응용데이터애널릭틱스
프로젝트 발표

Team1
김수빈
김주환
박주연
이채희

Contents

01 Executive Summary

- 얻을 수 있는 Domain 정보를 최대한 활용함.
- Visualization을 통해 얻은 데이터 내부의 정보를 이용.
- 각 모델마다 최적의 parameter를 찾기 위해 Grid search 활용.
- 모델 성능 Metric 을 검토하며 파생변수 생성을 다양하게 시도함.
- Decision Tree, Random Forest, AdaBoost를 전부 사용해보고 결과를 분석함.

01 Executive Summary

Results

Custom Testset	DT	RF	AF
MSE	2,533,982,395,323	927,412,197,156	780,692,867,541
MAE	1,032,578	632,290	591,752
R^2 (adjusted X)	0.9410	0.9784	0.9818
Error rate	14.87%	9.76%	8.92%

P1_sample	DT	RF	AF
MSE	3,023,895,000,000	2,330,743,430,222	901,920,000,000
MAE	1,318,500	1,239,853	822,000
R^2 (adjusted X)	0.8968	0.9204	0.9692
Error rate	19.51%	25.63%	15.08%

$$Error\ rate = \frac{|Ground\ Truth - Prediction|}{Ground\ Truth} * 100\ (\%)$$

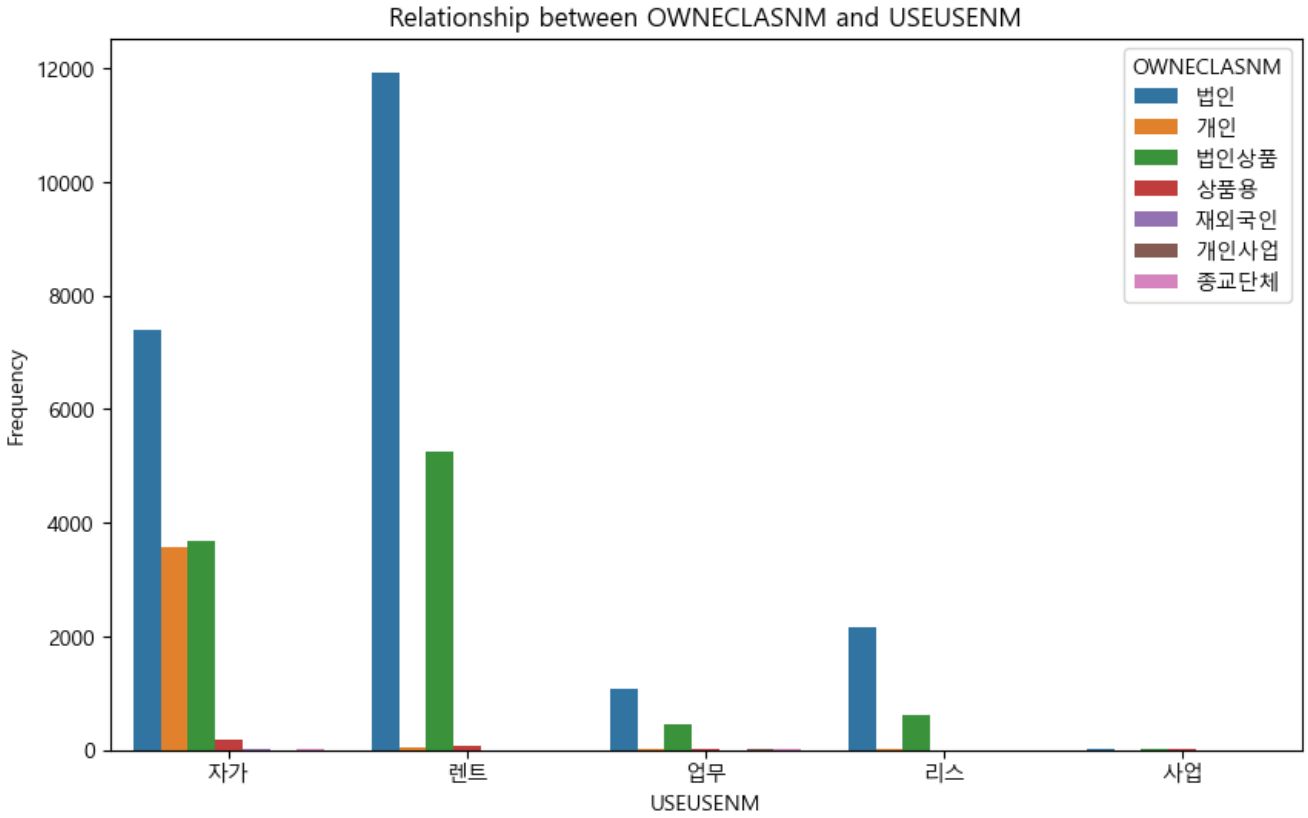
02 Feature Engineering

Missing Values in Discrete Variables

MISSNM	0
FUELNM	1
COLOR	0
USEUSENM	323
OWNECLASNM	13
INNEEXPOCLASCD_YN	0
SUCCYMD	0
CARNM	0
CHASNO	0
CARREGIYMD	1

02 Feature Engineering

Missing Values in Discrete Variables



OWECLASNM == 개인

↓

USEUSENM == 자가

*나머지 USEUSENM의 NA값들은 '기타' 로 처리

02 Feature Engineering

Missing Values in Continuous Variables

- 5개 연속형 변수들을 PCA 한 결과 4개로 축소됨.
- NEWCARPRIC가 1000 미만일 경우 NA로 판단. -> 24개 NaN 추가.
- 전체 중앙값으로 결측치 대체 or 각 모델별 중앙값으로 결측치 대체.

```
df_1['NEWCARPRIC'] = df_1['NEWCARPRIC'].replace(0, np.nan)
df_1['NEWCARPRIC'] = df_1['NEWCARPRIC'].replace(1, np.nan)
df_1['NEWCARPRIC'] = df_1['NEWCARPRIC'].apply(lambda x: np.nan if x < 1000 else x)
```

✓ 0.0s

	MJ_MODEL_KEY	DT_MODEL_KEY	MODEL
30557	46	1053	46-1053
22878	46	210	46-210
8898	46	210	46-210
31614	39	537	39-537
6258	46	210	46-210
...	...	...	...
16850	46	210	46-210
6265	38	166	38-166
11284	39	145	39-145
860	42	233	42-233
15795	71	93	71-93

