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 Solution도출 / 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 없을까?

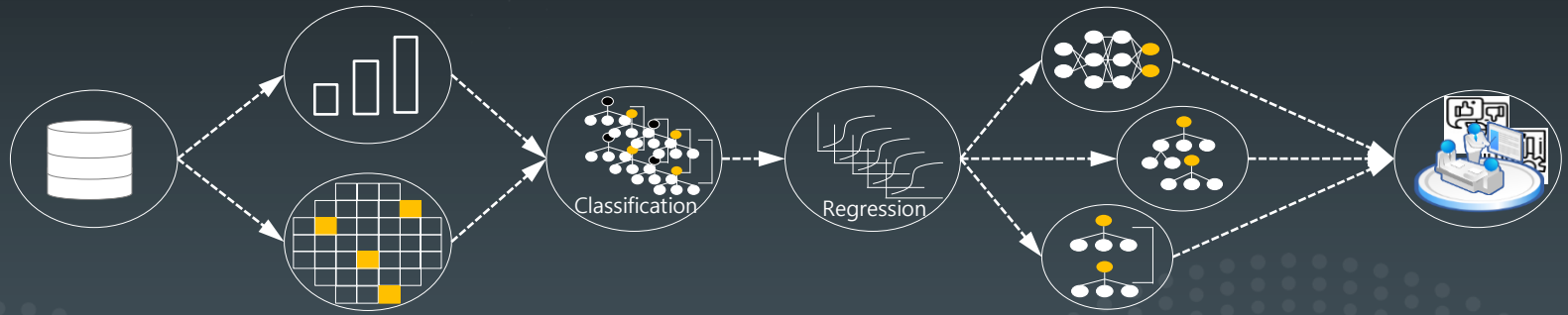
불량과 연관성이 높은 인자를 쉽게 찾을 수 없을까?

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

Data Preparation → **EDA** → ***CTQ Define & Model coef.*** → ***Model Ensemble*** → ••• → ***Eng'r Check***



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

DISCOVERY SUMMIT ONLINE

jmp

CTQ Define Engine Function

Data Preprocessing

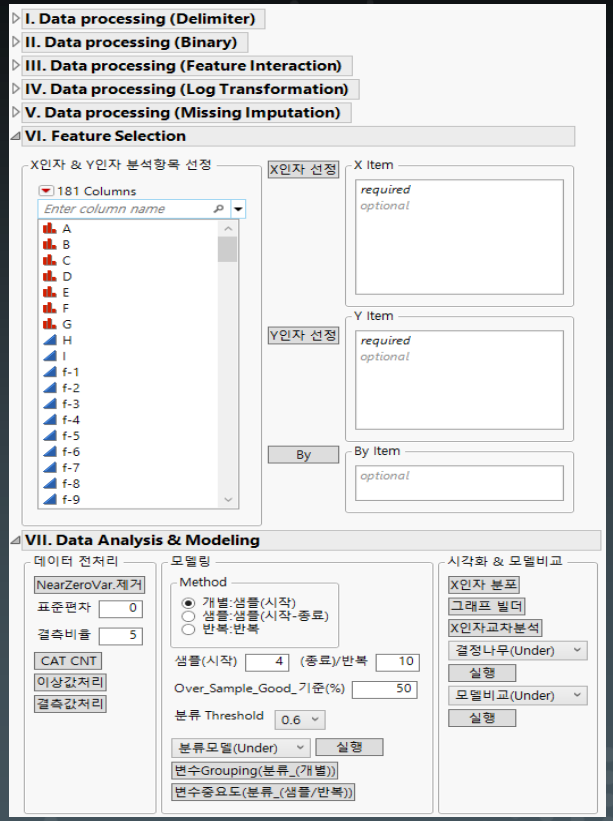
X, Y Variable Input

Data Preprocessing

Imbalance Data Sampling

Modeling & Ensemble

Data Visualization



[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

DISCOVERY SUMMIT ONLINE

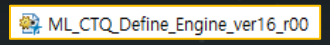
jmp

CTQ Define Engine Detail

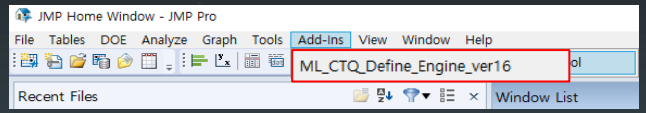
1. Add in 선택
2. Data Import
3. X,Y 인자 선정
4. Near Zero Var. 제거
5. Y 인자 별 발생 Count 확인
6. 이상값 처리 결측값 처리
7. Dataset 전처리

