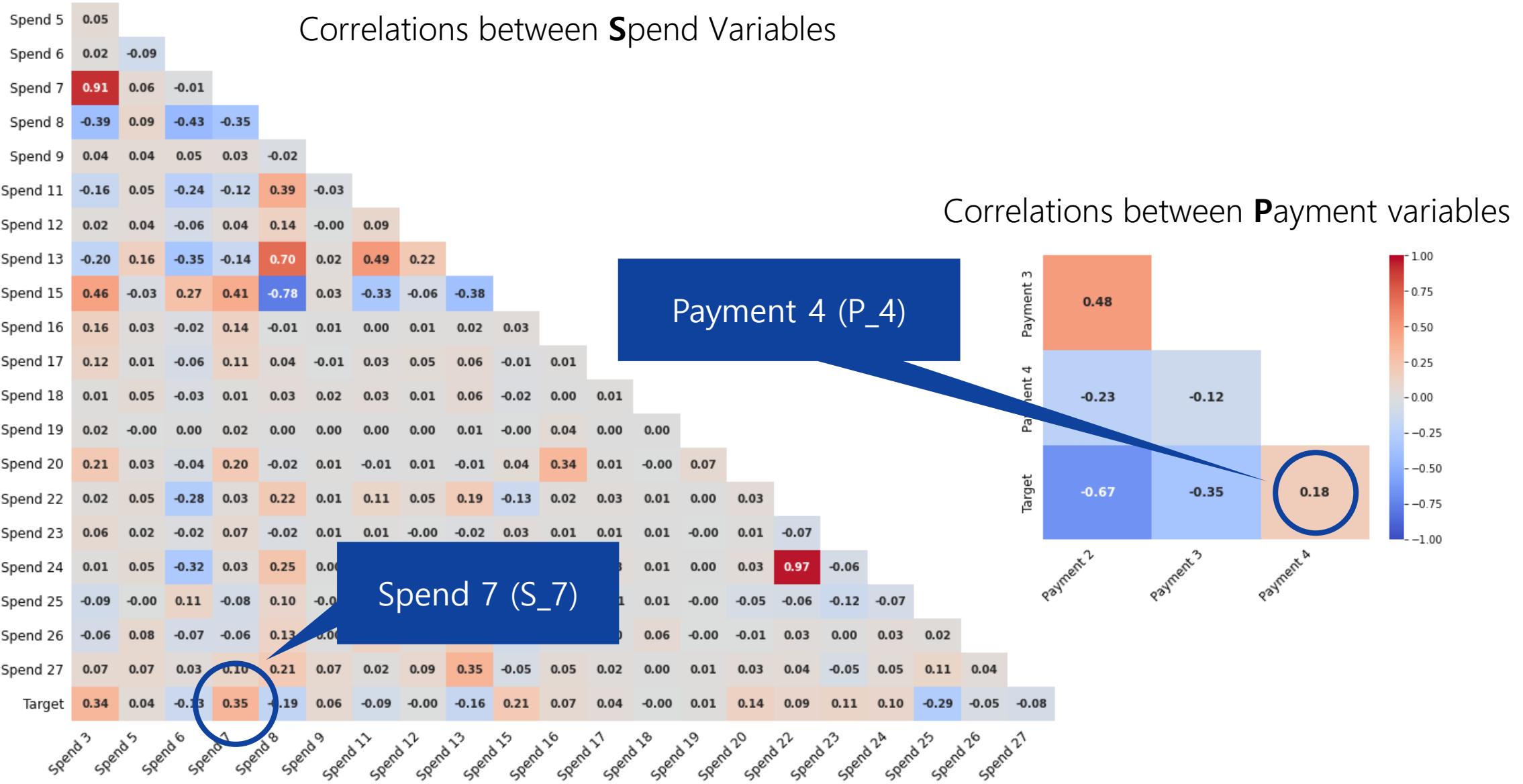
Out[38]:

P_2	P_2	1.000000
D_103	D_104	0.999779
D_77	D_62	0.999774
D_139	D_143	0.999629
	D_141	0.998261
D_143	D_141	0.997794
B_11	B_1	0.995475
B_23	B_7	0.995041
D_119	D_118	0.994861
B_1	B_37	0.993209
B_37	B_11	0.988153
D_75	D_74	0.987574
D_137	D_135	0.983981
S_22	S_24	0.978798
D_58	D_75	0.925743
	D_74	0.922671
D_138	D_137	0.918552
B_14	B_15	0.913476
B_33	B_2	0.912731
D_135	D_138	0.908674
D_137	D_136	0.907234
S_7	S_3	0.903646
D_135	D_136	0.903043
B_42	D_76	0.888606
B_20	B_16	0.888382
		dtype: float64

Part 2 >> Exploratory Data Analysis_ Statistical Values



Part 2 >> Exploratory Data Analysis_ Statistical Values



Part 2 >> Exploratory Data Analysis_ Statistical Values

Correlations between **Balance** variables

Correlations between Risk variables

Part 2 >> Exploratory Data Analysis_ Statistical Values

Correlations between **Balance** variables

Correlations between Balance variables

	Balance 1	Balance 2	Balance 3	Balance 4	Balance 5	Balance 6	Balance 7	Balance 8	Balance 9	Balance 10	Balance 11	Balance 12	Balance 13	Balance 14	Balance 15	Balance 16	Balance 17	Balance 18	Balance 19	Balance 20	Balance 21	Balance 22	Balance 23	Balance 24	Balance 25	Balance 26	Balance 27	Balance 28	Balance 29	Balance 30	Balance 31	Balance 32	Balance 33	Balance 34	Balance 35	Balance 36	Balance 37	Balance 38	Balance 39	Balance 40	Balance 41	Balance 42	Target				
Balance 1	-																																														
Balance 2	-0.63	-																																													
Balance 3	0.74	-0.72	-																																												
Balance 4	0.51	-0.53	0.57	-																																											
Balance 5	-0.06	0.11	-0.06	-0.01	-																																										
Balance 6	-0.08	0.12	-0.10	-0.10	0.01	-																																									
Balance 7	0.65	-0.62	0.65	0.69	-0.10	-0.12	-																																								
Balance 8	0.14	-0.22	0.12	0.11	-0.09	-0.02	0.15	-																																							
Balance 9	0.55	-0.64	0.57	0.45	-0.06	-0.09	0.51	0.29	-																																						
Balance 10	-0.03	0.04	-0.03	-0.03	0.01	0.05	-0.04	-0.00	-0.03	-																																					
Balance 11	1.00	-0.60	0.72	0.51	-0.06	-0.08	0.63	0.12	0.52	-0.02	-																																				
Balance 12	-0.04	0.05	-0.03	0.02	0.36	0.02	-0.06	-0.05	-0.02	0.01	-0.03	-																																			
Balance 13	-0.03	0.05	-0.04	0.03	0.42	0.04	-0.06	-0.05	-0.02	0.02	-0.02	0.91	-																																		
Balance 14	0.24	-0.18	0.23	0.30	0.43	-0.03	0.11	-0.08	0.16	-0.01	0.27	0.57	0.60	-																																	
Balance 15	0.01	-0.02	0.05	0.15	0.45	-0.00	-0.02	-0.11	-0.04	-0.00	0.02	0.61	0.63	0.90	-																																
Balance 16	0.59	-0.79	0.74	0.54	-0.06	0.12	0.59	0.14	0.59	-0.04	0.57	-0.03	-0.03	0.23	0.06	-																															
Balance 17	0.39	-0.69	0.47	0.33	-0.08	-0.14	0.40	0.11	0.42	-0.04	0.37	-0.07	-0.06	0.15	0.04	0.47	-																														
Balance 18	-0.58	0.86	-0.67	-0.66	0.07	0.15	-0.68	-0.20	0.60	0.05	-0.55	0.03	0.03	-0.23	-0.07	-0.79	-0.68	-																													
Balance 19	0.72	-0.64	0.80	0.49	-0.08	-0.09	0.64	0.17	0.54	-0.03	0.70	-0.05	-0.05	0.11	-0.04	0.66	0.33	0.58	-																												
Balance 20	0.63	-0.79	0.77	0.56	-0.06	-0.11	0.61	0.09	0.58	-0.03	0.61	-0.03	-0.04	0.24	0.07	0.90	0.54	0.76	0.67	-																											
Balance 21	0.10	-0.07	0.05	0.04	-0.01	-0.01	0.07	0.05	0.07	-0.00	0.10	-0.01	-0.00	0.01	-0.01	0.03	0.05	-0.06	0.05	0.02	-																										
Balance 22	0.59	-0.62	0.74	0.49	-0.05	-0.08	0.51	0.09	0.48	-0.02	0.59	-0.02	-0.02	0.25	0.09	0.59	0.37	-0.57	0.66	0.63	0.05	-																									
Balance 23	0.65	-0.61	0.65	0.73	-0.10	-0.11	0.99	0.13	0.51	-0.04	0.64	-0.05	-0.06	0.13	-0.01	0.59	0.40	0.68	0.64	0.61	0.06	0.52	-																								
Balance 24	0.06	-0.07	0.07	0.07	-0.02	-0.01	0.05	0.04	0.11	-0.00	0.06	0.01	0.02	0.07	0.05	0.08	0.04	-0.09	0.03	0.07	0.06	0.04	0.05	-																							
Balance 25	0.54	-0.36	0.42	0.38	0.06	-0.05	0.30	0.03	0.46	-0.01	0.57	0.06	0.08	0.47	0.04	0.40	0.24	-0.38	0.34	0.40	0.04	0.39	0.32	0.06	-																						
Balance 26	0.00	-0.05	0.02	0.04	0.08	-0.00	-0.01	-0.02	0.00	-0.00	0.00	0.14	0.16	0.27	0.29	0.01	0.01	-0.05	-0.01	0.01	0.00	0.01	-0.01	0.05	0.03	-																					
Balance 27	0.00	-0.01	0.01	0.02	0.00	-0.00	-0.00	-0.00	0.00	-0.00	0.00	0.06	0.09	0.13	0.15	0.00	0.00	-0.01	-0.00	0.00	0.00	0.00	-0.00	0.02	0.01	0.23	-																				
Balance 28	0.36	-0.35	0.40	0.72	0.23	-0.07	0.44	-0.06	0.28	-0.02	0.38	0.32	0.34	0.70	0.57	0.40	0.23	0.52	0.29	0.41	0.01	0.39	0.49	0.07	0.45	0.17	0.08	-																			
Balance 29	-0.01	-0.07	0.03	0.09	0.00	-0.01	-0.01	0.02	0.02	-0.01	-0.01	0.03	0.04	0.11	0.11	0.03	0.02	-0.10	-0.02	0.03	-0.00	0.03	-0.01	0.05	0.02	0.07	0.04	0.11	-																		
Balance 30	-0.05	0.06	-0.05	-0.05	0.01	0.01	-0.05	-0.03	-0.08	0.00	-0.05	0.00	-0.01	-0.03	-0.02	-0.05	-0.04	0.06	0.03	-0.05	-0.02	-0.05	0.57	-0.03	-0.03	0.01	-0.04	-0.03	-																		
Balance 31	0.13	-0.22	0.24	0.14	-0.03	-0.03	0.21	0.04	0.15	-0.01	0.11	-0.02	-0.02	-0.01	-0.02	0.23	0.10	-0.19	0.30	0.27	-0.01	0.16	0.21	-0.01	0.02	-0.01	0.00	0.06	-0.01	0.00	-																
Balance 32	-0.57	0.92	-0.65	-0.50	0.06	0.12	-0.55	-0.21	-0.61	0.04	-0.54	0.02	0.02	-																																	
Balance 33	0.07	-0.00	0.01	0.08	0.00	-0.01	0.03	0.02	0.06	-0.00	0.08	0.00	0.01	-																																	
Balance 34	0.99	-0.63	0.74	0.51	-0.06	-0.08	0.65	0.14	0.55	-0.03	0.99	-0.04	-0.03	-																																	
Balance 35	-0.42	0.03	-0.03	-0.21	0.07	0.10	-0.28	-0.05	0.53	0.06	-0.42	0.11	0.01	-																																	
Balance 36	0.03	-0.04	0.04	0.05	-0.01	-0.01	0.08	0.01	0.03	-0.00	0.03	0.00	0.01	-																																	
Balance 37	0.99	-0.63	0.74	0.51	-0.06	-0.08	0.65	0.14	0.55	-0.03	0.99	-0.04	-0.03	-																																	
Balance 38	-0.42	0.03	-0.03	-0.21	0.07	0.10	-0.28	-0.05	0.53	0.06	-0.42	0.11	0.01	-																																	
Balance 39	0.03	-0.04	0.04	0.05	-0.01	-0.01	0.08	0.01	0.03	-0.00	0.03	0.00	0.01	-																																	
Balance 40	0.03	-0.04	0.04	0.05	-0.01	-0.01	0.08	0.01	0.03	-0.00	0.03	0.00	0.01	-																																	
Balance 41	0.04	-0.07	0.05	0.05	-0.02	-0.01	0.04	0.05	0.10	-0.00	0.04	0.01	0.02	-																																	
Balance 42	-0.10	0.15	-0.12	-0.12	0.11	0.02	-0.14	-0.06	-0.10	0.04	0.04	0.01	0.01	-0.15	-0.15	0.18	-0.11	-0.13	-0.02	-0.10	-0.13	-0.03	-0.00	-0.01	-0.01	-0.06	-0.05	0.02	-0.03	0.13	-0.03	-0.10	0.12	-0.08	-0.03												
Target	0.45	-0.56	0.51	0.46	-0.08	-0.08	0.50	0.31	0.54	-0.03	0.43	-0.03	-0.03	0.11	0.01	0.47	0.33	0.55	0.45	0.46	0.09	0.41	0.50	0.14	0.22	0.04	0.02	0.27	0.09	-0.10	0.13	-0.52	0.05	0.45	-0.12	0.04	0.12	-0.01									

