

최재주, SK하이닉스

JMP Add-in 기능을 이용한 CTQ Define Engine 소개



Agenda

- *Introduction*
- *CTQ define engine Process*
- *CTQ define engine Main Function*
- *CTQ define engine Detail*
- *Summary*



Introduction

현재 반도체 공정은 생산성/품질 개선을 위해 지속적으로 *Shift Left* 활동 진행 중임.



허나, Eng'r의 경험 중심 업무 / 제한적 Data 활용한 Solution 도출 / Data 처리의 어려움
등으로 인해 현장 data 활용 한계 발생

■ Introduction

사내 다양한 분석 및 시각화 tool 운영 중이나
PNT 공정에 일반화된 CTQ Define Engine 요구됨.

많은 Data를 쉽게 Handling 할 수 있을까?

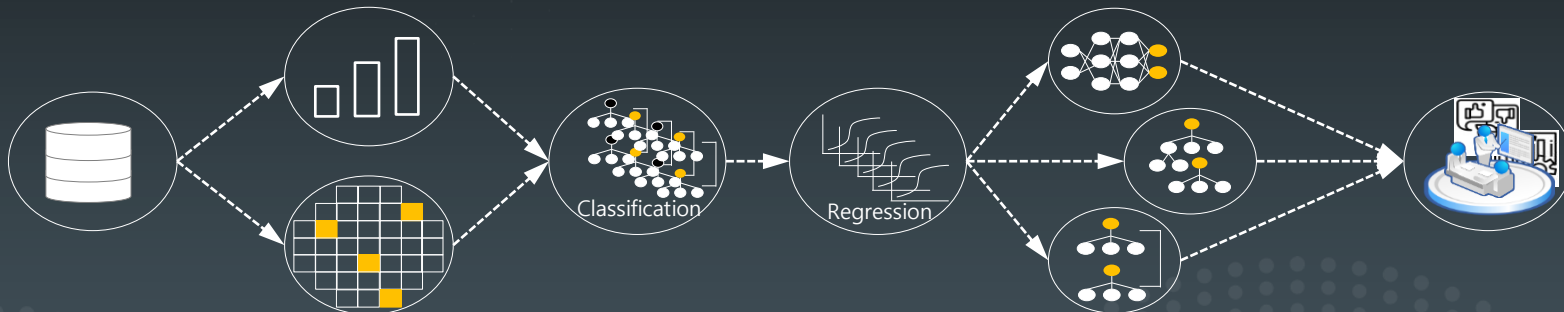
불량과 연관성이 높은 인자를 쉽게 찾을 수 있을까?



CTQ Define Engine Process

현장 Eng'r가 쉽게 활용할 수 있도록 Data 전처리 ~ CTQ 도출까지 JMP Add-in 구현

All in one by JMP Add-in



- ① Binary, Log 변환
- ② 파생 변수 생성
- ③ Outlier 처리
- ④ Missing Imputation
- ⑤ Category 변수 처리

- ① 통계량 파악
- ② Distribution
- ③ Graph Builder

- ① Imbalance data (over/under sampling)
- ② Feature Selection
- ③ Model coef. value

- ① Model Comparison
- ② Decision Boundary Review
- ③ Parameter interaction Review

- ① Output 결과 검토
- ② 현장 평가

CTQ Define Engine Function

Data Preprocessing

X, Y Variable Input

Data Preprocessing

Imbalance Data Sampling

Modeling & Ensemble

Data Visualization

I. Data processing (Delimiter)
 II. Data processing (Binary)
 III. Data processing (Feature Interaction)
 IV. Data processing (Log Transformation)
 V. Data processing (Missing Imputation)
 VI. Feature Selection

X인자 & Y인자 분석항목 선정

181 Columns

Enter column name

A B C D E F G H I f-1 f-2 f-3 f-4 f-5 f-6 f-7 f-8 f-9

X인자 선정

X Item
required
optional

Y인자 선정

Y Item
required
optional

By

By Item
optional

VII. Data Analysis & Modeling

데이터 전처리

NearZeroVar. 제거

표준편차 0

결측비율 5

CAT CNT

이상값처리

결측값처리

모델링

Method

개별:샘플(시작)

샘플:샘플(시작-종료)

반복:반복

샘플(시작) 4 (종료)/반복 10

Over_Sample_Good_기준(%) 50

분류 Threshold 0.6

분류모델(Under) 실행

변수Grouping(분류_(개별))

변수중요도(분류_(샘플/반복))

시각화 & 모델비교

X인자 분포

그래프 빌더

X인자교차분석

결정나무(Under)

실행

모델비교(Under)

실행

[Data Preprocessing – I]

- Data Delimiter : 변수 분리 기능
- Data Binary : 연속형 Value 대상 이진변수 생성
- Log Transformation : 선정된 변수 로그 변환
- Missing Imputation : 이전 data imputation (python ffill)

[X, Y Variable Input]

- X 인자 선정
- Y 인자 선정
- By : META 변수 지정 (group 별 별도 분석)

[Data Preprocessing – II]

- NearZeroVar. 제거 : Data 변동이 없거나 비어 있는 변수 제거
- CAT CNT : 종속변수의 Good/Bad 수량 산출
- 이상값 처리 / 결측값 처리 : JMP 내장 기능 연계

