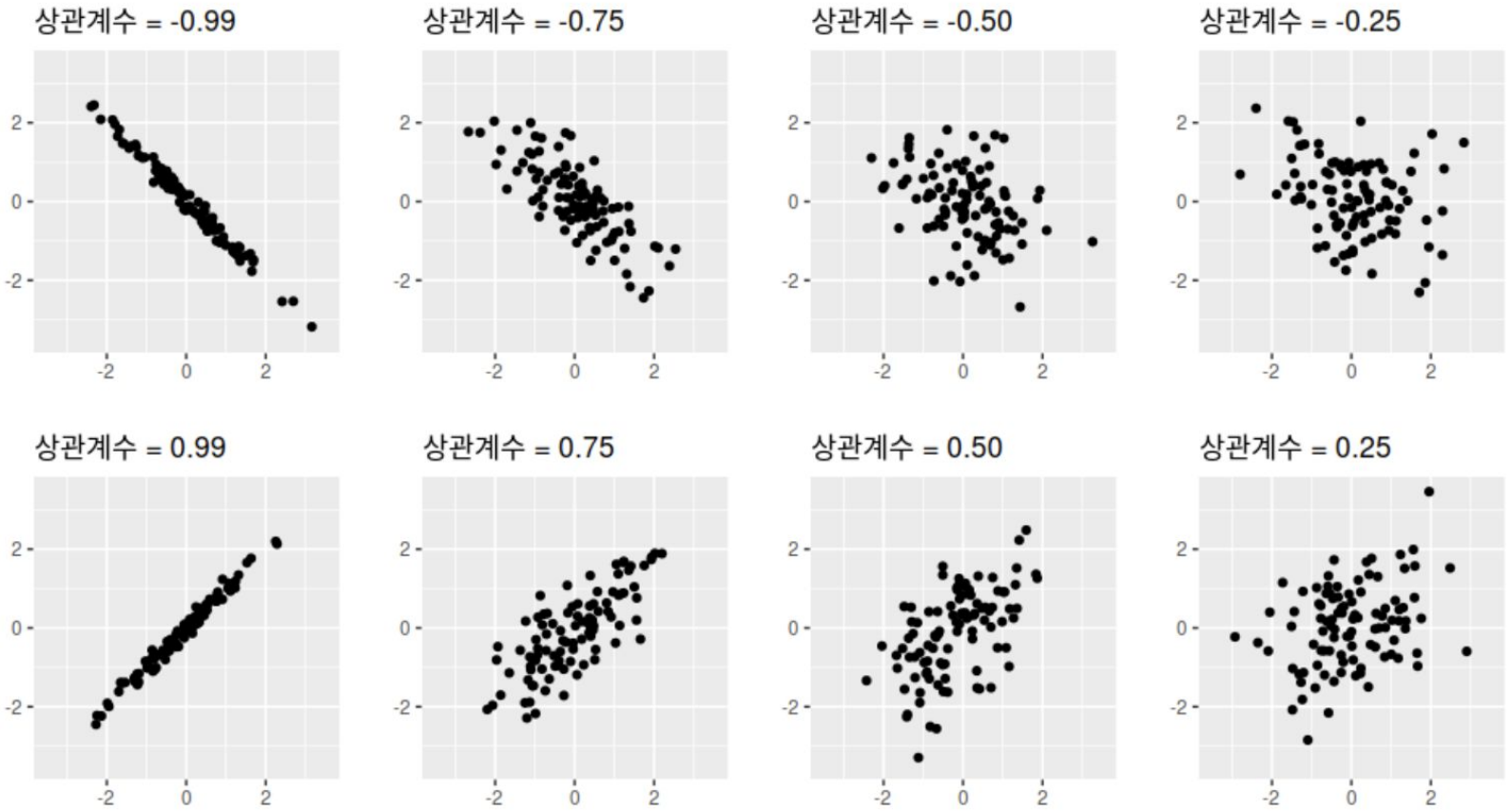
▣ 분석 개요

- 자동차 부품 생산과 관련된 데이터를 통해 부품의 품질 분석
  - 목표 : 다양한 변수들로 부터 부품의 강도를 예측하는 것
  - 연속형 변수인 부품의 강도를 통해 분류 레이블을 도입하여 failure type을 예측하는 분류 문제로도 분석 가능

- Dataset
  - 부품 번호, 공정 속성, 타겟 변수인 부품 강도 등으로 이루어진 데이터로 총 34,139개
  - 총 6개의 부품 중 부품 번호가 “90784-76001”인 부품만 분석에 사용(21,779개)
  - https://drive.google.com/file/d/1Z11TVxJA8sfxxybrsEQ72aRYTMVc7Nk2/view?usp=sharing

