

# 학생 직장인에 따른 우울증 상관분석 및 예측 모델 구축

쿠글 12기 2조  
강지윤 문준영 문형주 박슬옹

# Contents 목차

- 01 프로젝트 개요**
  - 목적
  - 데이터 소개
- 02 데이터 탐색 및 전처리**
  - 결측치 처리
  - 변수 선택
- 03 우울증 예측 모델 구축**
  - 머신러닝 모델 소개
  - 하이퍼파라미터 튜닝 및 모델 최적화
  - 모델 평가
- 04 결론**
  - 우울증 예측 중요 요인
  - 결론 및 한계점

# 1. 프로젝트 개요

## 목적

학생 / 직장인 각각의 특성을 반영하여 우울증 예측에 영향을 미치는 중요 요인(학업/직무 스트레스, 수면, 경제적 스트레스등 )  
규명

# 1. 프로젝트 개요 데이터 소개

[5]:

	id	Name	Gender	Age	City	Working Professional or Student	Profession	Academic Pressure	Work Pressure	CGPA	Study Satisfaction	Job Satisfaction	Sleep Duration	Dietary Habits	Degree
0	0	Aaradhya	Female	49	Ludhiana	Working Professional	Chef	NaN	5.0	NaN	NaN	2.0	More than 8 hours	Healthy	BHM
1	1	Vivan	Male	26	Varanasi	Working Professional	Teacher	NaN	4.0	NaN	NaN	3.0	Less than 5 hours	Unhealthy	LLB
2	2	Yuvraj	Male	33	Visakhapatnam	Student	NaN	5.0	NaN	8.97	2.0	NaN	5-6 hours	Healthy	B.Pharm
3	3	Yuvraj	Male	22	Mumbai	Working Professional	Teacher	NaN	5.0	NaN	NaN	1.0	Less than 5 hours	Moderate	BBA
4	4	Rhea	Female	30	Kanpur	Working Professional	Business Analyst	NaN	1.0	NaN	NaN	1.0	5-6 hours	Unhealthy	BBA
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
139685	139791	Prachi	Female	54	Kolkata	Working Professional	Teacher	NaN	3.0	NaN	NaN	5.0	7-8 hours	Healthy	M.Pharm
139686	139792	Atharv	Male	57	Nashik	Working Professional	NaN	NaN	1.0	NaN	NaN	3.0	5-6 hours	Moderate	Class 12
139687	139793	Darsh	Male	54	Ahmedabad	Working Professional	Pharmacist	NaN	2.0	NaN	NaN	3.0	More than 8 hours	Healthy	MD

# 1. 프로젝트 개요 데이터 소개

## columns - 학생

**Gender** - 성별

**Age** - 나이

**Academic Pressure** - 학업 스트레스 수준

**CGPA** - 학점(GPA)

**Study Satisfaction** - 학업 만족도

**Sleep Duration** - 수면 시간 (범주형: 5-6시간, 7-8시간 등)

**Dietary Habits** - 식습관  
(Healthy/Unhealthy/Moderate 등)

**Have you ever had suicidal thoughts?** - 자살 생각 경험 여부 (Yes/No)

**Work/Study Hours** - 하루 평균 학습 시간

**Financial Stress** - 경제적 스트레스 수준

**Family History of Mental Illness** - 가족 중 정신 질환 병력 여부 (Yes/No)

**Depression** - 우울증 위험군 여부 (0: 아님, 1: 위험군)

# 1. 프로젝트 개요 데이터 소개

## columns - 직장인

**Gender** - 성별

**Age** - 나이

**Work Pressure** - 직무 스트레스 수준

**Job Satisfaction** - 직무 만족도

**Sleep Duration** - 수면 시간 (범주형)

**Dietary Habits** - 식습관

(Healthy/Unhealthy/Moderate 등)

**Have you ever had suicidal thoughts?** - 자살  
생각 경험 여부 (Yes/No)

**Work/Study Hours** - 하루 평균 근무 시간

**Financial Stress** - 경제적 스트레스 수준

**Family History of Mental Illness** - 가족 중 정신  
질환 병력 여부 (Yes/No)

**Depression** - 우울증 위험군 여부 (0: 아님, 1: 위험  
군)

## 2. 데이터 탐색 및 전처리    데이터 전처리

```
# 독립변수 정의
student_features = [
    'Academic Pressure', 'CGPA', 'Study Satisfaction',
    'Sleep Duration', 'Dietary Habits',
    'Have you ever had suicidal thoughts ?',
    'Work/Study Hours', 'Financial Stress',
    'Family History of Mental Illness'
]

worker_features = [
    'Work Pressure', 'Job Satisfaction',
    'Sleep Duration', 'Dietary Habits',
    'Have you ever had suicidal thoughts ?',
    'Work/Study Hours', 'Financial Stress',
    'Family History of Mental Illness'
]

# 모델 학습 및 평가 함수
def evaluate_logistic_model(train_df, test_df, features, title='Model'):
    X_train = train_df[features].copy()
    y_train = train_df['Depression']
    X_test = test_df[features].copy()
    y_test = test_df['Depression']
```

### 독립변수 정의 / 모델 학습

- 일부 컬럼에서 비정상적 수치  
(40 시간 수면 등) 있었음
- 이상치는 전체 데이터 중 극히 일부,  
분석 방해 요소로 간주 / 삭제함

## 2. 데이터 탐색 및 전처리    데이터 전처리

```
# 결측값 처리
for col in X_train.select_dtypes(include='number').columns:
    X_train[col].fillna(X_train[col].mean(), inplace=True)
    X_test[col].fillna(X_train[col].mean(), inplace=True)

for col in X_train.select_dtypes(include='object').columns:
    X_train[col].fillna(X_train[col].mode()[0], inplace=True)
    X_test[col].fillna(X_train[col].mode()[0], inplace=True)
```

결측치 처리  
수치형은 평균값 대체, 범주형은 최빈값 대체

```
# 더미 변수화
X_train = pd.get_dummies(X_train, drop_first=True)
X_test = pd.get_dummies(X_test, drop_first=True)
X_test = X_test.reindex(columns=X_train.columns, fill_value=0)

# 스케일링
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

범주형 변수 변환 - 원 핫 인코딩 (**get dummies**) 통한 수치화  
특성 스케일링 - 표준화를 통한 모델 성능 향상



### 3. 우울증 예측 모델 구축    사용 모델 소개

1. 로지스틱 회귀 (기각) - 결측치 처리, 인코딩, 스케일링 등 전처리와 함께 모델 학습을 수행, 분류 성능 평가(정확도, 재현율, **F1-score**, 혼동행렬 등) 및 회귀계수 시각화로 변수 영향력 해석
2. **KNN** (기각) - 학생 및 직장인 데이터에서 우울증 위험군 예측을 위한 비교·보완용 머신러닝 분류기로 활용
3. **랜덤 포레스트 (채택)** - 비선형적 관계 및 변수 간 상호작용까지 반영해 우울증 예측, 변수 중요도 산출 및 성능 개선 목적

※ 학생의 우울증 예측 모델, 직장인의 우울증 예측 모델은 각각 구축하였음

(각 집단 특성에 맞는 별도의 예측 모델을 구축해야 정확한 예측과 해석이 가능하기 때문)

# 로지스틱 회귀

## ● 모델 학습과정

