Productivity Prediction of Garment Employees

▣ 분석 개요

- 의류 제품에 대한 거대한 글로벌 수요를 만족시키는 것은 주로 직원의 생산성에 달려 있음
  - 의류 산업은 수동 프로세스가 많은 노동 집약적인 산업
  - 목표 : 직원의 생산성을 예측

- Dataset
  - 의류 제조 공정의 중요한 특성과 업계 전문가가 검증한 직원의 생산성이 포함
  - 총 1,197개의 데이터와 15개의 변수 존재
  - https://drive.google.com/file/d/1MeLweI3qBAuIJiomVswlVybehL_BjTOU/view?usp=sharing

- Consideration
  - 서로 연관된 변수가 많아 보이므로 변수 간의 상관관계를 고려하여 적절한 변수 선택
  - 팀 또는 부서마다 생산성이 다를 것으로 예상됨으로 Cluster를 나누어 분석 진행

https://www.kaggle.com/ishadss/productivity-prediction-of-garment-employees

▣ 분석 개요

- Variables Description

Name	Description
date	날짜 (MM-DD-YYYY)
day	요일
quarter	한 달 등분 (Quarter1 ~ Quarter5)
department	소속 부서 (finishing, sewing)
team_no	팀 번호 (1 ~ 12)
no_of_workers	노동자 수
no_of_style_change	특정 제품의 스타일 변경 수
targeted_productivity	매일 각 팀에 할당된 목표 생산성
smv	Standard Minute Value(작업에 할당된 표준 시간)
wip	Work in progress
over_time	각 팀의 초과 근무 시간(분)
incentive	동기를 부여하는 재정적 인센티브(BDT)의 양
idle_time	여러 가지 이유로 생산이 중단된 기간
idle_men	생산 중단으로 인해 쉬고 있는 노동자의 수
actual_productivity	실제 생산성 비율

▣ 주요 활용 알고리즘 설명

- Linear Regression

- 회귀 알고리즘은 데이터가 주어졌을 때 데이터를 잘 설명하는 선을 찾고자 한다.

- 종속 변수 y와 한 개 이상의 독립 변수 (또는 설명 변수) X와의 선형 상관 관계를 모델링하는 회귀분석 기법이다.

- 한 개의 설명 변수에 기반한 경우에는 단순 선형 회귀(simple linear regression), 둘 이상의 설명 변수에 기반한 경우에는 다중 선형 회귀라 칭한다.

- 다중 선형 회귀는 주어진 데이터 집합에 대해, 종속 변수와 p개의 설명 변수 사이의 선형 관계를 모델링한다.

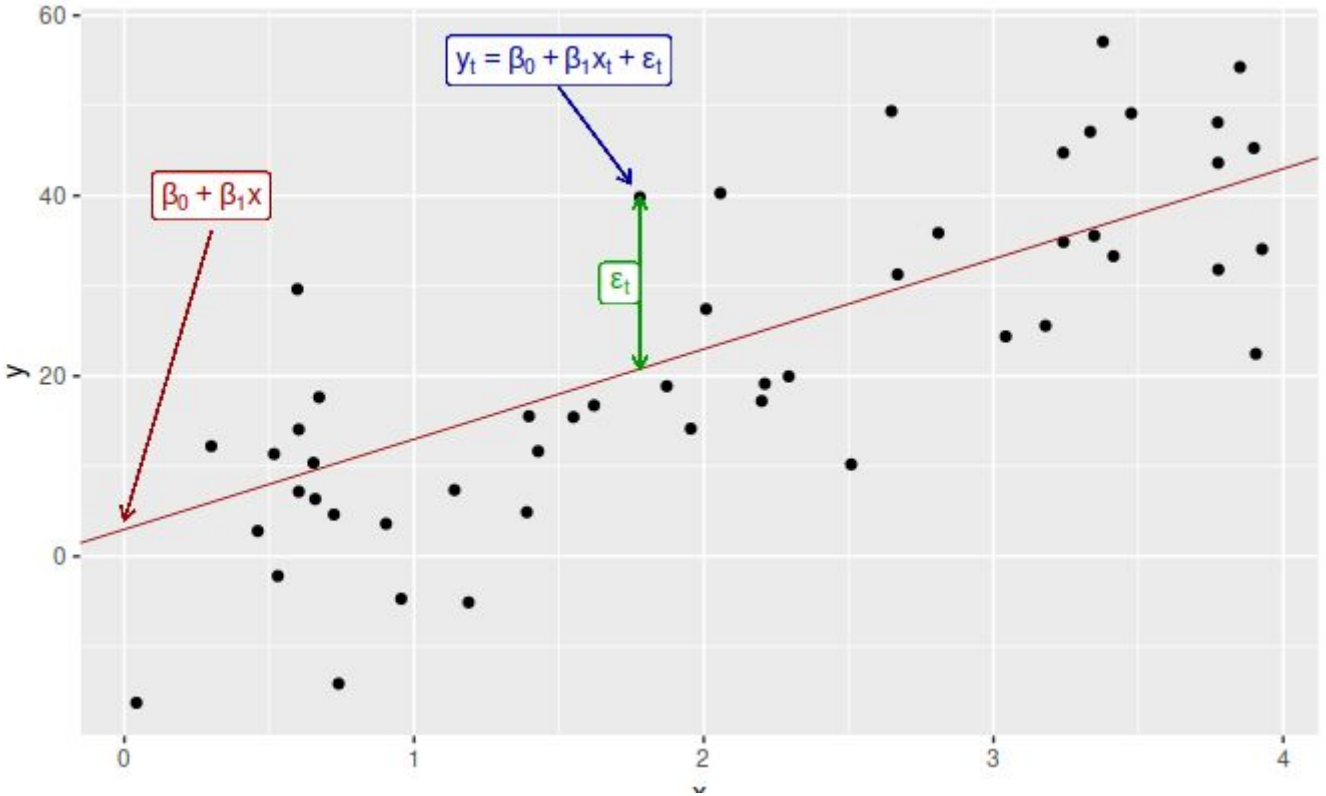
$$y_i = \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\mathrm{T}} \boldsymbol{\beta} + \varepsilon_i, \qquad i = 1, \ldots, n,$$

- 예측 변수에 대한 모든 값이 주어졌을 때, 수립한 선형 회귀 모델을 사용해 예측 변수 x가 응답 변수 y에 미치는 영향을 확인할 수 있다.

- β는 x가 한 단위 변했을 때, y의 기대 변화량을 의미한다.

▣ 주요 활용 알고리즘 설명

- Linear Regression



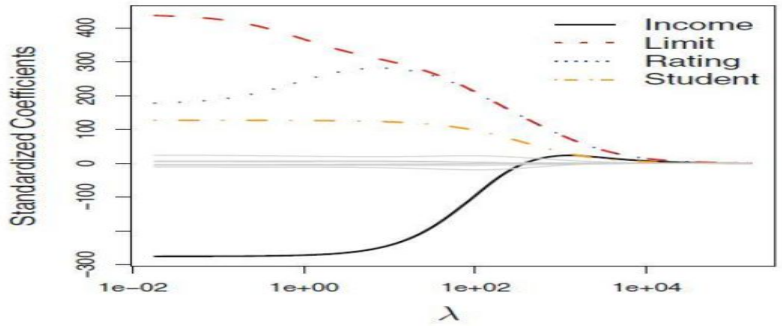
$$y_i = \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\mathrm{T}} \boldsymbol{\beta} + \varepsilon_i, \qquad i = 1, \ldots, n,$$

주요 활용 알고리즘 설명

Ridge Regression

- 기본 회귀 알고리즘을 사용하다 보면 Ovefitting이 발생할 수 있다. Overfitting된 경우 데이터에 매우 적합 되어 극단적인 그래프가 생성되며 선형 회귀의 계수가 매우 크게 나타난다.
- 이렇게 Variance 가 큰 상황을 막기 위해 계수 자체가 크면 페널티를 주는 수식을 추가 한 것이 Ridge Regression이다.
- 즉 오차를 최소화하는 함수에 페널티를 줌으로써 보다 부드럽게 계수를 선택하는 차이가 있다.
- λ가 크면 계수를 많이 줄이고 λ가 작으면 최소 제곱 법 문제를 풀게 된다. λ가커질수록 계수의 크기가 Shrink 되는 효과가 생기는 걸 볼 수 있다.