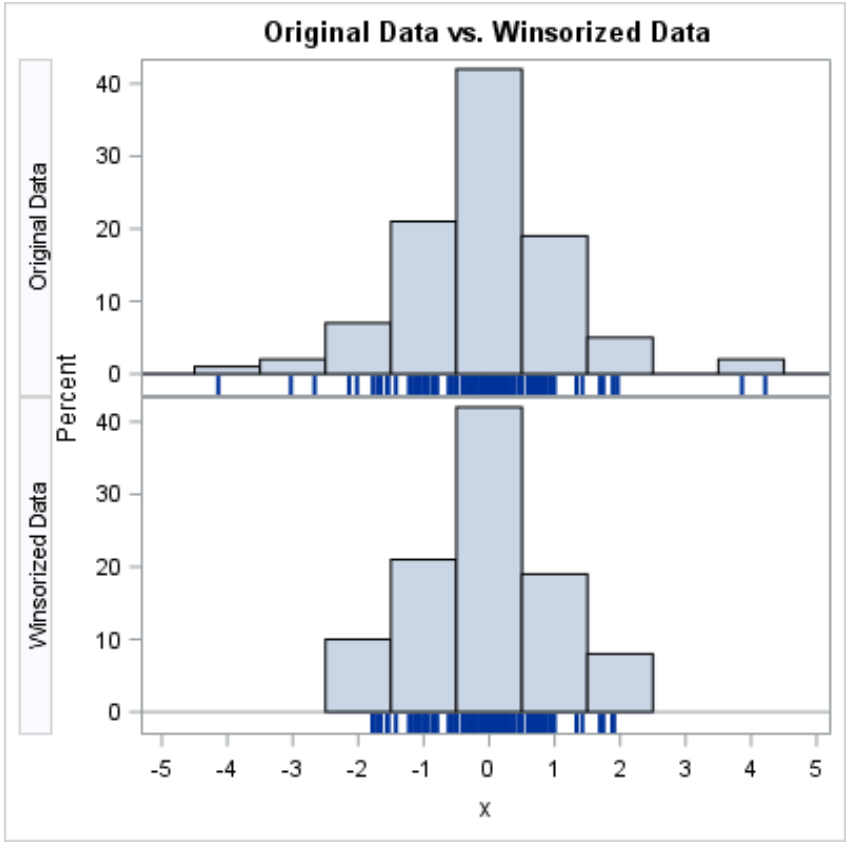
```
dict(price_model)
```

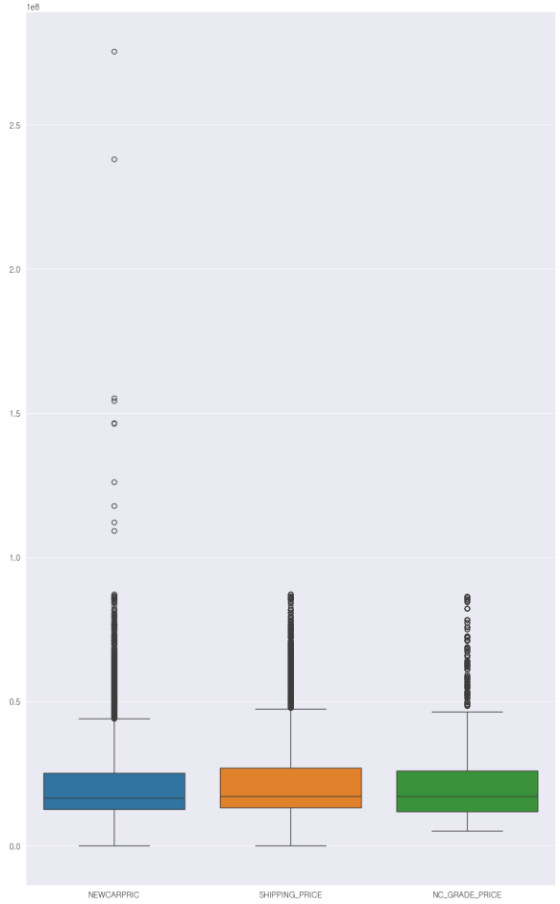
✓ 0.0s

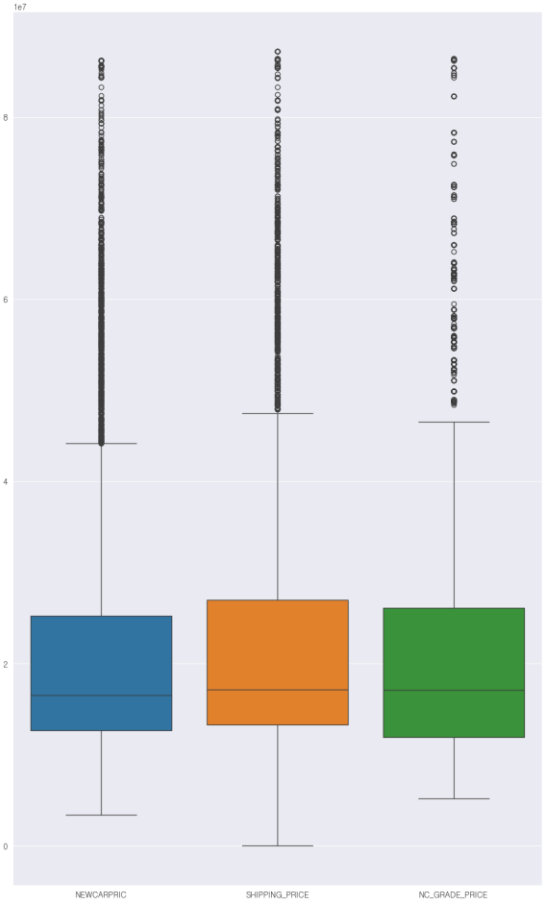
```
{'1160-2597': 49500000.0,
 '37-1588': 18380000.0,
 '37-2646': 25850000.0,
 '37-295': 16422277.0,
 '37-938': 19960000.0,
 '38-1560': 22080000.0,
 '38-1592': 30270000.0,
 '38-166': 18831000.0,
 '38-223': 29270000.0,
 '38-2635': 32640000.0,
 '38-2672': 36230000.0,
 '38-935': 17530000.0,
 '39-145': 31375403.0,
 '39-1598': 32770200.0,
 '39-2570': 41905000.0,
```