Correlations between Risk variables

Part 2 >> Exploratory Data Analysis_ Statistical Values

Verification_ Statistical value (count,mean,std,min,max,Quantiles)_target ' 0 '

```
ex_customer_data[ex_customer_data[ "target" ] == 0][b_cols[ :10 ]].describe()
```

	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
count	9032.000000	9032.000000	9032.000000	9032.000000	9032.000000	9032.000000	9032.000000	8.989000e+03
mean	0.081276	0.719796	0.081671	0.130928	0.097953	0.161655	0.133657	3.310980e-01
std	0.156970	0.355266	0.181874	0.182206	0.331847	0.319970	0.192023	4.688353e-01
min	-0.141469	0.000184	0.000003	0.000017	0.000007	-0.000552	-0.096913	2.764867e-07
25%	0.007554	0.620152	0.004455	0.019144	0.007463	0.037204	0.024911	3.603379e-03
50%	0.021453	0.817121	0.008435	0.055663	0.017671	0.140455	0.045870	7.330273e-03
75%	0.061725	1.003689	0.041836	0.161069	0.072235	0.199933	0.156390	1.002241e+00
max	1.320823	1.009999	1.171260	1.283849	12.974426	20.331217	1.252293	1.010181e+00

Part 2 >> Exploratory Data Analysis_ Statistical Values

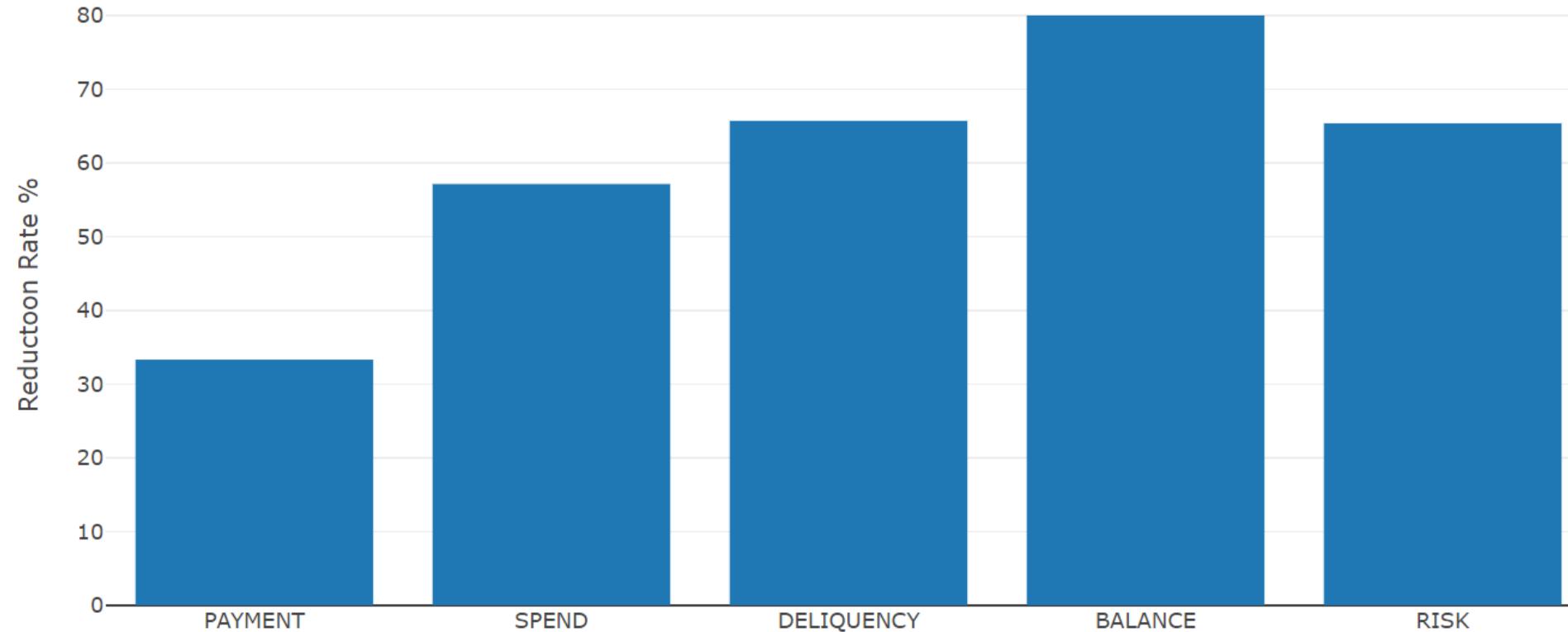
Verification_ Statistical value (count,mean,std,min,max,Quantiles)_target '1'

```
ex_customer_data[ex_customer_data[ "target" ] == 1][b_cols[ :10 ]].describe()
```

	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
count	3030.000000	3030.000000	3030.000000	3030.000000	3030.000000	3030.000000	3030.000000	3022.000000
mean	0.257518	0.299894	0.298945	0.328520	0.031458	0.043737	0.355820	0.734537
std	0.279484	0.375638	0.300259	0.296676	0.084002	0.085808	0.252973	0.444182
min	-0.046796	0.000034	0.000085	0.000082	0.000006	-0.002122	0.000510	0.000006
25%	0.054462	0.028687	0.036761	0.120258	0.007023	0.009662	0.159904	0.009335
50%	0.136546	0.066485	0.220646	0.251262	0.011993	0.017531	0.309124	1.003016
75%	0.378923	0.810611	0.455126	0.450153	0.023515	0.038585	0.510301	1.006530
max	1.323411	1.009960	1.258546	2.187350	2.188328	1.926984	1.252394	1.010065

Part 2 >> Exploratory Data Analysis_ Outlier detection for skewed data

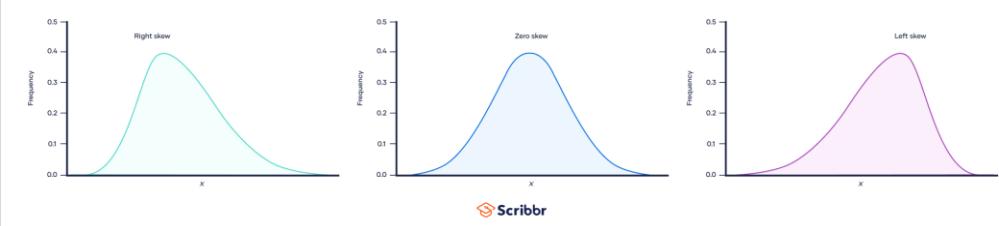
Verification_ Skewness Reduction Rate for each Variable Category



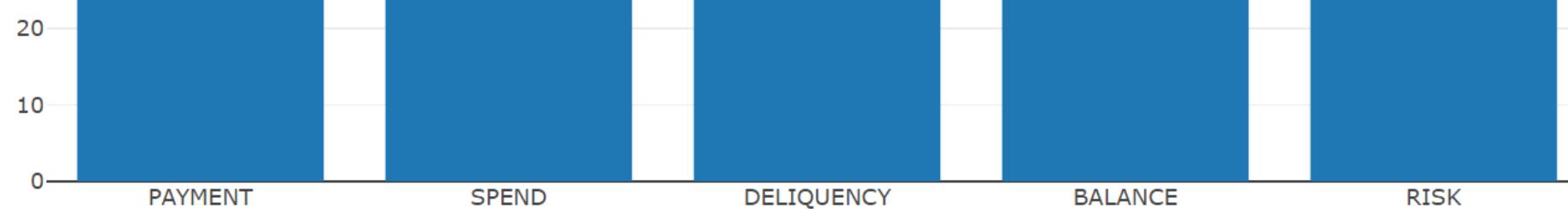
Part 2 >> Exploratory Data Analysis_ Outlier detection for skewed data

Verification_ Skewness Rate

Rate for each Variable Category

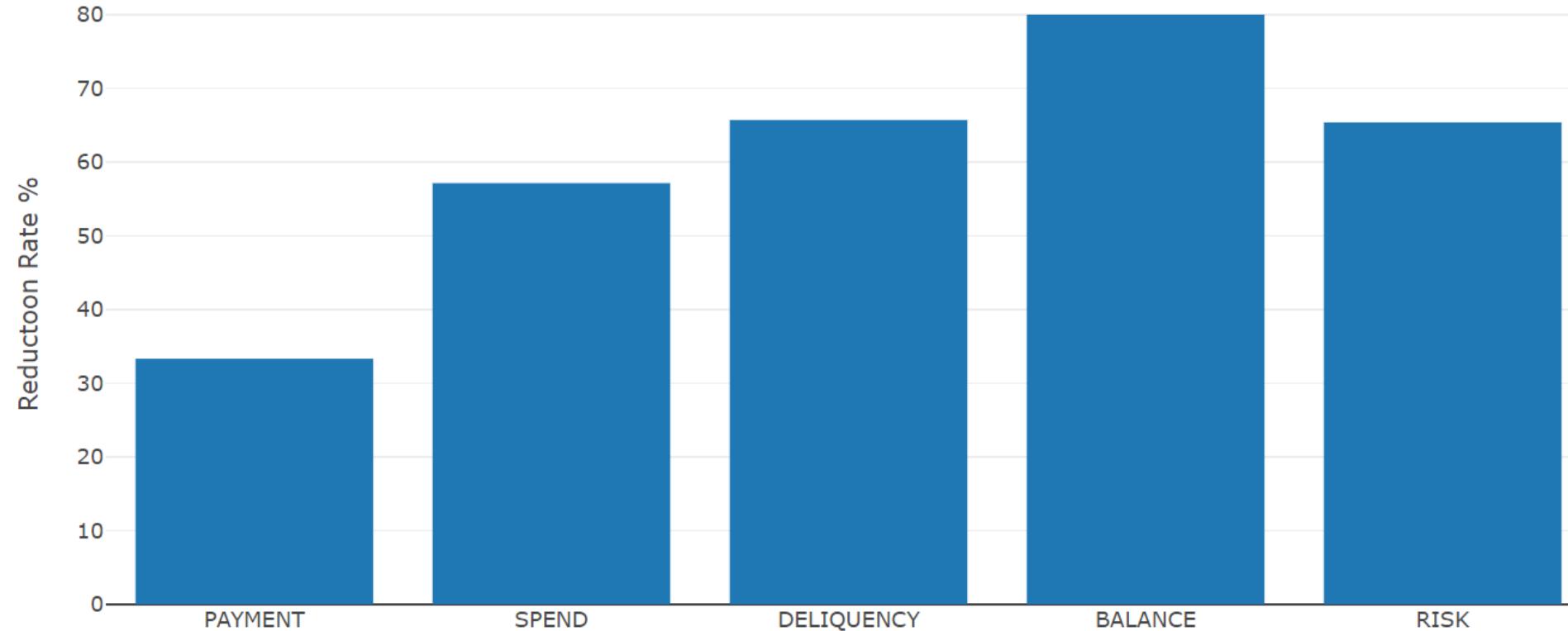


- deviate from the symmetrical bell curve
- Outliers(nonsense) → skewness



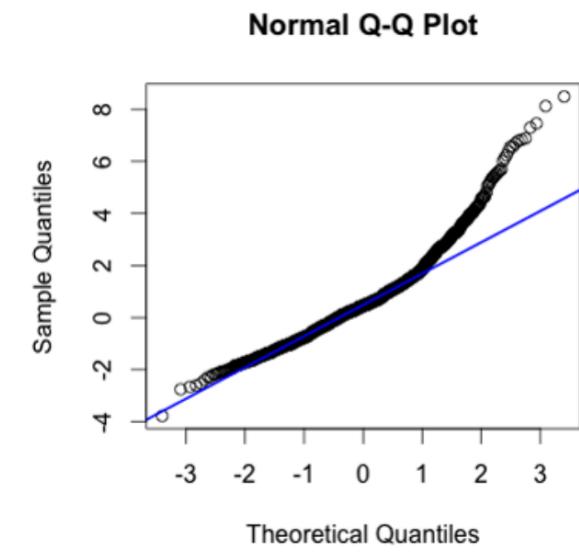
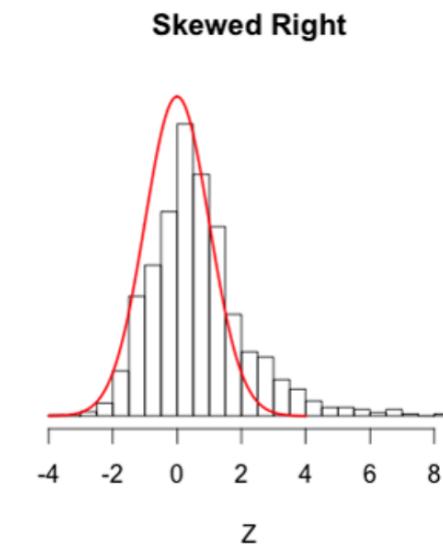
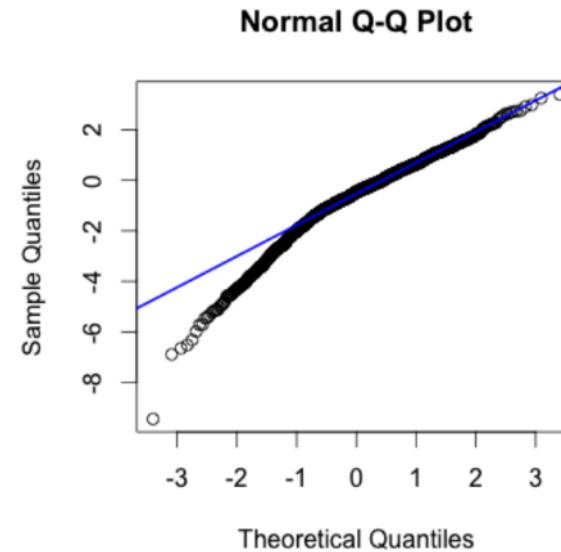
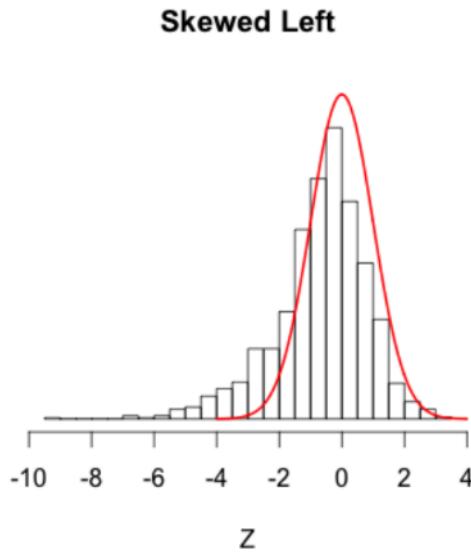
Part 2 >> Exploratory Data Analysis_ Outlier detection for skewed data