```
# 로지스틱 회귀 모델 학습
model = LogisticRegression(max_iter=1000)
model.fit(X_train_scaled, y_train)

# 예측
y_pred = model.predict(X_test_scaled)
y_proba = model.predict_proba(X_test_scaled)[: , 1]

# 평가
print(f"\n📊 {title} 결과")
print("정확도:", accuracy_score(y_test, y_pred))
print("ROC-AUC:", roc_auc_score(y_test, y_proba))
print("\n분류 보고서:\n", classification_report(y_test, y_pred))
print("혼동 행렬:\n", confusion_matrix(y_test, y_pred))
```

```
# ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_proba)
plt.figure()
plt.plot(fpr, tpr, label=f'{title} (AUC = {roc_auc_score(y_test, y_proba):.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title(f'ROC Curve - {title}')
plt.legend()
plt.grid(True)
plt.show()
```

# 모델 실행

```
evaluate_logistic_model(student_train, student_test, student_features, title='Logistic for Student')
evaluate_logistic_model(worker_train, worker_test, worker_features, title='Logistic for Professional')
```

정확도 - 전체 예측 중 올바른 비율

재현율 - 실제 양성 중 맞춘 비율

**ROC-AUC** - 모델의 판별력 지표 (1에 가까울수록 우수)

혼동행렬 - TP/TN/FP/FN 시각화

# 로지스틱 회귀

## ● 결과

### Logistic for Student 결과

정확도: 0.83640081799591

ROC-AUC: 0.9133470380679163

분류 보고서:

	precision	recall	f1-score	support
0.0	0.82	0.78	0.80	3442
1.0	0.85	0.88	0.86	4871
accuracy			0.84	8313
macro avg	0.83	0.83	0.83	8313
weighted avg	0.84	0.84	0.84	8313

혼동 행렬:

```
[[2672  770]
 [ 590 4281]]
```

- 학생
- 정확도 : 약 0.84
- 재현율 : 약 0.88

### Logistic for Professional 결과

정확도: 0.9329364488763209

ROC-AUC: 0.8990837547384356

분류 보고서:

	precision	recall	f1-score	support
0.0	0.94	0.99	0.96	30844
1.0	0.69	0.33	0.44	2751
accuracy			0.93	33595
macro avg	0.82	0.66	0.70	33595
weighted avg	0.92	0.93	0.92	33595

혼동 행렬:

```
[[30447  397]
 [ 1856  895]]
```

- 직장인
- 정확도 : 약 0.93
- 재현율 : 약 0.33



# 로지스틱 회귀

- 독립변수 유의성 검정

```
import statsmodels.api as sm

def logistic_regression_significance(df, features):
    X = df[features].copy()
    y = df['Depression']

    # 결측값 처리
    for col in X.select_dtypes(include='number').columns:
        X[col].fillna(X[col].mean(), inplace=True)
    for col in X.select_dtypes(include='object').columns:
        X[col].fillna(X[col].mode()[0], inplace=True)

    # 원-핫 인코딩 + 숫자형 강제 변환
    X = pd.get_dummies(X, drop_first=True)
    X = X.astype(float) # ★ 중요: 모든 열을 float로 변환
```

```
# 상수항 추가
X = sm.add_constant(X)

# 로지스틱 회귀 분석
model = sm.Logit(y, X)
result = model.fit()

# 결과 출력
print(result.summary())
return result
```

특정 **feature**들을 사용해 '**우울증(Depression)**' 여부를 예측하는  
로지스틱 회귀 모델을 학습하고, 각 변수의 유의성을 분석

# 로지스틱 회귀

## ● 학생

학업 압박, 경제적 스트레스, 수면 부족, 나쁜 식습관, 자살 생각 경험 등이 우울증과 강하게 관련

특히 자살 생각 경험이 우울증과 매우 높은 관련성을 보임  
(coefficient 2.48).

수면시간이 많거나, 학업만족도가 높을수록 우울증 위험이 감소하는 경향이 있음.

Logit Regression Results						
Dep. Variable:	Depression	No. Observations:	27707			
Model:	Logit	Df Residuals:	27694			
Method:	MLE	Df Model:	12			
Date:	Thu, 24 Apr 2025	Pseudo R-squ.:	0.4623			
Time:	03:47:00	Log-Likelihood:	-10105.			
converged:	True	LL-Null:	-18794.			
Covariance Type:	nonrobust	LLR p-value:	0.000			
	coef	std err	z	P> z	[0.025	0.975]
const	-6.6783	0.144	-46.234	0.000	-6.961	-6.395
Academic Pressure	0.8259	0.015	56.911	0.000	0.798	0.854
CGPA	0.0551	0.012	4.524	0.000	0.031	0.079
Study Satisfaction	-0.2277	0.013	-17.191	0.000	-0.254	-0.202
Work/Study Hours	0.1153	0.005	23.713	0.000	0.106	0.125
Financial Stress	0.5595	0.013	42.811	0.000	0.534	0.585
Sleep Duration_7-8 hours	0.0308	0.051	0.608	0.543	-0.069	0.130
Sleep Duration_Less than 5 hours	0.3562	0.050	7.109	0.000	0.258	0.454
Sleep Duration_More than 8 hours	-0.2290	0.053	-4.316	0.000	-0.333	-0.125
Dietary Habits_Moderate	0.4563	0.044	10.363	0.000	0.370	0.543
Dietary Habits_Unhealthy	1.1022	0.045	24.387	0.000	1.014	1.191
Have you ever had suicidal thoughts ?_Yes	2.4889	0.038	65.399	0.000	2.414	2.564
Family History of Mental Illness_Yes	0.2402	0.036	6.737	0.000	0.170	0.310

# 로지스틱 회귀

## ● 직장인

직장 압박, 경제적 스트레스, 수면 부족, 나쁜 식습관, 자살 생각 경험은 모두 우울증 위험을 높임

특히 자살 생각 경험과 나쁜 식습관이 강력한 영향력을 가지고 있음

직업 만족도와 충분한 수면(8시간 이상)은 우울증 위험을 줄이는 보호요인임.

Logit Regression Results						
Dep. Variable:	Depression	No. Observations:	111982			
Model:	Logit	Df Residuals:	111970			
Method:	MLE	Df Model:	11			
Date:	Thu, 24 Apr 2025	Pseudo R-squ.:	0.3600			
Time:	03:47:01	Log-Likelihood:	-20309.			
converged:	True	LL-Null:	-31731.			
Covariance Type:	nonrobust	LLR p-value:	0.000			
	coef	std err	z	P> z	[0.025	0.975]
const	-8.5767	0.086	-99.248	0.000	-8.746	-8.407
Work Pressure	0.6502	0.011	60.924	0.000	0.629	0.671
Job Satisfaction	-0.4395	0.010	-44.077	0.000	-0.459	-0.420
Work/Study Hours	0.1451	0.004	40.409	0.000	0.138	0.152
Financial Stress	0.5195	0.010	51.958	0.000	0.500	0.539
Sleep Duration_7-8 hours	0.0245	0.039	0.633	0.527	-0.051	0.100
Sleep Duration_Less than 5 hours	0.6475	0.036	17.977	0.000	0.577	0.718
Sleep Duration_More than 8 hours	-0.2354	0.042	-5.591	0.000	-0.318	-0.153
Dietary Habits_Moderate	0.3233	0.036	9.026	0.000	0.253	0.393
Dietary Habits_Unhealthy	0.9951	0.033	29.710	0.000	0.929	1.061
Have you ever had suicidal thoughts ?_Yes	2.2870	0.035	64.862	0.000	2.218	2.356
Family History of Mental Illness_Yes	0.1003	0.026	3.833	0.000	0.049	0.152

# 로지스틱 회귀

## • 다중공선성 체크