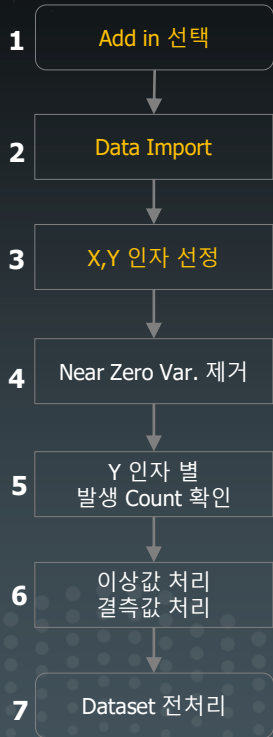
[Imbalance Data Sampling & Modeling]

- 분석 기법 : Bootstrap forest / Boosted tree / Logistic Regression 사용
- Sampling : Random under sampling / Random over sampling / SMOTE
- 변수 중요도 : 모델 별 importance Score 활용하여 변수를 Raking함.

[Data Visualization & Ensemble]

- X인자 분포 : Distribution / Graph Builder 활용
- X인자 교차분석 : X인자 별 관계 분석 시 활용
- 결정나무 분석 : 도출된 CTQ의 Decision Boundary을 반복 도출하여 추가 정보 제공
- 모델 비교 : 모델 별 성능 비교 및 예측 결과 Ensemble을 통해 CTQ Item 검증

CTQ Define Engine Detail

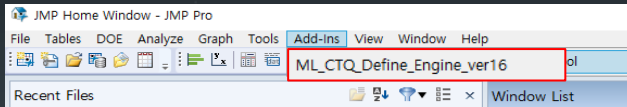


1. Add in 선택

- JMP 실행 후 Addin File 실행 및 install

ML_CTQ_Define_Engine_ver16_r00

- JMP 메뉴에 생성된 Add-Ins 선택



2. Data Import

- Open Data File 창이 뜨면 저장된 데이터 테이블 Open

Sample data (Encoding)

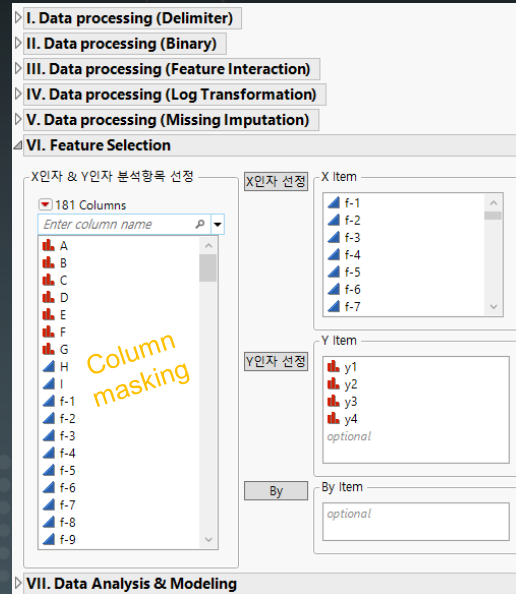
	A	B	C	D	E	F	G	H	I
49	1	8	19	1	2	5	12	30	
49	1	8	19	1	2	5	16	29	
49	1	8	19	1	2	5	17	21	
49	1	8	19	1	2	5	17	26	
49	1	8	19	1	2	5	18	29	
49	1	8	19	1	2	5	16	37	
49	1	8	19	1	3	5	19	19	
49	1	8	19	1	2	5	19	29	
49	1	8	19	1	3	5	20	17	
49	1	8	19	1	3	5	20	18	
49	1	8	19	1	3	5	20	23	
49	1	8	19	1	3	5	20	26	
49	1	8	19	1	3	5	20	28	
49	1	8	19	1	3	5	20	33	
49	1	8	19	1	3	5	20	36	
49	1	8	19	1	3	5	20	36	

3. X,Y 인자 선정

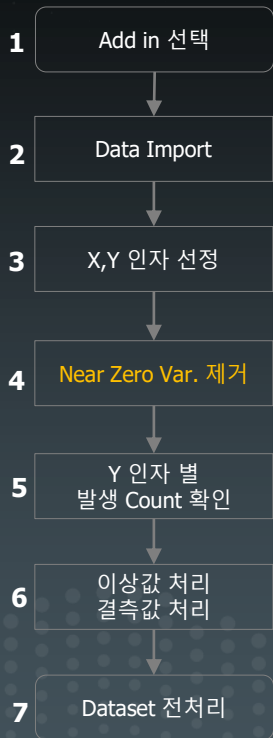
- 분석에 필요한 X인자와 Y 인자를 선정

→ Y 인자 : Nominal & Continuous 모두 분석 가능

(전처리가 필요할 경우 Data processing 기능 사용)



CTQ Define Engine Detail



4. Near Zero Var. 제거

- Data의 편차가 없는 변수와 결측이 많은 변수 제거

VI. Feature Selection

X인자 & Y인자 분석항목 선정

X인자 선정: X Item (f-1 to f-9)

Y인자 선정: Y Item (y1 to y4, optional)