Verification_ Skewness Reduction Rate for each Variable Category



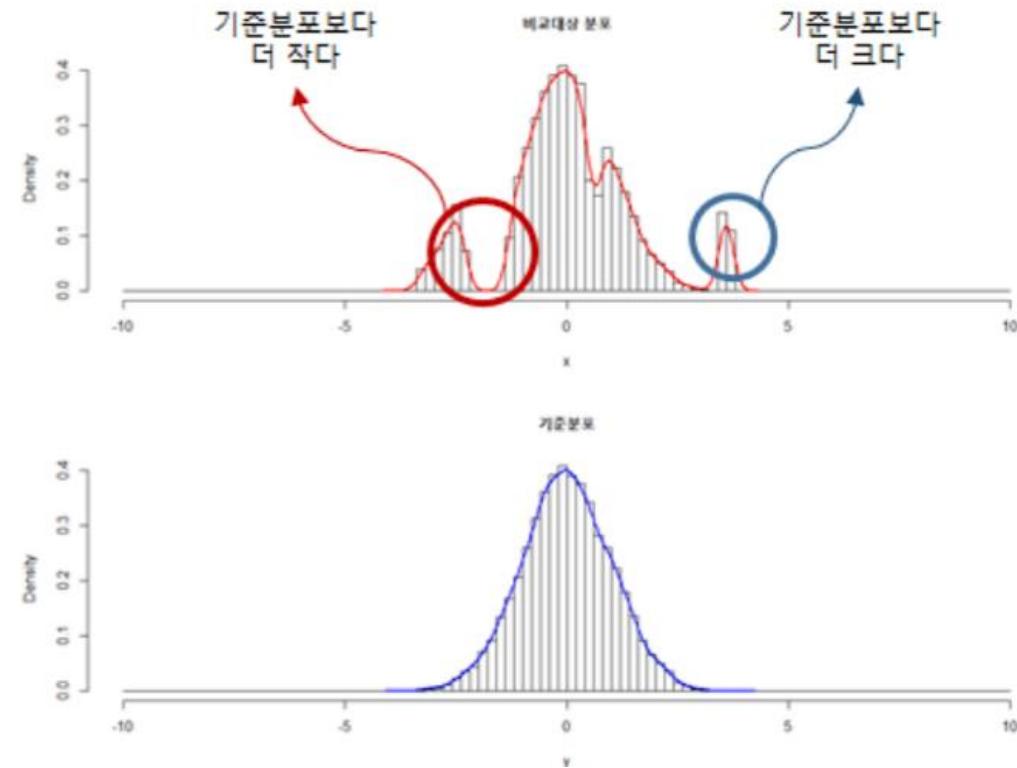
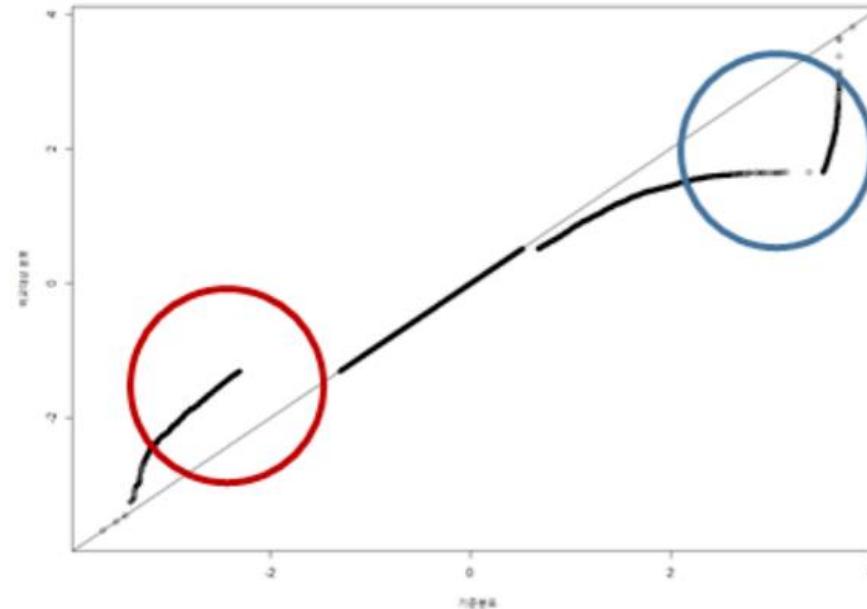
Part 2 >> Exploratory Data Analysis_ Understanding Features with Quantiles

Verification_ Quantile-Quantile plot



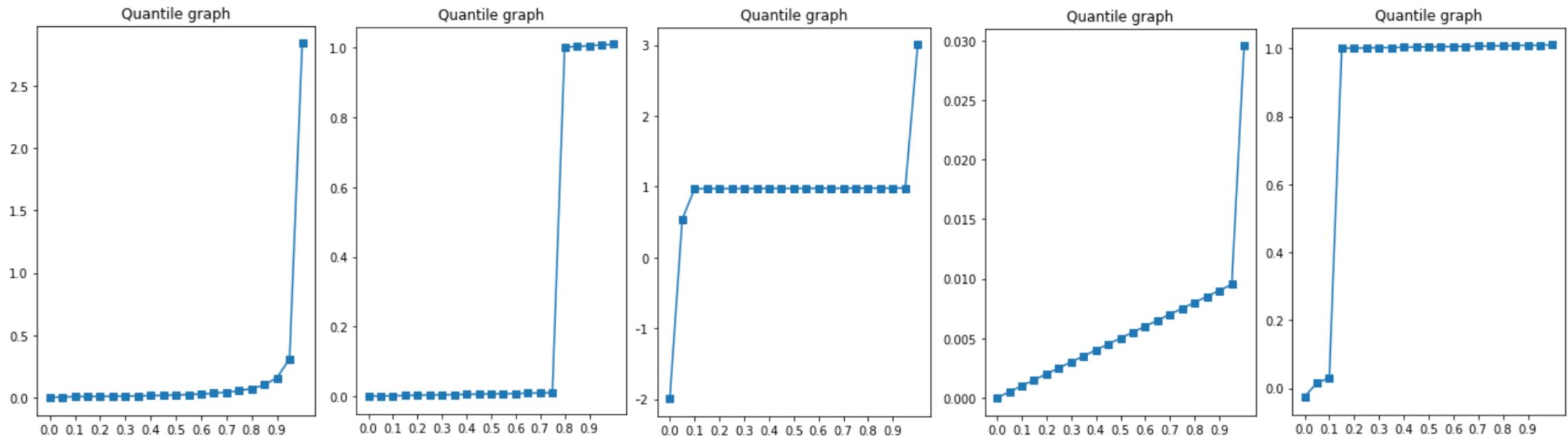
Part 2 >> Exploratory Data Analysis_ Understanding Features with Quantiles

Verification_ Quantile-Quantile plot



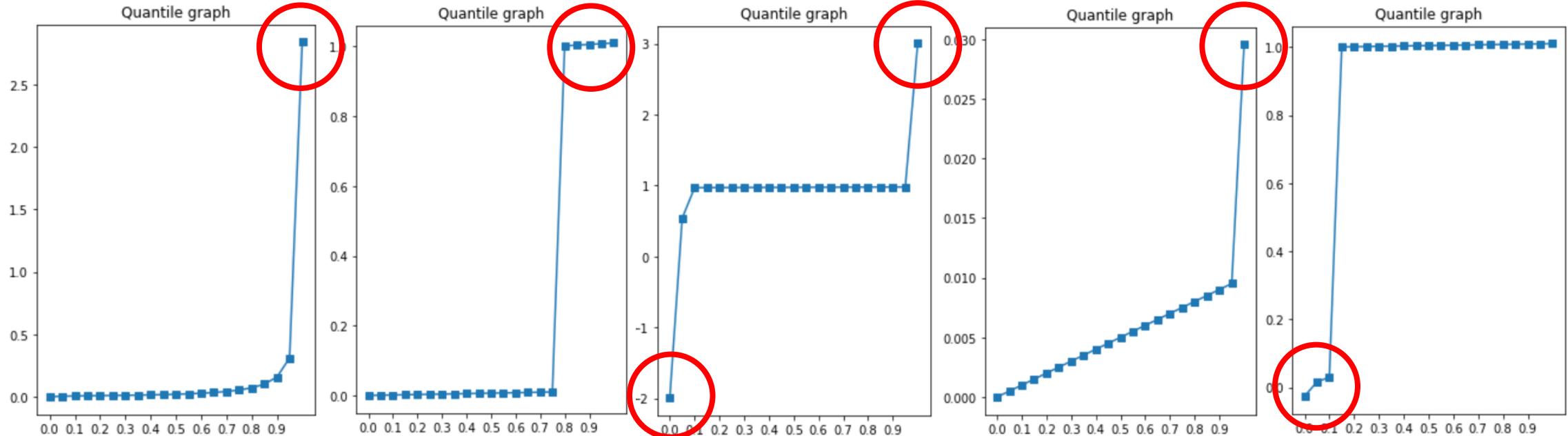
Part 2 >> Exploratory Data Analysis_ Understanding Features with Quantiles

Verification_ Quantile graph for Outlier detection



Part 2 >> Exploratory Data Analysis_ Understanding Features with Quantiles

Verification_ Quantile graph for Outlier detection



Part 2 >> Exploratory Data Analysis_ Understanding Features with Quantiles

Verification_ Quantile graph for Outlier detection

“

Outlier(이상치) 갖는 feature 많음

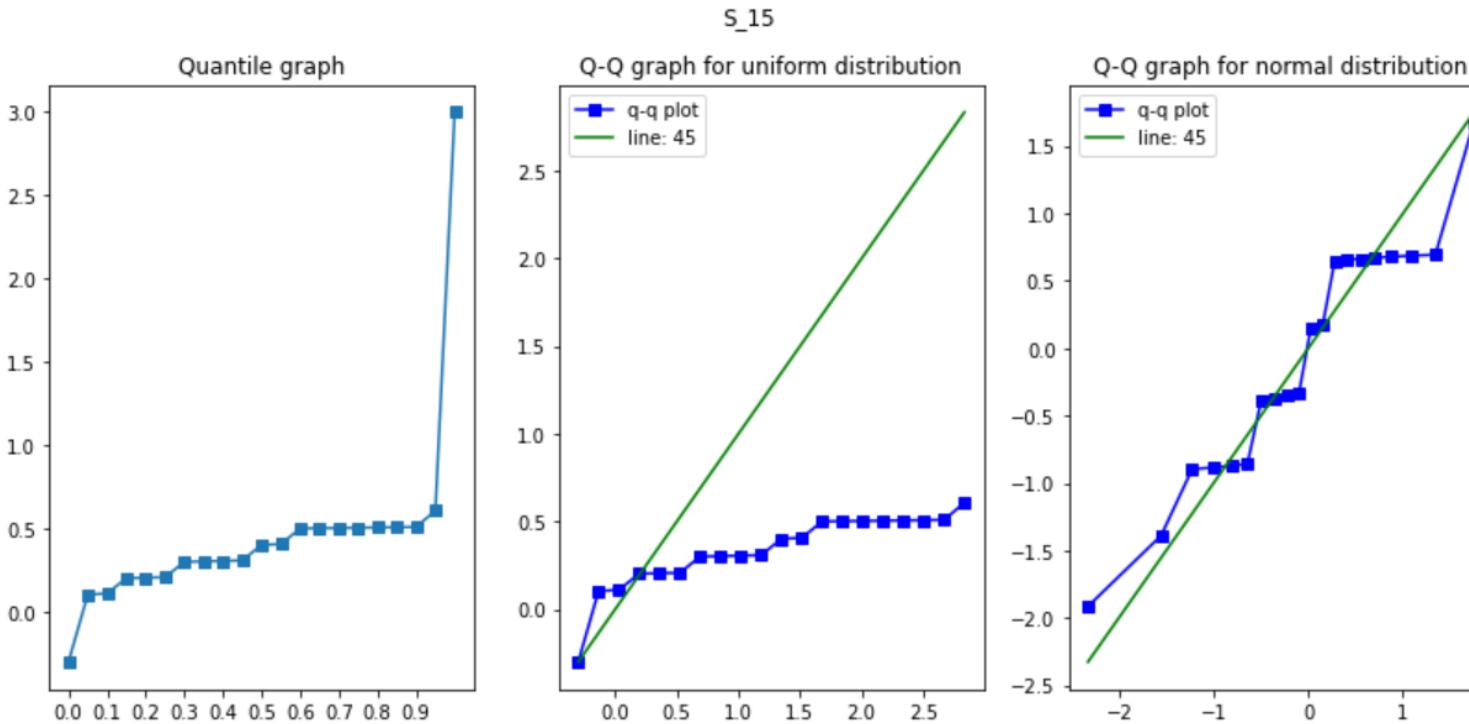


”