```
# NaN 비율이 높은 열을 제거하거나
nan_ratio = X_train_df.isnull().mean()
print(nan_ratio.sort_values(ascending=False))

# 또는 간단히 평균값 등으로 채우기
X_train_df_filled = X_train_df.fillna(X_train_df.mean())

from statsmodels.stats.outliers_influence import variance_inflation_factor

vif_data = pd.DataFrame()
vif_data['Feature'] = X_train_df_filled.columns
vif_data['VIF'] = [
    variance_inflation_factor(X_train_df_filled.values, i)
    for i in range(X_train_df_filled.shape[1])
]

print("VIF (다중공선성) 결과:\n")
print(vif_data.sort_values(by="VIF", ascending=False))
```

다중공선성은 독립변수들끼리 너무 비슷해서, 모델을 불안정하게 만드는 현상

서로 비슷한 의미를 가진 변수가 많으면 모델이 불안정해질 수 있기 때문에, **VIF**가 높은 변수를 찾고, 제거하거나 조정할지를 결정



# 로지스틱 회귀

- 다중공선성

VIF (다중공선성) 결과:

	Feature	VIF
9	Depression	1.581727
2	Academic Pressure	1.296623
8	Financial Stress	1.151504
1	Age	1.055197
7	Work/Study Hours	1.046832
5	Study Satisfaction	1.032489
4	CGPA	1.004469
0	id	1.000709
6	Job Satisfaction	1.000629
3	Work Pressure	NaN

**VIF가 1에 가까우면 → 독립성 OK (문제 없음)**

**VIF가 5 이상이면 → 약간 문제 있음**


**VIF가 10 이상이면 → 심각한 다중공선성 문제**

대부분 **VIF가 1~1.5 사이**

**→ 다중공선성 거의 없음 (좋음)**

# 로지스틱 회귀

## ● 결과

 Logistic for Student 결과  
정확도: 0.83640081799591  
ROC-AUC: 0.9133470380679163


- 학생
- 정확도 : 약 0.84
- 재현율 : 약 0.88

분류 보고서:

	precision	recall	f1-score	support
0.0	0.82	0.78	0.80	3442
1.0	0.85	0.88	0.86	4871
accuracy			0.84	8313
macro avg	0.83	0.83	0.83	8313
weighted avg	0.84	0.84	0.84	8313

혼동 행렬:

```
[[2672  770]
 [ 590 4281]]
```

 Logistic for Professional 결과  
정확도: 0.9329364488763209  
ROC-AUC: 0.8990837547384356

- 직장인
- 정확도 : 약 0.93
- 재현율 : 약 0.33

분류 보고서:

	precision	recall	f1-score	support
0.0	0.94	0.99	0.96	30844
1.0	0.69	0.33	0.44	2751
accuracy			0.93	33595
macro avg	0.82	0.66	0.70	33595
weighted avg	0.92	0.93	0.92	33595

혼동 행렬:

```
[[30447  397]
 [ 1856  895]]
```

결과: 정확도는 높게 나오지만, 직장인의 재현율이 **0.33**로 매우 낮음 (부적합)

# KNN

## ● 모델 학습과정

```
# 학생과 직장인 구분
students = df[df['Working Professional or Student'] == 'Student'].copy()
workers = df[df['Working Professional or Student'] == 'Working Professional'].copy()

# 타겟 변수 정의
target = 'Depression'

# 공통 전처리 함수 정의
def preprocess_and_train_knn(data, features, target, k=5):
    # 1. 필요한 열만 선택
    data = data[features + [target]].copy()

    # 2. 결측값 처리
    # 수치형은 평균으로, 범주형은 최빈값으로
    num_cols = data.select_dtypes(include=np.number).columns.tolist()
    cat_cols = data.select_dtypes(exclude=np.number).columns.tolist()

    imputer_num = SimpleImputer(strategy='mean')
    imputer_cat = SimpleImputer(strategy='most_frequent')

    data[num_cols] = imputer_num.fit_transform(data[num_cols])
    data[cat_cols] = imputer_cat.fit_transform(data[cat_cols])

    # 3. 범주형 변수 인코딩
    label_encoders = {}
    for col in cat_cols:
        le = LabelEncoder()
        data[col] = le.fit_transform(data[col])
        label_encoders[col] = le
```

데이터 전처리와 모델 훈련을 위한 준비 단계

**preprocess\_and\_train\_knn** 함수는  
주어진 데이터에 대해 전처리를 진행하고,  
**KNN** 분류 모델을 훈련

# KNN

## ● 모델 학습과정

```
# 4. 학습/테스트셋 분리
X = data[features]
y = data[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# 5. 정규화
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# 6. KNN 분류
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train_scaled, y_train)
y_pred = knn.predict(X_test_scaled)
```

```
# 7. 결과 출력
report = classification_report(y_test, y_pred, output_dict=True)
return report
```