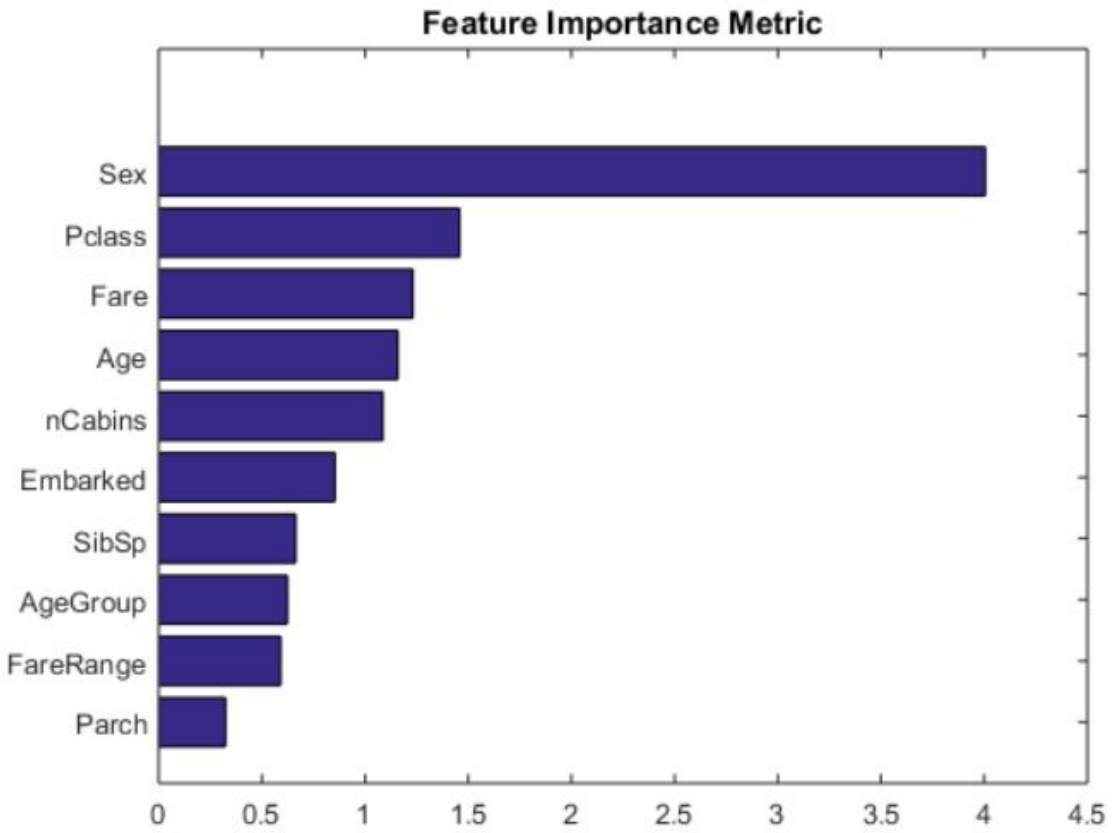
▣ 사전 학습

- 상관관계: 변수간의 관계의 정도와 방향을 하나의 수치로 요약해 표시해 주는 지표



▣ 사전 학습

- **Feature Importance**: 의사 결정 나무에서 특정 **feature**가 트리를 분할하는데 얼마나 기여를 했는지에 대한 중요도
- 아래와 같은 결과가 도출되었다면 성별이 가장 중요한 요인이라고 판단할 수 있음