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2=\mathrm{RSS}+\lambda\sum_{j=1}^{p}\beta_j^2$$

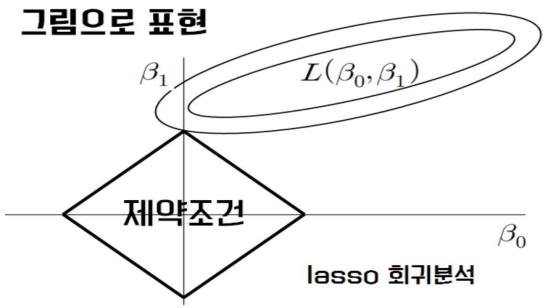


▣ 주요 활용 알고리즘 설명

- Lasso Regression

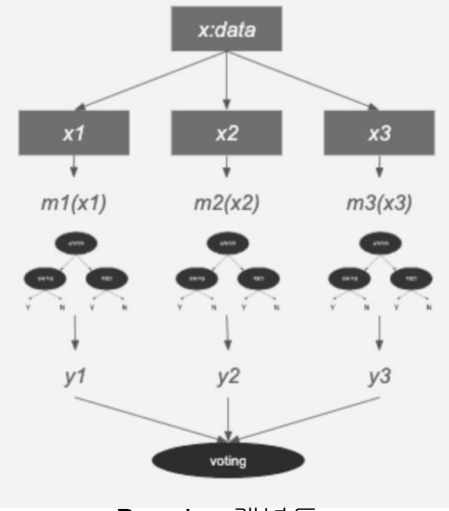
  - Lasso Regression의 경우 Ridge Regression과 비슷하게 생겼지만 페널티 항에 절대값의 합을 주는 방식으로 진행된다.
  - Lasso Regression은 몇몇 유의미하지 않은 변수들에 대해 계수를 0에 가깝게 추정하여 변수 선택의 효과를 가지게 한다.
  - 파라미터 크기에 관계없이 같은 수준의 Regularization을 적용하기 때문에 작은 값의 파라미터를 0으로 만들어 해당 변수를 모델에서 삭제하고 따라서 모델을 단순하게 만들어주고 해석에 용이하게 만들어준다.

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j| = \mathrm{RSS}+\lambda\sum_{j=1}^{p}|\beta_j|.$$

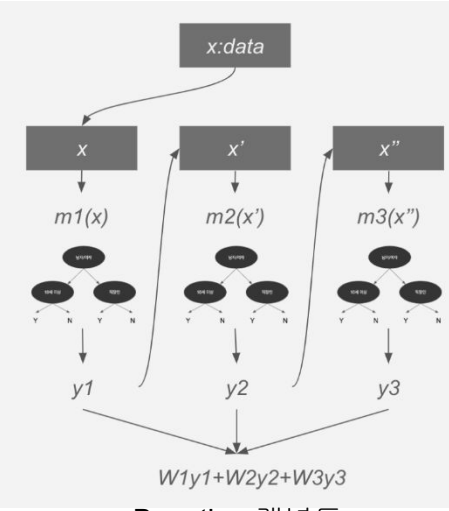


▣ 주요 활용 알고리즘 설명

- XGBoost
  - Bagging 방식의 Random Forest와 달리 여러 개의 의사결정나무를 Boosting 방식을 사용하여 ensemble한 모델이다.
  - 예를 들어, 첫 번째 의사결정 나무를 통해 예측을 수행하고 이것의 오차 값을 다음 모델에 가중치를 반영하여 모델을 개선해 나가는 방식이다.
  - GradientBoost(GBM) 알고리즘과 같은 방식이나, 병렬 학습이 지원되어 더 빠른 학습이 가능하고 과적합 방지를 위한 규제가 포함되어 있다.



<Bagging 개념도>

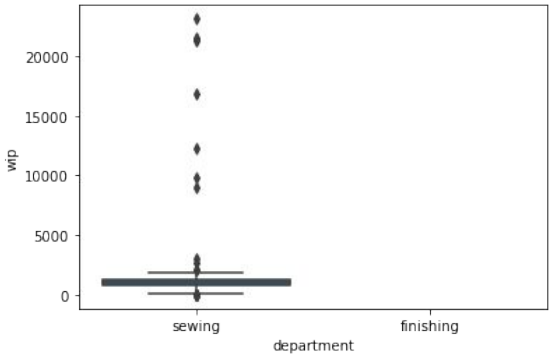


<Boosting 개념도>

▣ 전처리(1/2)

- 형변환
  - int형인 team변수를 object형으로 변환

- 결측치 확인
  - wip 변수에만 506건의 결측치 존재
  - 모든 결측치는 마무리 부서에 속함. 마무리 부서는 재봉 부서에서 작업을 이어 받아야 하기에 재봉부의 작업이 끝날 때 까지 마무리 부서는 대기하여야 함. 따라서 null 값을 0으로 대체 가능



<부서에 따른 결측치 파악>

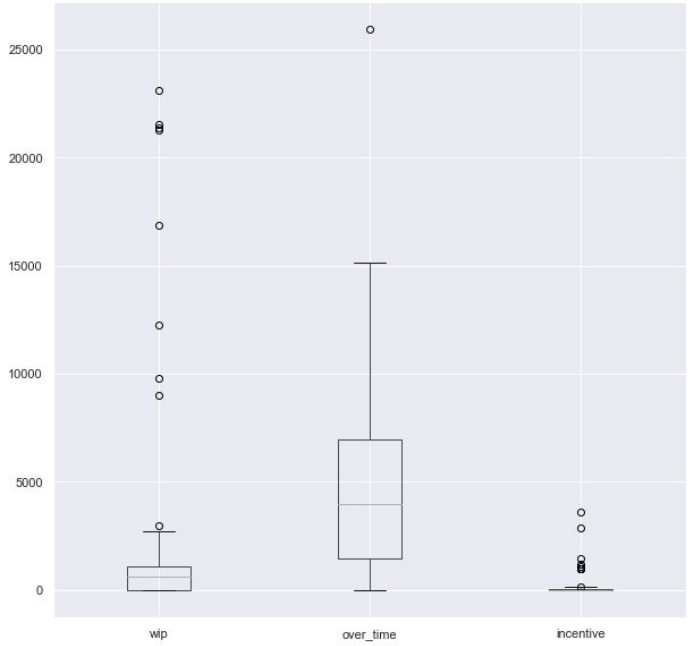
▣ 전처리(2/2)

- 더미 변수 생성
  - 회귀 모델은 설명변수가 연속형 변수여야 사용할 수 있기 때문에 범주형 변수를 더미변수로 변환
  - 범주형 변수인 day, department, quarter, team 변수에 대해 더미변수화 진행
  - 더미변수는 해당 더미에 속하면 1 아니면 0의 값을 가짐
  - 더미변수는 원래의 범주 개수보다 1개 적게 생성
    - 원래 변수가 성별(남,여) 이라면 남성여부 또는 여성여부 둘 중 하나만 만들면 두 범주 모두 표현 가능

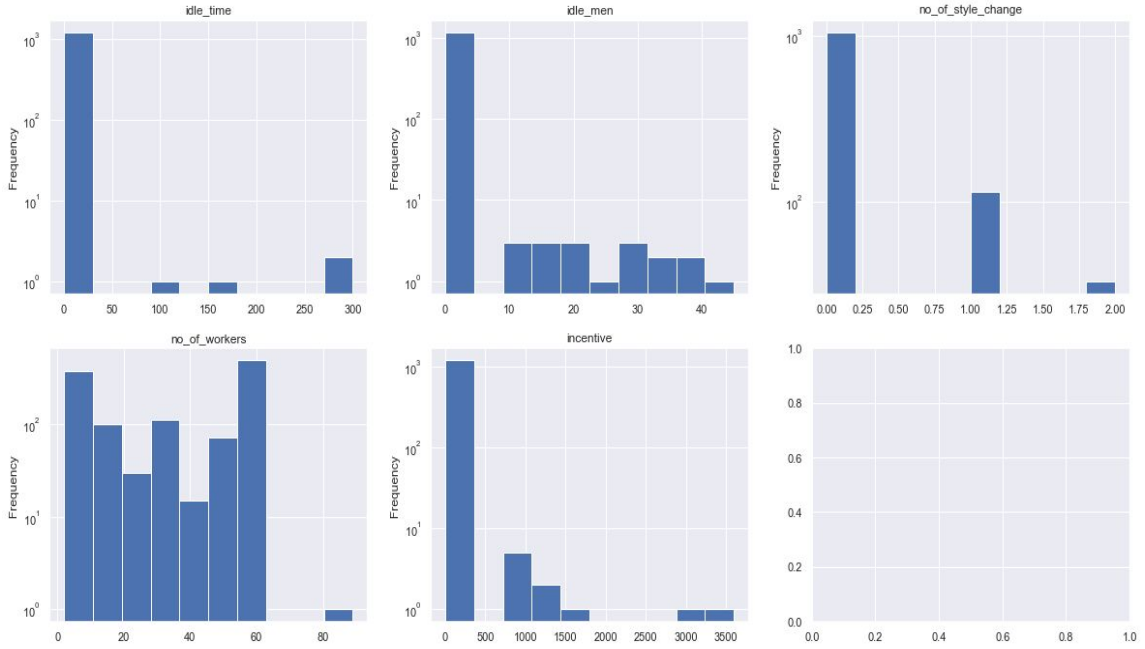
남주인공 역 연예인의 본업	시간대
가수	오후
배우	오후
개그맨	오전
배우	오전
가수	오후

남주인공 _가수	남주인공 _개그맨	시간대_오전
1	0	0
0	0	0
0	1	1
0	0	1
1	0	0

▣ EDA(1/6)