```
# 학생과 직장인 각각에 대해 KNN 모델 학습 및 평가
student_report = preprocess_and_train_knn(students, student_features, target)
worker_report = preprocess_and_train_knn(workers, worker_features, target)

student_report, worker_report
```

**knn.predict(X\_test\_scaled)**는 학습한 모델을 사용하여 테스트셋에 대한 예측을 수행

**classification\_report** 함수는 모델의 성능을 평가하는 지표 출력(**precision, recall, f1-score, accuracy** 등)

# KNN

## ● 학생 결과

```
{('0.0': {'precision': 0.7976851851851852,
  'recall': 0.7426724137931034,
  'f1-score': 0.7691964285714287,
  'support': 2320},
 '1.0': {'precision': 0.8251830161054172,
  'recall': 0.8657450076804916,
  'f1-score': 0.8449775112443778,
  'support': 3255},
 'accuracy': 0.8145291479820628,
 'macro avg': {'precision': 0.8114341006453012,
  'recall': 0.8042087107367974,
  'f1-score': 0.8070869699079033,
  'support': 5575},
 'weighted avg': {'precision': 0.813739972565518,
  'recall': 0.8145291479820628,
  'f1-score': 0.8134417064369801,
  'support': 5575}},
```

클래스 0 (**Depression = 0**, 우울증 없음)

**Precision: 0.7977**

**Recall: 0.7427**

**F1-score: 0.7692**

**Support: 2320명**

클래스 1 (**Depression = 1**, 우울증 있음)

**Precision: 0.8252**

**Recall: 0.8657**

**F1-score: 0.8450**

**Support: 3255명**

전체 **Accuracy (정확도): 0.8145 (81.45%)**

우울증 있는 사람(**1.0**)의 재현율: **86.6%**

# KNN

- 직장인 결과

```
{ '0.0': { 'precision': 0.9411627690300941,
  'recall': 0.9772177614394892,
  'f1-score': 0.9588514475557665,
  'support': 20674 },
  '1.0': { 'precision': 0.5634847080630213,
  'recall': 0.324959914484233,
  'f1-score': 0.41220338983050847,
  'support': 1871 },
  'accuracy': 0.9230871590153027,
  'macro avg': { 'precision': 0.7523237385465578,
  'recall': 0.6510888379618611,
  'f1-score': 0.6855274186931375,
  'support': 22545 },
  'weighted avg': { 'precision': 0.9098194267338248,
  'recall': 0.9230871590153027,
  'f1-score': 0.9134853568037612,
  'support': 22545 } }
```

클래스 0 (**Depression = 0**, 우울증 없음)

**Precision: 0.9412**

**Recall: 0.9772**

**F1-score: 0.9589**

**Support: 20674명**

클래스 1 (**Depression = 1**, 우울증 있음)

**Precision: 0.5635**

**Recall: 0.3250**

**F1-score: 0.4122**

**Support: 1871명**

전체 **Accuracy (정확도): 92.31%**

실제 우울증 있는 사람 중 **32.5%**만 정확하게 예측함.

# 랜덤 포레스트

## ● 모델 학습과정 -직장인

```

1 #직장인
2 import pandas as pd
3 from sklearn.model_selection import train_test_split
4 from sklearn.ensemble import RandomForestClassifier
5 from sklearn.metrics import classification_report, accuracy_score, r2_score
6 from sklearn.preprocessing import LabelEncoder
7
8 # 1. 데이터 불러오기
9 df = pd.read_csv("C:\Users\seulo\OneDrive바탕 화면\KUGGLE\1차 프로젝트\df_new.csv")
10
11 # 2. 인코딩할 범주형 변수들
12 cols_to_encode = ['Sleep Duration', 'Dietary Habits',
13                  'Have you ever had suicidal thoughts?',
14                  'Family History of Mental Illness']
15 le = LabelEncoder()
16 for col in cols_to_encode:
17     df[col] = le.fit_transform(df[col])
18

```

# KNN

- 혼동행렬

```
def plot_confusion(y_test, y_pred, title):
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(5, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=['Not Depressed', 'Depressed'],
                yticklabels=['Not Depressed', 'Depressed'])
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title(title)
    plt.tight_layout()
    plt.show()

# 학생 혼동 행렬
y_test_student, y_pred_student = get_predictions(students, student_features)
plot_confusion(y_test_student, y_pred_student, "Confusion Matrix - Students")