1. Add in 선택

-. JMP 실행 후 Addin File 실행 및 install

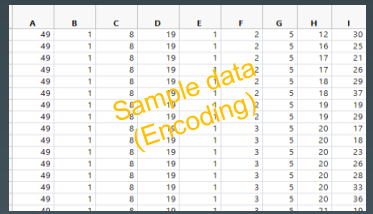


-. JMP 메뉴에 생성된 Add-Ins 선택



2. Data Import

-. Open Data File 창이 뜨면 저장된 데이터 테이블 Open

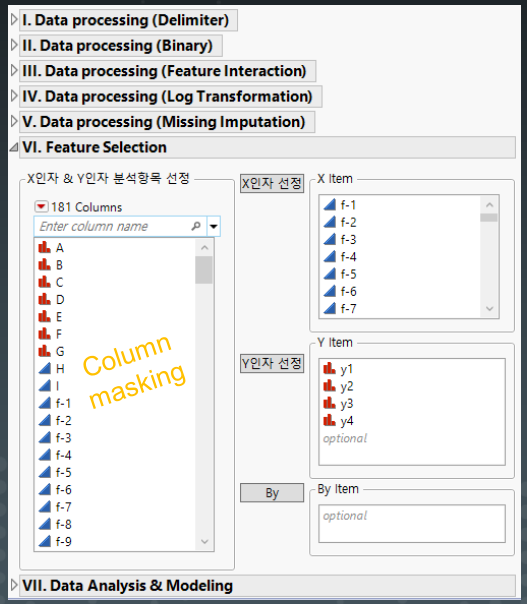


3. X,Y 인자 선정

-. 분석에 필요한 X인자와 Y 인자를 선정

→ Y 인자 : Nominal & Continuous 모두 분석 가능

(전처리가 필요할 경우 Data processing 기능 사용)