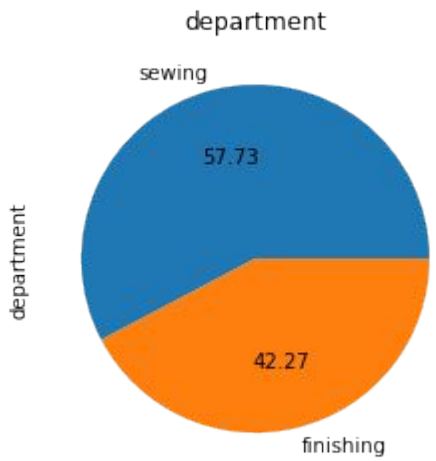


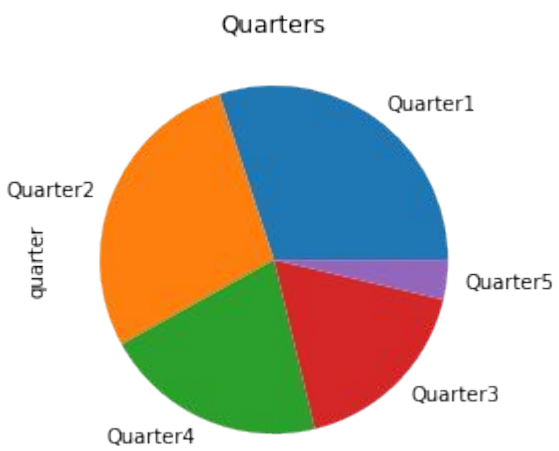
<BoxPlot>

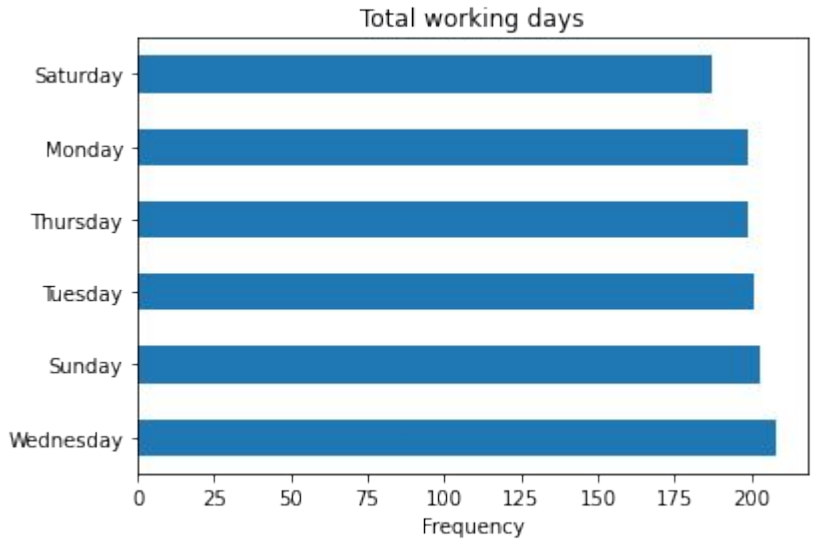


<Histogram(log scale)>

EDA(2/6)

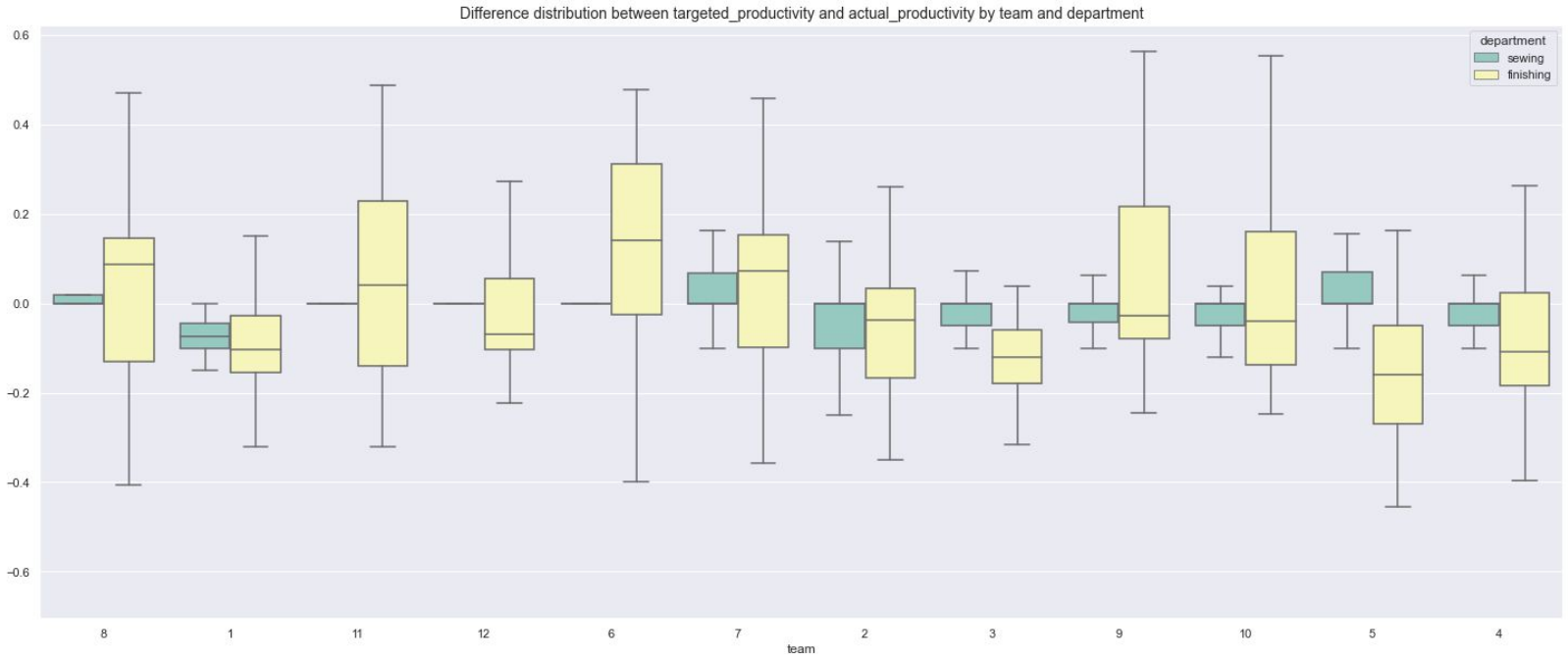

department
sewing
57.73
42.27
finishing
department


Quarters
Quarter1
Quarter2
Quarter5
Quarter3
Quarter4
quarter


Total working days
Saturday
Monday
Thursday
Tuesday
Sunday
Wednesday
0
25
50
75
100
125
150
175
200
Frequency

▣ EDA(3/6)

- Actual Productivity vs Targeted Productivity
  - 팀, 부서별로 차이가 존재하는 모습
  - 마무리 부서의 경우 목표 생산량과 실제 생산량 간의 차이가 큰 편

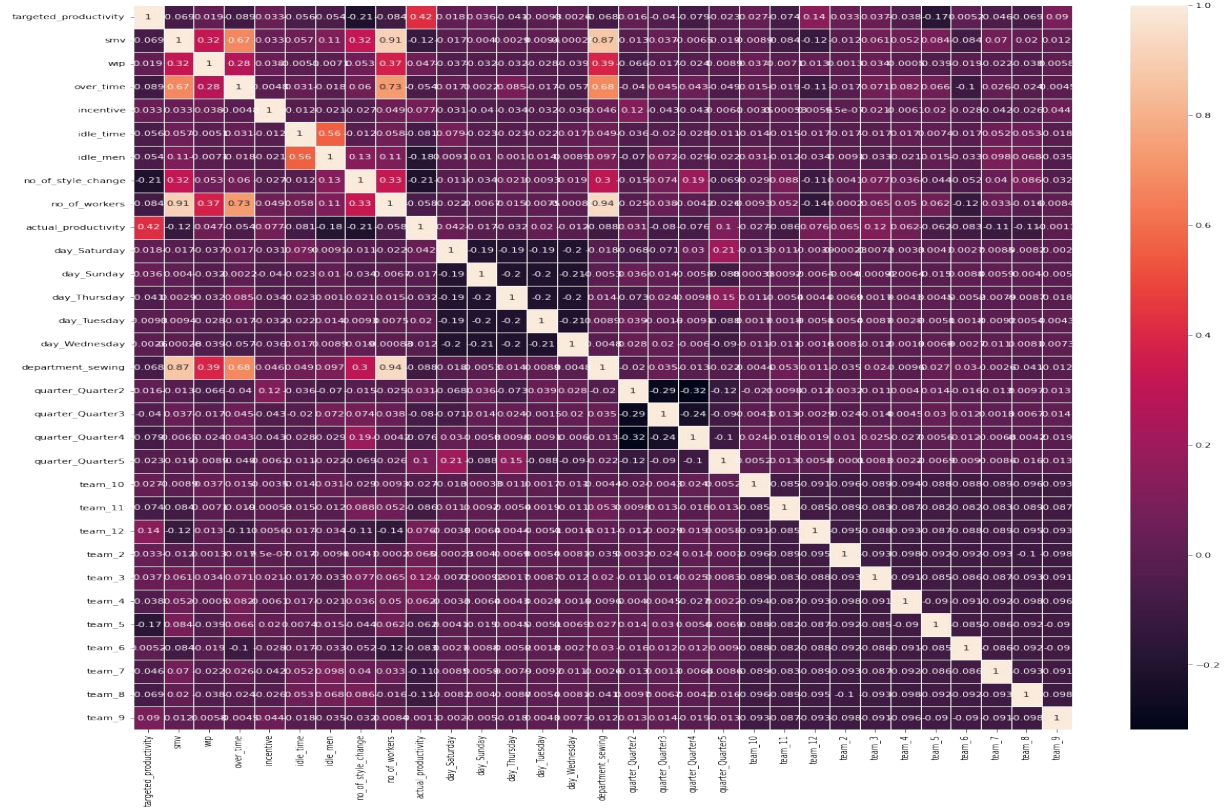


▣ EDA(4/6)