# 직장인 혼동 행렬
y_test_worker, y_pred_worker = get_predictions(workers, worker_features)
plot_confusion(y_test_worker, y_pred_worker, "Confusion Matrix - Workers")
```

혼동행렬 (4가지 요소)

**True Positive (TP):** 실제 양성(**True**)이고 모델이 양성(**Positive**)으로 예측한 경우

**False Positive (FP):** 실제 음성(**False**)인데 모델이 양성(**Positive**)으로 잘못 예측한 경우

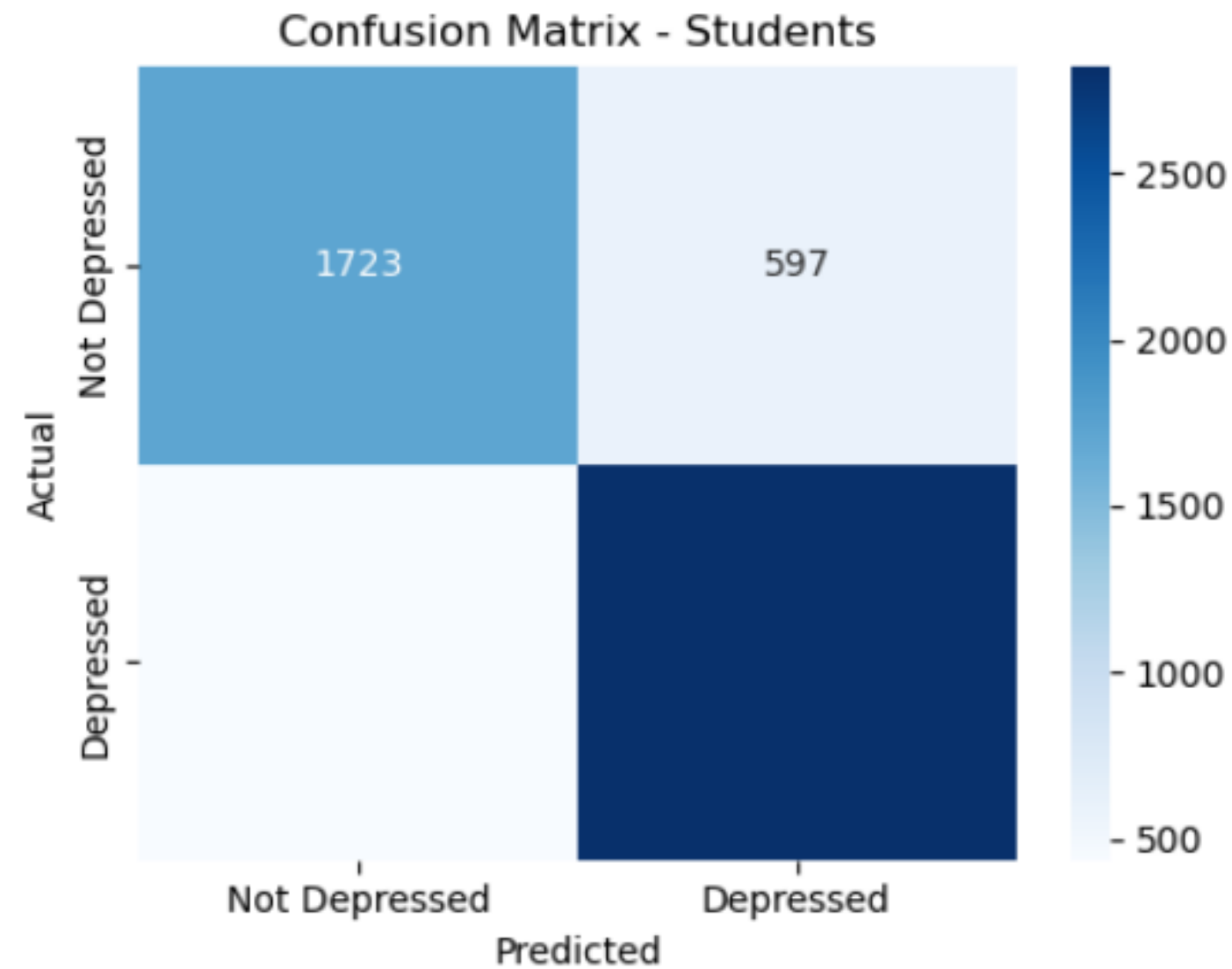
**True Negative (TN):** 실제 음성(**False**)이고 모델이 음성(**Negative**)으로 예측한 경우

**False Negative (FN):** 실제 양성(**True**)인데 모델이 음성(**Negative**)으로 잘못 예측한 경우

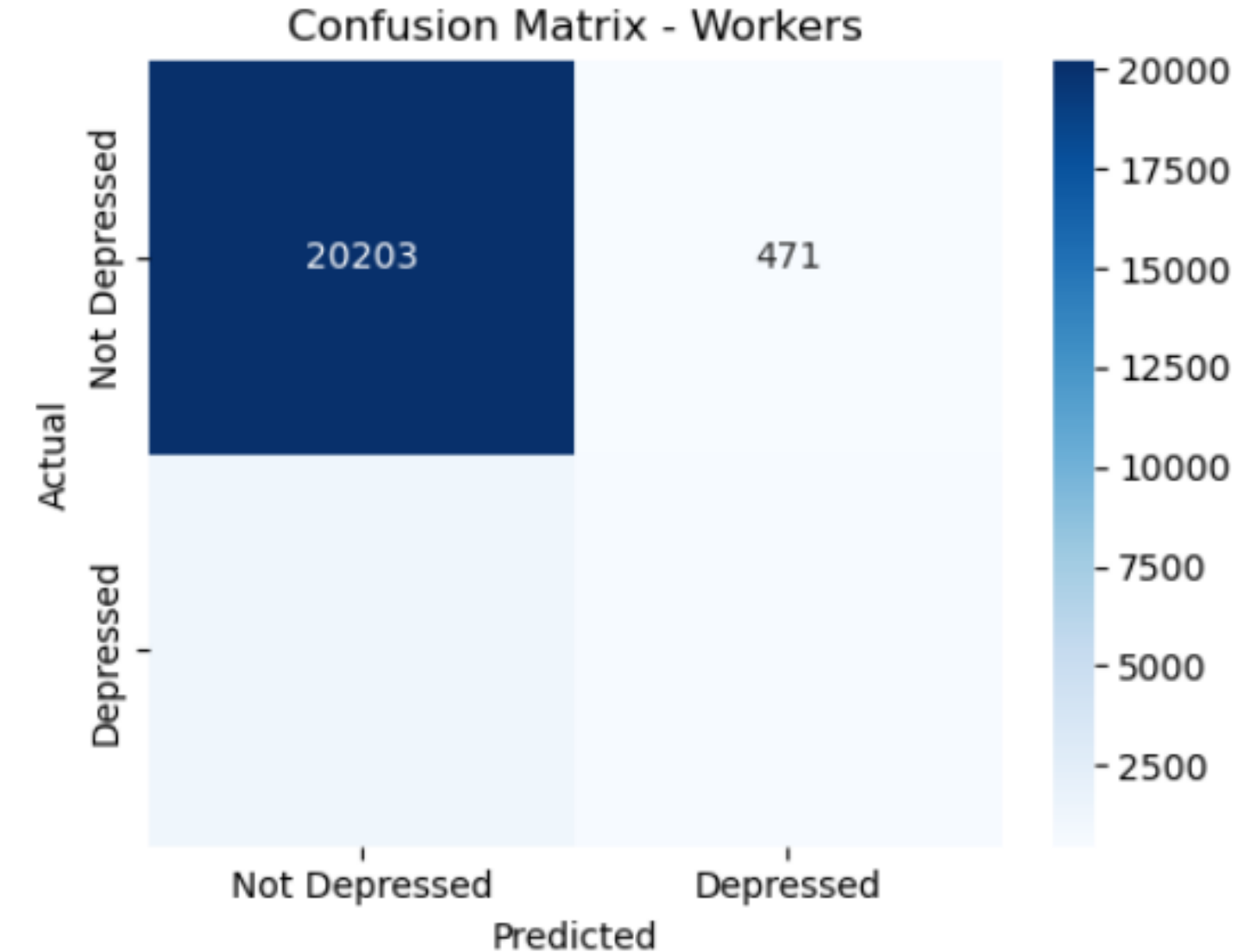


# KNN

## 혼동행렬 시각화



- 학생 혼동행렬



- 직장인 혼동행렬

**True Positive (TP)와 True Negative (TN)가 많으면 모델이 예측을 잘했다는 의미**  
**False Positive (FP)와 False Negative (FN)가 많으면 모델이 자주 실수를 한다는 의미**

# KNN

- 결과

\*학생

정확도: **81.45%**

재현율: **86.6%**

\*직장인

정확도: **92.31%**

재현율: **32.5%**

전체 정확도는 높지만, 직장인 재현율이 **32.5%**로 진짜 중요한 우울증 환자 (1번 클래스)를 놓치는 비율이 꽤 큼.

=실제 우울증 있는 사람 중 **32.5%**만 정확하게 예측함.

결과: 정확도는 높게 나오지만, 직장인의 재현율이 **0.325**로 매우 낮음 (부적합)

# 랜덤 포레스트

## ● 모델 학습과정 -학생