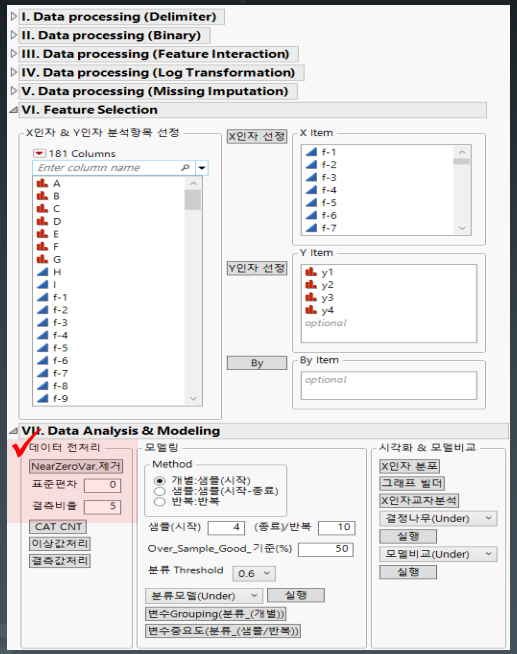


CTQ Define Engine Detail

1. Add in 선택
2. Data Import
3. X,Y 인자 선정
4. Near Zero Var. 제거
5. Y 인자 별 발생 Count 확인
6. 이상값 처리 결측값 처리
7. Dataset 전처리

4. Near Zero Var. 제거

-. Data의 편차가 없는 변수와 결측이 많은 변수 제거



- 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

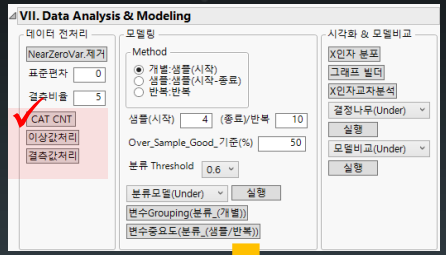
```
missing_rate = neb4 << Get; //neb: number edit box
ncols_rows = N Rows( dt );
missing_n = Floor( ncols_rows * (missing_rate / 100) ); //결측치를 %로 환산
For( z = 1, z <= ncols, z++,
Column( var_x[z] ) << Set Selected( 1 )
);
cc_select = dt << Get Selected Columns();
ncols1 = N Items( cc_select );
For( 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

1. Add in 선택
2. Data Import
3. X,Y 인자 선정
4. Near Zero Var. 제거
5. Y 인자 별 발생 Count 확인
6. 이상값 처리 결측값 처리
7. Dataset 전처리

5. Y 인자 별 발생 Count 확인

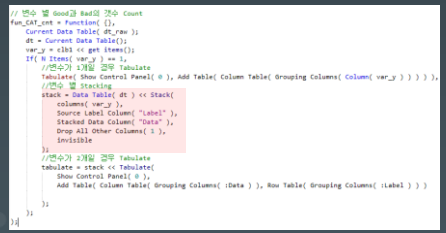
-. 여러 Y's 발생 수준을 파악할 경우 Modeling 전 확인



Tabulate

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

Y 변수 별 데이터의 불균형 수준을 파악

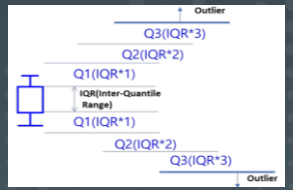


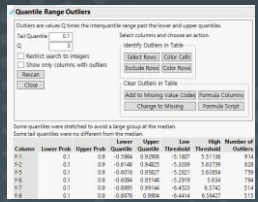
6-1. 결측값 처리

- 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 사이)의 Q배를 초과 시 outlier로 판단함.





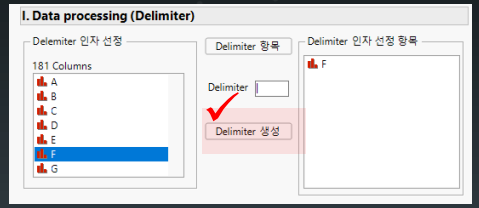
CTQ Define Engine Detail

1. Add in 선택
2. Data Import
3. X,Y 인자 선정
4. Near Zero Var. 제거
5. Y 인자 별 발생 Count 확인
6. 이상값 처리 결측값 처리
7. Dataset 전처리

7.1 Delimeter (별도 구현)

-. 특정 변수의 경우 구분자 '|' 로 구분되어 있음.

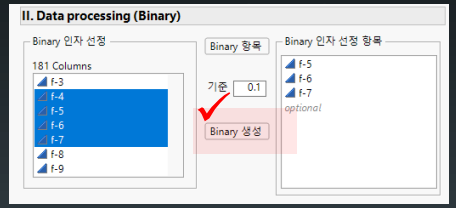
→ '|' 구분자를 기준으로 내부 값을 구분함.



7-2. Binary (별도 구현)

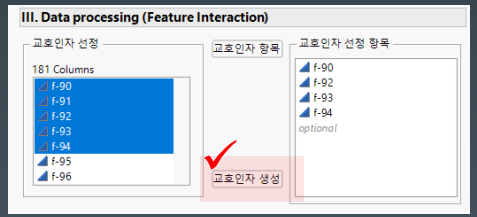
-. 연속형 Value 대상으로 이진변수 생성이 필요할 경우 사용

→ 분류 모델 적용할 경우 사용자 설정값을 기준으로 이진분류 가능



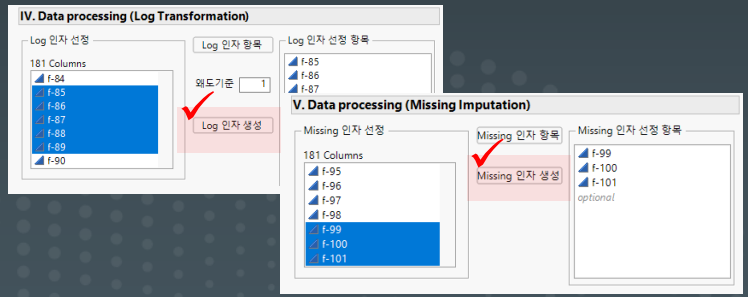
7-3. Feature Interaction (별도 구현)

-. 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

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

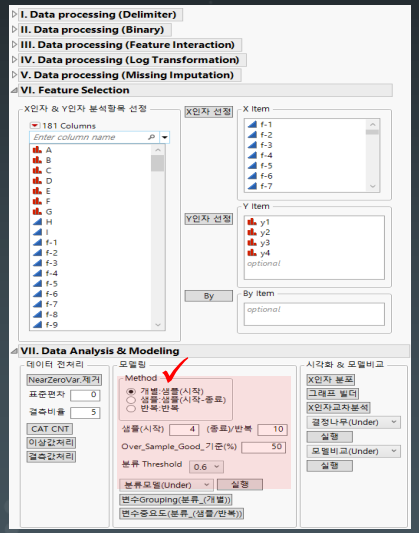
14 추가 검토 (분기점, para 관계)

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

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

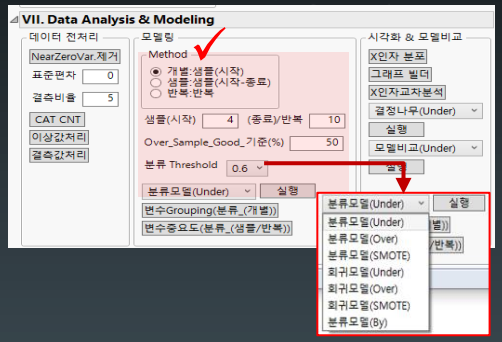
13 적용 Model Coef. 도출

14 추가 검토 (분기점, para 관계)

9. Y 별 분류 모델 샘플 Loop

-. CTQ 도출을 위한 사용 알고리즘

Random Forest & Logistic Regression

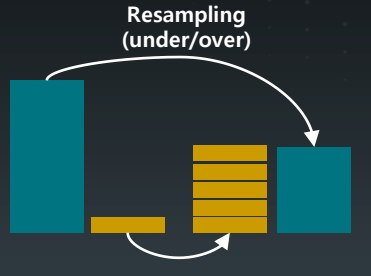