VII. Data Analysis & Modeling

데이터 전처리

NearZeroVar.제거

표준편차: 0

결측비율: 5

Method: 개별 샘플(시작) 샘플*샘플(시작-종료) 반복/반복

샘플(시작): 4 (종료)/반복: 10

Over_Sample_Good_기준(%): 50

분류 Threshold: 0.6

분류모형(Under):

변수Grouping(분류_개별):

변수중요도(분류_샘플/반복):

시각화 & 모델비교

X인자 분포:

X인자교차분석:

모델비교(Under):

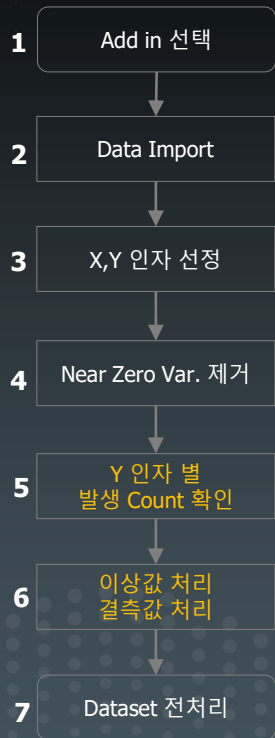
- 표준편차 0 : 각 열(변수)의 값이 모두 동일하다면 그 열(변수) 제거
(예) 한 변수의 값이 모두 1인 경우
→ 표준편차 기준값 변경이 필요한 경우 Input Data 변경 가능

```
std_value = neb3 << Get; //neb: number edit box
ncols1_std = N Items( cc_select_std );
For( j = 1, j <= ncols1_std, j++, //변수 별 표준편차가 기준값 이하일 경우 변수 제거
If( Col Std Dev( Column( cc_select_std[j] ) ) <= std_value,
Column( cc_select_std[j] ) << Exclude( 1 ),
Column( cc_select_std[j] ) << Exclude( 0 )
));
```

- 결측비율 0: 각 변수(열) 내 결측값이 하나라도 있으면 그 변수(열) 제거
→ 결측비율 기준값 변경이 필요한 경우 조건 변경 가능

```
missing_rate = neb4 << Get; //neb: number edit box
ncols_rows = N Rows( dt );
missing_n = Floor( ncols_rows * (missing_rate / 100) ); //결측치를 %로 환산
For( z = 1, z <= ncols, z++,
Column( var_x[z] ) << Set Selected( 1 )
);
cc_select = dt << Get Selected Columns();
ncols1 = N Items( cc_select );
For( j = 1, j <= ncols1, j++, //결측치가 기준값보다 높을 경우 변수 제거
If( Col N Missing( Column( cc_select[j] ) ) > missing_n,
Column( cc_select[j] ) << Exclude( 1 ),
Column( cc_select[j] ) << Exclude( 0 )
));
```


CTQ Define Engine Detail



5. Y 인자 별 발생 Count 확인

- 여러 Y's 발생 수준을 파악할 경우 Modeling 전 확인

Tabulate		
	Data	
Label	0	1
y1	185843	2817
y2	184894	3748
y3	171767	16892
y4	185664	2981

Y 변수 별 데이터의 불균형 수준을 파악

```

// 변수 별 Good과 Bad의 개수 Count
Proc Data = Femsim(1);
Current Data Table( dt_row );
dt = Current Data Table();
var_3 = 1 <= get itest();
if (N Stack( var_3 ) > 4,
// 변수가 3개일 경우 Tabulate
Tabulate( Show Control Panel( 0 ), Add Table( Column Table( Grouping Columns( Column( var_3 ) ) ) ),
//데이터 별 stacking
stack = Data Table( dt ) << Stack(
column( var_3 );
Source Label Column( "Label" ),
Stacked Data Column( "Data" ),
Drop all other Columns( 1 ),
Invisible
);
//변수가 3개일 경우 Tabulate
tabulate = stack << Tabulate(
Show Control Panel( 0 ),
Add Table( Column Table( Grouping Columns( Data ), Row Table( Grouping Columns( Label ) ) )
);
    
```

6-1. 결측값 처리

- 행 별 결측값 확인 시 jmp 내장 기능과 연계하여 처리함.

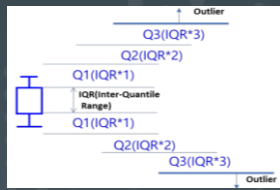
6-2. 이상값 처리

- 행 별 이상값 확인 시 jmp 내장 기능과 연계하여 처리함.

→ Tail Quantile과 Q를 조절하여 이상치 제거

→ Tail Quantile : 분포의 전체 확률이 1일 때, 꼬리 부분의 확률

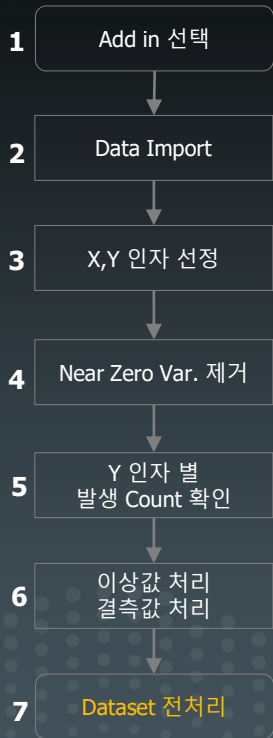
→ Q : 사분위 범위(Tail Quantile과 1-Tail Quantile 사이의) Q배를 초과 시 outlier로 판단함.