- 상관관계
  - no_of_workers와 smv간 상관관계가 매우 높음(0.91)
    - 작업에 할당된 시간이 많을 수록 근무자가 많이 투입
  - no_of_workers와 over_time간 상관관계가 높음(0.73)
    - smv와 over_time간 상관관계로 인함
  - smv와 over_time간 상관관계가 높음(0.67)
    - 작업에 할당된 시간이 많을 수록 초과근무 발생

EDA(5/6)

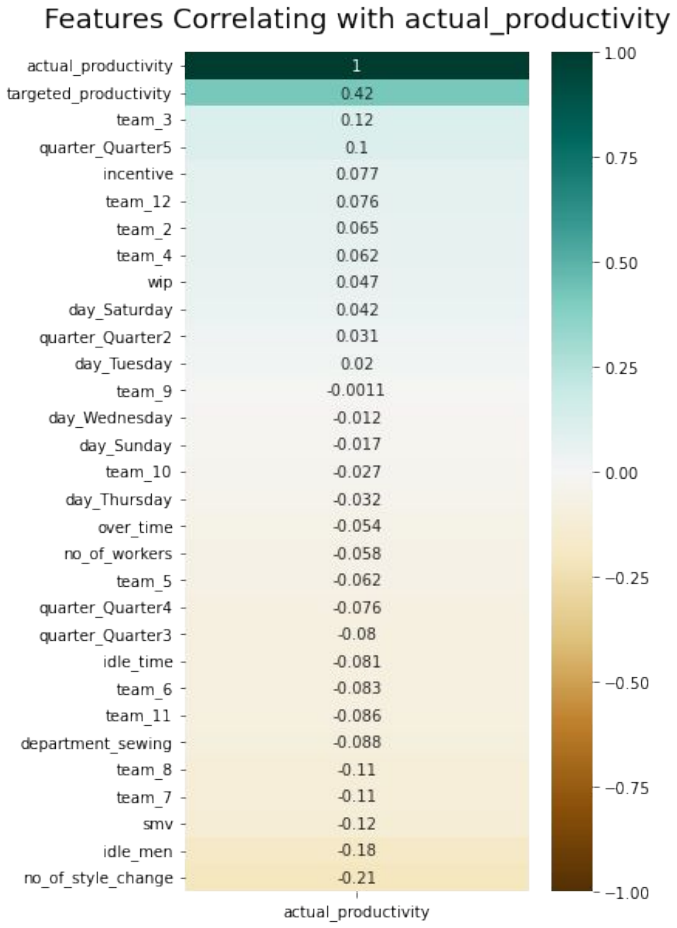
- 상관관계(더미변수 포함)
  - 봉재작업과 **smv, no_of_workers, over_time**간의 상관관계가 매우 큼
    - 해당 작업에 소요 시간과 인력이 많이 소모됨



▣ EDA(6/6)

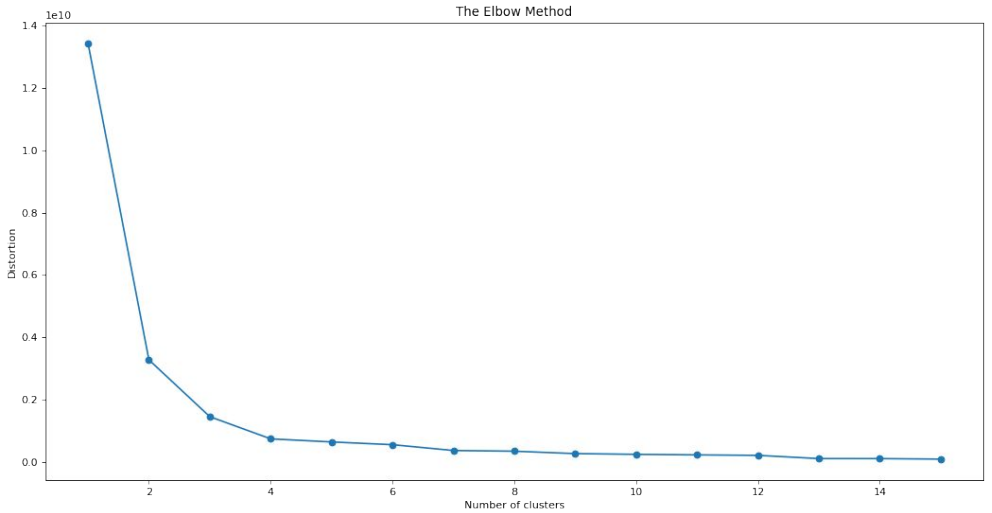
- 타겟 변수와의 상관관계(더미변수 포함)
  - 목표 생산량과 실제 생산성간에 양의 상관관계 존재
  - 스타일의 변화와 실제 생산성간에 음의 상관관계 존재
  - 쉬는 노동자와 실제 생산성간에 음의 상관관계 존재

- 변수 선택
  - 타겟 변수와의 상관관계가 0.05 이상인 변수만 포함
  - no_of_workers, department_sewing 제거
    - smv와의 높은 상관관계로 인함
  - idle_time, idle_men 제거
    - 0값의 빈도가 많기 때문



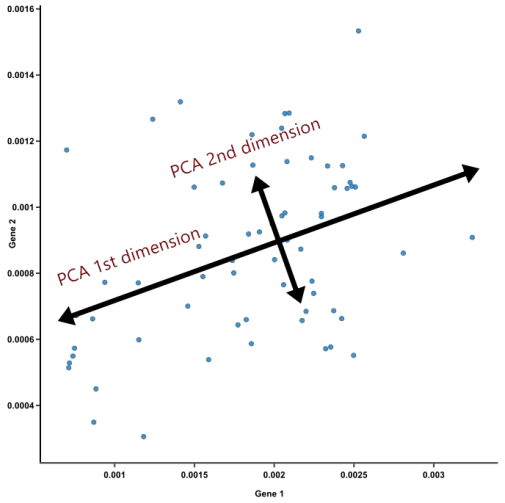
▣ 분석 과정(1/9)

- Clustering
  - K-means 알고리즘 사용
    - 클러스터 내 오차제곱합의 값이 최소가 되도록 클러스터의 중심을 결정해 나가는 방법
  - Elbow Chart 활용
    - Cluster 간 거리의 합을 나타내는 inertia가 급격히 떨어지는 구간을 최적의 군집 개수 K로 설정
    - 해당 분석에서는 4개의 Cluster가 적합할 것으로 판단됨

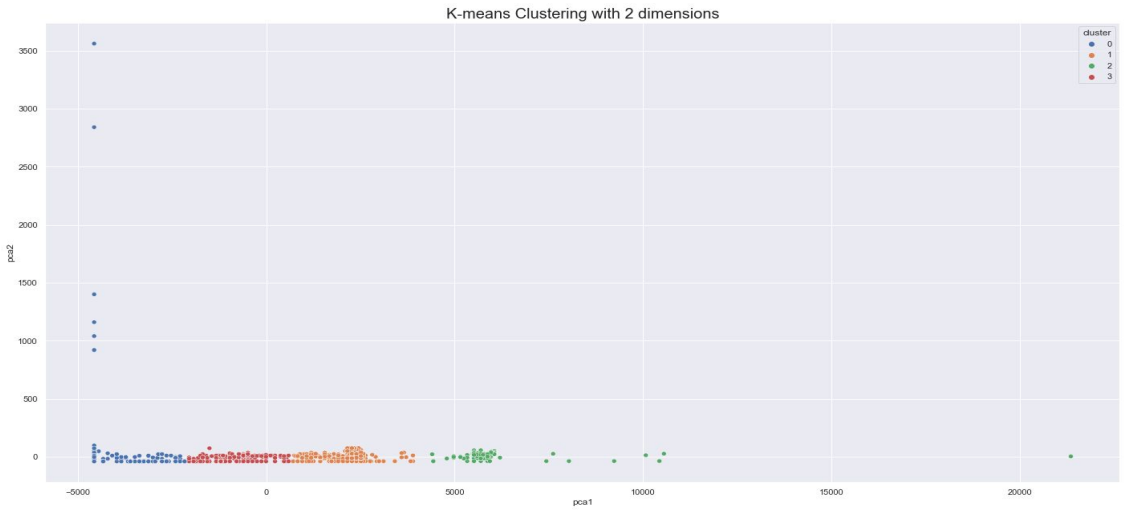


▣ 분석 과정 (2/9)

- Cluster 시각화(PCA 사용)
  - PCA(Principal Component Analysis)는 데이터 하나 하나에 대한 성분을 분석하는 것이 아니라, 여러 데이터들이 모여 하나의 분포를 이룰 때 이 분포의 주 성분을 통해 여러 변수를 주 성분의 개수로 축소시킬 수 있음
  - PCA에서는 분산이 최대인 축을 찾고(첫번째 주성분), 이 첫번째 축에 직교하고 남은 분산을 최대한 보존하는 두번째 축(두번째 주성분)을 찾으면서 주성분을 추출
  - PCA를 통해 시각화한 결과 4개의 클러스터로 잘 나뉘는 것을 확인



<PCA 개념>



<PCA를 통한 Cluster 시각화>

▣ 분석 과정 (3/9)