```
RF_sample = Sample_subset << Bootstrap Forest(
Y( Eval( var_y_new ) ), X( Eval( var_x_new ) ),
Validation Portion( 0.3 ),
Specify Profit Matrix( [0 -1.5, -1 0, . .], "0", "1", "Undecided" ),
Method( "Bootstrap Forest" ), Column Contributions( 1 ), ROC Curve( 1 ),
Portion Bootstrap( 1 ), Number Terms(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

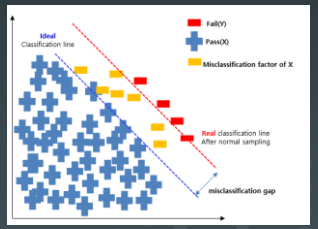
```
Sample_subset << Fit Model(
Y( Eval( var_y_new ) ), Effects( Eval( var_x ) ),
Personality( "Nominal Logistic" ),
Run( Positive Level( "0" ), ROC Curve( 1 ), Confusion Matrix( 1 ),
Specify Profit Matrix( [0 -1.5, -1 0, . .], "0", "1", "Undecided" )));
```

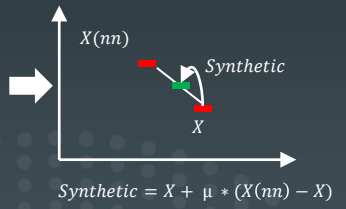
-. 반도체 Data의 경우 클래스 불균형 데이터가 일반적임. (과소/과대적합 발생)