Column	Lower Prob.	Upper Prob.	Lower	Upper	Low	High	Number of
6-1	0.1	0.9	-0.5386	0.9206	-5.1807	5.3138	914
6-2	0.1	0.9	-0.4148	0.8423	-0.2324	3.6779	821
6-3	0.1	0.9	-0.4016	0.9507	-5.2021	5.8304	759
6-4	0.1	0.9	-0.4206	0.9168	-5.2319	5.814	784
6-7	0.1	0.9	-0.4885	0.9946	-4.4233	6.5742	514
6-8	0.1	0.9	-0.2473	0.9264	-4.4444	4.5627	521

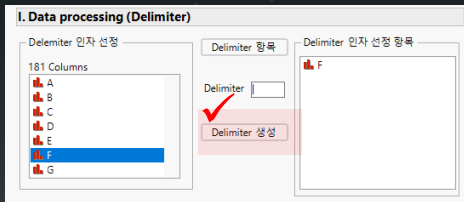


CTQ Define Engine Detail



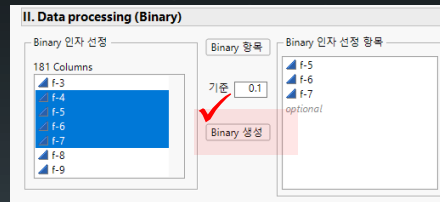
7.1 Delimiter (별도 구현)

- 특정 변수의 경우 구분자 '|' 로 구분되어 있음.
→ '|' 구분자를 기준으로 내부 값을 구분함.



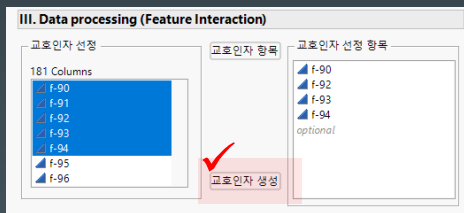
7-2. Binary (별도 구현)

- 연속형 Value 대상으로 이진변수 생성이 필요할 경우 사용
→ 분류 모델 적용할 경우 사용자 설정값을 기준으로 이진분류 가능



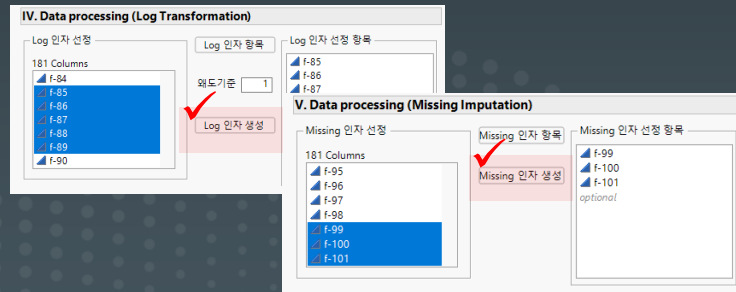
7-3. Feature Interaction (별도 구현)

- 다항차수 & 교호작용 신규변수 생성 가능



7-4. Log 변환 / 결측치 보간 (별도 구현)

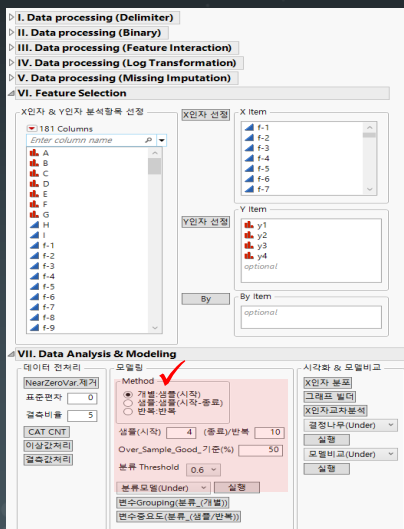
- 선정된 변수 로그 변환 및 Python의 pandas 내 ffill 기능을 구현



CTQ Define Engine Detail



8. X 인자 분포 & 그래프 확인
- 각 X,Y인자 분포 형태 및 기술 통계량 확인 가능
9. Y 별 분류 모델 샘플 Loop
- Sample 증가와 반복 기반 모델 구축 및 성능 검토



- Sampling Method : Bad 개수 대비 Good 데이터 Sampling(표본) 배수 선택
 - 샘플(시작) : Bad 개수 * 샘플(배수) 만큼
 - 샘플(시작-종료) : Bad 개수 * 샘플(시작) ~ Bad 개수 * 샘플(종료) 만큼
- Bad 개수 * 4, Bad 개수 * 5, ... , Bad 개수 * 9, Bad 개수 * 10 Good sampling



```

neb_value = neb << Get;
neb_value2 = neb2 << Get;
For( q = 1, q <= N Items( var_y ), q++,
For( r = neb_value, r <= neb_value2, r++,
sample_ratio = r;
var_y_new = var_y[q];
yy = Column( var_y_new ) << Get Values;
yy1 = Sum( yy );
If( yy1 == 0, Continue() );
yy_sum = yy1 * sample_ratio; //Sample 비율 별 Bad값 계산
Sample_subset = dt << Subset( Sample Size( yy_sum ),
Selected columns only( 0 ), Stratify( var_y_new ), invisible );
//Sample 비율 별 Dataset 생성
    
```