```
#학생
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, r2_score
from sklearn.preprocessing import LabelEncoder

# 데이터 불러오기
df = pd.read_csv("C:\Users\seulo\OneDrive\바탕 화면\KUGGLE\1차 프로젝트\df_new.csv")

# 인코딩할 범주형 변수
cols_to_encode = ['Sleep Duration', 'Dietary Habits',
                  'Have you ever had suicidal thoughts?',
                  'Family History of Mental Illness']

le = LabelEncoder()
for col in cols_to_encode:
    df[col] = le.fit_transform(df[col])
```

# 랜덤 포레스트

## ● 모델 학습과정 -학생

# 설명변수와 종속변수 지정

```
features = ['Academic Pressure', 'CGPA', 'Study Satisfaction', 'Sleep Duration',
            'Dietary Habits', 'Have you ever had suicidal thoughts ?',
            'Work/Study Hours', 'Financial Stress', 'Family History of Mental Illness']
```

```
target = 'Depression'
```

```
X = df[features]
```

```
y = df[target]
```

# 데이터 분할

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

# 모델 생성 및 학습

```
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
```

```
rf_model.fit(X_train, y_train)
```

# 랜덤 포레스트

## ● 학생 결과

```
36 # 예측 및 평가
37 y_pred_rf = rf_model.predict(X_test)
38 print(f'Accuracy: {accuracy_score(y_test, y_pred_rf)}')
39 print(f'R² Score: {r2_score(y_test, y_pred_rf)}')
40 print(classification_report(y_test, y_pred_rf))
41
```

Accuracy: 0.9019879796578826

R² Score: 0.333965242725933

	precision	recall	f1-score	support
0	0.92	0.97	0.94	23077
1	0.80	0.61	0.69	5042
accuracy			0.90	28119
macro avg	0.86	0.79	0.82	28119
weighted avg	0.90	0.90	0.90	28119

### 클래스 0 (Not Depressed)

정확도 (Precision): 0.92

재현율 (Recall): 0.97

F1-Score: 0.94

### 클래스 1 (Depressed)

정확도 (Precision): 0.80

재현율 (Recall): 0.61

F1-Score: 0.69

전체 정확도 0.90

재현율 0.61

# 랜덤 포레스트

## ● 모델 학습과정 -직장인

```

1 #직장인
2 import pandas as pd
3 from sklearn.model_selection import train_test_split
4 from sklearn.ensemble import RandomForestClassifier
5 from sklearn.metrics import classification_report, accuracy_score, r2_score
6 from sklearn.preprocessing import LabelEncoder
7
8 # 1. 데이터 불러오기
9 df = pd.read_csv("C:\Users\seul0\OneDrive\바탕 화면\KUGGLE\1차 프로젝트\df_new.csv")
10
11 # 2. 인코딩할 범주형 변수들
12 cols_to_encode = ['Sleep Duration', 'Dietary Habits',
13                  'Have you ever had suicidal thoughts ?',
14                  'Family History of Mental Illness']
15 le = LabelEncoder()
16 for col in cols_to_encode:
17     df[col] = le.fit_transform(df[col])
18

```

# 랜덤 포레스트

- 모델 학습과정 -직장인

# 3. 새로운 설명변수 조합 설정

```
features = ['Work Pressure', 'Job Satisfaction', 'Sleep Duration', 'Dietary Habits',
            'Have you ever had suicidal thoughts ?', 'Work/Study Hours',
            'Financial Stress', 'Family History of Mental Illness']
target = 'Depression'
```

```
X = df[features]
y = df[target]
```

# 4. 데이터 분할

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

# 5. 모델 학습

```
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

# 랜덤 포레스트

## 직장인 결과

```
35 # 6. 예측 및 평가
36 y_pred = rf_model.predict(X_test)
37
38 print("✅ [새로운 변수 조합 결과]")
39 print("Accuracy:", accuracy_score(y_test, y_pred))
40 print("R² Score:", r2_score(y_test, y_pred))
41 print(classification_report(y_test, y_pred))
42
```

✅ [새로운 변수 조합 결과]

Accuracy: 0.8889007432696753

R² Score: 0.24503171925827816

	precision	recall	f1-score	support
0	0.93	0.94	0.93	23077
1	0.70	0.67	0.68	5042
accuracy			0.89	28119
macro avg	0.81	0.80	0.81	28119
weighted avg	0.89	0.89	0.89	28119

클래스 0 (Not Depressed)

정확도 (Precision): 0.93

재현율 (Recall): 0.94

F1-Score: 0.93

클래스 1 (Depressed)

정확도 (Precision): 0.70

재현율 (Recall): 0.67

F1-Score: 0.68

전체 정확도 88.89%

재현율 67%



## 랜덤 포레스트

### ● 결론

\*학생

정확도: **0.90**

재현율: **0.61**

\*직장인

정확도 **88.89%**

재현율 **67%**

학생 재현율 **61%**로 상대적으로 낮은 편.

직장인 재현율 **67%**로 낮은 편이지만, 로지스틱, **KNN**보다 두 배 가까이 높음.

=> 하이퍼파라미터 튜닝 통해 재현율 높이기로 결정

# 랜덤 포레스트

## ● 하이퍼파라미터 튜닝 전 다중공선성 검사 - 학생

```
# 하이퍼파라미터 튜닝 전, 학생 데이터의 VIF(분산팽창계수) 분석
import pandas as pd
from statsmodels.stats.outliers_influence import variance_inflation_factor
import numpy as np

# 설명변수 데이터 준비
X = df[features].copy()

# 결측치와 무한값 처리
X.replace([np.inf, -np.inf], np.nan, inplace=True)
X.dropna(inplace=True) # 결측치가 있는 행 제거

# 상수항 추가
X_const = X.copy()
X_const['Intercept'] = 1

# VIF 계산
vif_data = pd.DataFrame()
vif_data["Feature"] = X_const.columns
vif_data["VIF"] = [variance_inflation_factor(X_const.values, i) for i in range(X_const.shape[1])]
```