Part 2 >> Exploratory Data Analysis_ Understanding Features with Quantiles

Verification_ Highly discrete values

insight

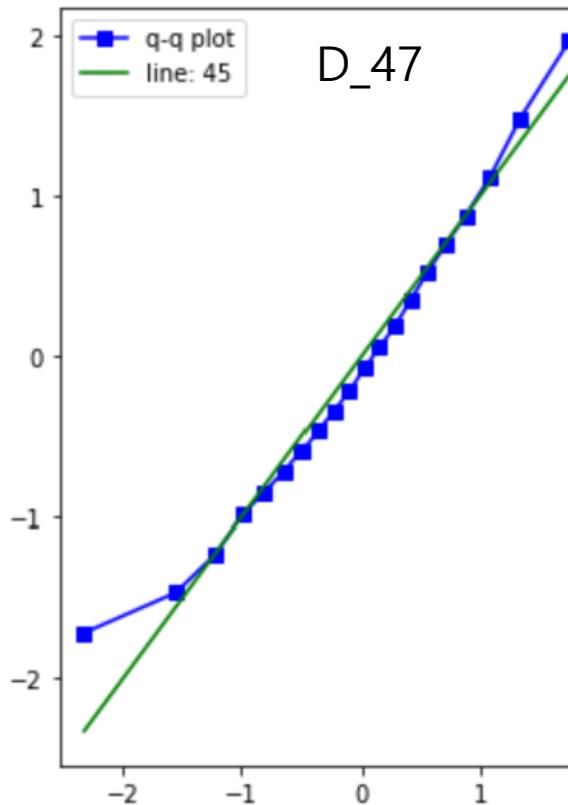
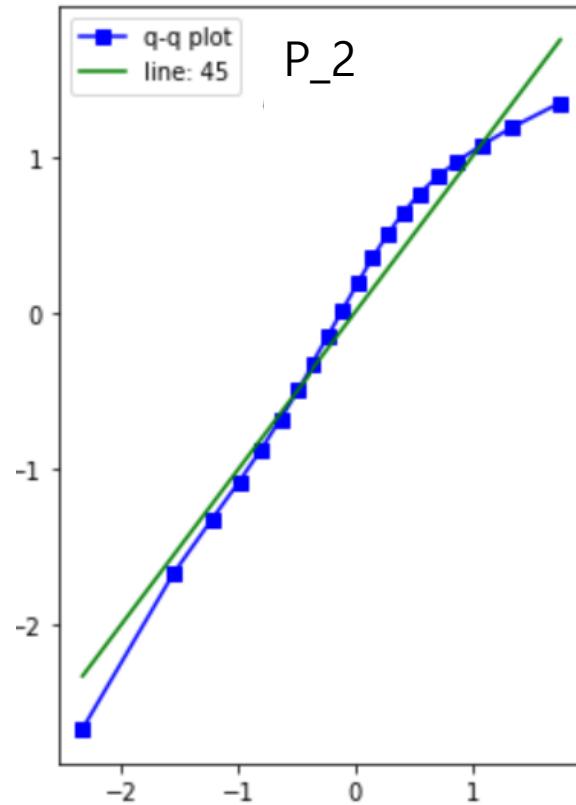


S15는 최소 7개의 이산 값들
가지는 것으로 보임

Part 2 >> Exploratory Data Analysis_ Understanding Features with Quantiles

Verification_ Q-Q graph for Normal Distribution

insight



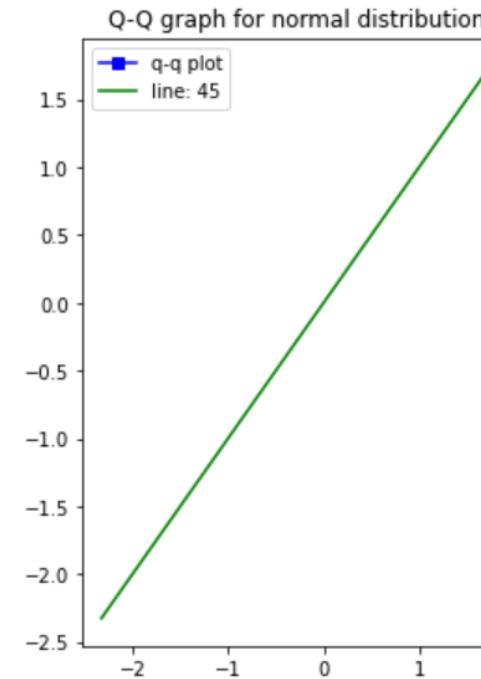
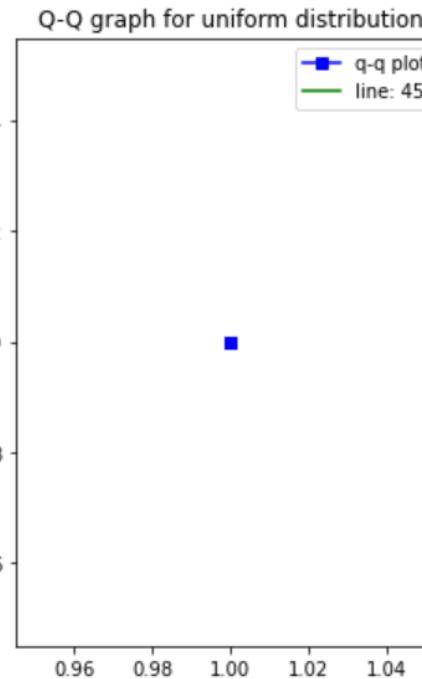
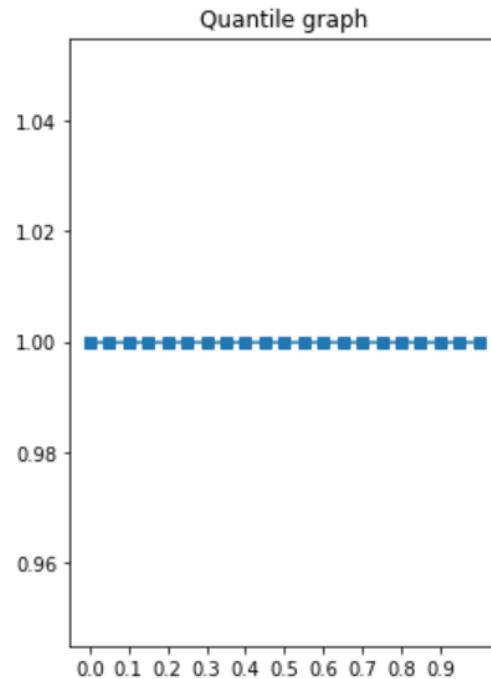
P_2, D_47은
가우시안분포(=정규분포)
따르는 것으로 판단됨

Part 2 >> Exploratory Data Analysis_ Understanding Features with Quantiles

Verification_ Q-Q graph for some Features

insight

B_31



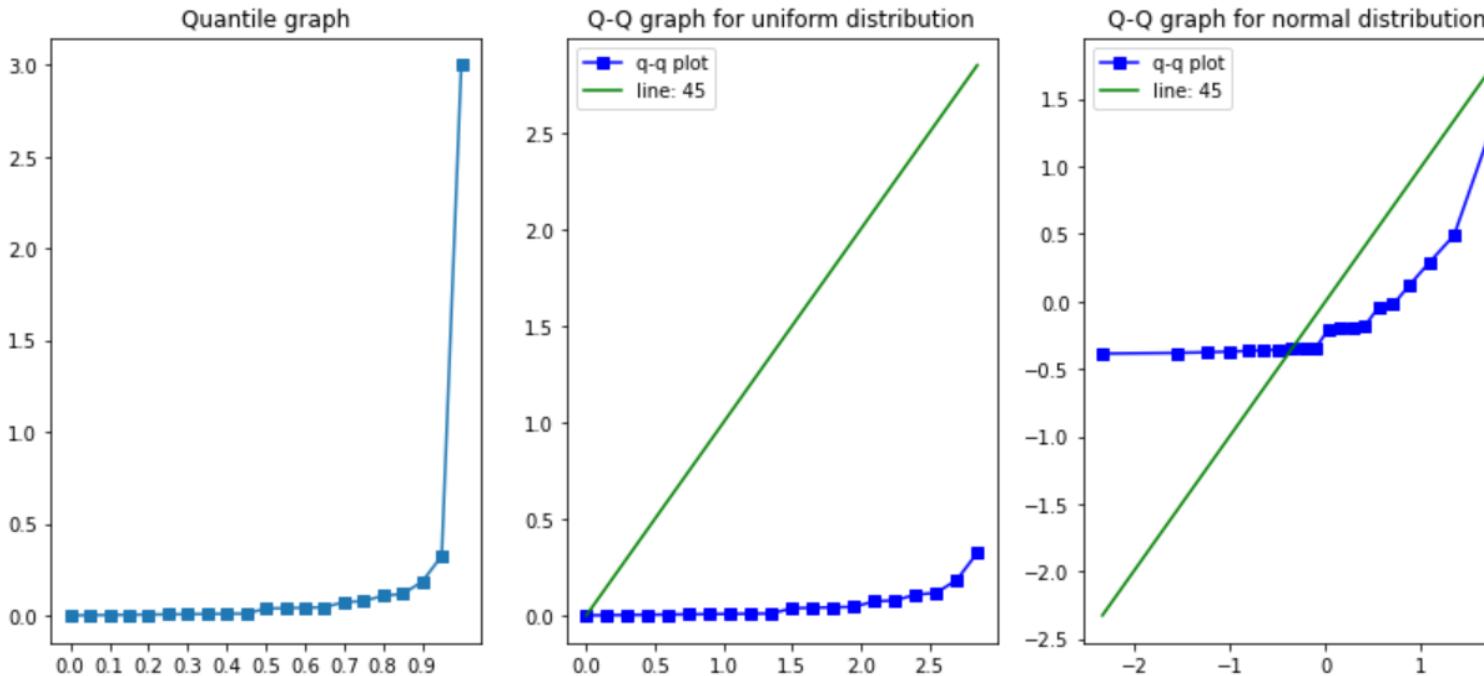
B_31 는 모든 분위 수에서
단일 값을 가진다고 판단됨

Part 2 >> Exploratory Data Analysis_ Understanding Features with Quantiles

Verification_ Q-Q graph for some Features

insight

R_26



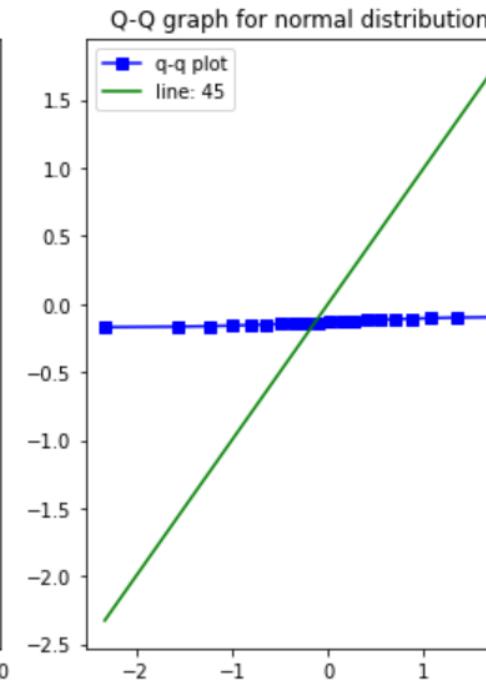
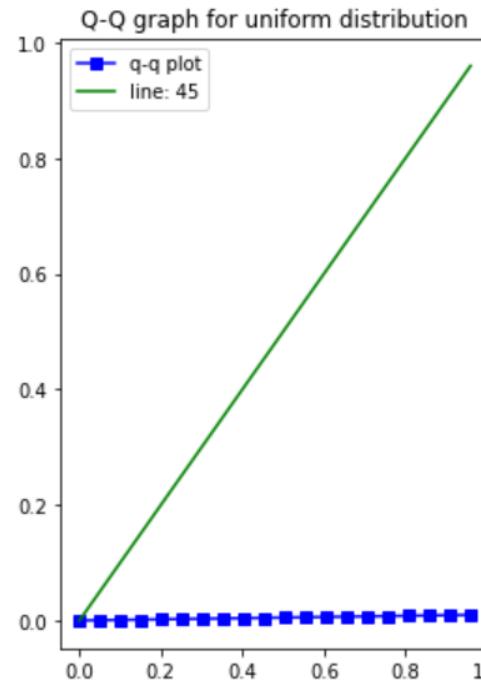
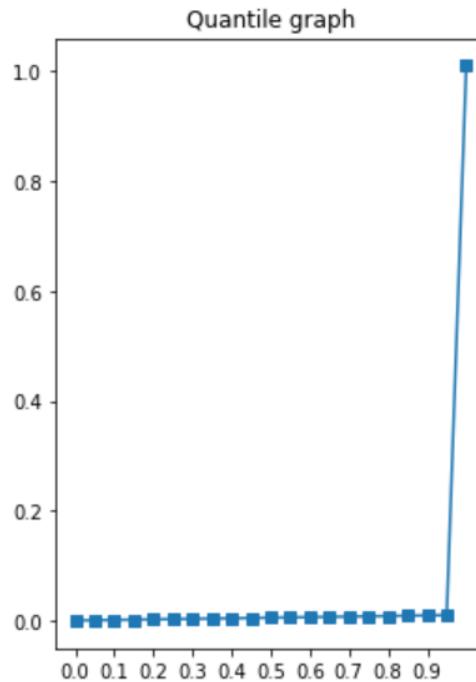
R_26 는 확대 시 6개의
discrete value가 있는 것
으로 판단됨