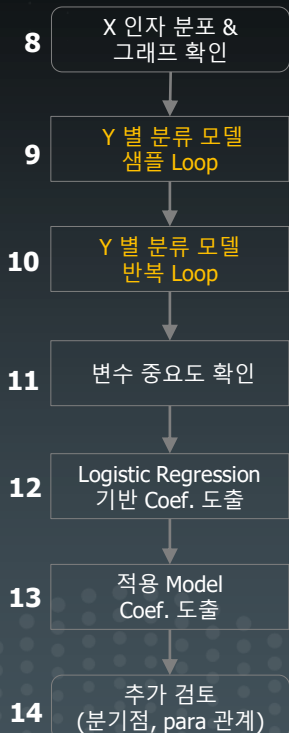
- 반복 : Bad 개수 * 샘플(배수)만큼 반복해서 모델 실행
- Bad 개수 * 4 Good sampling을 10번 반복



```

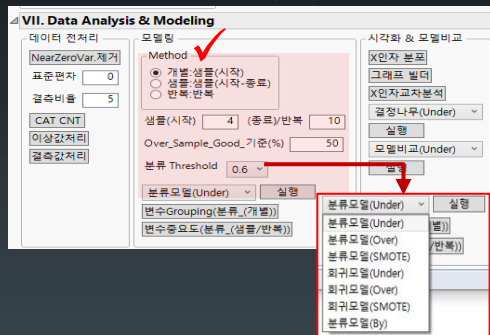
neb_value = neb << Get;
neb_value2 = neb2 << Get;
For( q = 1, q <= N Items( var_y ), q++,
For( r = 1, r <= neb_value2, r++,
sample_ratio = neb_value; // Sample 비율을 고정
var_y_new = var_y[q];
yy = Column( var_y_new ) << Get Values;
yy1 = Sum( yy );
If( yy1 == 0, Continue() );
yy_sum = yy1 * sample_ratio;
Sample_subset = dt << Subset( Sample Size( yy_sum ),
Selected columns only( 0 ), Stratify( var_y_new ), invisible );
    
```

CTQ Define Engine Detail



9. Y 별 분류 모델 샘플 Loop

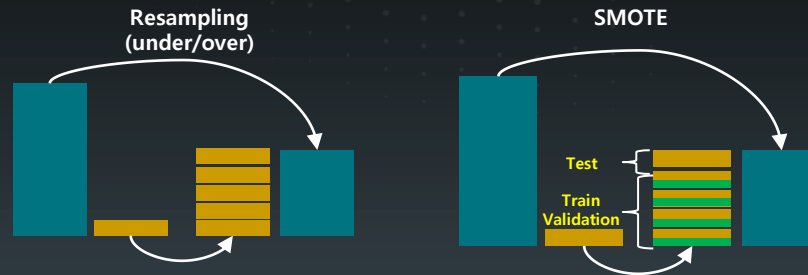
- CTQ 도출을 위한 사용 알고리즘
Random Forest & Logistic Regression



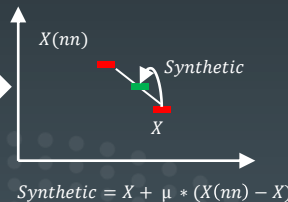
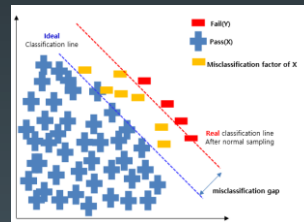
```
RF_sample = Sample_subset << Bootstrap Forest(
  Y( Eval( var_y_new ) ), X( Eval( var_x_new ) ),
  Validation Portion( 0.3 ),
  Specify Profit Matrix( [0 -1.5, -1 0, .], "0", "1", "Undecided" ),
  Method( "Bootstrap Forest" ), Column Contributions( 1 ), ROC Curve( 1 ),
  Portion Bootstrap( 1 ), Number Terms( Sqrt( N Items( var_x_new ) ) ),
  Number Trees( 100 ), Go);
```

```
Sample_subset << Fit Model(
  Y( Eval( var_y_new ) ), Effects( Eval( var_x ) ),
  Personality( "Nominal Logistic" ),
  Run( Positive Level( "0" ), ROC Curve( 1 ), Confusion Matrix( 1 ),
  Specify Profit Matrix( [0 -1.5, -1 0, .], "0", "1", "Undecided" ));
```

- 반도체 Data의 경우 클래스 불균형 데이터가 일반적임. (과소/과대적합 발생)
- 이를 위해 Under / Over sampling / SMOTE 방법을 통한 모델 성능 저하 방지 기능 구현