▣ 전처리

- 분석할 부품 코드 선택
  - ‘prod_no‘ 변수가 ‘90784-76001’인 부품만 선택

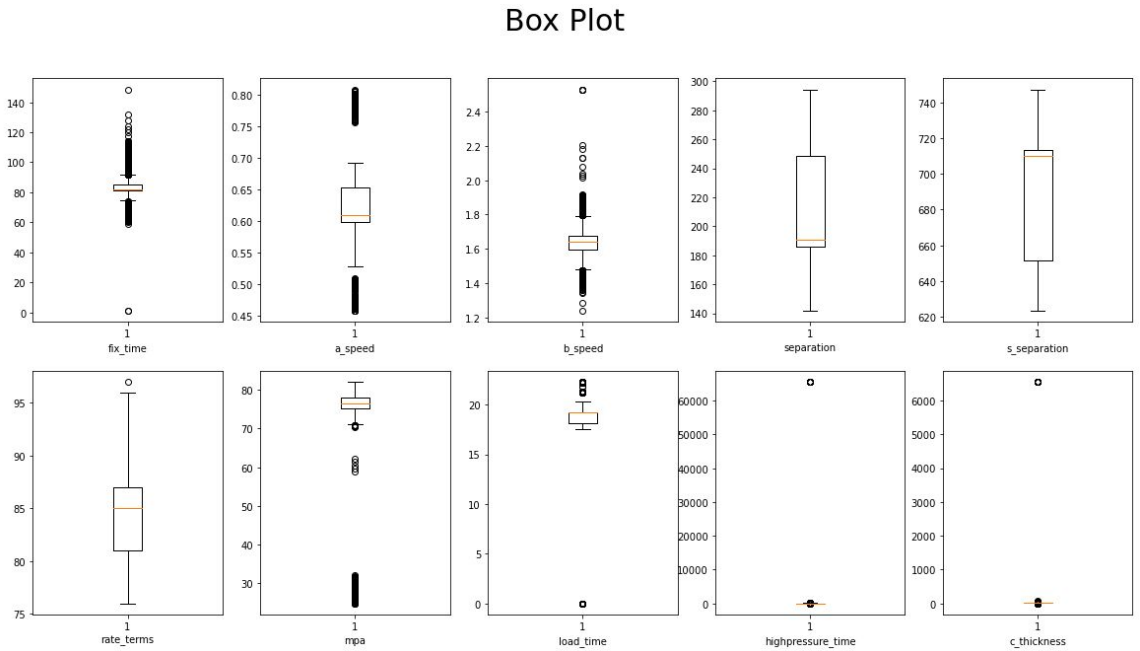
- 결측치 확인
  - 결측치는 없는 것으로 파악됨

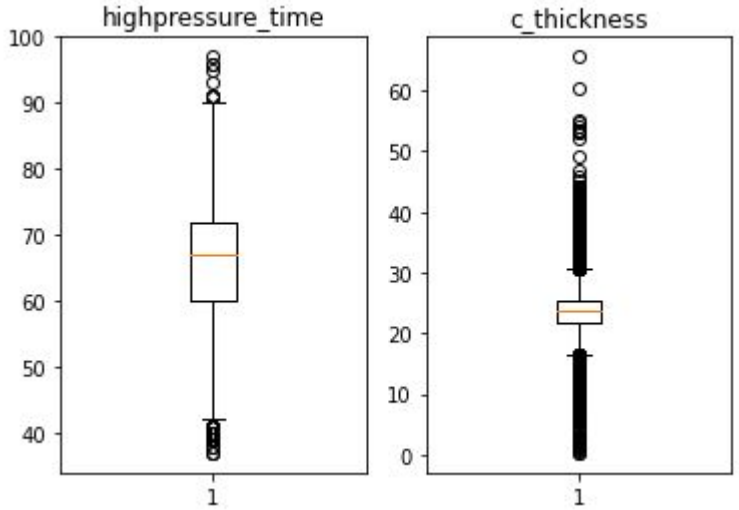
- 이상치 확인
  - 기초 통계량에서 max값과 4분위수를 살펴보면 highpressure_time, c_thickness 변수에서 극단적인 이상치가 존재하는 것으로 판단됨
  - 시각화를 통해 확인이 필요함

	fix_time	a_speed	b_speed	separation	s_separation	rate_terms	mpa	load_time	highpressure_time	c_thickness
count	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000
mean	83.141287	0.618894	1.643940	214.502112	685.851453	84.532715	74.209964	18.679903	96.361954	27.438216
std	3.206389	0.048995	0.081503	33.953957	32.910181	3.822717	10.918193	1.703842	1402.568755	153.279417
min	1.000000	0.457000	1.240000	141.600000	623.300000	76.000000	24.800000	0.000000	37.000000	0.300000
25%	81.000000	0.598000	1.597000	185.900000	651.600000	81.000000	75.300000	18.100000	60.000000	21.800000
50%	82.100000	0.609000	1.640000	190.700000	710.300000	85.000000	76.600000	19.200000	67.000000	23.800000
75%	85.400000	0.652000	1.676000	248.700000	713.600000	87.000000	78.100000	19.200000	72.000000	25.400000
max	148.600000	0.808000	2.528000	294.500000	747.300000	97.000000	82.100000	22.300000	65534.000000	6553.400000

▣ 시각화

- Box Plot
  - highpressure_time, c_thickness 변수에서 극단적인 이상치가 존재하는 것을 확인
  - 극단적 이상치 제거 후 박스 모양이 온전하게 나타나는 것을 확인 가능