Part 2 >> Exploratory Data Analysis_ Understanding Features with Quantiles

Verification_ Q-Q graph for some Features

insight

D_94

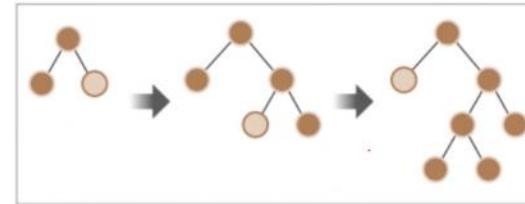
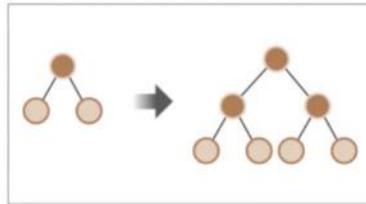


D_94, S_26 기울기 거의 0
→ 이상치 값이 매우 크다
고 판단됨

3

Insight about Machine Learning Model

Part 3 >> Generate Machine Learning Model



1 Random Forest

ensemble ⊃ bagging ⊃ RF

- subsample에 대해
다수의 결정 트리
classifier를 최적화
- 여러 추론 결과 평균
→ 예측 정확도를 개선
→ 과적합방지
- hyperparameter
xgbm보다 적음

2 XGBoost

Ensemble ⊃ boosting ⊃ GBM ⊃ XGBoost, LightGBM

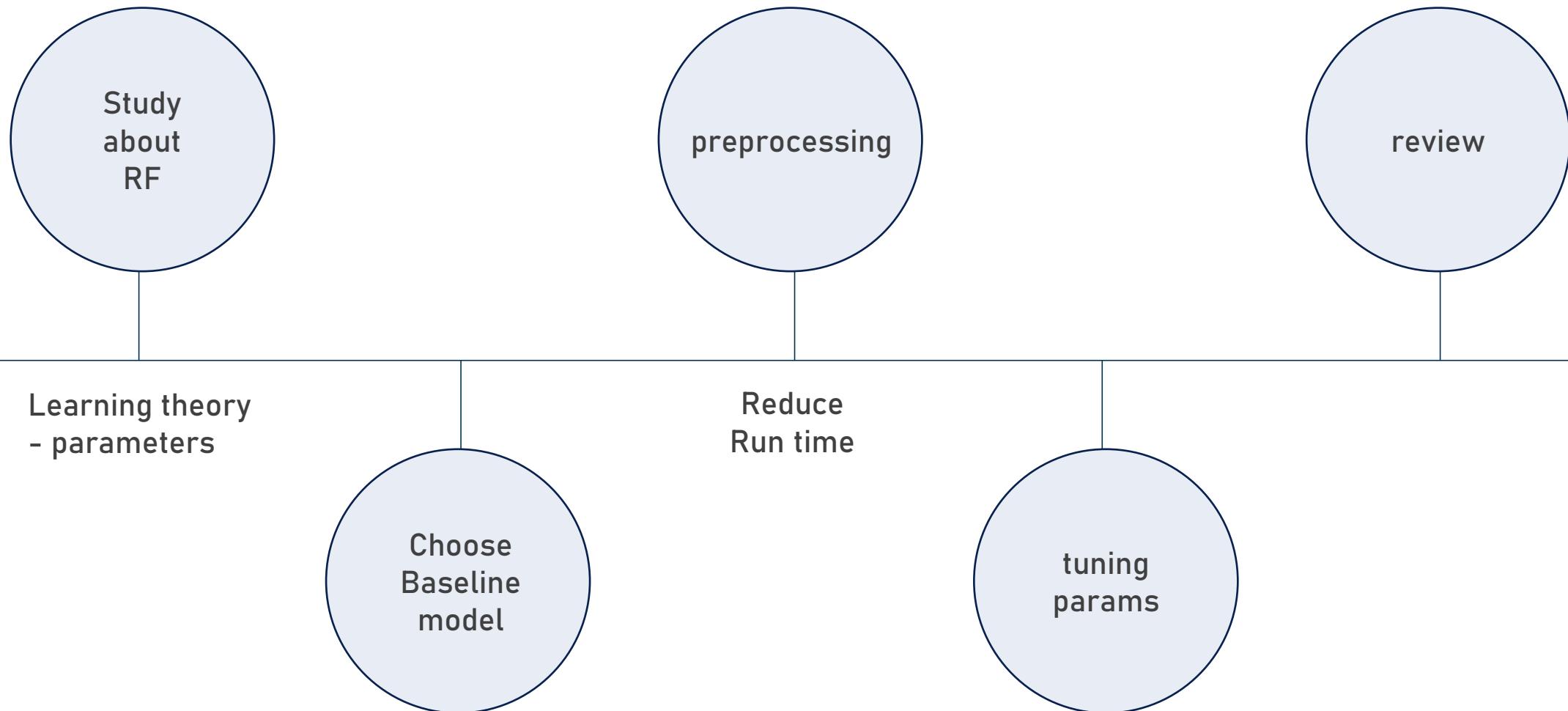
- 트리기반 앙상블
균형트리분할방식
- 병렬학습이 가능
→ GBM의 단점인 오랜
수행시간을 극복
- 예측 성능 우수
- 가지치기 기능 존재
조기중단 기능 존재
- 트리기반 앙상블
리프중심트리분할방식
- XGBoost보다 더 빠르고
메모리 사용량 적음
- 예측 성능 우수
- 10000건 이하의 dataset
처리시 과적합 가능성 ↑

3 LightGBM

4 CatBoost

- Categorical Boosting
범주형 feature 처리중점
- Gradient Boosting 기반
XGB, LGBM보다 우수
- Feature 자동 타깃인코딩
- 망각 결정 트리 사용
(oblivious decision tree)

Part 3 >> Generate Machine Learning Model



Part 3 >> Generate Machine Learning Model_ Random Forest

Missing Value Processing_ Remove Columns

In [12]:

```
#Lets remove columns if there are >90% of missing values.  
#Given that there are many columns with large number of missing values, it is impractical to go through every single one of them to determine whether it is useful.  
#Furthermore, we do not have information on the feature (e.g. actual name of the feature) except the type of variable  
#Brute force is thus a practical option to weed out columns with too many missing values.  
#Since about 33.1% of the data are defaults(66.9% non-defaults), it is safe to say that columns with >90% missing data are not useful.  
train=train_data.dropna(axis=1, thresh=int(0.90*len(train_data)))  
  
#Checking the shape of new train data  
train.shape  
## We are now left with 152 columns
```

Out[12]:

```
(5531451, 152)
```

Part 3 >> Generate Machine Learning Model_ Random Forest

Missing Value Processing_ Remove Columns

In [12]:

```
#Lets remove columns if there are >90% of missing values.  
#Given that there are many columns with large number of missing values, it is impractical to go through every single one of them to determine whether it is useful.  
#Furthermore, we do not have information on the feature (e.g. actual name of the feature) except the type of variable  
#Brute force is thus a practical option to weed out columns with too many missing values.  
#Since about 33.1% of the data are defaults(66.9% non-defaults), it is safe to say that columns with  
>90% missing data are not useful.  
train=train_data.dropna(axis=1, thresh=int(0.90*len(train_data)))
```

```
#Checking the shape of new train data  
train.shape  
## We are now left with 152 columns
```

Out[12]:

(5531451, 152)

191 columns → 152 columns

Part 3 >> Generate Machine Learning Model_ Random Forest

Missing Value Processing_ Fill Missing Values, Drop S_2, groupby

In [13]:

```
# There are multiple transactions. Lets take only the latest transaction from each customer.  
# Latest transaction may have missing values, we will perform forward fill for those missing values.  
# We perform forward fill as the last known value is likely to be brought forward to the next transaction.  
# We then do a backfill if the first row happens to be NA.  
train=train.set_index(['customer_ID'])  
train=train.ffill().bfill()  
train=train.reset_index()  
train=train.groupby('customer_ID').tail(1)  
train=train.set_index(['customer_ID'])  
  
#Drop date column since it is no longer relevant  
train.drop(['S_2'],axis=1,inplace=True)  
#Check for number of rows  
train.shape  
# We now have 458913 rows, which corresponds to the number of unique customers.
```

Out[13]:

(458913, 150)

Part 3 >> Generate Machine Learning Model_ Random Forest

Missing Value Processing_ Fill Missing Values, Drop S_2, groupby

In [13]:

```
# There are multiple transactions. Lets take only the latest transaction from each customer.  
# Latest transaction may have missing values, we will perform forward fill for those missing values.  
# We perform forward fill as the last known value is likely to be brought forward to the next transaction.  
# We then do a backfill if the first row happens to be NA.  
train=train.set_index(['customer_ID'])  
train=train.ffill().bfill()  
train=train.reset_index()  
train=train.groupby('customer_ID').tail(1)  
train=train.set_index(['customer_ID'])  
  
#Drop date column since it is no longer relevant  
train.drop(['S_2'],axis=1,inplace=True)  
#Check for number of rows  
train.shape  
# We now have 458913 rows, which corresponds to the number of unique customers.
```

Out[13]:

(458913, 150)

Rows : 5531451 → 458913

Columns : 152 → 150

Part 3 >> Generate Machine Learning Model_ Random Forest

one-hot Encoding for D_63 and D_64

- 문자열인 범주형 변수를 원핫인코딩

```
train_D63 = pd.get_dummies(train[['D_63']])
```

```
train = pd.concat([train, train_D63], axis=1)
```

```
train = train.drop(['D_63'], axis=1)
```

```
train_D64 = pd.get_dummies(train[['D_64']])
```

```
train = pd.concat([train, train_D64], axis=1)
```

```
train = train.drop(['D_64'], axis=1)
```

Part 3 >> Generate Machine Learning Model_ Random Forest

Remove Highly Correlated Features

- 상관관계 높은 feature 제거
- absolute correlation $\geq 90\%$ 이상인 열 drop

```
train_without_target=train.drop(['target'],axis=1)
cor_matrix = train_without_target.corr().abs()
upper_tri = cor_matrix.where(np.triu(np.ones(cor_matrix.shape),k=1).astype(np.bool))
#Drop out columns with absolute correlation of more than 90%
to_drop = [column for column in upper_tri.columns if any(upper_tri[column] > 0.90)]
train_drop_highcorr=train.drop(to_drop, axis=1)
train_drop_highcorr.shape
#We are now left with 145 columns, which is still significant
```

(458913, 145)

Part 3 >> Generate Machine Learning Model_ Random Forest

Remove Highly Correlated Features

- 상관관계 높은 feature 제거
- absolute correlation $\geq 90\%$ 이상인 열 drop

```
train_without_target=train.drop(['target'],axis=1)
cor_matrix = train_without_target.corr().abs()
upper_tri = cor_matrix.where(np.triu(np.ones(cor_matrix.shape), k=1).astype(np.bool))
#Drop out columns with absolute correlation of more than 90%
to_drop = [column for column in upper_tri.columns if any(upper_tri[column] > 0.90)]
train_drop_highcorr=train.drop(to_drop, axis=1)
train_drop_highcorr.shape
#We are now left with 145 columns, which is still significant
```

(458913, 145)

Columns : 150 → 145

Part 3 >> Generate Machine Learning Model_ Random Forest

Remove Columns with Low Variance = 0.1

```
from sklearn.feature_selection import VarianceThreshold
from itertools import compress
def fs_variance(df, threshold:float=0.1):
    features = list(df.columns)

    # Initialize and fit the method
    vt = VarianceThreshold(threshold = threshold)
    _ = vt.fit(df)

    # Get which column names which pass the threshold
    feat_select = list(compress(features, vt.get_support()))

    return feat_select
columns_to_keep=fs_variance(train_drop_highcorr)
# We are left with 54 columns (excluding target), which passed the threshold.
train_final=train[columns_to_keep]
len(columns_to_keep)
```