- Linear Regression - 모델 학습 및 테스트
  - 0, 1번 클러스터의 정확도가 낮게 나타나는 모습
  - PCA를 통한 Cluster 시각화 그래프에서 군집의 중심에서 크게 벗어난 데이터 포인트가 존재하기 때문

```
Linear Regression for Cluster #0:
R2 square: -0.05109049115183373
MAE:  0.14118468741088888
MSE:  0.032176437969241176

Linear Regression for Cluster #1:
R2 square: 0.7310274580528067
MAE:  0.06695440332121363
MSE:  0.00813287848797873

Linear Regression for Cluster #2:
R2 square: 0.798309490336124
MAE:  0.02825078259191317
MSE:  0.0018082450560791773

Linear Regression for Cluster #3:
R2 square: 0.38931590989885556
MAE:  0.11982502560683106
MSE:  0.02594961399224148
```

<Test Score>

▣ 분석 과정 (4/9)

- Lasso Regression - 모델 학습 및 테스트
  - 전체적으로 Linear Regression보다 성능이 매우 하락
  - 변수 제거의 효과를 보이기 때문에 성능이 하락한 것으로 파악됨

```
Lasso Regression for Cluster #0:
R2 square: -0.004539965604514862
MAE:  0.14585206760156383
MSE:  0.030751413092393997

Lasso Regression for Cluster #1:
R2 square: 0.3435042798407272
MAE:  0.09977857567131936
MSE:  0.019850353055672447

Lasso Regression for Cluster #2:
R2 square: 0.1297892133690759
MAE:  0.06619640919116886
MSE:  0.0078018264483268015

Lasso Regression for Cluster #3:
R2 square: 0.00797708978362932
MAE:  0.16159198261394167
MSE:  0.042153728922774504
```

<Test Score>

▣ 분석 과정 (5/9)

- Ridge Regression – 모델 학습 및 테스트
  - Linear Regression보다 다소 하락한 성능
  - 유의미하지 않은 변수의 계수를 아예 0으로 제한시키는 Lasso Regression보다는 나은 모습

```
Ridge Regression for Cluster #0:
R2 square: -0.04367908282440247
MAE:  0.13998410733614805
MSE:  0.031949556723221174

Ridge Regression for Cluster #1:
R2 square: 0.6890509922329333
MAE:  0.07287289758137018
MSE:  0.009402113977208878

Ridge Regression for Cluster #2:
R2 square: 0.7196567494687665
MAE:  0.03114399131971725
MSE:  0.002513401833448108

Ridge Regression for Cluster #3:
R2 square: 0.36526348744278303
MAE:  0.12021368626303042
MSE:  0.026971666291344322
```

<Test Score>

▣ 분석 과정 (6/9)

- Decision Tree Regression - 모델 학습 및 테스트
  - Linear Regression 보다 0, 1번 클러스터의 성능은 상승하였지만, 2, 3번 클러스터의 성능은 하락

```
DT Regressor for Cluster #0:
R2 square: -0.20113643567552586
MAE:  0.14446681360798122
MSE:  0.036769709497377734

DT Regressor for Cluster #1:
R2 square: 0.680638594251612
MAE:  0.057370511861111105
MSE:  0.009656478270602202

DT Regressor for Cluster #2:
R2 square: 0.6819472416018195
MAE:  0.023537502612000008
MSE:  0.0028514843306425624

DT Regressor for Cluster #3:
R2 square: 0.5215114408542859
MAE:  0.08470207654838711
MSE:  0.02033226935301129
```

<Test Score>

▣ 분석 과정 (7/9)

- Random Forest Regression - 모델 학습 및 테스트
  - 모든 클러스터에서 가장 좋은 성능을 보임

```
RF Regressor for Cluster #0:
R2 square: 0.10083346111150582
MAE:  0.12819069022156726
MSE:  0.02752567605369345

RF Regressor for Cluster #1:
R2 square: 0.7597367661425267
MAE:  0.05230251654732096
MSE:  0.00726479986375441

RF Regressor for Cluster #2:
R2 square: 0.790051611501554
MAE:  0.022156407124928116
MSE:  0.001882280609865011

RF Regressor for Cluster #3:
R2 square: 0.6011855792605241
MAE:  0.08335718365534998
MSE:  0.016946700332433284
```

<Test Score>

▣ 분석 과정 (8/9)

- XGBoost Regression - 모델 학습 및 테스트
  - Random Forest 모델 보다 0, 1, 2번 클러스터에서 성능 하락을 보임
  - 3번 클러스터에 대해서는 Random Forest 모델 대비 성능 향상

```
XGB Regressor for Cluster #0:
R2 square: 0.007018002567908188
MAE:  0.13853893814794424
MSE:  0.030397595558051268

XGB Regressor for Cluster #1:
R2 square: 0.703255348412993
MAE:  0.056895842345690145
MSE:  0.008972619196901248

XGB Regressor for Cluster #2:
R2 square: 0.6435325775816138
MAE:  0.02779458516054688
MSE:  0.003195888866142237

XGB Regressor for Cluster #3:
R2 square: 0.7360884689178431
MAE:  0.06695660949028062
MSE:  0.011214312720262853
```

<Test Score>

▣ 분석 과정 (9/9)

- General Model - 모델 학습 및 테스트
  - 가장 좋은 성능을 보인 Random Forest 모델을 최종 모델로 선정
  - 성능이 좋지 않은 0번 클러스터를 제외한 1, 2, 3번 클러스터를 모두 활용하여 모델 학습 진행
  - Train set / Test set을 0.85 : 0.15 비율로 분리
  - n_estimators = 300

```
R2 square: 0.7226378296002253
MAE:  0.046405762492953806
MSE:  0.007778087169517802
```

<Test Score>

▣ 결과 분석 (1/2)

- 모델 성능 평가
  - 4개 클러스터의 평균 R2 score는 Random Forest 모델이 가장 우수
  - 앞선 EDA과정에서 이미 변수를 선택하는 과정을 거쳤기 때문에 Lasso와 Ridge모델의 성능이 하락한 것으로 추정

	Linear Reg	Lasso Reg	Ridge Reg	Decision Tree Regressor	Random Forest Regression	XGBRegressor
Cluster0	-0.051090	-0.004540	-0.043679	-0.201136	0.100833	0.007018
Cluster1	0.731027	0.343504	0.689051	0.680639	0.759737	0.703255
Cluster2	0.798309	0.129789	0.719657	0.681947	0.790052	0.643533
Cluster3	0.389316	0.007977	0.365263	0.521511	0.601186	0.736088

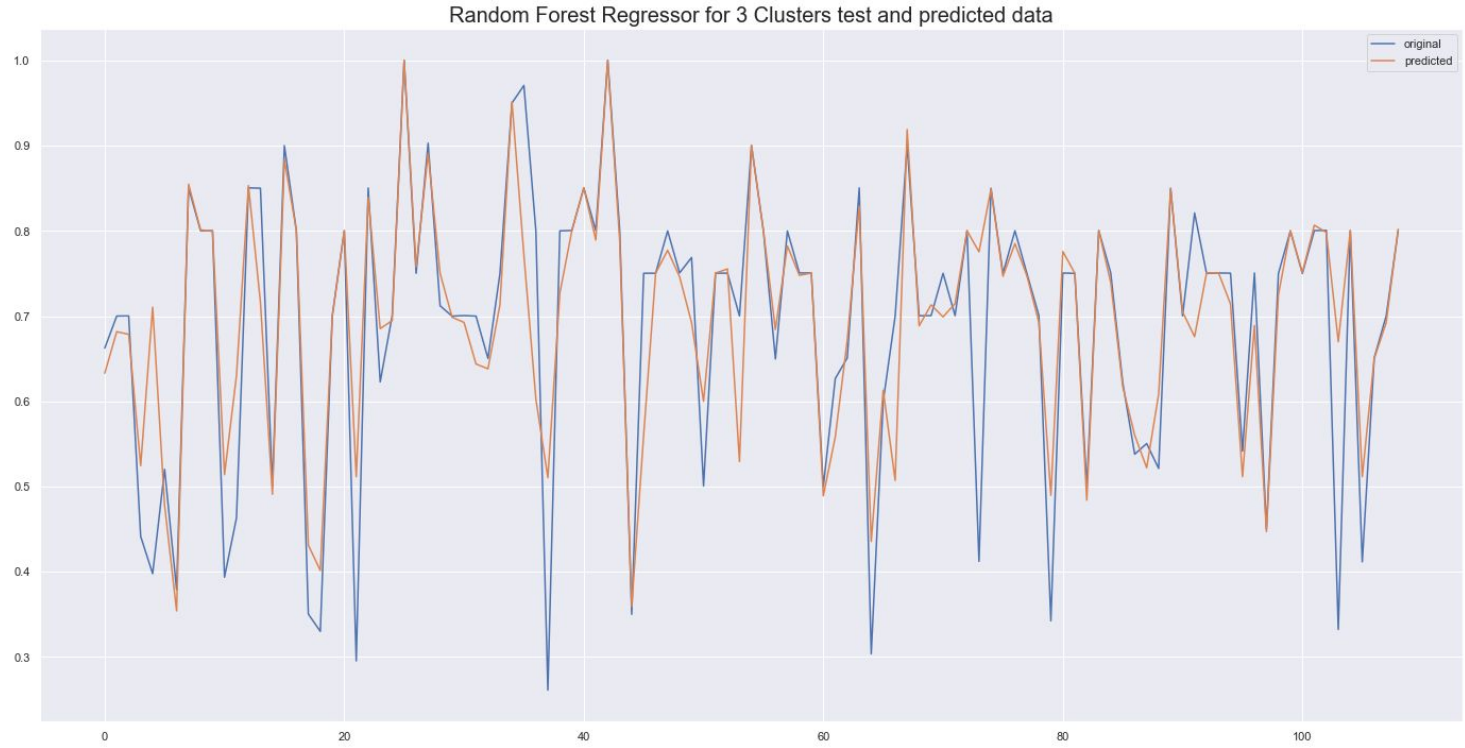
<Test Score 요약>

```
Linear Reg                  0.466891
Lasso Reg                   0.119183
Ridge Reg                   0.432573
Decision Tree Regressor     0.420740
Random Forest Regression    0.562952
XGBRegressor                0.522474
```