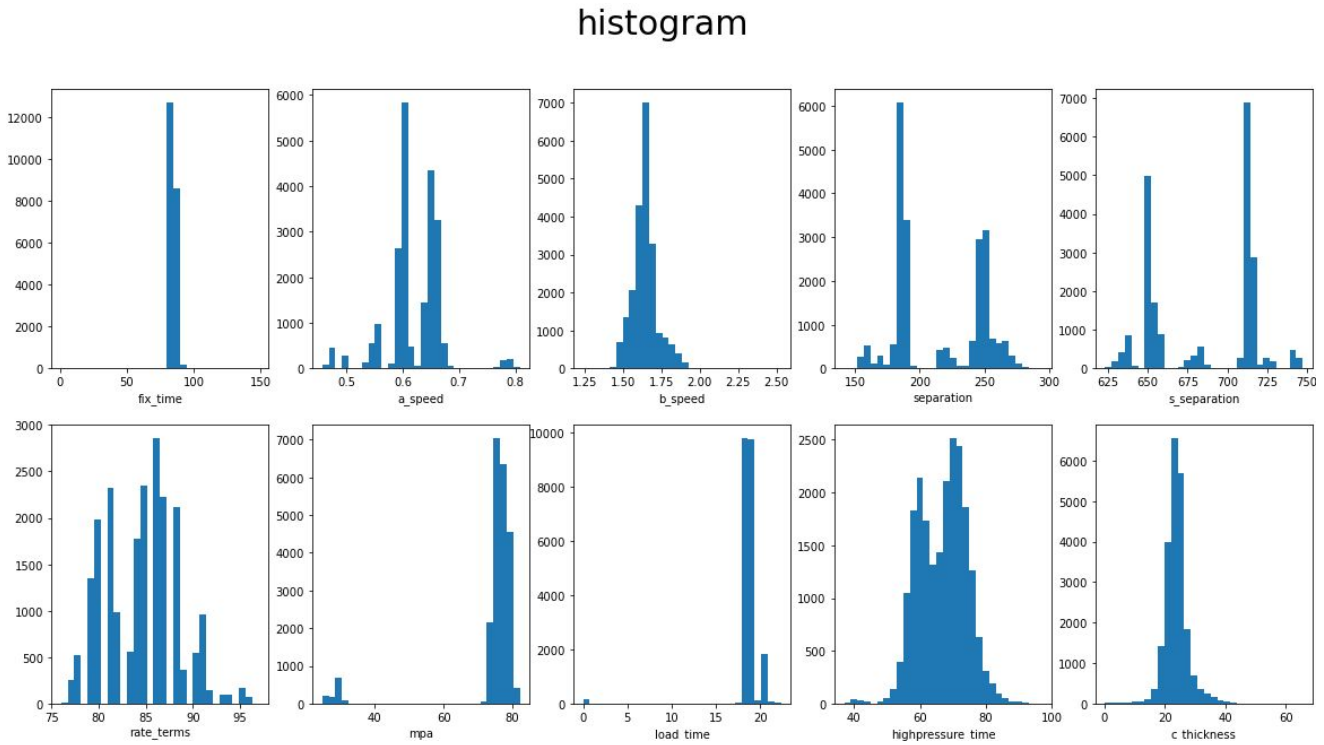




▣ 시각화

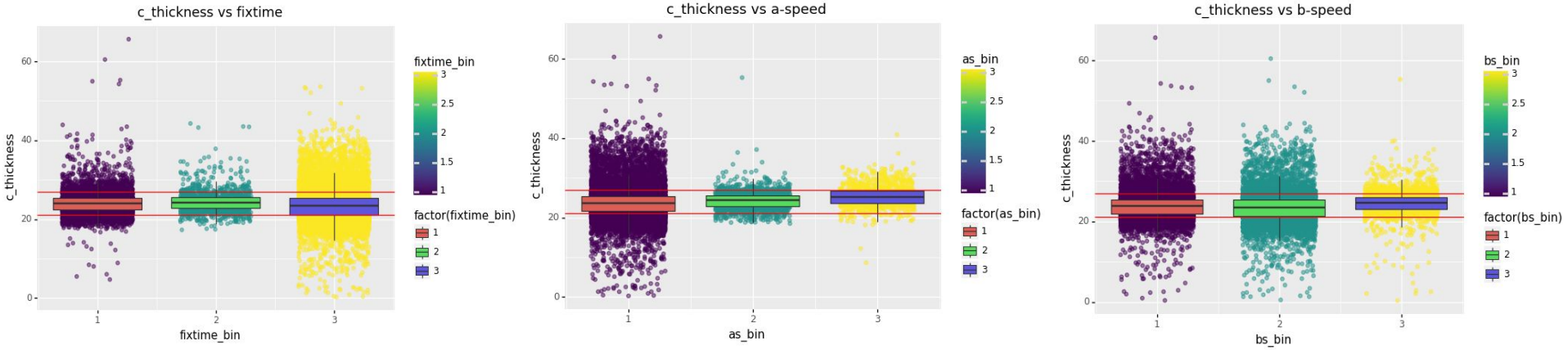
- Histogram

  - 히스토그램을 보면 양봉 분포를 가진 변수들이 많음을 확인 가능



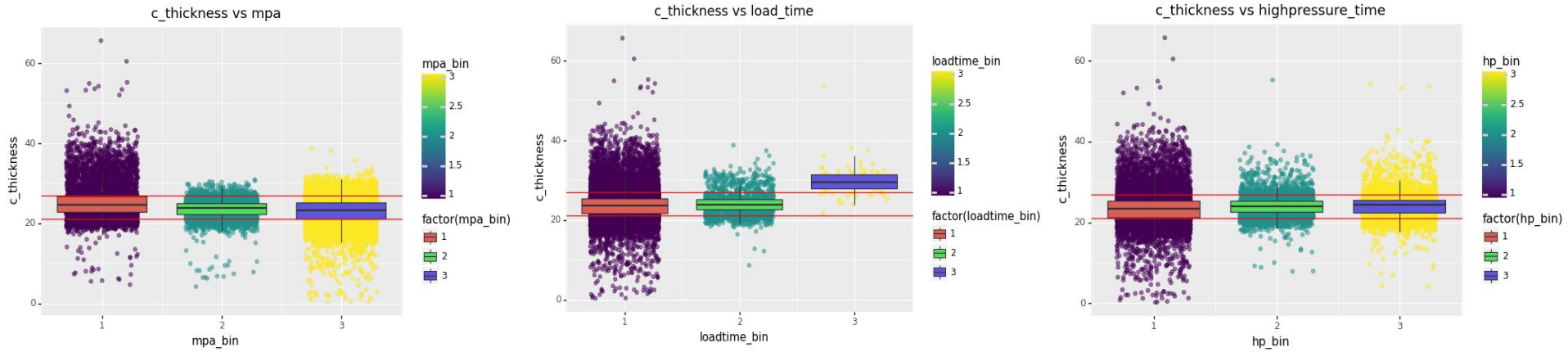
▣ 시각화

- Box Plot(interval)
  - 각 독립변수들을 구간으로 나눠 종속변수와의 관계 분포를 Box Plot으로 확인
  - as_bin의 경우, 다른 변수들에 비해 범주간 분포에서 차이 발생



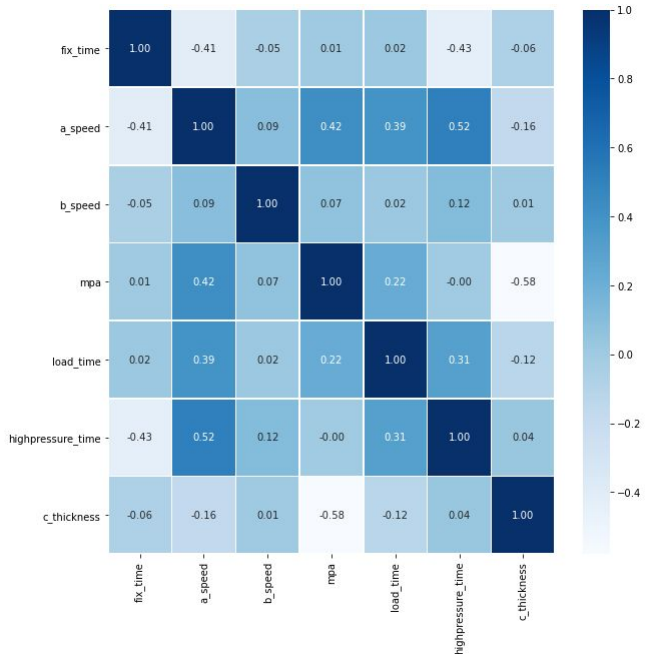
▣ 시각화

- Box Plot(interval)
  - 각 독립변수들을 구간으로 나눠 종속변수와의 관계 분포를 Box Plot으로 확인
  - load_time 변수가 다른 변수들에 비해 범주간 분포에서 차이 발생