SMOTE concept



code

```
// KNN 실행
KNN_result = dt_bad << K Nearest Neighbors( Y( Eval( var_y_new ) ), X( Eval( var_x_new ) ), K( 3 ), invisible );
KNN_result << Save Near Neighbor Rows;
KNN_col = dt_bad << Get Column Names;

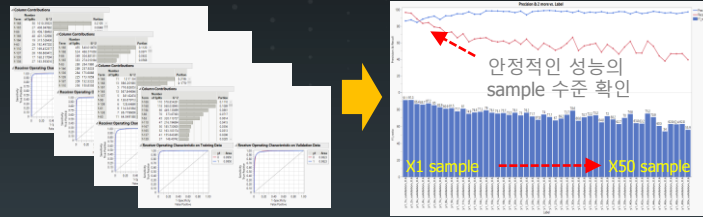
bad_mat_KNN1 = bad_mat + ((bad_mat - bad_table_KNN1_mat) * 0.7);
bad_mat_KNN2 = bad_mat + ((bad_mat - bad_table_KNN2_mat) * 0.7);
bad_mat_KNN3 = bad_mat + ((bad_mat - bad_table_KNN3_mat) * 0.7);
```

CTQ Define Engine Detail

- 8 X 인자 분포 & 그래프 확인
- 9 Y 별 분류 모델 샘플 Loop
- 10 Y 별 분류 모델 반복 Loop
- 11 변수 중요도 확인
- 12 Logistic Regression 기반 Coef. 도출
- 13 적용 Model Coef. 도출
- 14 추가 검토 (분기점, para 관계)

10. Y 별 분류 모델 반복 Loop

- Sample 증가와 반복을 통한 모델 도출 및 세부 성능 지표 검토
- Threshold 변경에 따른 목표 성능 조절 가능



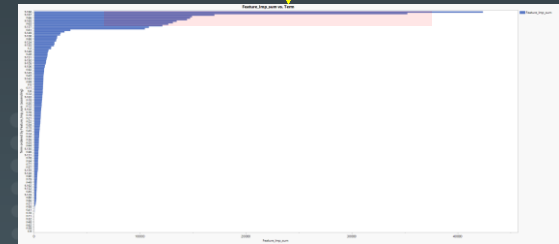
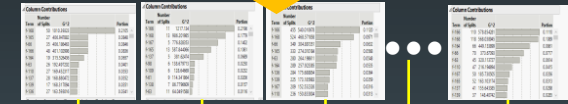
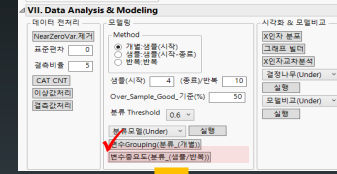
Resampling 기반 modeling 반복 차수 별 model 성능 도출

Label	g0_g0	g1_g1	g2_g2	g3_g3	g0_sum	g1_sum	g2_sum	g3_sum	total_sum	Actual_Bat_Rate	Predict_Bat_Rate	Accuracy	Precision	Recall	F1_Score	AUC
g0_bat_train_0.5	7254	33	448	1727	7887	1973	8100	1760	8660	20.0011	3.3704	0.7194	98.123	0.1317	0.2581	0.9563
g0_bat_validation_0.5	3129	32	190	454	3381	644	3519	708	4235	19.9783	5.59926	94.2752	92.8246	77.8882	84.3871	0.96176
g1_bat_train_0.6	7740	142	105	1868	7887	1973	7920	2010	8860	20.0701	1.1779	97.4869	93.9033	84.6762	83.7968	0.95643
g1_bat_validation_0.6	3227	124	114	790	3381	644	3371	824	4225	19.9783	3.81776	94.9669	91.4007	84.4629	82.9002	0.96116
g2_bat_train_0.8	9548	12	154	1617	9880	1971	12052	1629	11831	16.4596	3.46991	98.0064	99.2634	82.0596	89.8313	0.99629
g2_bat_validation_0.8	4294	31	156	588	4235	648	4462	606	5071	16.46811	5.78174	98.44881	98.5137	89.5935	83.8447	0.99096
g3_bat_train_0.6	9791	69	144	1827	9880	1971	9935	1896	11831	16.4596	1.44942	98.1996	98.3608	82.6941	84.4919	0.99638
g3_bat_validation_0.6	4147	79	162	684	4235	648	4398	762	5071	16.46811	3.79971	98.26822	99.7838	80.0117	85.0768	0.99066
g4_bat_train_0.5	11747	64	400	1563	11831	1972	12174	1627	13860	14.2867	3.15907	94.9864	79.2596	88.8375	0.99228	
g4_bat_validation_0.5	4703	58	222	623	5071	645	5258	761	5916	14.2883	4.22495	95.6581	84.5111	73.7278	82.3737	0.9845
g5_bat_train_0.9	11891	140	255	1717	11831	1972	11946	1957	13805	14.2867	3.15907	92.4081	92.401	87.009	89.648	0.99228
g5_bat_validation_0.9	4990	81	158	627	5071	645	5148	760	5916	14.2883	3.04911	95.9801	89.4311	81.3018	83.1629	0.98905
g6_bat_train_0.5	11911	24	324	1446	11885	1972	14205	1470	15279	12.4881	3.66926	96.5281	98.2921	71.4071	84.0968	0.99687
g6_bat_validation_0.5	5903	11	276	571	5914	647	6179	582	6761	12.5277	4.68674	95.7571	98.11	67.4744	79.978	0.98229
g7_bat_train_0.8	11744	61	208	1622	11885	1972	14242	1733	15279	12.4881	3.1222	97.7342	94.6473	81.8711	83.0467	0.99487
g7_bat_validation_0.8	5873	41	198	649	5914	647	6071	690	6761	12.5277	3.14141	96.485	94.258	76.6334	84.4502	0.98229
g8_bat_train_0.5	18756	20	576	1395	1579	1972	16332	1415	17247	11.1061	3.2666	96.6437	98.5666	70.7763	82.3861	0.99641
g8_bat_validation_0.5	10740	14	296	550	6700	649	7042	564	7496	11.1238	4.20335	95.8343	97.5177	85.0118	87.0142	0.97845
g9_bat_train_0.6	16879	103	139	1612	1579	1971	16632	1715	17247	11.1061	2.23827	97.3967	93.9462	81.7859	87.4661	0.99481
g9_bat_validation_0.6	8702	58	229	617	6700	646	6931	676	7496	11.1238	3.2666	96.2697	97.4074	82.2614	81.5368	0.97845
g0_bat_train_0.3	17719	29	592	1980	17747	1972	18319	1409	19719	10.0005	3.23231	96.8598	97.8418	69.9797	81.8327	0.99272
g0_bat_validation_0.3	7384	32	252	583	7696	645	7836	615	8451	9.99982	3.17981	96.9578	94.4238	70.1773	81.3209	0.98214
g1_bat_train_0.6	17651	96	411	1561	17747	1972	18062	1657	19719	10.0005	3.2751	97.4289	94.2964	79.5862	85.0602	0.99272
g1_bat_validation_0.6	7391	68	190	638	1806	645	1790	651	9066	9.99982	2.69581	97.9961	97.979	75.3683	82.7564	0.98214
g2_bat_train_0.5	18677	42	601	1371	19719	1972	20278	1413	21691	9.09133	2.9638	97.0196	97.0276	69.5333	81.0044	0.99678
g2_bat_validation_0.5	8420	23	320	516	8431	645	8791	539	9296	9.08993	3.76989	95.2144	95.7230	62.0051	74.5663	0.97517
g3_bat_train_0.8	19804	115	471	1501	19719	1972	20075	2169	90933	2.8461	97.2964	92.8817	71.1546	81.861	0.99019	
g3_bat_validation_0.8	8384	67	385	500	8431	645	8849	647	9296	9.09991	3.04994	95.8945	88.6991	77.348	80.97517	