<Test Score 평균>

▣ 결과 분석 (2/2)

- 모델 성능 평가 – General Model
  - 대부분 실제 값과 거의 유사하게 맞추는 모습



▣ 분석 개요

- 기계 예측 유지보수분류 데이터 세트이며 분석을 통해 고장유형과 실패 여부를 검출하는 과정
  - 기계의 고장여부 혹은 어떤 유형으로 고장유형이 나타나는지 예측하기위한 데이터 셋이다
  - 대기온도 , 프로세스 온도 , 회전 속도 등 고장 유발에 관련된 변수들이 존재

- Dataset
  - 14개로 이루어진 열이 있으며 10000개의 행으로 이루어져 있다.
  - 고장 여부에는 Failure 및 No Failure 이며 고장 유형은 6개로 이루어져 있다.
  - https://drive.google.com/file/d/1uvwHXJfuVCsanbpHnq7dWrhZLJ0-IzNX/view?usp=sharing

- Consideration
  - 고장 여부에 어떠한 변수가 영향을 미치는지에 대해서 고려해 봐야 하며 또한 고장 유형에 어떤 변수가 영향을 미치는지 변수간 상관관계를 고려하여 분류 모델을 설계해야 한다.
  - 고장 여부 및 고장 유형의 레이블 불균형이 심하여 사전 처리가 필요하다
  - 따라서 Over Sampling 또는 Under Sampling 등 레이블 불균형을 조절하는 과정이 필요

https://www.kaggle.com/datasets/shivamb/machine-predictive-maintenance-classification

▣ 주요 활용 알고리즘 설명

- Naïve Bayesian Classification
  - 데이터가 각 클래스에 속할 확률을 구하는 조건부 확률 기반의 분류 방법
  - Feature들은 서로 독립관계라는 가정하에 계산
  - 사전확률 P(B|A)을 통해 사후확률 P(A|B) 을 계산할 수 있음
    - 장점

    -Feature 간의 연관 관계를 고려하지 않아 계산이 간단하고 빠르며 효율적

    -Training 시 데이터의 크기에 관계 없이 잘 처리함
    - 단점

    - Feature 간의 독립성이 있어야함 실제 데이터에선 Feature 간 독립인 경우가 적음

▣ 주요 활용 알고리즘 설명

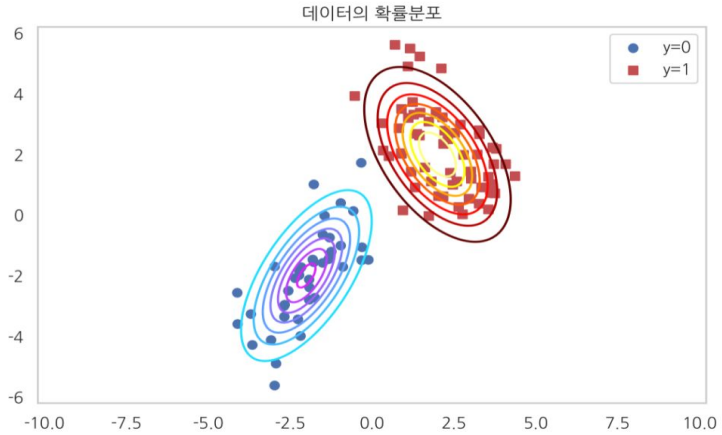
- Naïve Bayesian Classification
  - Bayesian 의 사후확률을 공식으로 나타낼 수 있다.

우도 사전확률

사후확률

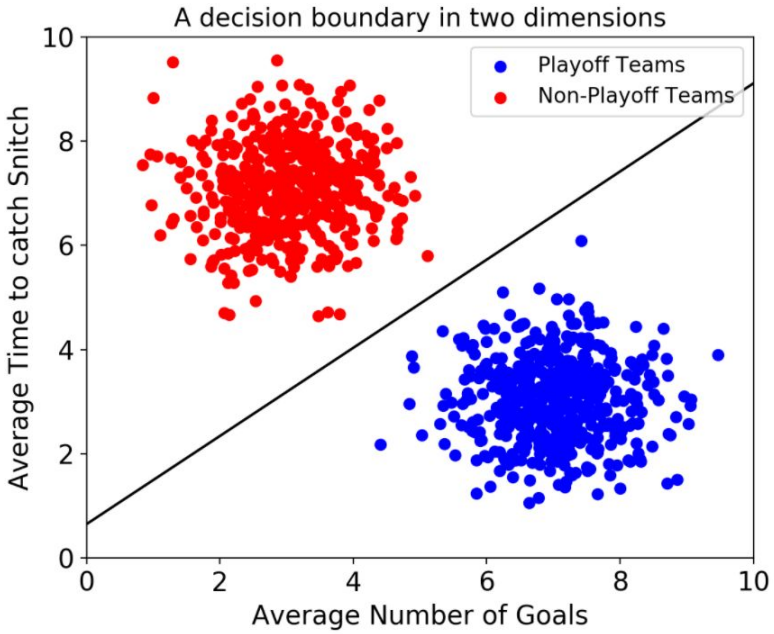
$$P(A_k|B) = \frac{P(B|A_k)P(A_k)}{P(B)}$$

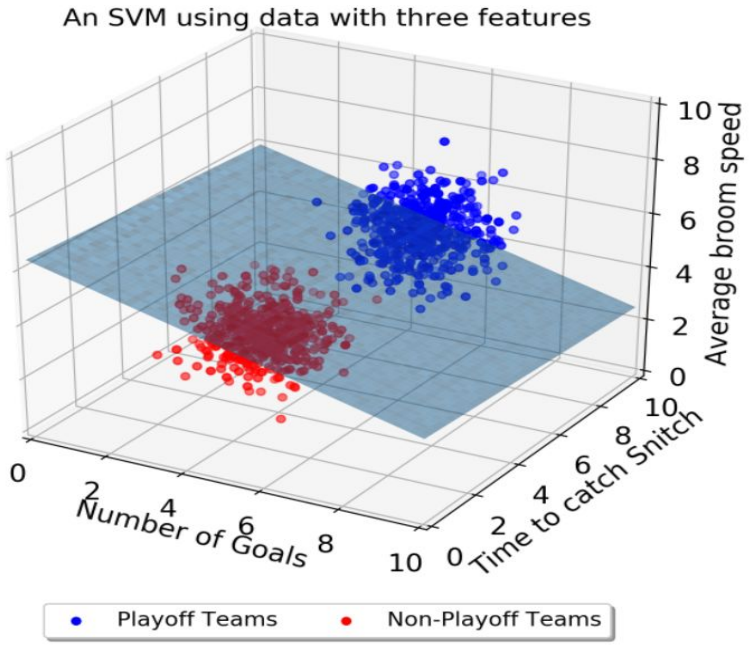
주변우도(Marginal Likelihood)



▣ 주요 활용 알고리즘 설명

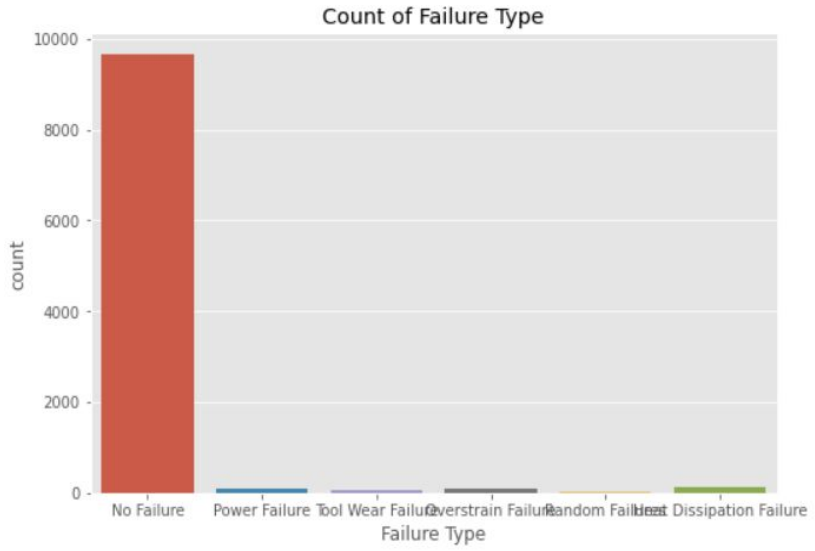
- Support Vector Classification
  - 다양한 데이터 분포에도 잘 작동하는 분류기법 중 최상의 기법
  - 정확도 측면에서 다른 분류기법보다 우수하다.
  - SVM 알고리즘은 주어진 데이터 집합을 바탕으로 하여 새로운 데이터가 어느 카테고리에 속할지 판단하는 비 확률적 이진 선형 모델을 만든다.