▣ 시각화

- 상관관계
  - a_speed는 mpa, load_time, highpressure_time과 상관관계가 다소 있는 것으로 파악됨
  - a_speed와 fix_time 간 음의 상관관계를 보임
  - mpa와 load_time 간 약한 상관관계를 보임
  - mpa와 c_thickness 간 음의 상관관계를 보임
  - load_time과 highpressure_time과 약한 상관관계를 보임



▣ 분석 과정 (1/3)

- Classification Problem
  - 회귀 문제를 분류 문제로 변환하기 위해 종속 변수의 일정 값을 기준으로 이진화
  - 종속 변수인 c_thickness의 값이 32초과, 20미만 인 경우 불량으로 간주
  - 정상 범주가 18,921개, 불량 범주가 2,836개

	fix_time	a_speed	b_speed	mpa	load_time	highpressure_time	c_thickness	failure
0	85.5	0.611	1.715	78.2	18.1	58	24.7	0
1	86.2	0.606	1.708	77.9	18.2	58	22.5	0
2	86.0	0.609	1.715	78.0	18.1	82	24.1	0
3	86.1	0.610	1.718	78.2	18.1	74	25.1	0
4	86.1	0.603	1.704	77.9	18.2	56	24.5	0
...	...	...	...	...	...	...	...	...
34134	82.7	0.594	1.578	75.9	20.2	68	19.9	1
34135	82.7	0.597	1.577	75.9	20.2	72	21.3	0
34136	82.5	0.591	1.581	75.9	20.2	72	19.8	1
34137	82.5	0.588	1.584	75.9	20.2	69	20.1	0
34138	82.5	0.597	1.576	75.9	20.2	69	21.1	0

```
class_car['failure'].value_counts()
```

```
0    18921
1     2836
Name: failure, dtype: int64
```

▣ 분석 과정 (2/3)

- Decision Tree – 모델 학습 및 테스트
  - train set, test set을 각각 8:2의 비율로 학습 및 테스트
  - 불순도 측정 지표를 entropy, 최대 깊이를 3으로 설정
  - Accuracy를 평가 지표로 활용

```
tree = tree.DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=2022)
tree.fit(class_train_x, class_train_y)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=2022)
```

```
y_pred_tr = tree.predict(class_test_x)
print('Accuracy: %.2f' % accuracy_score(class_test_y, y_pred_tr))
```

```
Accuracy: 0.89
```

▣ 분석 과정 (3/3)

- Random Forest – 모델 학습 및 테스트
  - train set, test set을 각각 8:2의 비율로 학습 및 테스트
  - n_estimators : 결정트리의 개수를 지정(시간과 성능 간 trade-off)
  - max_features : 최적의 분할을 위해 고려할 최대 feature 개수
  - oob_score : 학습 시 부트스트랩 샘플링에서 선택되지 않은 샘플들을 기반으로 테스트 진행
  - Accuracy를 평가 지표로 활용

```
rf = RandomForestClassifier(n_estimators=600, oob_score=True, random_state=2022, max_features=6)
rf.fit(class_train_x, class_train_y)
```

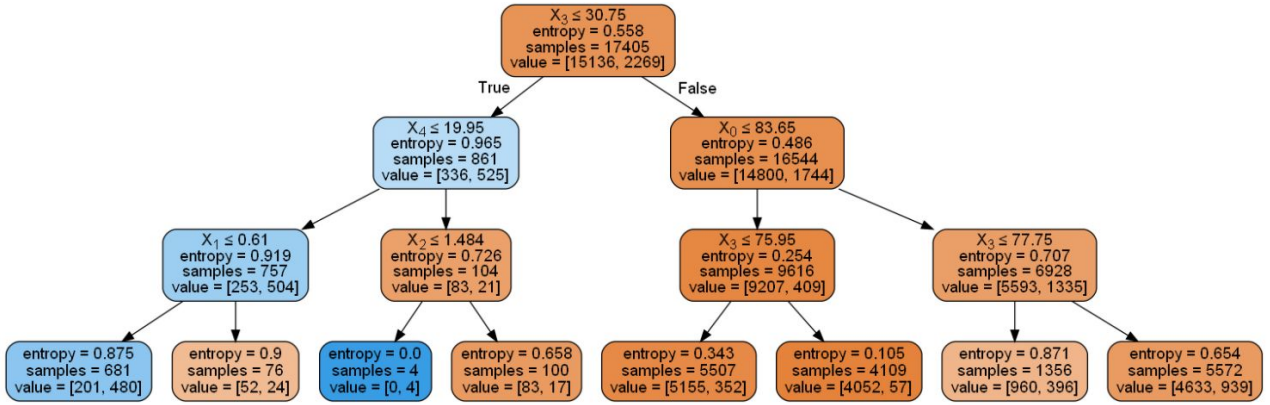
```
RandomForestClassifier(max_features=6, n_estimators=600, oob_score=True,
                       random_state=2022)
```

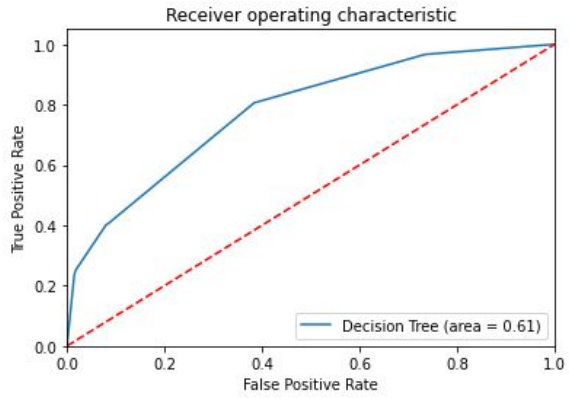
```
rf_predict = rf.predict(class_test_x)
accuracy = accuracy_score(class_test_y, rf_predict)

print(f'Out-of-bag score estimate: {rf.oob_score_:.3}')
print(f'Mean accuracy score: {accuracy:.3}')
```

```
Out-of-bag score estimate: 0.885
Mean accuracy score: 0.888
```