성능 지표 검토

11. 변수 중요도 확인

- 반복 모델링하여 얻은 Feature Importance 값의 sum을 통해 주요 변수 선정함.



전체 model 변수 중요도



CTQ Define Engine Detail



12. Logistic Regression 기반 Coef. 도출
 - Y 인자와 선정된 CTQ 인자에 대해 반복적으로 회귀 모델링하여 Coef. 변화 수준 확인함.

Model 별 coef. 도출

Iterations					
Whole Model Test					
Model	-LogLikelihood	DF	ChiSquare	Pr>ChiSq	
Difference	2515.1051	4	5030.21	<.0001*	
Full	2421.0541				
Reduced	4936				

Iterations					
Whole Model Test					
Model	-LogLikelihood	DF	ChiSquare	Pr>ChiSq	
Difference	2661.4910	4	5322.982	<.0001*	
Full	2997.8005				
Reduced	5659.2995				

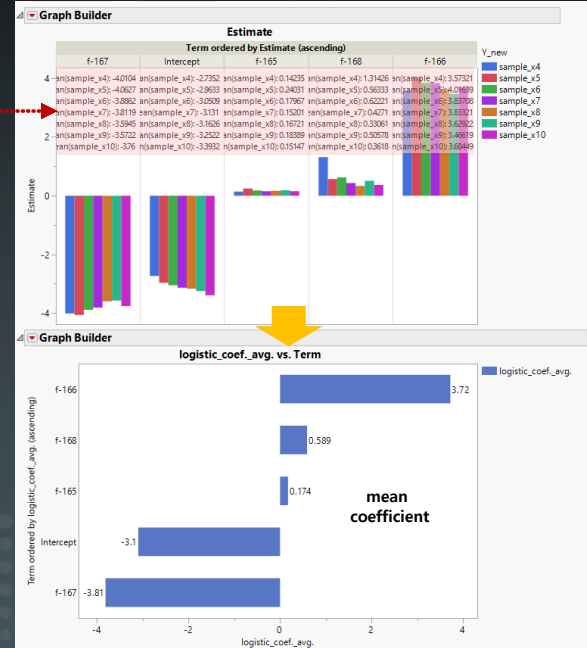
Iterations					
Whole Model Test					
Model	-LogLikelihood	DF	ChiSquare	Pr>ChiSq	
Difference	2645.0853	4	5290.171	<.0001*	
Full	3296.7609				
Reduced	5941.8464				

Iterations					
Whole Model Test					
Model	-LogLikelihood	DF	ChiSquare	Pr>ChiSq	
Difference	2619.3016	4	5238.003	<.0001*	
Full	2711.5563				
Reduced	5330.8579				