Part 3 >> Generate Machine Learning Model_ Random Forest

Remove Columns with Low Variance = 0.1

```
from sklearn.feature_selection import VarianceThreshold
from itertools import compress
def fs_variance(df, threshold:float=0.1):
    features = list(df.columns)

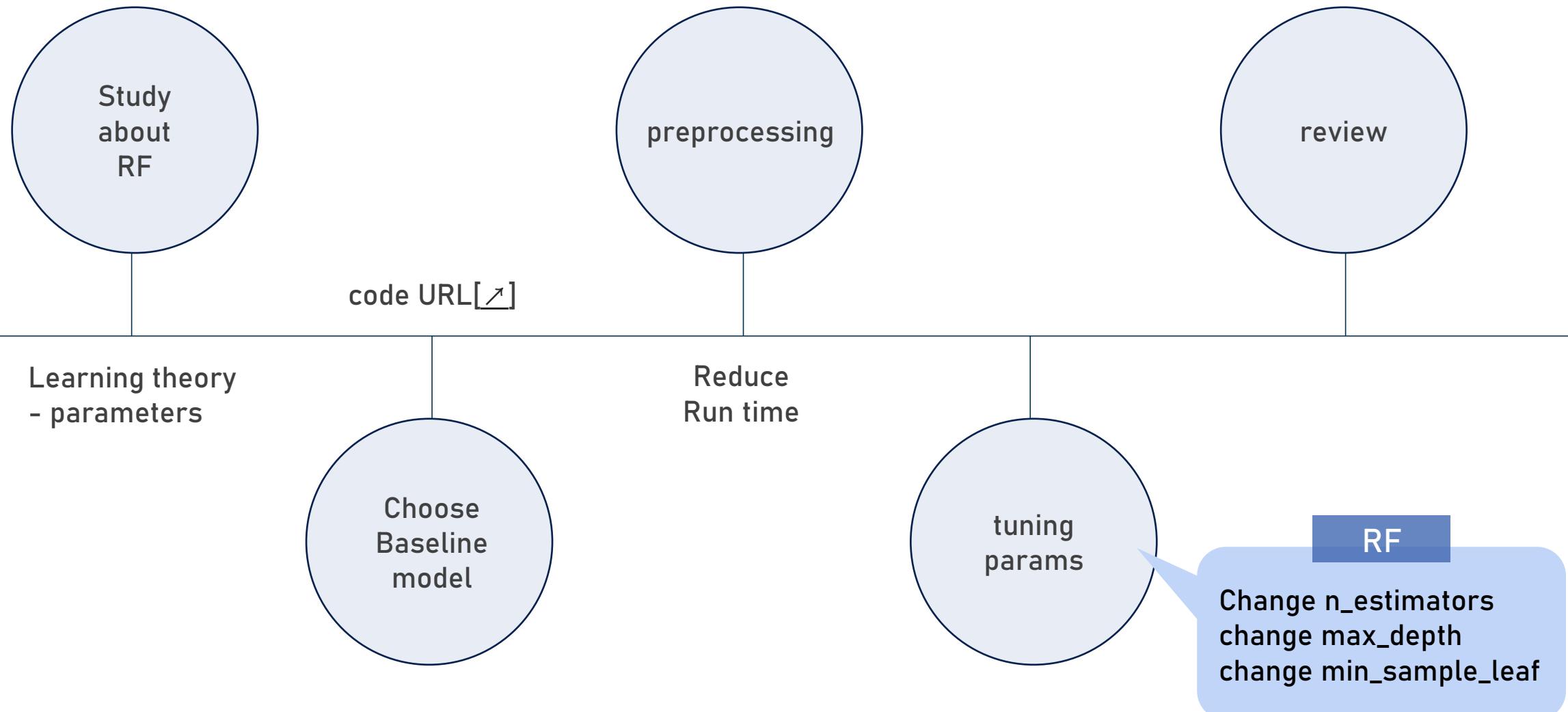
    # Initialize and fit the method
    vt = VarianceThreshold(threshold = threshold)
    _ = vt.fit(df)

    # Get which column names which pass the threshold
    feat_select = list(compress(features, vt.get_support()))

    return feat_select
columns_to_keep=fs_variance(train_drop_highcorr)
# We are left with 54 columns (excluding target), which passed the threshold.
train_final=train[columns_to_keep]
len(columns_to_keep)
```

54 columns left (except target)

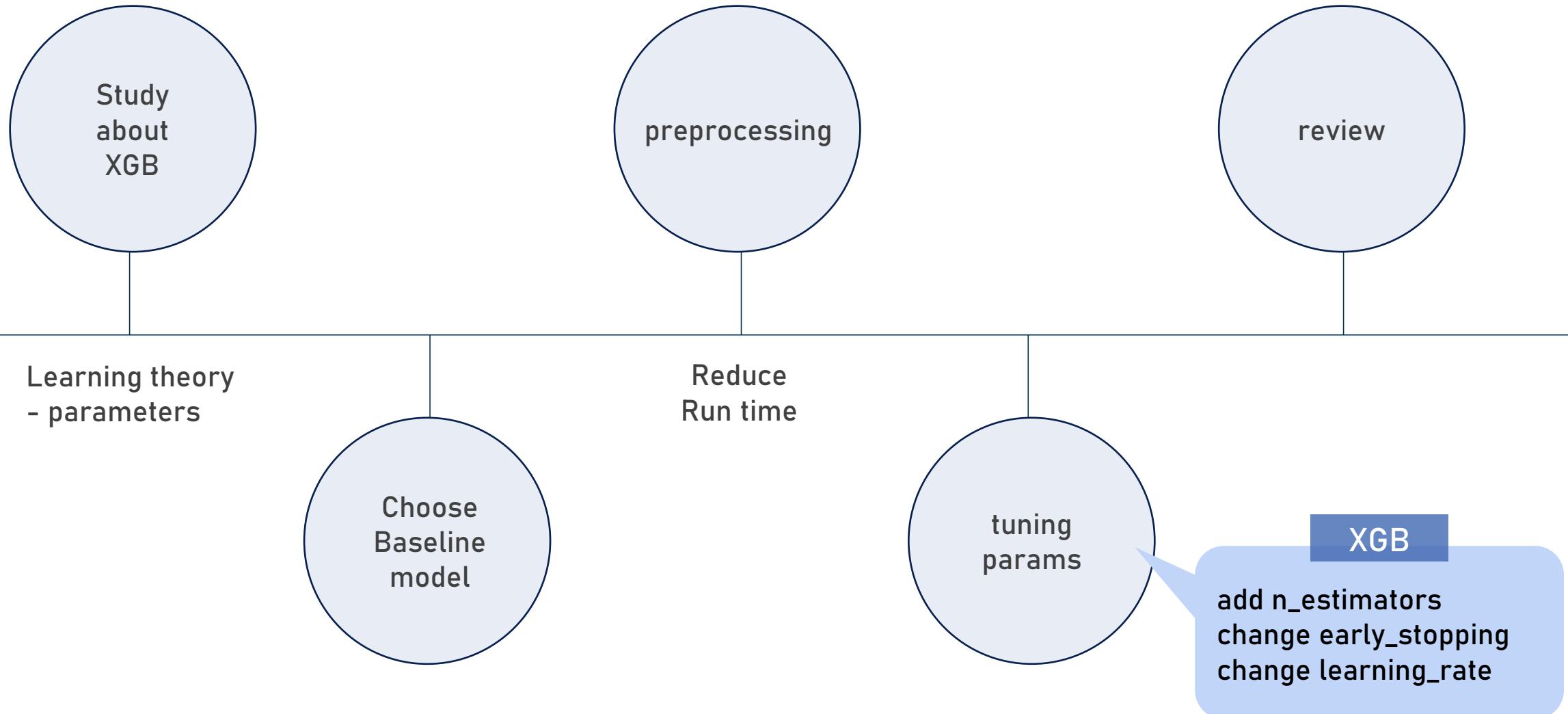
Part 3 >> Generate Machine Learning Model_ Random Forest



Part 3 >> Generate Machine Learning Model_ Random Forest

No.	내용요약	소요시간	parameters	scores
1	Base RandomForest model	1058.8s - GPU	n_estimators=400, max_features='sqrt', bootstrap=True(default), max_depth=30, min_samples_leaf=1(default), min_samples_split=5, n_jobs=-1	score: 0.729
2		477.1s - CPU		
3	change n_estimators (v1)	290.8s - CPU	n_estimators → 200	score: 0.727 / ↓
4		437.6s - CPU	n_estimators → 500	score: 0.729 / -
5	change max_depth (v2)	547.8s - CPU	max_depth → 100	score: 0.728 / ↓
6		538.3s - CPU	max_depth → 40	score: 0.729 / -
7		535.3s - CPU	max_depth → 50	score: 0.729 / -
8		244.6s - CPU	max_depth → 10	score: 0.720 / ↓
9		303.5s - CPU	max_depth → 20	score: 0.742 / ↑
10	add random_state (v3)	383.9s - CPU	random_state = 77	score: 0.730 / ↑
11	change min_samples_leaf (v4)	349.5s - CPU	min_samples_leaf → 2	score: 0.744 / ↑

Part 3 >> Generate Machine Learning Model_ XGBoost



Part 3 >> Generate Machine Learning Model_ XGBoost

No.	내용요약	소요시간	parameters	scores
1	basic XGB model	190.8s(GPU)	(n_estimators : default 100, early_stopping_rounds=100, learning_rate=0.05)	
2	add n_estimators	186.6s(GPU)	n_estimators=110	
3		176.8s(GPU)	n_estimators=120	
4		190.4s(GPU)	n_estimators=130	
5		191.8s(GPU)	n_estimators=140	
6		196.2s(GPU)	n_estimators=150	
7		183.4s - GPU	n_estimators=200	
8	change early_stopping option	192.0s - GPU	n_estimators=200, early_stopping_rounds=150	
9	change n_estimators	201.1s - GPU	n_estimators=300, early_stopping_rounds=150	
10	change learning_rate option	179.0s - GPU	n_estimators=200, learning_rate=0.1, early_stopping_rounds=100	
11		246.3s - GPU	n_estimators=200, learning_rate=0.2, early_stopping_rounds=100	변동없음

Part 3 >> Generate Machine Learning Model_XGBoost_Parameters

Find out the Best Parameters_ Define Parameter Range

In [13]:

```
#define parameter range
learning_rate=np.linspace(0.01,0.1,10)
max_depth=np.arange(2, 18, 2)
colsample_bytree=np.arange(0.3, 0.8, 0.1)
iterations=np.arange(50, 1000, 50)
l2_leaf_reg=np.arange(0,10)
bagging_temperature=np.arange(0,100,10)
n_estimators=np.arange(50,500,50)
```

Part 3 >> Generate Machine Learning Model_XGBoost_Parameters

Define Parameter Space, Fit Condition, and Ross Function

```
xgb_cat_params = {  
    'learning_rate': hp.choice('learning_rate', learning_rate),  
    'max_depth': hp.choice('max_depth', max_depth),  
    'colsample_bytree': hp.choice('colsample_bytree', colsample_bylevel),  
    'n_estimators': hp.choice('n_estimators', n_estimators),  
    'loss_function': 'logloss',  
    'nan_mode': 'Min',  
    'task_type': 'GPU'  
}  
xgb_fit_params = {  
    'eval_metric': 'logloss',  
    'early_stopping_rounds': 10,  
    'verbose': False  
}  
xgb_para = dict()  
xgb_para['cls_params'] = xgb_cat_params  
xgb_para['fit_params'] = xgb_fit_params  
xgb_para['loss_func'] = lambda y, pred: np.sqrt(mean_squared_error(y, pred))
```

Part 3 >> Generate Machine Learning Model_XGBoost_Parameters

Define Parameter Space, Fit Condition, and Ross Function

```
#calling the XGBoost function by passing XGB parameter space
xgb_opt = obj.process(fn_name='xgb_cls', space=xgb_para, trials=Trials(), algo=tpe.suggest,
max_evals=2)
```

In [24]:

```
#Save best parameters in a dictionary
best_param_xgb={}
best_param_xgb['learning_rate']=learning_rate[xgb_opt['learning_rate']]
best_param_xgb['colsample_bytree']=colsample_bylevel[xgb_opt['colsample_bytree']]
best_param_xgb['max_depth']=max_depth[xgb_opt['max_depth']]
best_param_xgb['n_estimators']=n_estimators[xgb_opt['n_estimators']]
```