▣ 결과 분석 (1/2)

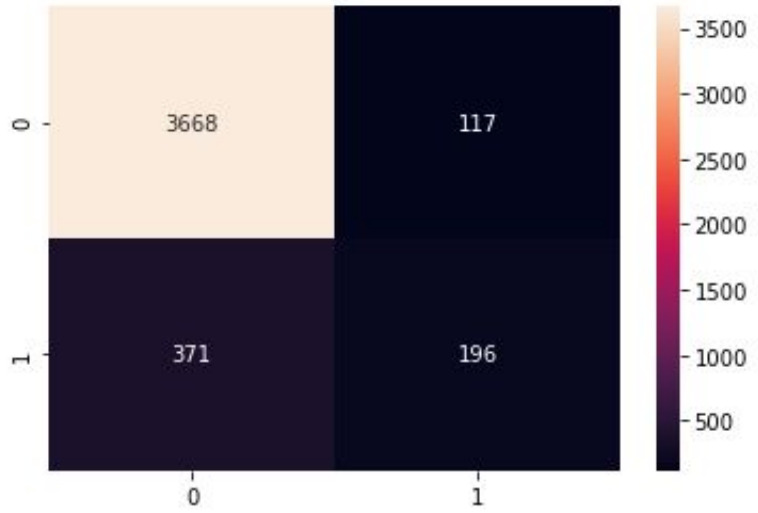
- Decision Tree
  - Accuracy : 0.89 (Logistic Regression의 결과보다 약간 개선)
  - AUC : 0.61 (Logistic Regression의 결과보다 약간 개선)
  - Tree Plot을 통해 마지막 노드를 살펴보면 몇몇 노드에서 entropy 지수가 상당히 큰 것을 확인 가능
    => 분기가 완벽히 이루어지지 않음
  - max_depth를 3보다 크게 잡으면 불순도가 조금 더 낮아질 것으로 예상됨

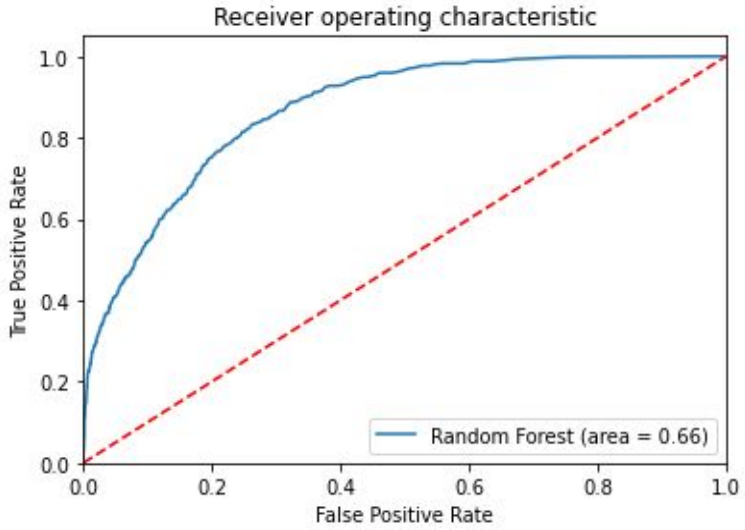




▣ 결과 분석 (2/2)

- Random Forest
  - Accuracy : 0.888 (Decision Tree의 결과와 비슷함)
  - Confusion Matrix : FN의 비율이 줄고 FP의 비율이 증가함
  - AUC : 0.66 (Decision Tree의 결과보다 크게 개선)





▣ 분석 개요

- 생산 기계 데이터를 통해 기계의 고장 예측
  - 목표1 : 분류 모델을 통해 고장을 예측하는 것
  - 목표2 : 고장에 가장 큰 영향을 미치는 지표를 제시하는 것

- Dataset
  - 58개의 지표가 있지만 이름을 모두 알지 못함, 총 900,000개의 데이터 존재
  - 타겟은 0(정상),1(고장) 로 이루어진 분류 문제
  - https://drive.google.com/file/d/1sJzU2RJR76TRcLPOX6EAIOfX_AHY1J0J/view?usp=sharing
  - https://drive.google.com/file/d/1xrZXp5RO9s8sjffk-LszQ4bKZOUA_pgy/view?usp=sharing

- Consideration
  - 결측치가 거의 모든 행마다 존재하므로 적절한 처리 필요
  - 독립변수가 58개로 많기 때문에 적절한 변수를 선택하여 모델링하는 방안 고려

https://www.kaggle.com/ivanloginov/the-broken-machine

▣ 전처리(1/2)

- 결측치 처리
  - 대부분의 행이 결측치를 포함하고 있음
  - 결측치를 앞이나 뒤의 값으로 간단히 대체
  - 평균이나 그룹 별 대표값으로 대체하는 방안도 존재

```
print('Initial size: {}'.format(X.shape))
print('After NaN omit size: {}'.format(X.dropna().shape))
```

```
Initial size: (900000, 58)
After NaN omit size: (2462, 58)
```

	1	2	3	4	5	6	7	8	9	10	...
0	-42.822536	NaN	12.0	NaN	1.0	2.0	24.0	-45.025510	NaN	1.0	...
1	-13.478816	13.0	12.0	75.132502	0.0	2.0	24.0	-49.213545	7.0	0.0	...
2	51.702721	13.0	12.0	63.459270	0.0	3.0	24.0	-58.777043	8.0	0.0	...
3	NaN	12.0	13.0	-15.492561	1.0	1.0	23.0	0.624258	9.0	0.0	...
4	7.633273	NaN	13.0	59.862681	0.0	3.0	NaN	-61.395319	NaN	0.0	...