02 Feature Engineering

Dealing With Outliers







02 Feature Engineering

Making Derived Variables

```
# SUCCYMD(낙찰일자) - CARREGIYMD(차량등록일) 을 새로운 변수 Days로 저장
df_3['SUCCYMD'] = pd.to_datetime(df_3['SUCCYMD'], format='%Y%m%d')
df_3['CARREGIYMD'] = pd.to_datetime(df_3['CARREGIYMD'], format='%Y%m%d')
df_3['Days'] = (df_3['SUCCYMD'] - df_3['CARREGIYMD']).dt.days.astype(int)
```

✓ 0.0s

	SUCCYMD	CARREGIYMD	Days
30557	2018-10-16	2015-07-06	1198
22878	2018-02-27	2012-03-20	2170
8898	2016-11-03	2014-06-20	867
31614	2018-11-13	2015-11-06	1103
6258	2016-08-05	2011-08-24	1808
...	...	...	...
16850	2017-07-21	2011-02-07	2356
6265	2016-08-05	2013-06-18	1144
11284	2017-01-12	2010-06-29	2389
860	2016-02-05	2013-01-28	1103
15795	2017-06-13	2010-10-06	2442

제8조(사용년 계수) 차량의 사용연수에 따른 계수는 아래[표]와 같고, 사용연수 산출은 연도만으로 계산한다.

(예) 최초등록일이 2014.12.15이고, 평가일이 2020.07.12.인 경우,

① 사용연수 : 2020년 - 2014년 = 6년

② 사용월수 : (6년 × 12개월) - 5개월 = 67개월

KAIWA 한국자동차진단보증협회 Korea Automotive Inspection and Warranty Association

한국자동차진단보증협회 가격조사산정기준서
→사용연수가 중고차량의 가격에 영향을 줌.

02 Feature Engineering

Making Derived Variables

3. 자동차성능·상태점검자가 거짓으로 자동차성능·상태 점검을 하거나 실제 점검한 내용과 다른 내용을 제공한 경우 「자동차관리법」 제80조제9호의2에 따라 2년 이하의 징역 또는 2천만원 이하의 벌금에 처하며, 「자동차관리법 제21조제2항 등의 규정에 따른 행정처분의 기준과 절차에 관한 규칙」 제5조제1항에 따라 1차 사업정지 30일, 2차 사업정지 90일, 3차 사업장 폐쇄의 행정처분을 받습니다.

4. ㉮ 사고이력 인정은 사고로 자동차 주요 골격 부위의 판금, 용접수리 및 교환이 있는 경우로 한정합니다. 단, 쿼터패널, 루프패널, 사이드실패널 부위는 절단, 용접 시에만 사고로 표기합니다.
(후드, 프론트펜더, 도어, 트렁크리드 등 외판 부위 및 범퍼에 대한 판금, 용접수리 및 교환은 단순수리로서 사고에 포함되지 않습니다)

5. 자동차성능·상태점검은 국토교통부장관이 정하는 자동차성능·상태점검 방법에 따라야 합니다.

6. 「자동차관리법 시행규칙」 제120조제2항에 따라 자동차성능·상태점검기록부는 해당 기록부의 발급일을 기준으로 120일 이내에 이루어진 자동차 성능·상태점검으로 한정합니다

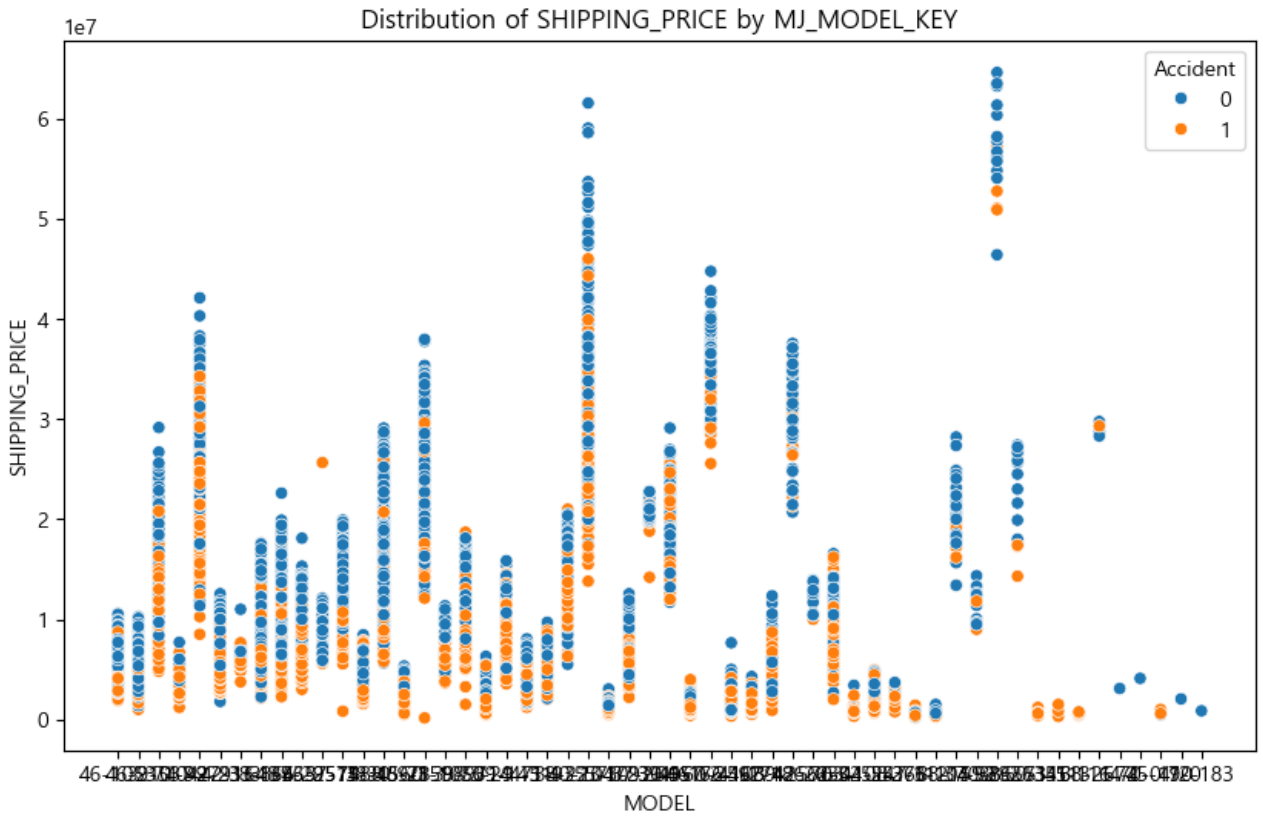
7. 체크항목 판단기준(예시)