# 랜덤 포레스트

## • 하이퍼파라미터 튜닝 전 다중공선성 검사 - 직장인

```
# 하이퍼파라미터 튜닝 전, 직장인 데이터의 VIF(분산팽창계수) 분석
from statsmodels.stats.outliers_influence import variance_inflation_factor
import numpy as np

# 설명변수(X) 복사
X_vif = X.copy()

# 결측치 및 무한값 처리
X_vif.replace([np.inf, -np.inf], np.nan, inplace=True)
X_vif.dropna(inplace=True)

# 상수항 추가
X_vif['Intercept'] = 1

# VIF 계산
vif_data = pd.DataFrame()
vif_data["Feature"] = X_vif.columns
vif_data["VIF"] = [variance_inflation_factor(X_vif.values, i) for i in range(X_vif.shape[1])]
```

# 랜덤 포레스트

## ● 학생 다중공선성

```
22 # 결과 출력
23 print(vif_data)
24
25
```

	Feature	VIF
	Academic Pressure	1.103055
	CGPA	1.003796
	Study Satisfaction	1.020419
	Sleep Duration	1.002437
	Dietary Habits	1.020759
	Have you ever had suicidal thoughts ?	1.128236
	Work/Study Hours	1.023549
	Financial Stress	1.064541
	Family History of Mental Illness	1.001436
	Intercept	52.392412

대부분 **VIF**가 1~1.5 사이  
→ 다중공선성 거의 없음 (좋음)

## ● 직장인 다중공선성

```
20 # 결과 출력
21 print("📊 [직장인 모델 - VIF 분석 결과]")
22 print(vif_data)
23
```

📊 [직장인 모델 - VIF 분석 결과]

	Feature	VIF
0	Work Pressure	1.004443
1	Job Satisfaction	1.006365
2	Sleep Duration	1.000231
3	Dietary Habits	1.004210
4	Have you ever had suicidal thoughts ?	1.010375
5	Work/Study Hours	1.002694
6	Financial Stress	1.004717
7	Family History of Mental Illness	1.000433
8	Intercept	22.167364

대부분 **VIF**가 1~1.5 사이  
→ 다중공선성 거의 없음 (좋음)

# 랜덤 포레스트

## • 하이퍼파라미터 튜닝 통해 재현율 개선 - 학생

```
#하이퍼파라미터 튜닝 후, 재현율 개선 (학생)
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score, r2_score
from sklearn.preprocessing import LabelEncoder

# 1. 데이터 불러오기
df = pd.read_csv("C:\\Users\\seulo\\OneDrive\\바탕 화면\\KUGGLE\\1차 프로젝트\\df_new.csv")

# 2. Label Encoding
cols_to_encode = ['Sleep Duration', 'Dietary Habits',
                  'Have you ever had suicidal thoughts?',
                  'Family History of Mental Illness']
le = LabelEncoder()
for col in cols_to_encode:
    df[col] = le.fit_transform(df[col])

# 3. 변수 설정
features = ['Academic Pressure', 'CGPA', 'Study Satisfaction', 'Sleep Duration',
            'Dietary Habits', 'Have you ever had suicidal thoughts?',
            'Work/Study Hours', 'Financial Stress', 'Family History of Mental Illness']
target = 'Depression'
X = df[features]
y = df[target]

# 4. 훈련/테스트 데이터 분할
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

데이터를 불러오고,  
범주형 데이터를 숫자로 변환한 후,  
훈련과 테스트 데이터를 분리

# 랜덤 포레스트

- 하이퍼파라미터 튜닝 통해 재현율 개선 - 학생

```
# 5. 하이퍼파라미터 후보 설정
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5, 10],
    'class_weight': ['balanced']
}

# 6. GridSearchCV 수행 (Recall 기준으로)
grid_search = GridSearchCV(RandomForestClassifier(random_state=42),
                           param_grid,
                           scoring='recall',
                           cv=3,
                           n_jobs=-1,
                           verbose=2)

grid_search.fit(X_train, y_train)

# 7. 최적 모델로 예측
best_model_student = grid_search.best_estimator_
y_pred_best = best_model_student.predict(X_test)

# 8. 결과 출력
print("Best Parameters:", grid_search.best_params_)
print("Accuracy:", accuracy_score(y_test, y_pred_best))
print("R² Score:", r2_score(y_test, y_pred_best))
print(classification_report(y_test, y_pred_best))
```

하이퍼파라미터 튜닝을 통해  
최적의 모델을 찾고,  
그 모델을 사용해 테스트 데이터를 예측한 후  
성능을 평가

# 랜덤 포레스트

## ● 학생 결과

Fitting 3 folds for each of 18 candidates, totalling 54 fits

Best Parameters: {'class\_weight': 'balanced', 'max\_depth': 10, 'min\_samples\_split': 10, 'n\_estimators': 200}

Accuracy: 0.8630463387744941

R<sup>2</sup> Score: 0.06933967697299259

	precision	recall	f1-score	support
0	0.96	0.87	0.91	23077
1	0.58	0.81	0.68	5042
accuracy			0.86	28119
macro avg	0.77	0.84	0.80	28119
weighted avg	0.89	0.86	0.87	28119

전체 정확도 (Accuracy): 86.3%

클래스 1 재현율 : 81%

튜닝 전 '정확도: 0.90, 재현율: 0.61' 에 비해  
재현율이 20%나 올라간 것을 알 수 있음.

# 랜덤 포레스트

## • 하이퍼파라미터 튜닝 후 재현율 개선 - 직장인

```
#하이퍼파라미터 튜닝 후, 재현율 개선(직장인)
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score, r2_score
from sklearn.preprocessing import LabelEncoder

# 1. 데이터 불러오기
df = pd.read_csv("C:\\Users\\seulo\\OneDrive\\바탕 화면\\KUGGLE\\1차 프로젝트\\df_new.csv")

# 2. 범주형 변수 라벨 인코딩
cols_to_encode = ['Sleep Duration', 'Dietary Habits',
                  'Have you ever had suicidal thoughts?',
                  'Family History of Mental Illness']
le = LabelEncoder()
for col in cols_to_encode:
    df[col] = le.fit_transform(df[col])

# 3. 직장인용 설명변수 설정
features = ['Work Pressure', 'Job Satisfaction', 'Sleep Duration', 'Dietary Habits',
            'Have you ever had suicidal thoughts?', 'Work/Study Hours',
            'Financial Stress', 'Family History of Mental Illness']
target = 'Depression'

X = df[features]
y = df[target]