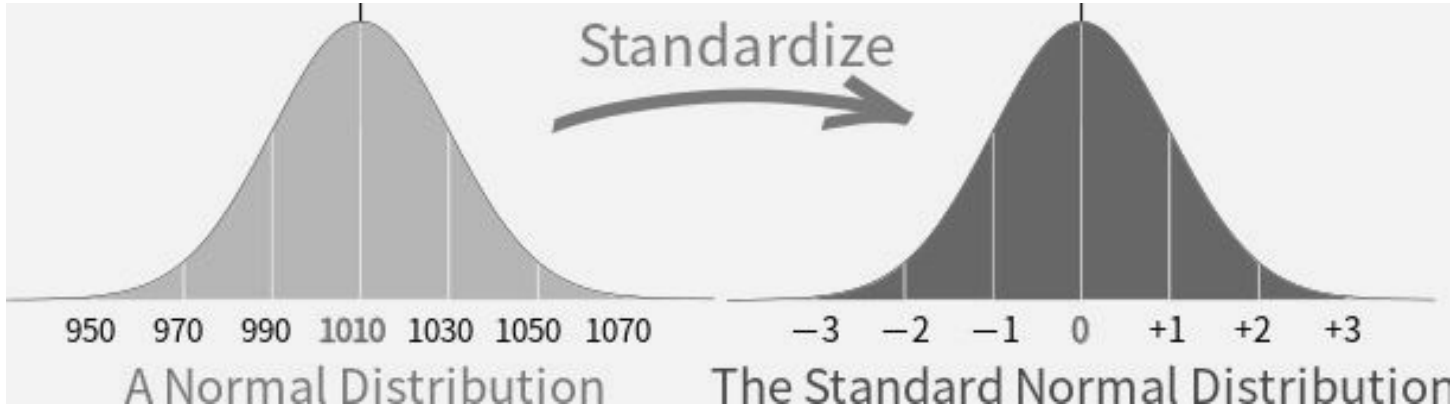
< 결측치 처리 전 >

	1	2	3	4	5	6	7	8	9	10	...
0	-42.822536	13.0	12.0	75.132502	1.0	2.0	24.0	-45.025510	7.0	1.0	...
1	-13.478816	13.0	12.0	75.132502	0.0	2.0	24.0	-49.213545	7.0	0.0	...
2	51.702721	13.0	12.0	63.459270	0.0	3.0	24.0	-58.777043	8.0	0.0	...
3	7.633273	12.0	13.0	-15.492561	1.0	1.0	23.0	0.624258	9.0	0.0	...
4	7.633273	13.0	13.0	59.862681	0.0	3.0	23.0	-61.395319	7.0	0.0	...

< 결측치 처리 후 >

▣ 전처리(2/2)

- 표준화
  - 평균을 제거하고 데이터를 단위 분산으로 조정
  - 모든 변수가 수치형이기 때문에 표준화를 통해 이상치 및 변수간 분포 차이를 줄여 안정적인 학습이 가능하게 함



▣ 분석 과정 (1/5)

- Random Forest – 모델 학습 및 테스트
  - train set, test set을 각각 8:2의 비율로 학습 및 테스트
  - n_estimators : 결정트리의 개수를 지정(시간과 성능 간 trade-off)
  - criterion : 불순도 지표로 gini 계수 사용
  - max_depth : 최대 분기 개수
  - oob_score : 학습 시 부트스트랩 샘플링에서 선택되지 않은 샘플들을 기반으로 테스트 진행
  - Accuracy를 평가 지표로 활용

```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators=20, oob_score=True,
                             criterion='gini', max_depth=5, random_state=42)
rfc.fit(X_train, y_train)
```