   ㅇ 미세누유(미세누수): 해당부위에 오일(냉각수)이 비치는 정도로서 부품 노후로 인한 현상
   ㅇ 누유(누수): 해당부위에서 오일(냉각수)이 맺혀서 떨어지는 상태

02 Feature Engineering

Making Derived Variables

```
# Outer_frame에 해당하는 값이 1이 하나라도 있으면 1, 없으면 0으로 설정
df_3['Accident'] = df_3[outer_frame].max(axis=1)
```



02 Feature Engineering

Making Derived Variables

3. 사고이력에 따른 랭크 분류

랭크분류	적용 부위	평 가 기 준
1랭크	1. 후드	- 교환여부(X) - 볼트가 전부 풀렸거나, 해당부품 색 차이 여부(X) - 판금, 용접수리 여부(W)
	2. 프런트 펜더	
	3. 도어	
	4. 트렁크 리드	
	5. 라디에이터 서포트(볼트체결)	
2랭크	6. 쿼터 패널(리어펜더)	- 교환여부(X) - 용접수리 여부(W) - 해당 부품이 구겨진 흔적이나 망치로 핀 자국이 있는 경우(W)
	7. 루프 패널	
	8. 사이드 실 패널	
A랭크	9. 프런트 패널	- 교환여부 (X) - 용접수리 여부(W) - 해당 부품이 구겨진 흔적이나 망치로 핀 자국이 있는 경우(W)
	10. 크로스 멤버(용접부품)	
	11. 인사이드 패널	
	17. 트렁크 플로어 패널	
	18. 리어 패널	
B랭크	12. 사이드 멤버	- 교환여부 (X) - 용접수리 여부(W) - 해당 부품이 구겨진 흔적이나 망치로 핀 자국이 있는 경우(W)
	13. 휠 하우스	
	14. 필러 패널	
	19. 패키지 트레이	
C랭크	15. 대쉬 패널	- 교환여부 (X) - 용접수리 여부(W) - 해당 부품이 구겨진 흔적이나 망치로 핀 자국이 있는 경우(W)
	16. 플로어 패널	

```
##사고 유형 분류
# 외판부위 1랭크
outer_panels_rank1 = ['BONET', 'FRONT_LEFT_FENDER', 'FRONT_RIGHT_FENDER', 'FRONT_LEFT_DOOR', 'FRONT_RIGHT_DOOR',
                      'BACK_LEFT_DOOR', 'BACK_RIGHT_DOOR', 'TRUNK', 'FRONT_PANNEL']

# 외판부위 2랭크
outer_panels_rank2 = ['LEFT_REAR_FENDER', 'RIGHT_REAR_FENDER', 'LEFT_QUARTER', 'RIGHT_QUARTER']

# 주요골격 A랭크
main_frame_rankA = ['FRONT_PANNEL', 'TRUNK_FLOOR', 'LEFT_INSIDE_PANEL', 'RIGHT_INSIDE_PANEL']

# 주요골격 B랭크
main_frame_rankB = ['SIDE_MEMBER_FRAME', 'LEFT_WHEEL_HOUSE', 'RIGHT_WHEEL_HOUSE', 'LEFT_FILER_A', 'RIGHT_FILER_A',
                    'LEFT_FILER_B', 'RIGHT_FILER_B', 'LEFT_FILER_C', 'RIGHT_FILER_C']

# 주요골격 C랭크
main_frame_rankC = ['DASH_PANEL', 'FLOOR_PANEL']
```

```
# 외판, 골격 세분화
df_3['outer_1'] = df_3[outer_panels_rank1].sum(axis=1)
df_3['outer_2'] = df_3[outer_panels_rank2].sum(axis=1)
df_3['main_A'] = df_3[main_frame_rankA].sum(axis=1)
df_3['main_B'] = df_3[main_frame_rankB].sum(axis=1)
df_3['main_C'] = df_3[main_frame_rankC].sum(axis=1)
```

02 Feature Engineering

Making Derived Variables

	MJ_GRADE_KEY	DT_GRADE_KEY	NC_GRADE_KEY	SUCCPRIC	SUCCPRIC_MEAN_MJ_GRADE_KEY	SUCCPRIC_MEAN_DT_GRADE_KEY	SUCCPRIC_MEAN_NC_GRADE_KEY
0	10,912	24,318	21,215	4,770,000	6,315,437	5,327,506	5,327,506
1	9,101	20,963	8,599	5,610,000	4,900,656	4,775,172	4,619,680
2	9,100	20,959	9,861	6,040,000	3,591,906	3,624,607	3,584,820
3	2,532	22,345	15,617	18,150,000	16,385,068	16,330,661	16,330,661
4	9,101	20,966	8,600	5,500,000	4,900,656	5,937,889	5,221,726
5	8,728	0	0	4,980,000	4,640,000	2,118,818	4,110,643
6	1,168	21,419	13,110	14,600,000	22,809,105	20,230,357	20,230,357
7	9,159	21,243	9,326	5,210,000	5,926,138	5,097,935	5,097,935
8	17,276	33,596	34,716	6,000,000	6,492,391	6,492,391	6,492,391
9	5,782	20,504	10,891	6,950,000	6,396,387	6,978,568	7,052,961

*MJ_GRADE_KEY, DT_GRADE_KEY, NC_GRADE_KEY에 따른 평균 SUCCPRIC 계산

→ 새 column으로 추가

→ 모델별 평균 SUCCPRIC 순서대로 구간화(binning)하려 했으나 실수로 인해 이 부분이 반영되지 않음

```
# # 구간을 정의하고 각 카테고리를 구간에 따라 인코딩
# bins = [0, 12000000, 24000000, 36000000, 48000000, np.inf]
# labels_MJ = [0, 1, 2, 3, 4]
# train['SUCCPRIC_MEAN_MJ_GRADE_KEY'] = pd.cut(df_3['SUCCPRIC_MEAN_MJ_GRADE_KEY'], bins=bins, labels=labels_MJ)
# labels_DT = [0, 1, 2, 3, 4]
# train['SUCCPRIC_MEAN_DT_GRADE_KEY'] = pd.cut(df_3['SUCCPRIC_MEAN_DT_GRADE_KEY'], bins=bins, labels=labels_DT)
# labels_NC = [0, 1, 2, 3, 4]
# train['SUCCPRIC_MEAN_NC_GRADE_KEY'] = pd.cut(df_3['SUCCPRIC_MEAN_NC_GRADE_KEY'], bins=bins, labels=labels_NC)
```