-. 이를 위해 Under / Over sampling / SMOTE 방법을 통한 모델 성능 저하 방지 기능 구현



SMOTE concept





$$Synthetic = X + \mu * (X(nn) - X)$$

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);
```

DISCOVERY SUMMIT ONLINE

jmp

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

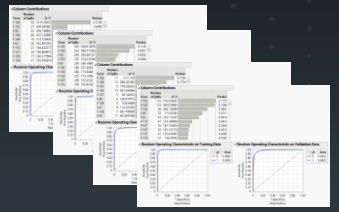
13 적용 Model Coef. 도출

14 추가 검토 (분기점, para 관계)

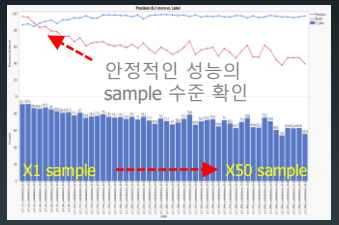
10. Y 별 분류 모델 반복 Loop

-. Sample 증가와 반복을 통한 모델 도출 및 세부 성능 지표 검토

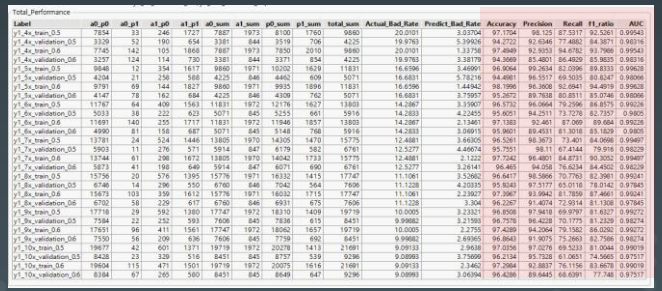
→ Threshold 변경에 따른 목표 성능 조절 가능



Resampling 기반 modeling 반복



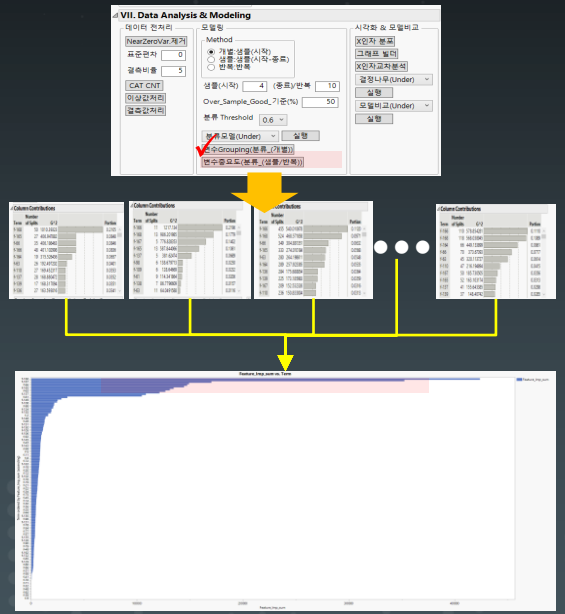
차수 별 model 성능 도출



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

8 X 인자 분포 & 그래프 확인

9 Y 별 분류 모델 샘플 Loop

10 Y 별 분류 모델 반복 Loop

11 변수 중요도 확인

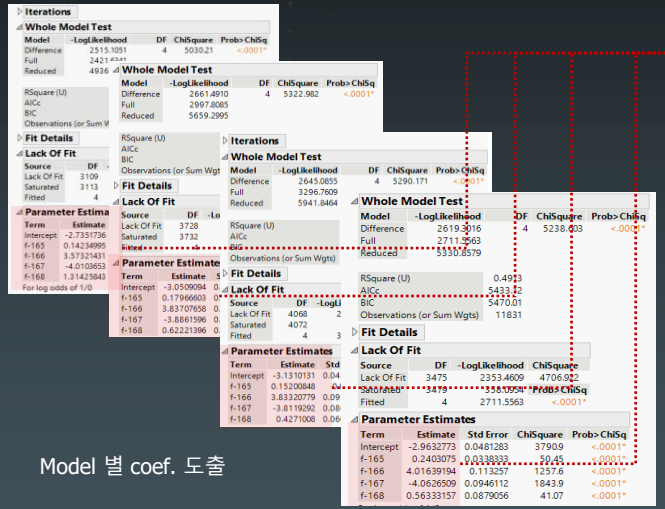