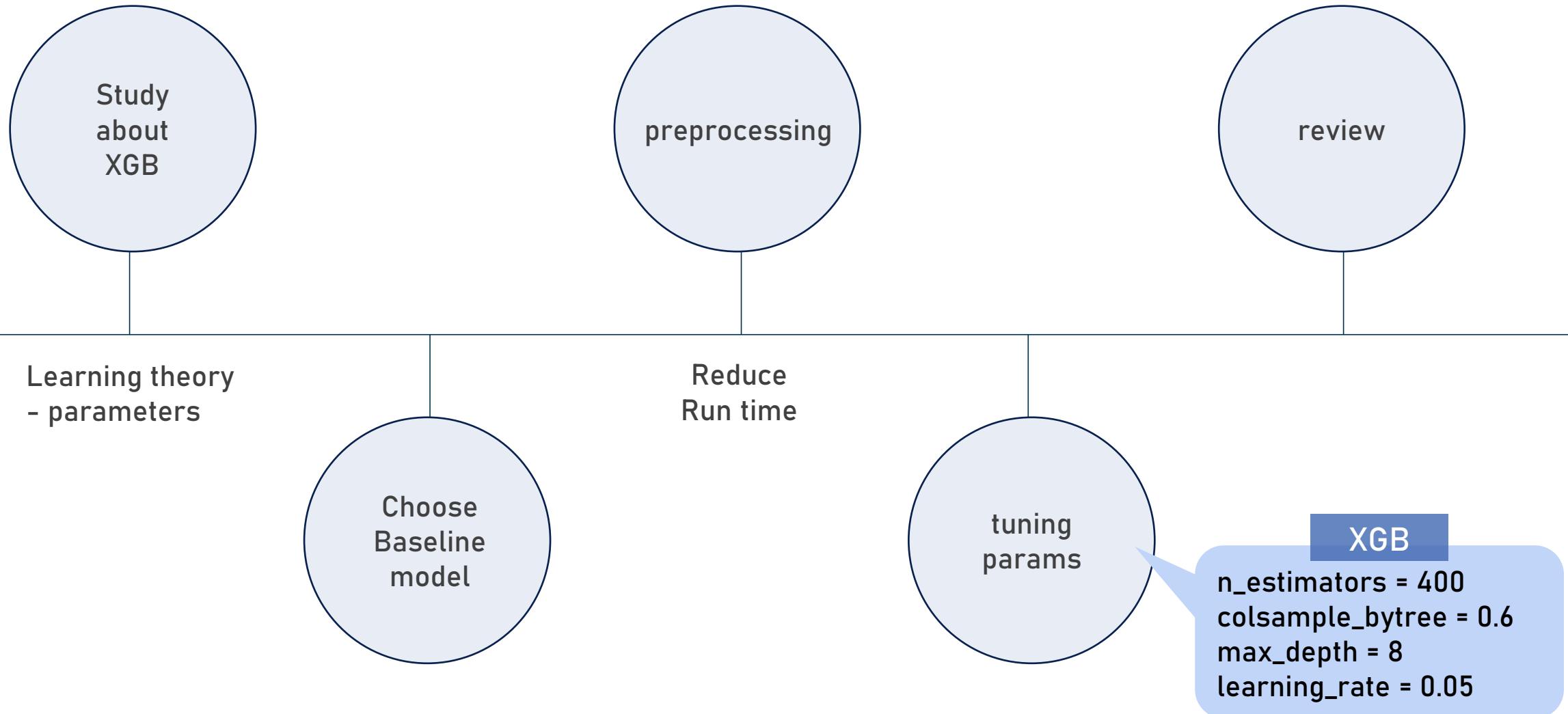
In [25]:

```
best_param_xgb
```

Out[25]:

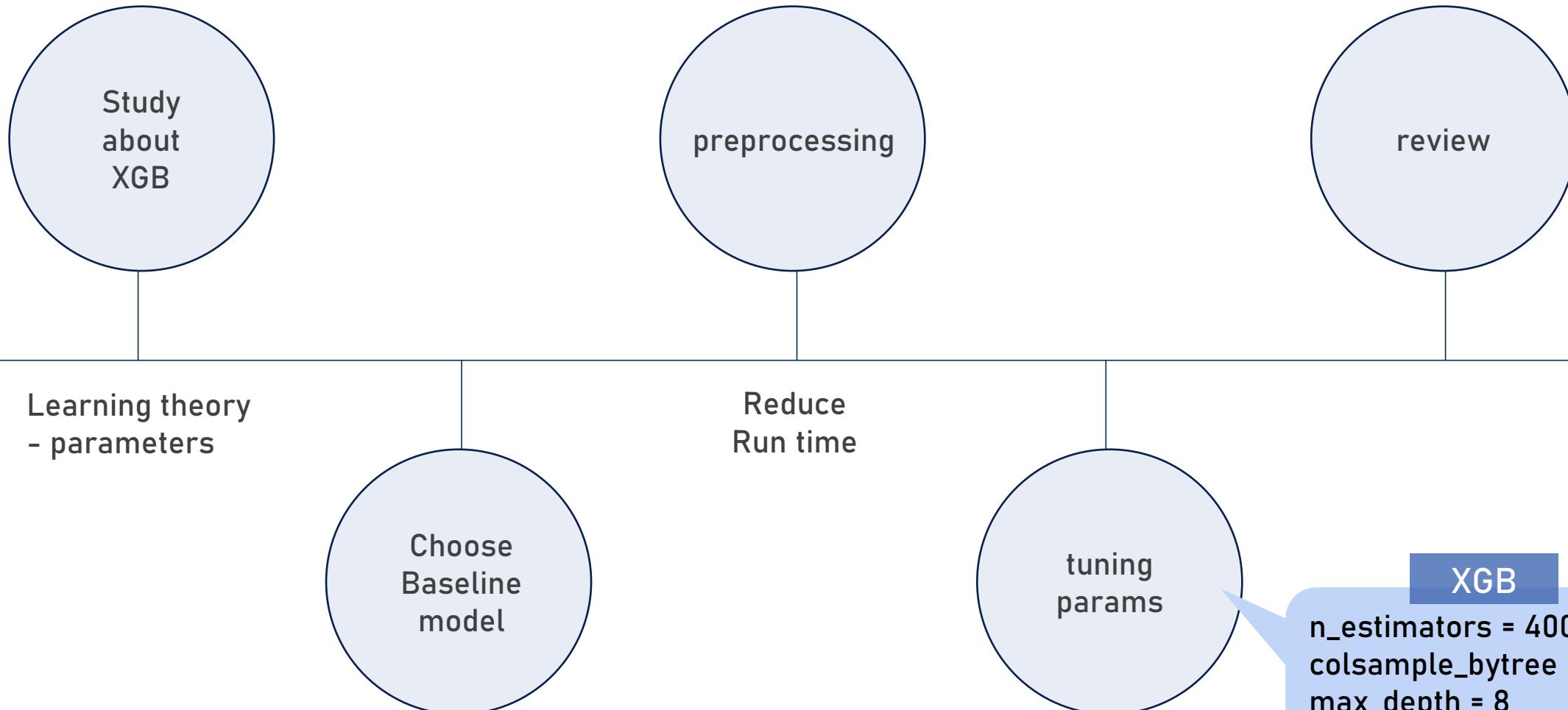
```
{'learning_rate': 0.0500000000000001,
'colsample_bytree': 0.6000000000000001,
'max_depth': 8,
'n_estimators': 400}
```