02 Feature Engineering

Encoding Categorical Variables

*Label Encoding -> FUELNM, USEUSENM, OWNECLASNM

- N개의 범주형 데이터를 0~n-1의 연속적인 정수로 표현
- 두 개의 범주일 때 독립적인 의미로 사용 가능
- DecisionTree 분류 모델에서는 범주가 3개 이상이어도 사용 가능

*OneHot Encoding -> YEARCHK(Y/N), INNEEXPOCLASCD_YN(O/X)

- N개의 범주형 데이터를 n개의 (0,1)벡터로 표현함
- 서로 다른 범주에 대해서는 벡터 내적을 취했을 때 내적 값 0
- 서로 다른 범주 데이터를 독립적인 의미로 사용 가능
- 여러 범주형 변수를 한 번에 인코딩 가능

02 Feature Engineering

Encoding Categorical Variables

```
MISSNM
M/T    3216799
CVT    6936938
A/T    8907228
Name: SUCCPRIC, dtype: int32
```

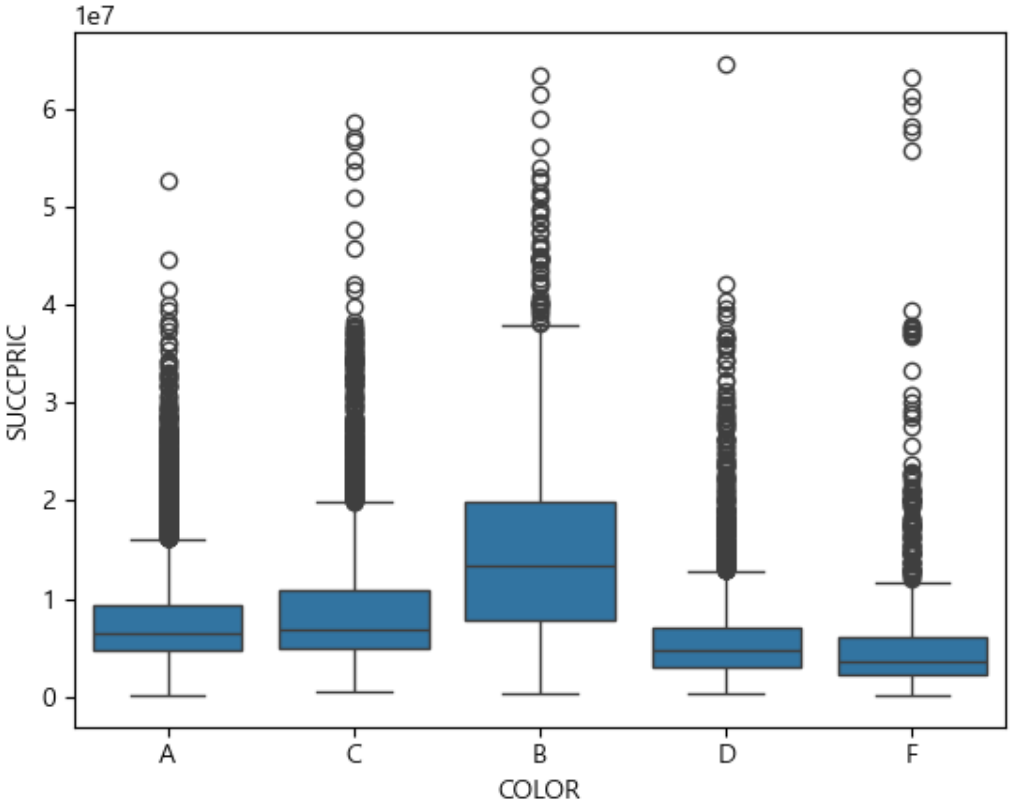
```
missnm_order = ['M/T', 'CVT', 'A/T']
encoder = OrdinalEncoder(categories=[missnm_order])
df_4['MISSNM'] = encoder.fit_transform(df_4[['MISSNM']])
```

*MISSNM의 평균 SUCCPRIC 순위에 따른 Ordinal Encoding

A/T: 자동변속기
CVT: 무단변속기
M/T: 수동변속기

02 Feature Engineering

Encoding Categorical Variables



*COLOR의 SUCCPRIC(boxplot)에 따른 Ordinal Encoding

```
color_order = ['B', 'C', 'A', 'D', 'F']
encoder = OrdinalEncoder(categories=[color_order])
df_4['COLOR'] = encoder.fit_transform(df_4[['COLOR']])
```

02 Feature Engineering

Encoding Categorical Variables

```
for item in new_one_hot_list:
    left_col = f'LEFT_{item}'
    right_col = f'RIGHT_{item}'
    new_col_name = item

    df_4[new_col_name] = df_4[left_col] + df_4[right_col]
    df_4 = df_4.drop([left_col, right_col], axis=1)

df_4[new_one_hot_list][df_4[new_one_hot_list]==2]=1
```