12 Logistic Regression 기반 Coef. 도출

13 적용 Model Coef. 도출

14 추가 검토 (분기점, para 관계)

12. Logistic Regression 기반 Coef. 도출

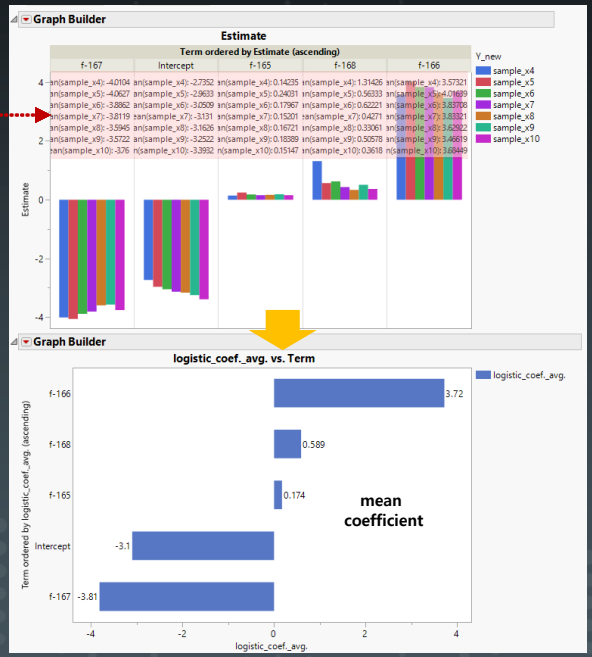
-. Y인자와 선정된 CTQ 인자에 대해 반복적으로 회귀 모델링하여 Coef. 변화 수준 확인함.



Model 별 coef. 도출

13. 적용 Model Coef. 도출

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



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

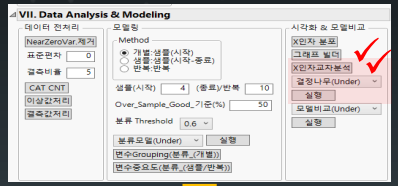
12 Logistic Regression 기반 Coef. 도출

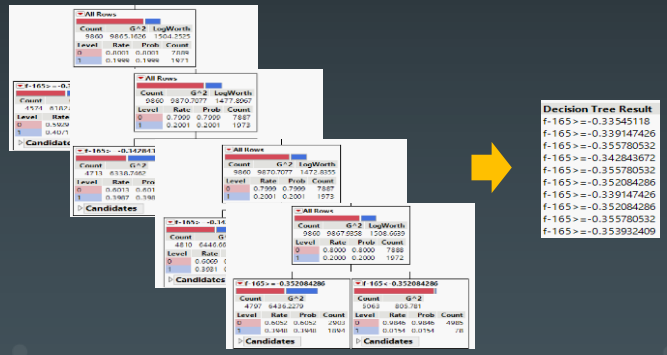
13 적용 Model Coef. 도출

14 추가 검토 (분기점, para 관계)

14. 추가 검토 (분기점, para 관계)

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

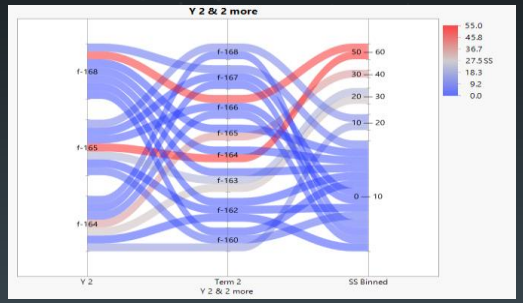




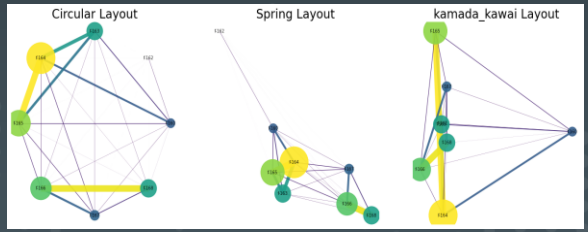
Decision Tree 활용 주요 변수 분기점 파악
(자동 반복 기반 output 도출)

-. 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

Thank You