Part 3 >> Generate Machine Learning Model_ XGBoost

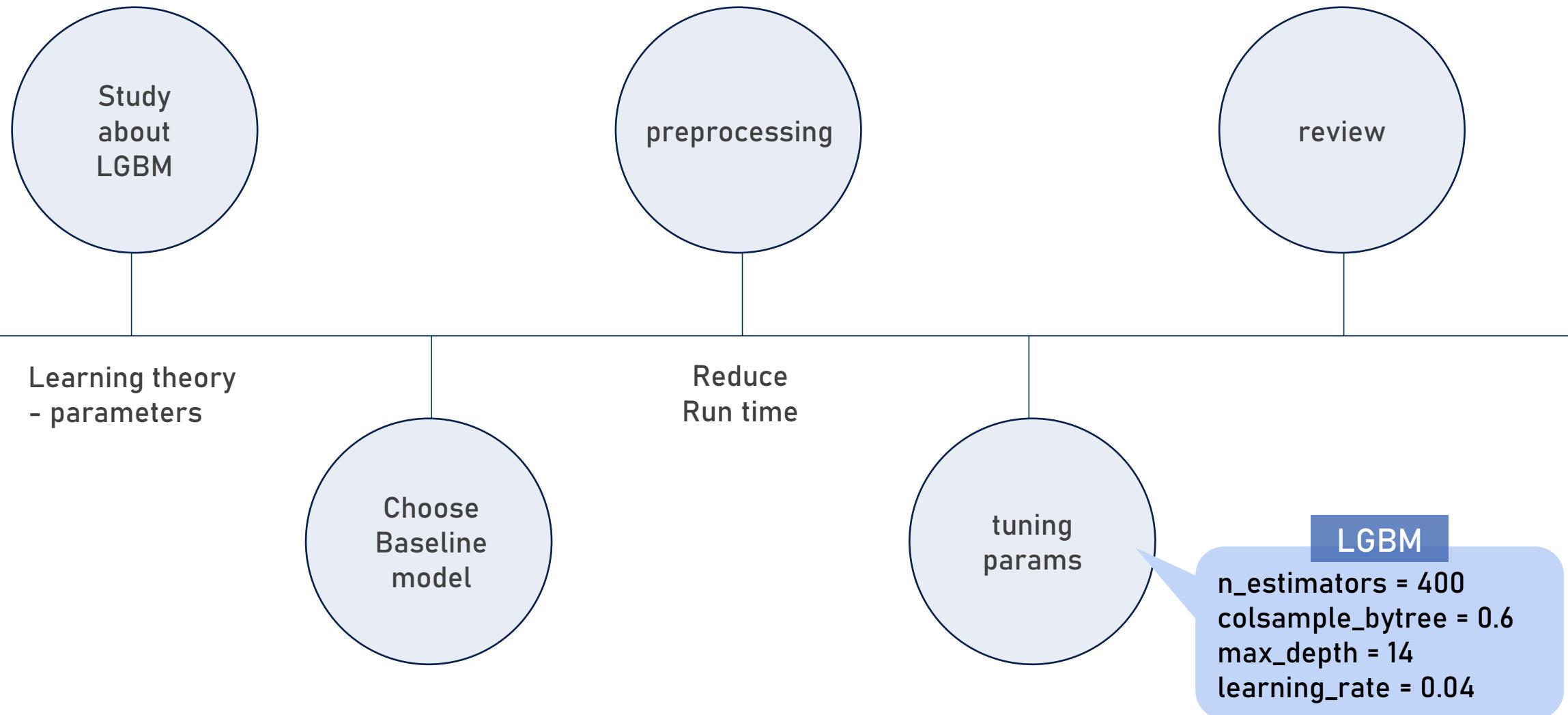


Part 3 >> Generate Machine Learning Model_ XGBoost

F1-score of XGBoost : 0.81

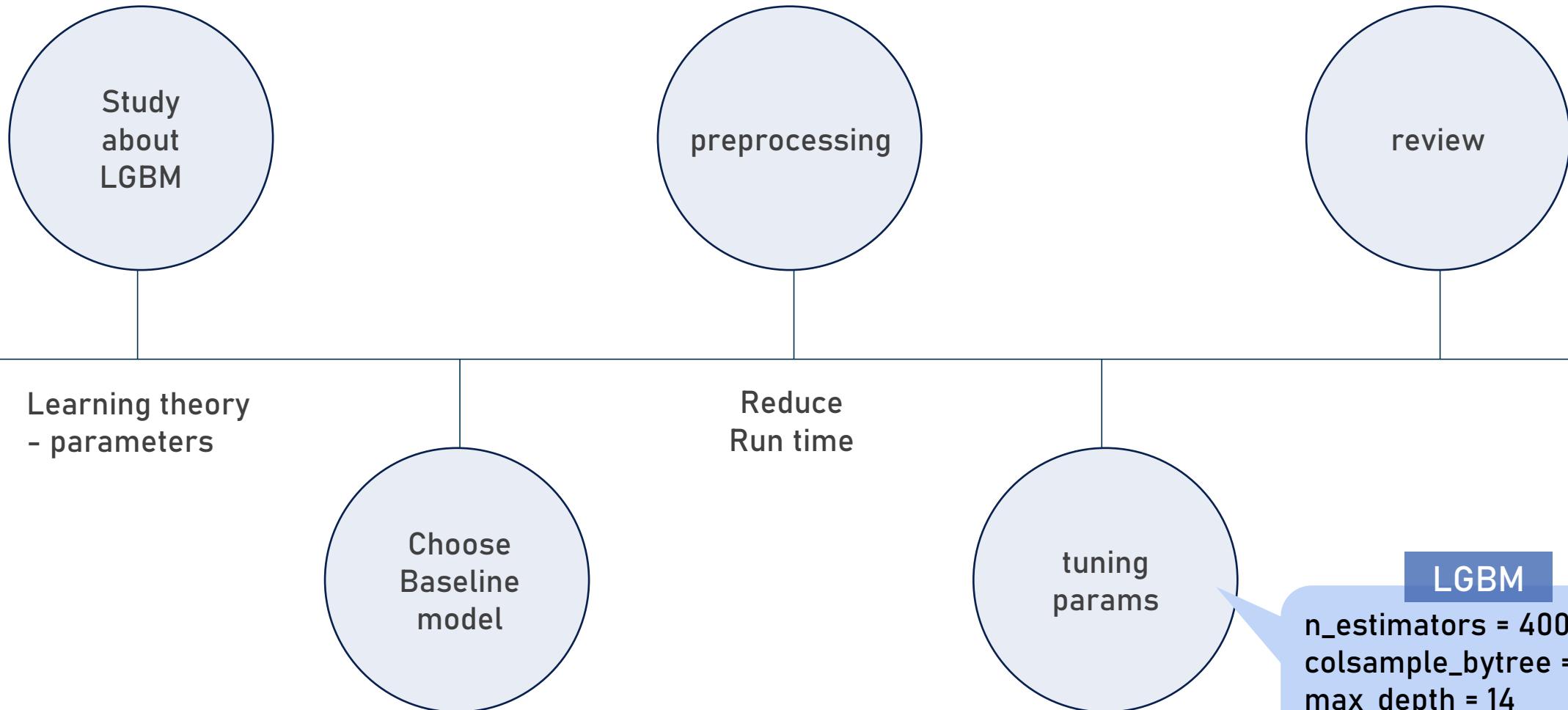


Part 3 >> Generate Machine Learning Model_ LightGBM



Part 3 >> Generate Machine Learning Model_ LightGBM

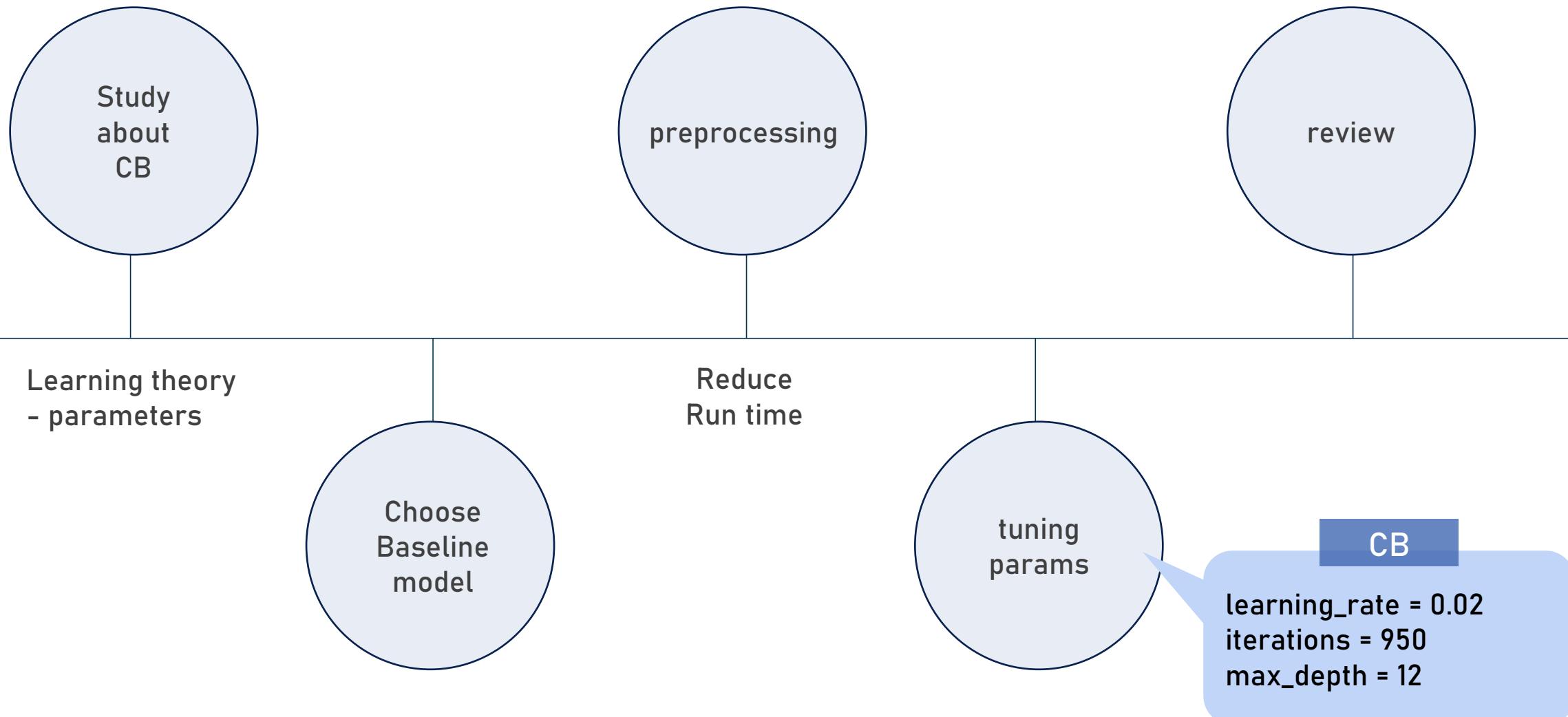
F1-score of LGBM : 0.81



LGBM

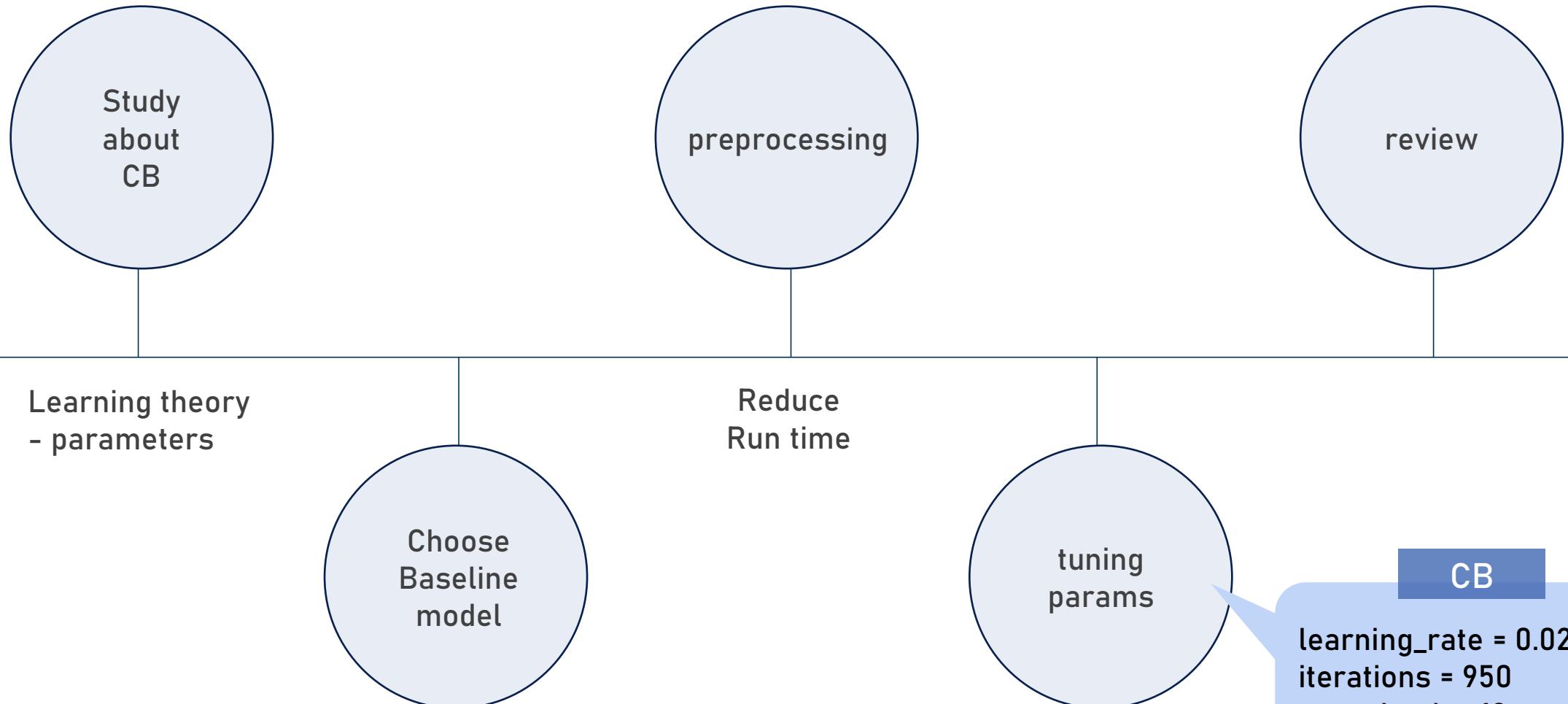
`n_estimators = 400
colsample_bytree = 0.6
max_depth = 14
learning_rate = 0.04`

Part 3 >> Generate Machine Learning Model_ CatBoost



Part 3 >> Generate Machine Learning Model_ CatBoost

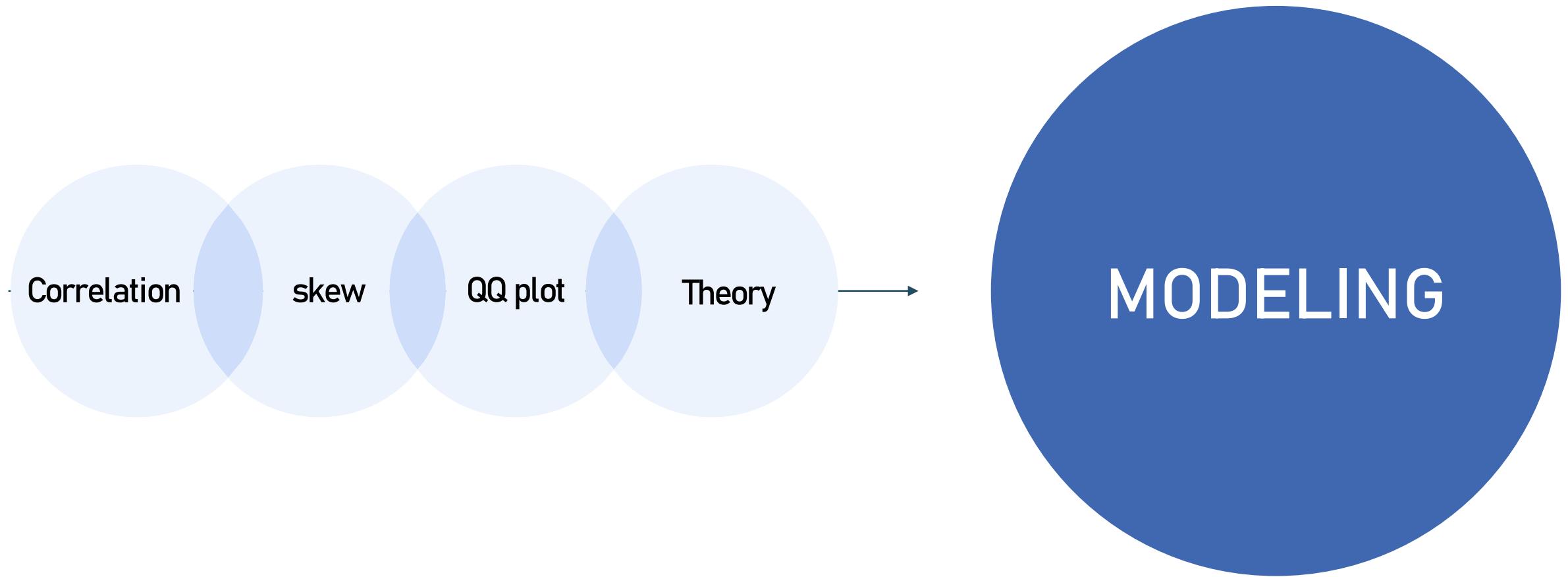
F1-score of Catboost: 0.80



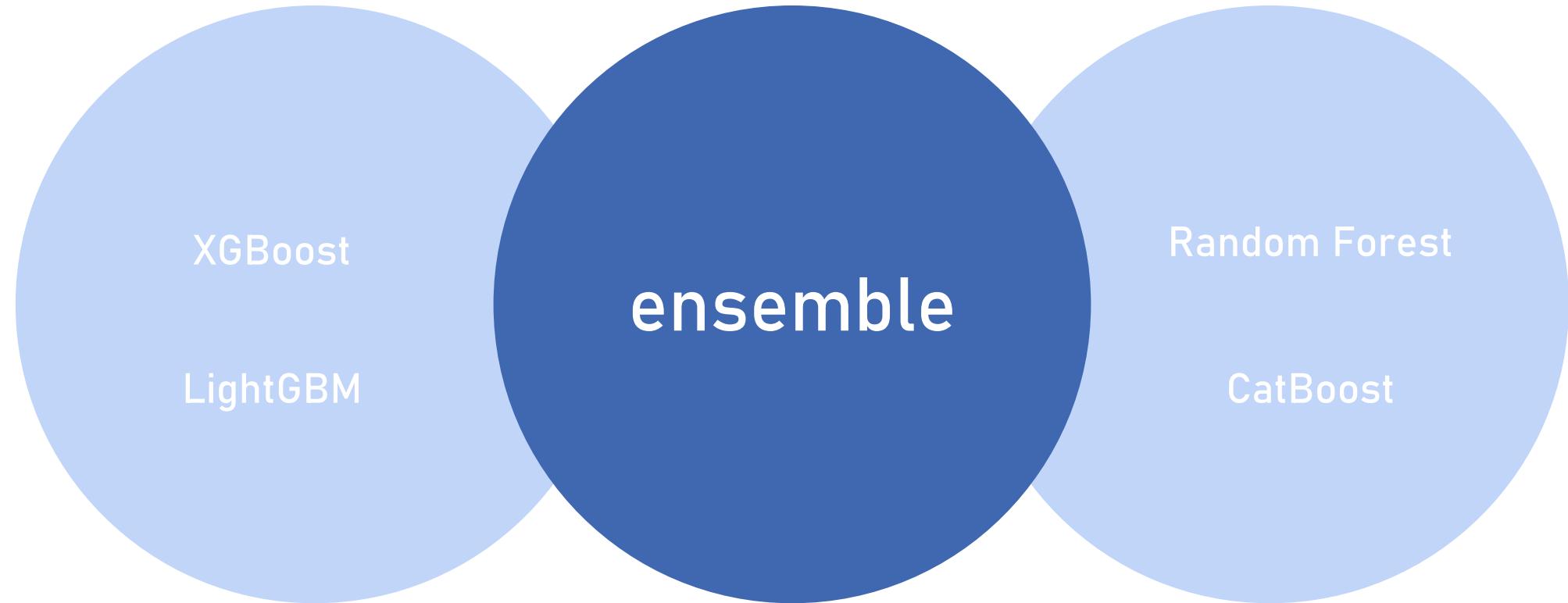
4

Conclusion

Part 4 >> Conclusion



Part 4 >> Conclusion



Q&A