✓ 0.4s

```
df_4.shape
```

✓ 0.0s

```
(29423, 109)
```

→

```
df_4.shape
```

✓ 0.0s

```
(29423, 87)
```

*Dimension reduction

03 Algorithm

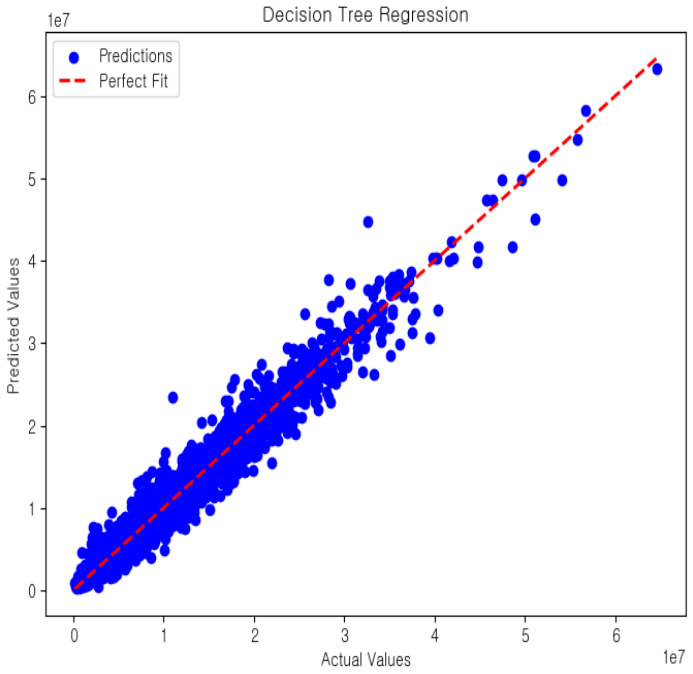
*Hyperparmeter

- DecisionTree
  - Max_depth = [30, 40, 50]
  - Max_features = [0.3, 0.5, 0.7]
- RandomForest
  - Max_depth = [50, 75, 100]
  - N_estimator = [50, 100, 150]
- AdaBoost
  - Max_depth = [5, 10, 30]
  - N_estimator = [50, 100, 150]

04 Result & Discussions

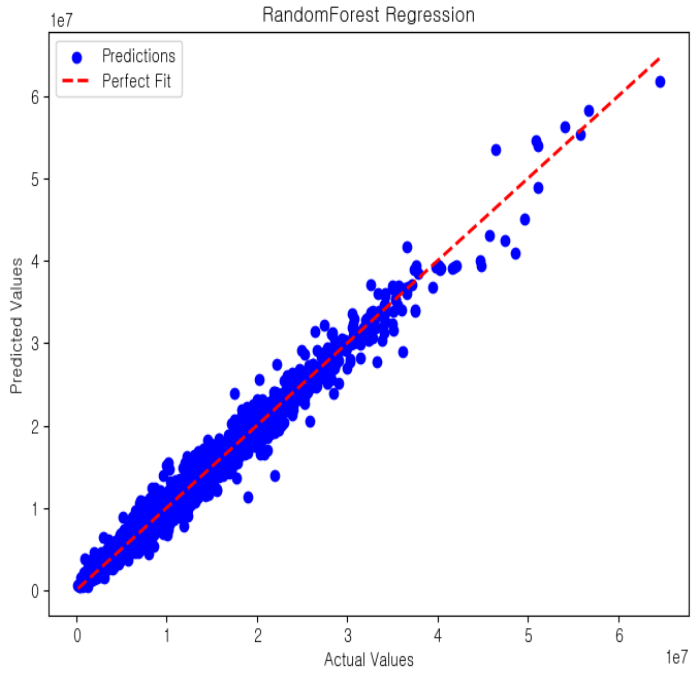
1. Best Hyperparameter

DEICISION TREE



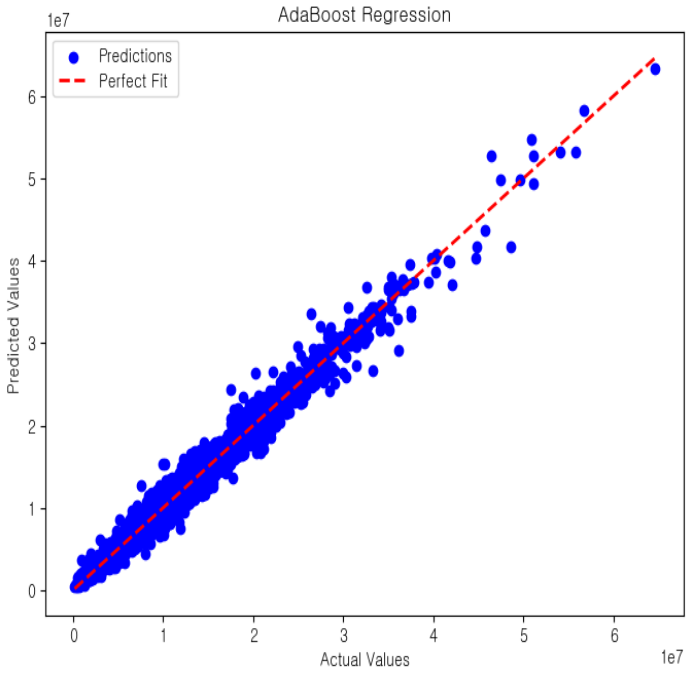
- 'max_depth': 40, 'max_features': 0.7
- Best MSE : 1,677,614,953,271.028