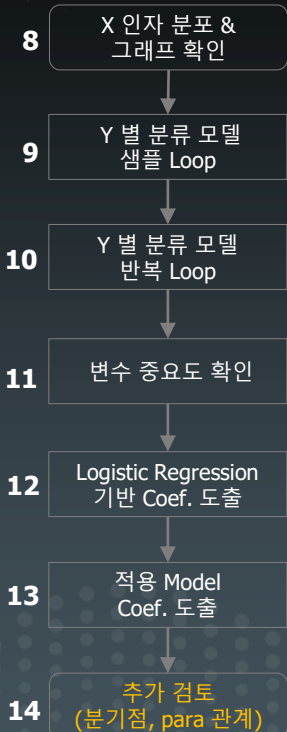
Iterations					
Whole Model Test					
Model	-LogLikelihood	DF	ChiSquare	Pr>ChiSq	
Difference	3130.9131	0.04	2353.4609	4705.922	
Full	0.15200848				
Saturated	3.83320779	0.09			
Fitted	-2.8119092	0.09			
Observations (or Sum Wgts)	4	3	2711.5563	<.0001*	

Iterations					
Whole Model Test					
Model	-LogLikelihood	DF	ChiSquare	Pr>ChiSq	
Difference	-2.9632773	0.0481283	3790.9	<.0001*	
Fitted	0.2403075	0.0333333	1257.6	<.0001*	
Saturated	4.01639194	0.13257			
Fitted	-4.0626509	0.0946112	1843.9	<.0001*	
Observations (or Sum Wgts)	4	3	2711.5563	<.0001*	

13. 적용 Model Coef. 도출
 - 반복 회귀 모델링하여 얻은 Coef. 값들의 평균값 산출 (해당 값을 기준으로 현장 활용)

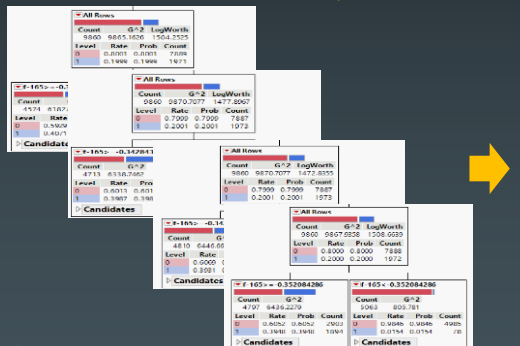
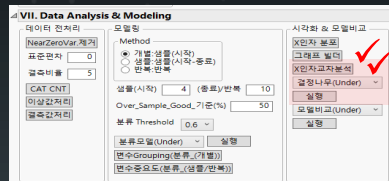


CTQ Define Engine Detail



14. 추가 검토 (분기점, para 관계)

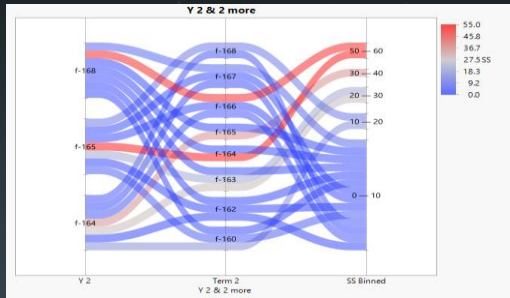
- 선정한 Y와 CTQ 인자를 사용하여 의사결정나무 모델 기반으로 반복 모델링하여 Main Item에 대한 분기 기점 검토함.



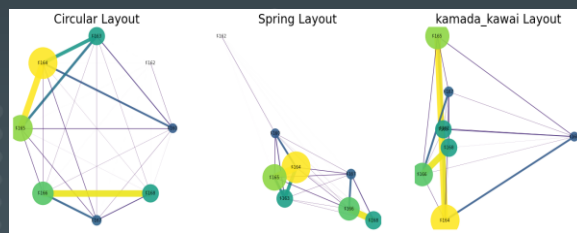
Decision Tree Result
 f-165 > = -0.33545118
 f-165 > = -0.339147426
 f-165 > = -0.35780532
 f-165 > = -0.342843672
 f-165 > = -0.35780532
 f-165 > = -0.352084286
 f-165 > = -0.339147426
 f-165 > = -0.352084286
 f-165 > = -0.35780532
 f-165 > = -0.353932409

Decision Tree 활용 주요 변수 분기점 파악 (자동 반복 기반 output 도출)

- Feature importance 기반 Parameter Grouping을 통한 관계 분석 가능
 → Parallel Plot in graph builder & network chart with python



Parallel Plot in graph builder



network chart with python

Summary

- > Eng'r 현장 data 활용 한계 극복 필요 (경험 중심 업무 / 제한적 Data 활용 / Data 처리 불가 등)
- > 사용자 편의를 갖춘 JMP Add-in 기반 tool 개발 (고용량 data handling & ML 기반 CTQ Define Engine)
- > 신속한 현장 적용 (data 전처리, CTQ 도출, model coef. 도출 등 별도 coding 없이 click으로 강건한 결과 확보)
- > CTQ 변수의 분기점 파악이 가능하며 graph builder 및 parameter 간 관계 분석 등 다양한 visualization 가능

DISCOVERY
SUMMIT

ONLINE

Thank You