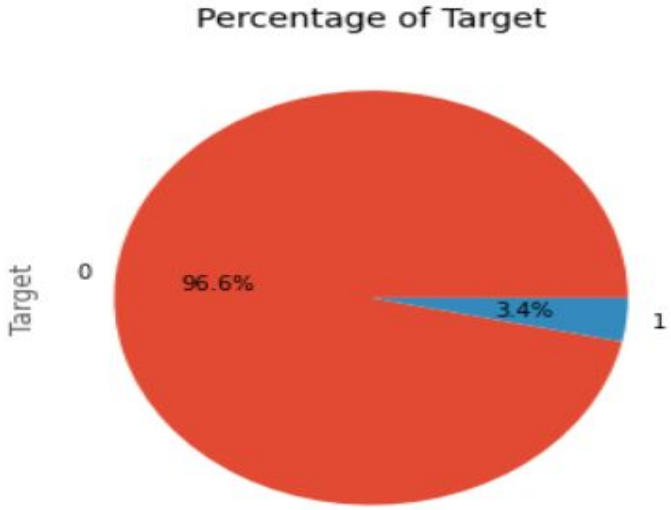




▣ 전처리(1/2)

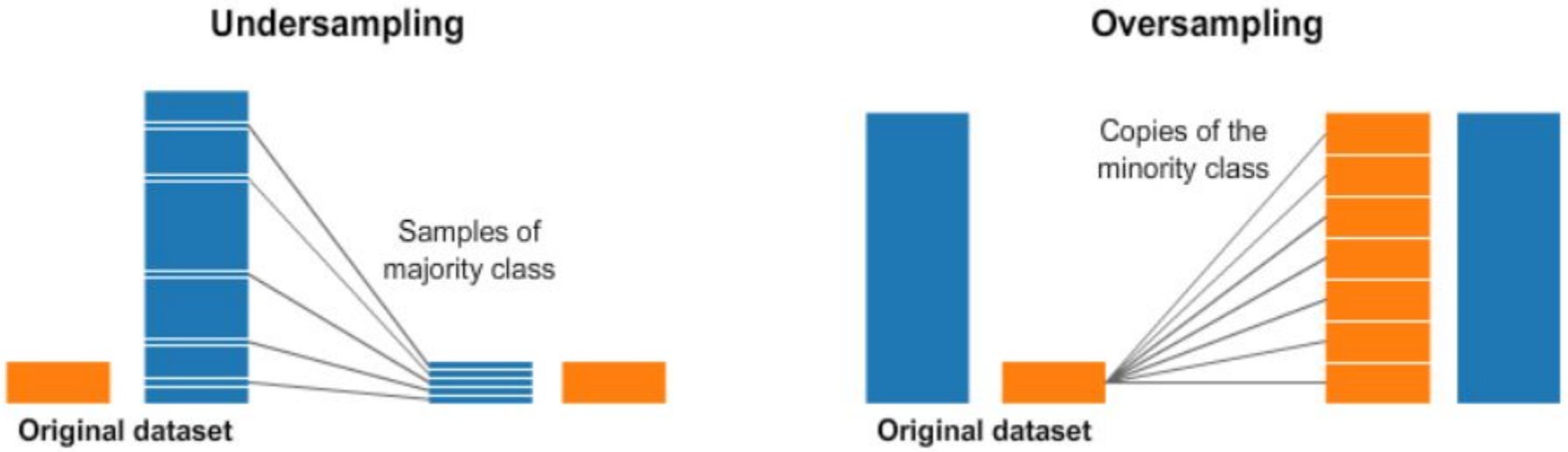
- 레이블 불균형 해결
  - 타겟 변수의 불균형한 레이블을 데이터 증강 혹은 데이터 감소로 해결하여 모델 학습을 용이하게 한다.





▣ 전처리(1/2)

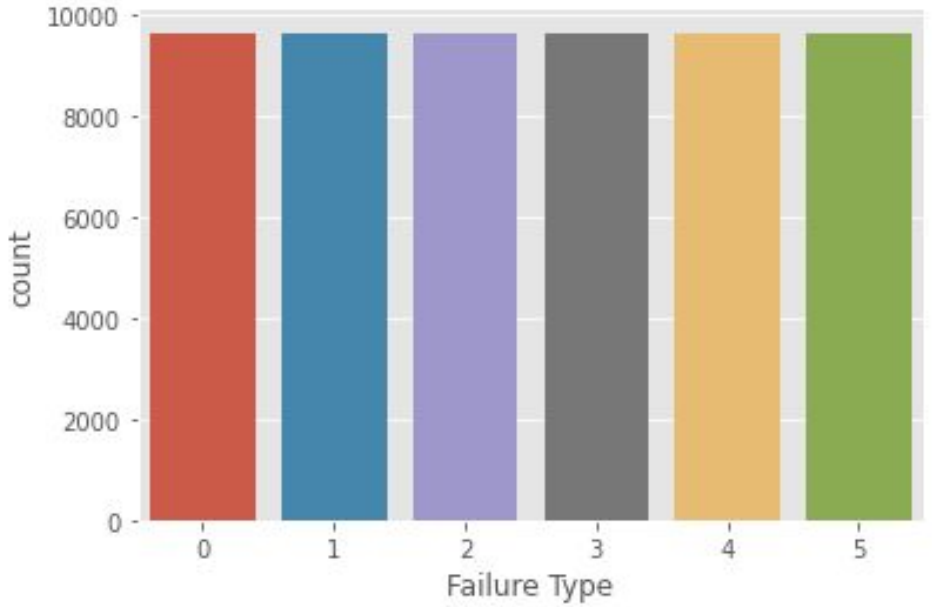
- SMOTE
  - SMOTE는 대표적인 오버 샘플링 기법 중 하나이므로 먼저 오버 샘플링과 언더 샘플링의 개념을 알아보고자 한다. 언더 샘플링과 오버 샘플링은 클래스 불균형을 해결하기 위한 기법 중 하나이다.
  - 본 분석에선 오버 샘플링을 통해 클래스 불균형을 해결한다



▣ 전처리(1/2)

- SMOTE
  - Over Sampling 후 Failure Type 에 대한 Label 이 조정 된 것을 볼 수 있다.

```
#OverSampling 진행

df_train=df.drop(df.columns[-3],axis=1)

df_train_val=df_train.values

smk=SMOTETomek(random_state=42)

X_res,y_res=smk.fit_resample(df_train_val,y)
```



▣ 전처리(2/2)

- Robust Scaler
  - Robust Scaler 란 중앙값 0 , 사분위수의 제3사분위수에서 제1사분위수를 뺀 값인 IQR이 1 이 되도록 변환하는 방법을 말한다.
  - 이상치 의 영향을 최소화한다는 특징을 가진다.
  - 본 분석에서 Target (고장 여부 , 고장 타입) 이외에는 연속형 변수이므로 Robuster Sclaer 로 변수간 Scale 조정한다.

$$\frac{n\sum_{i=1}^{n}(x_i - Q)^2(1-u_i^2)^4 I(|u_i|<1)}{\left(\sum_i (1-u_i^2)(1-5u_i^2)I(|u_i|<1)\right)^2},$$

$$u_i = \frac{x_i - Q}{9 \cdot \mathrm{MAD}}.$$

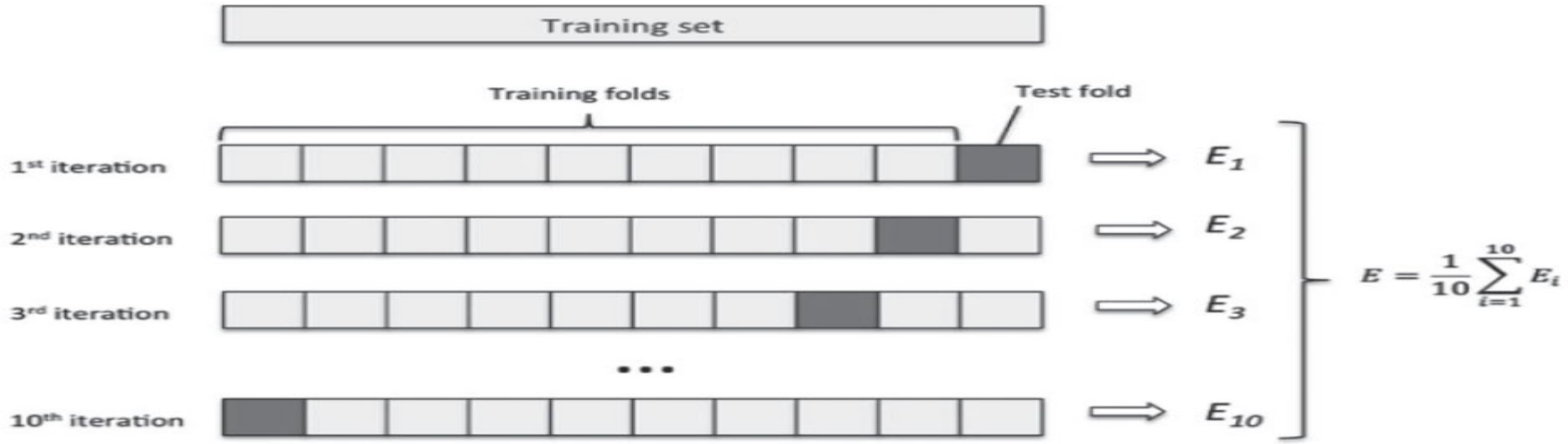
```
ro_scaler=RobustScaler()
X_train=ro_scaler.fit_transform(X_train)
X_valid=ro_scaler.transform(X_valid)
```