Random Forest



- 'n_estimators': 150, 'max_depth': 50
- Best MSE : 790,743,764,764.6559

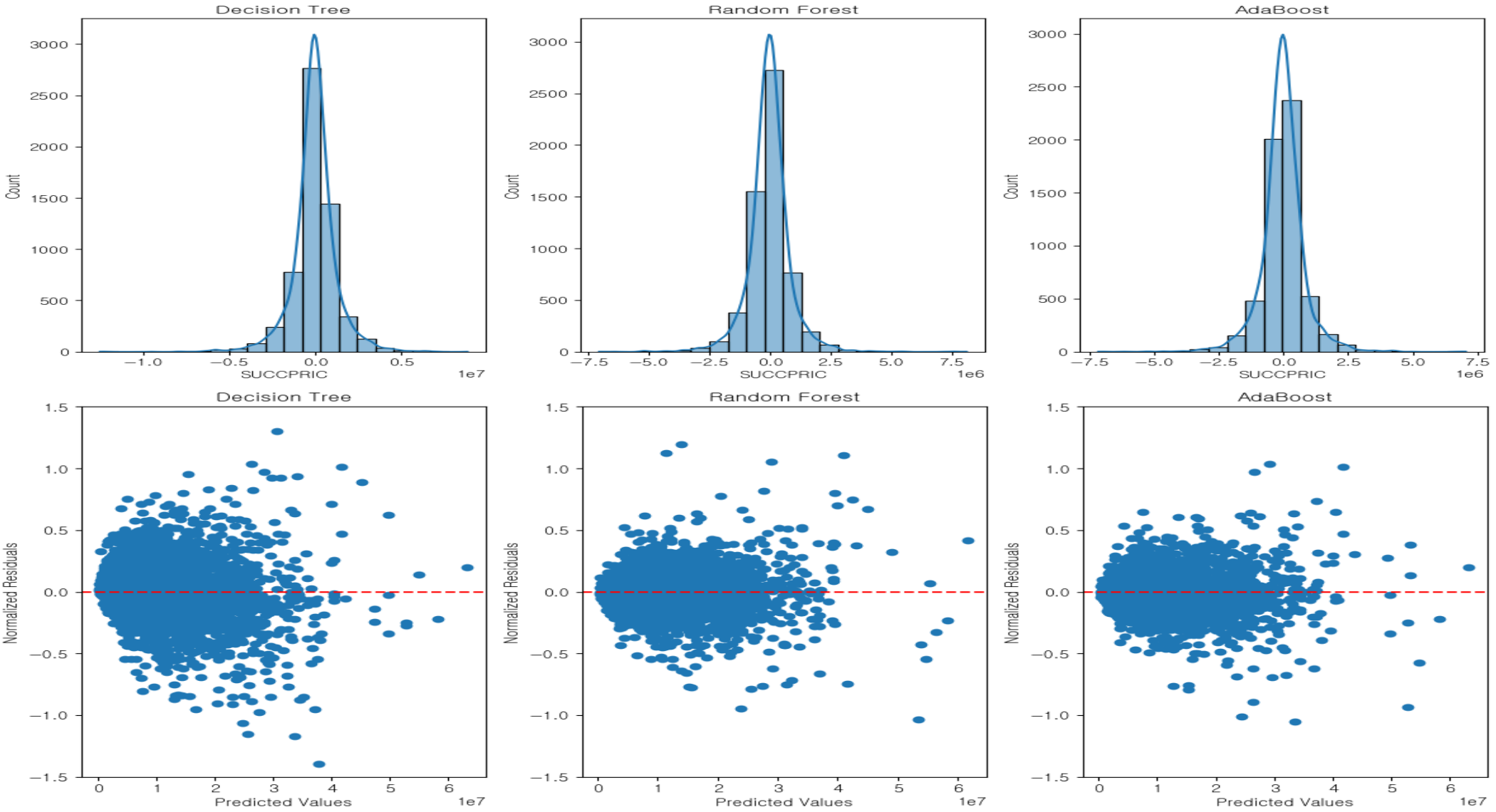
Ada Boost



- 'n_estimators': 150, 'max_depth': 30
- Best MSE : 733,739,605,132.7374

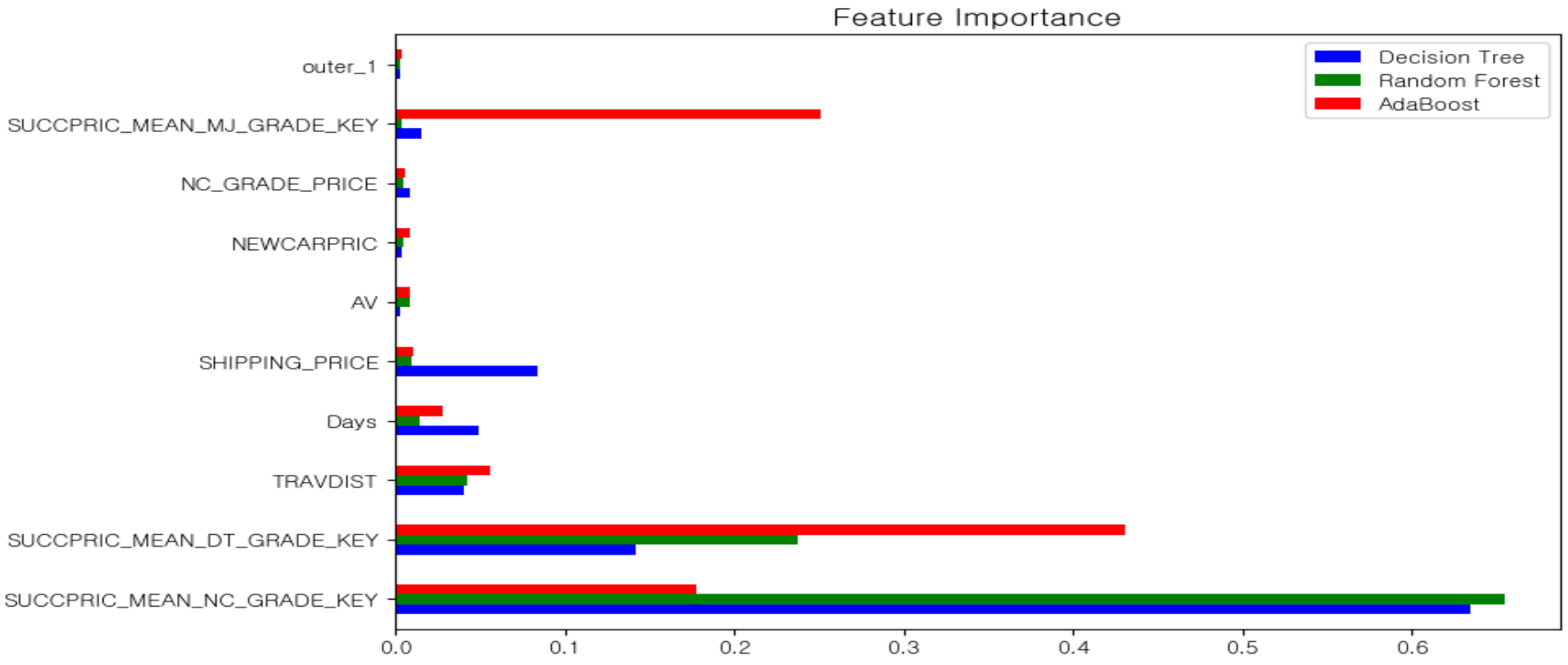
04 Result & Discussions

2. Residual Distribution



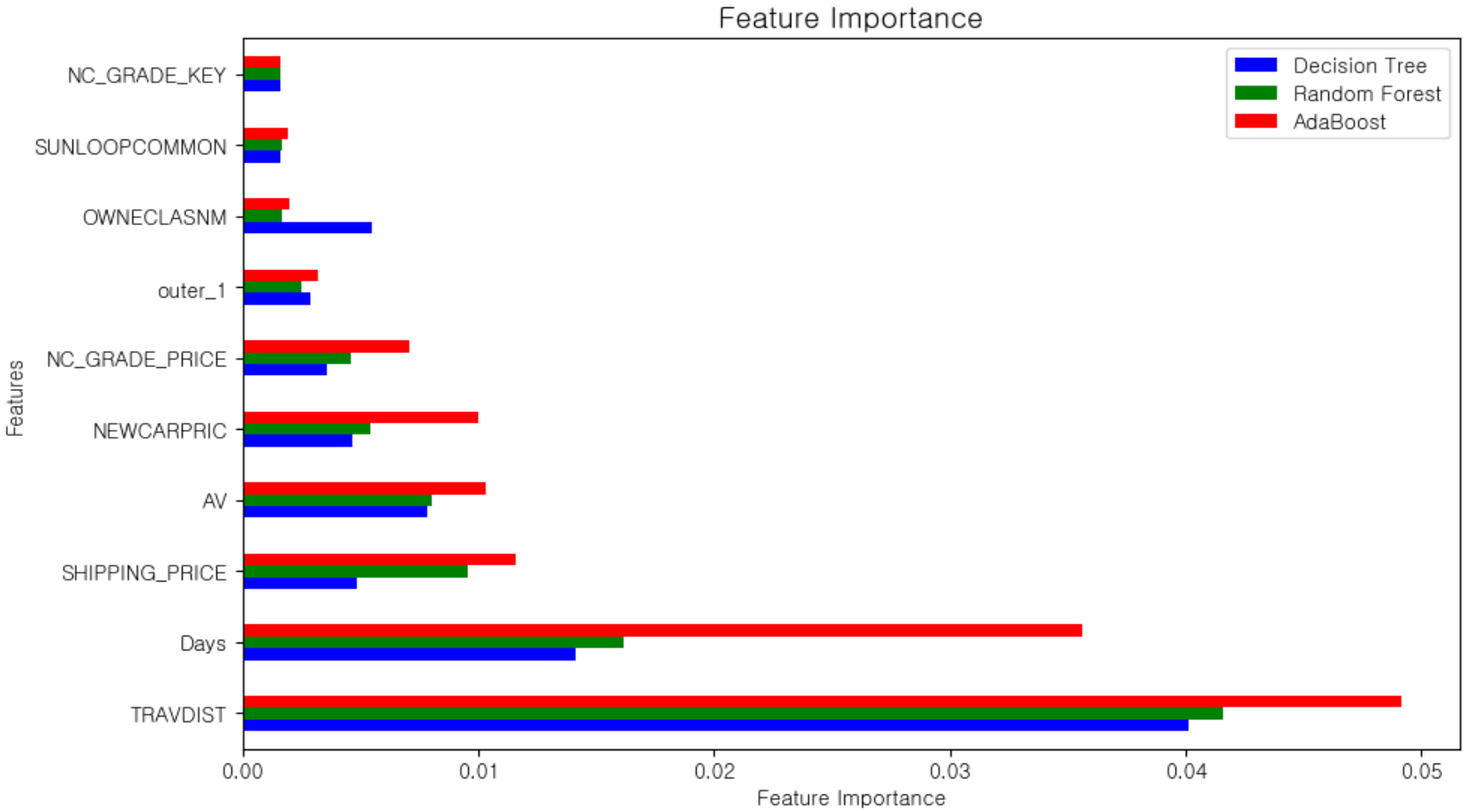
04 Result & Discussions

3. Feature Importance



04 Result & Discussions

3. Feature Importance



Feature Importance

- 모델별로 변수 중요도는 다름.
- Days, SHIPPING_PRICE, TRAVDIST 는 모든 세 모델에서 비슷하게 비슷한 중요도를 가짐.

04 Result & Discussions

4. System implementation 결과

Custom Testset	DT	RF	AF
MSE	2,533,982,395,323	927,412,197,156	780,692,867,541
MAE	1,032,578	632,290	591,752
R^2 (adjusted X)	0.9410	0.9784	0.9818
Error rate	14.87%	9.76%	8.92%

P1_sample	DT	RF	AF
MSE	3,023,895,000,000	2,330,743,430,222	901,920,000,000
MAE	1,318,500	1,239,853	822,000
R^2 (adjusted X)	0.8968	0.9204	0.9692
Error rate	19.51%	25.63%	15.08%

System implementation 결과

- 모델별 예측 성능 : Decision Tree < Random Forest < Ada Boost
- Custom Test/P1_Sample 결과와 비교했을 때 성능이 저하됨.
- Ada Boost는 두 데이터 세트 모두에서 지속적으로 비슷한 성능을 보임.