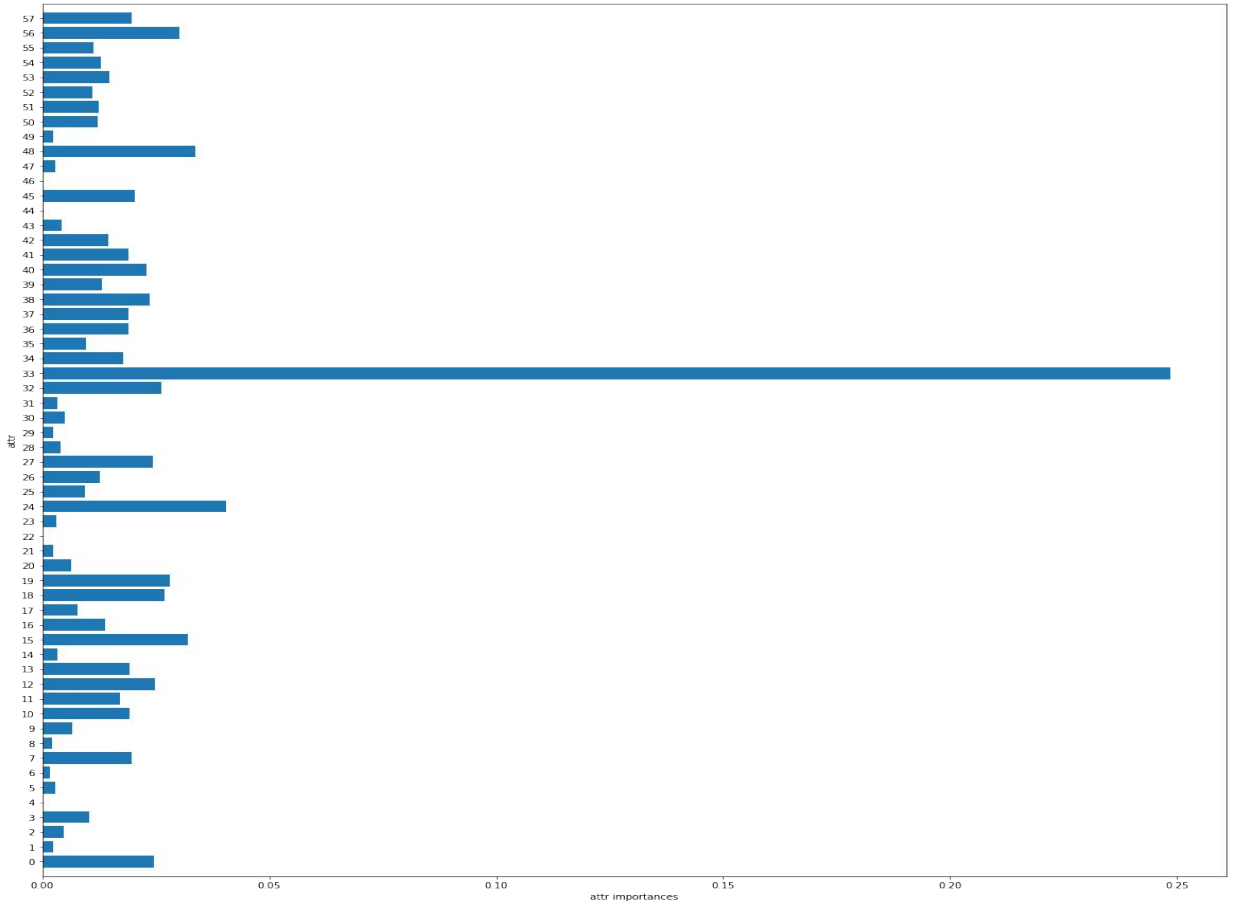
```
RandomForestClassifier(max_depth=5, n_estimators=20, oob_score=True,
                       random_state=42)
```

```
print(rfc.score(X_test,y_test))
```

```
0.6946666666666667
```

▣ 분석 과정 (2/5)

- Random Forest – Feature Importance
  - 결정에 각 특성이 얼마나 중요한 영향을 미쳤는지 평가하는 것 (원래 변수 이름 = index + 1)



▣ 분석 과정 (3/5)

- Feature Selection
  - 변수가 많기 때문에 feature importance의 상위 20개 변수만 사용

- Random Forest – 모델 학습 및 테스트(Reduced Feature)
  - 변수의 개수를 제한했음에도 성능에 변화가 없음

```
X_train, X_test, y_train, y_test = train_test_split(new_X, y, test_size=0.2, random_state=0)

rfc = RandomForestClassifier(n_estimators=20, oob_score=True, \
                             criterion='gini', max_depth=5, random_state=42)
rfc.fit(X_train, y_train)
```

```
RandomForestClassifier(max_depth=5, n_estimators=20, oob_score=True,
                       random_state=42)
```

```
print(rfc.score(X_test,y_test))
```

```
0.6946666666666667
```

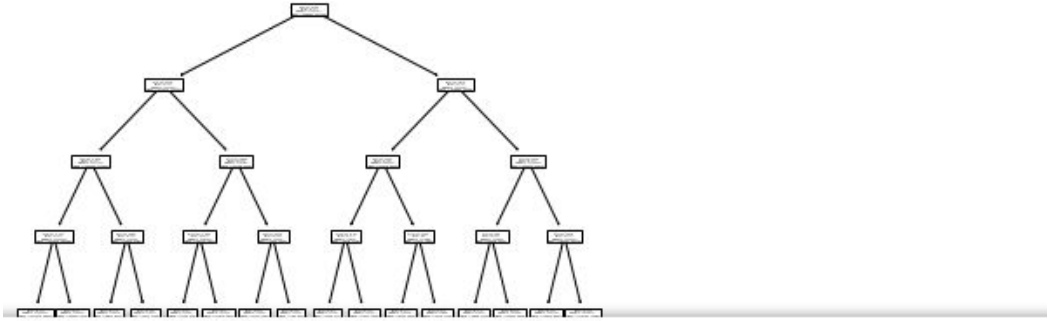
▣ 분석 과정 (4/5)

- Random Forest – Feature Importance(Reduced Feature)
  - 34번 변수의 중요도가 매우 큰 것으로 확인됨 (원래 변수 이름 = index + 1)

▣ 분석 과정 (5/5)

- Decision Tree – 모델 학습 및 테스트
  - criterion : 불순도 지표로 gini 계수 사용
  - max_depth : 최대 분기 개수

```
from sklearn import tree
clf_tree = DecisionTreeClassifier(criterion = 'gini', max_depth=4, random_state=0)
tree.plot_tree(clf_tree.fit(X_train, y_train))
```



```
train_acc = clf_tree.score(X_train,y_train)
test_acc = clf_tree.score(X_test,y_test)

print(f'Train_Accuracy: {train_acc}')
print(f'Test_Accuracy: {test_acc}')
```

```
Train_Accuracy: 0.6934444444444444
Test_Accuracy: 0.6951833333333334
```

▣ 결과 분석

- Decision Tree – Feature Importance
  - 34, 42 순으로 중요한 변수로 판단됨
  - max depth를 4로 지정했을 때 34변수와 42변수로 분기가 모두 이루어짐
  - 마지막 노드들의 gini 계수가 높은 것으로 보아 max depth를 더 늘려도 될 것으로 판단됨