▣ 분석 과정

- Stratified K-fold

  - 데이터를 K개의 분할로 나누고 K개의 모델을 만들어 K-1 개의 분할에서 훈련하고 나머지 분할에서 평가하는 방법이다. 모델의 검증 평균 점수는 K개의 검증 점수의 평균이 된다.

```
from sklearn.naive_bayes import GaussianNB
acc_Gauss=[]
kf=model_selection.StratifiedKFold(n_splits=5)
for fold , (trn_,val_) in enumerate(kf.split(X=df_new,y=y_res)):
```



▣ 분석 과정 (1/3)

- 모델학습 (Target)
  - Support Vector Classification

```
svc     = SVC()
svc_clf = MultiOutputClassifier(estimator=svc)

svc_clf.fit(X_train, y_train)

print("Multi-Output Training Accuracy: ", svc_clf.score(X_train, y_train)*100, "%")
```

▣ 분석 과정 (2/3)

- 모델학습 (Failure Type)
  - Naïve Bayesian Classification
  - Stratified K-fold 로 레이블 비율 일정하게

5 Fold 로 데이터 분할 후 검증

```
from sklearn.naive_bayes import GaussianNB
acc_Gauss=[]
kf=model_selection.StratifiedKFold(n_splits=5)
for fold , (trn_,val_) in enumerate(kf.split(X=df_new,y=y_

    X_train=df_new.loc[trn_,feature_col]
    y_train=df_new.loc[trn_,target_col]

    X_valid=df_new.loc[val_,feature_col]
    y_valid=df_new.loc[val_,target_col]

    ro_scaler=RobustScaler()
    X_train=ro_scaler.fit_transform(X_train)
    X_valid=ro_scaler.transform(X_valid)

    clf=GaussianNB()
    clf.fit(X_train,y_train)
    y_pred=clf.predict(X_valid)
    print(f"The fold is : {fold} : ")
    print(classification_report(y_valid,y_pred))
```

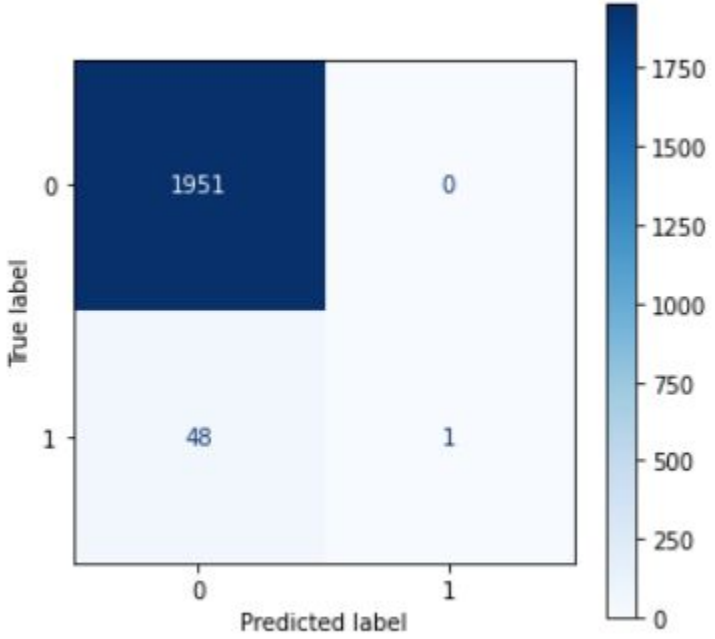
▣ 분석 과정 (3/3)

- 모델학습 (Faliure Type)
  - Support Vector Classification
  - SVC 경우 Kernel을 지정 해줘야 한다.

```
from sklearn.svm import SVC
acc_svm_poly=[]
kf=model_selection.StratifiedKFold(n_splits=5)
for fold , (trn_,val_) in enumerate(kf.split(X=df_new,y=y_res)):

    X_train=df_new.loc[trn_,feature_col]
    y_train=df_new.loc[trn_,target_col]

    X_valid=df_new.loc[val_,feature_col]
    y_valid=df_new.loc[val_,target_col]

    ro_scaler=RobustScaler()
    X_train=ro_scaler.fit_transform(X_train)
    X_valid=ro_scaler.transform(X_valid)

    clf=SVC(kernel="poly")
    clf.fit(X_train,y_train)
    y_pred=clf.predict(X_valid)
    print(f"The fold is : {fold} : ")
    print(classification_report(y_valid,y_pred))
```

▣ 분석 결과 (1/3)

- Support Vector Classification (Target)
  - Accuracy : 0.9755




```
cm            = confusion_matrix(y_test[:,0],
                                 y_pred_svc[:,0])

disp          = ConfusionMatrixDisplay(confusion_matrix=cm,)

fig, ax       = plt.subplots(figsize = (5,5))

disp.plot(cmap = plt.cm.Blues,
          ax   = ax)
```

▣ 분석 결과 (2/3)

- Naïve Baysian Classification (Failure Type)
  - 각 폴드 별 약 0.77의 Accuracy를 가진다

The fold is : 0 :

	precision	recall	f1-score	support
0	0.79	0.92	0.85	1930
1	0.69	0.45	0.54	1925
2	0.81	0.92	0.86	1930
3	0.84	0.76	0.80	1929
4	0.59	0.67	0.63	1928
5	0.88	0.89	0.88	1929
accuracy			0.77	11571
macro avg	0.77	0.77	0.76	11571
weighted avg	0.77	0.77	0.76	11571

The fold is : 1 :

	precision	recall	f1-score	support
0	0.79	0.91	0.85	1929
1	0.68	0.47	0.56	1925
2	0.82	0.92	0.86	1931
3	0.86	0.76	0.81	1929
4	0.59	0.65	0.62	1928
5	0.88	0.91	0.89	1929
accuracy			0.77	11571
macro avg	0.77	0.77	0.77	11571
weighted avg	0.77	0.77	0.77	11571

The fold is : 2 :

	precision	recall	f1-score	support
0	0.79	0.92	0.85	1929
1	0.70	0.48	0.57	1925
2	0.82	0.92	0.86	1931
3	0.86	0.75	0.80	1929
4	0.60	0.67	0.64	1928
5	0.87	0.90	0.88	1929
accuracy			0.77	11571
macro avg	0.77	0.77	0.77	11571
weighted avg	0.77	0.77	0.77	11571

The fold is : 3 :

	precision	recall	f1-score	support
0	0.81	0.94	0.87	1930
1	0.69	0.47	0.56	1924
2	0.82	0.92	0.86	1930
3	0.87	0.74	0.80	1930
4	0.60	0.68	0.64	1929
5	0.87	0.91	0.89	1928
accuracy			0.78	11571
macro avg	0.78	0.78	0.77	11571
weighted avg	0.78	0.78	0.77	11571

▣ 분석 결과 (3/3)

- Support Vector Classification (Failure Type)

  - 각 폴드 별 약 0.88 의 정확도를 가진다

The fold is : 0 :

	precision	recall	f1-score	support
0	0.97	0.93	0.95	1930
1	1.00	0.67	0.80	1925
2	0.74	1.00	0.85	1930
3	0.99	0.92	0.95	1929
4	0.78	0.92	0.85	1928
5	0.92	0.83	0.87	1929
accuracy			0.88	11571
macro avg	0.90	0.88	0.88	11571
weighted avg	0.90	0.88	0.88	11571

The fold is : 1 :

	precision	recall	f1-score	support
0	0.98	0.94	0.96	1929
1	1.00	0.66	0.80	1925
2	0.77	1.00	0.87	1931
3	0.99	0.93	0.96	1929
4	0.78	0.94	0.85	1928
5	0.92	0.85	0.89	1929
accuracy			0.89	11571
macro avg	0.91	0.89	0.89	11571
weighted avg	0.91	0.89	0.89	11571

The fold is : 2 :

	precision	recall	f1-score	support
0	0.98	0.92	0.95	1929
1	1.00	0.67	0.80	1925
2	0.74	1.00	0.85	1931
3	0.99	0.94	0.96	1929
4	0.79	0.94	0.86	1928
5	0.92	0.83	0.87	1929
accuracy			0.88	11571
macro avg	0.90	0.88	0.88	11571
weighted avg	0.90	0.88	0.88	11571

The fold is : 3 :

	precision	recall	f1-score	support
0	0.97	0.93	0.95	1930
1	1.00	0.65	0.79	1924
2	0.74	1.00	0.85	1930
3	0.99	0.92	0.95	1930
4	0.78	0.93	0.85	1929
5	0.91	0.84	0.88	1928
accuracy			0.88	11571
macro avg	0.90	0.88	0.88	11571
weighted avg	0.90	0.88	0.88	11571