- P1_dataset_sample은 크기가 작아 실제 데이터 분포를 정확하게 반영하지 못할 수 있음.
- 모델들이 과적합되었을 가능성이 있음.

04 Result & Discussions

4. 원래 의도대로 했을 때 결과

Custom Testset	DT	RF	AF
MSE	1,666,291,394,779	763,079,765,611	737,414,165,081
MAE	850,623	579,384	569,462
R^2 (adjusted X)	0.9612	0.9822	0.9828
Error rate	12.80%	9.07%	8.70%

P1_sample	DT	RF	AF
MSE	1,804,125,000,000	2,249,654,180,444	1,674,350,000,000
MAE	816,500	971,646	1,016,000
R^2 (adjusted X)	0.9384	0.9232	0.9428
Error rate	9.94%	12.27%	15.78%

System implementation 결과

- 모델별 예측 성능 : Decision Tree < Ada Boost < Random Forest
- Custom Test/P1_Sample 결과와 비교했을 때 성능이 저하됨.
- P1_dataset_sample은 크기가 작아 실제 데이터 분포를 정확하게 반영하지 못할 수 있음.
- 원래 의도대로 파생변수 생성시 결과가 바뀌나, 비슷하다고 해석하는 것이 타당해 보임.

05 Conclusions

1. 전체 소결

프로젝트 목적

- Decision Tree, Random Forest, 그리고 AdaBoost 를 이용한 중고차 시세 예측

결과 요약

- AdaBoost가 전반적으로 가장 뛰어난 성능을 보임.
- Train/Test 데이터에서는 Decision Tree와 Random Forest가 좋은 성능을 보였지만, System implementation 결과에서는 높은 MAE 값을 보임.

추가 개선 방안

- 의도와 다르게 들어간 파생변수 수정.
- 변수들의 상관관계를 고려한 변수의 수 축소.
- 추가한 파생변수들 사이에서 강한 상관관계가 나타남. -> 대표등급키에 대해서만 파생변수 생성.
- 과적합 가능성을 고려한 모델 학습.

감사합니다