# 4. 학습/테스트 데이터 분할
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

데이터를 모델에 적합한 형태로 변환하고,  
학습 및 테스트 데이터셋을 분할하여  
이후 모델링을 위한 준비



# 랜덤 포레스트

- 하이퍼파라미터 튜닝 후 재현율 개선 - 직장인

```
# 5. 하이퍼파라미터 후보
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5],
    'class_weight': ['balanced']
}

# 6. GridSearchCV 수행 (재현율 기준)
grid_search = GridSearchCV(RandomForestClassifier(random_state=42),
                           param_grid,
                           scoring='recall',
                           cv=3,
                           n_jobs=-1,
                           verbose=2)

grid_search.fit(X_train, y_train)

# 7. 최적 모델 선택
best_model_worker = grid_search.best_estimator_

# 8. 평가
y_pred_best = best_model_worker.predict(X_test)

print("✅ [튜닝된 직장인 모델 결과]")
print("Best Parameters:", grid_search.best_params_)
print("Accuracy:", accuracy_score(y_test, y_pred_best))
print("R² Score:", r2_score(y_test, y_pred_best))
print(classification_report(y_test, y_pred_best))
```

하이퍼파라미터 튜닝을 통해 최적의 모델을 찾고,  
재현율을 개선하는 모델을 선택한 뒤,  
테스트 데이터에서 모델을 평가

# 랜덤 포레스트

## ● 직장인 결과

Fitting 3 folds for each of 12 candidates, totalling 36 fits

✓ [튜닝된 직장인 모델 결과]

Best Parameters: {'class\_weight': 'balanced', 'max\_depth': 10, 'min\_samples\_split': 5, 'n\_estimators': 100}

Accuracy: 0.8717593086525125

R<sup>2</sup> Score: 0.1285481368903172

	precision	recall	f1-score	support
0	0.96	0.88	0.92	23077
1	0.60	0.83	0.70	5042
accuracy			0.87	28119
macro avg	0.78	0.86	0.81	28119
weighted avg	0.90	0.87	0.88	28119

전체 정확도: **87.18%**

클래스1 재현율 (Recall): **83%**

튜닝 전 '정확도 **88.89%**, 재현율 **67%**'에 비해  
재현율이 **16%**나 올라간 것을 알 수 있음.

# 랜덤 포레스트

## ● 교차검증(3-fold)-학생

```

1 # 교차검증을 통한 일반화 성능 분석 (학생) (3-fold)
2 from sklearn.model_selection import cross_val_score
3
4 # 9. 3-Fold 교차검증으로 Recall 평가
5 cv_scores = cross_val_score(
6     best_model_student, # GridSearchCV를 통해 얻은 최적 모델
7     X, y,                # 전체 데이터셋을 대상으로
8     cv=3,               # 3-Fold 교차검증
9     scoring='recall',
10    n_jobs=-1
11 )
12
13 print("=== 3-Fold CV Recall ===")
14 print(f"Mean Recall: {cv_scores.mean():.4f}")
15 print(f"Std Recall: {cv_scores.std():.4f}")

```

```

=== 3-Fold CV Recall ===
Mean Recall: 0.8111
Std Recall: 0.0031

```

### Mean Recall: 0.8111

모델의 평균 재현율. 3번의 교차검증을 통해 모델이 전체 데이터에 대해 평균적으로 81.11%의 재현율을 기록했다는 의미

### Std Recall: 0.0031

재현율의 표준편차. 표준편차가 작기 때문에, 모델이 여러 번 평가해도 재현율 값의 변동이 적다는 것을 의미. 안정적인 성능을 보여줌.

# 랜덤 포레스트

- 교차검증(5-fold)-학생

1 # 교차검증을 통한 일반화 성능 분석 (학생) (5-fold)

```
1 from sklearn.model_selection import cross_val_score
2
3 # 9. 5-Fold 교차검증으로 Recall 평가
4 cv_scores = cross_val_score(
5     best_model_student, # GridSearchCV를 통해 얻은 최적 모델
6     X, y,               # 전체 데이터셋을 대상으로
7     cv=5,               # 5-Fold 교차검증
8     scoring='recall',
9     n_jobs=-1
10 )
11
12 print("=== 5-Fold CV Recall ===")
13 print(f"Mean Recall: {cv_scores.mean():.4f}")
14 print(f"Std Recall: {cv_scores.std():.4f}")
```

```
=== 5-Fold CV Recall ===
Mean Recall: 0.8139
Std Recall: 0.0057
```

**Mean Recall: 0.8139**

5번의 교차검증을 통해 모델이 전체 데이터에 대해 평균적으로 **81.39%**의 재현율을 기록했다는 의미.

**Std Recall: 0.0057**

표준편차가 매우 낮아 성능의 변동성이 적고 안정적인 모델임을 보여줌.

# 랜덤 포레스트

- 교차검증(3-fold)-직장인

```

1 # 교차검증을 통한 일반화 성능 분석 (직장인) (3-fold)
2 from sklearn.model_selection import cross_val_score
3
4 # 9. 3-Fold 교차검증으로 Recall 평가
5 cv_scores = cross_val_score(
6     best_model_worker, # GridSearchCV를 통해 얻은 최적 모델
7     X, y,               # 전체 데이터셋을 대상으로
8     cv=3,               # 3-Fold 교차검증
9     scoring='recall',
10    n_jobs=-1
11 )
12
13 print("=== 3-Fold CV Recall ===")
14 print(f"Mean Recall: {cv_scores.mean():.4f}")
15 print(f"Std Recall: {cv_scores.std():.4f}")

```

```

=== 3-Fold CV Recall ===
Mean Recall: 0.8357
Std Recall: 0.0022

```

**Mean Recall (평균 재현율): 0.8357**  
 전체적으로 재현율이 약 **83.57%**. (다소 높음)

**Std Recall (재현율 표준편차): 0.0022**  
 표준편차가 매우 낮음. 안정적.

# 랜덤 포레스트

## ● 교차검증(5-fold)-직장인

```

1 # 교차검증을 통한 일반화 성능 분석 (직장인) (5-fold)
2 from sklearn.model_selection import cross_val_score
3
4 # 9. 5-Fold 교차검증으로 Recall 평가
5 cv_scores = cross_val_score(
6     best_model_worker, # GridSearchCV를 통해 얻은 최적 모델
7     X, y,               # 전체 데이터셋을 대상으로
8     cv=5,               # 5-Fold 교차검증
9     scoring='recall',
10    n_jobs=-1
11 )
12
13 print("=== 5-Fold CV Recall ===")
14 print(f"Mean Recall: {cv_scores.mean():.4f}")
15 print(f"Std Recall: {cv_scores.std():.4f}")

```

평균 재현율 (Mean Recall): 0.8132

재현율 표준편차 (Std Recall): 0.0057

모델이 직장인 집단에서 우울증을 예측하는데 상당히 좋은 성능을 보였고, 성능의 일관성도 높다는 것을 의미

```

=== 5-Fold CV Recall ===
Mean Recall: 0.8132
Std Recall: 0.0057

```

# 랜덤 포레스트

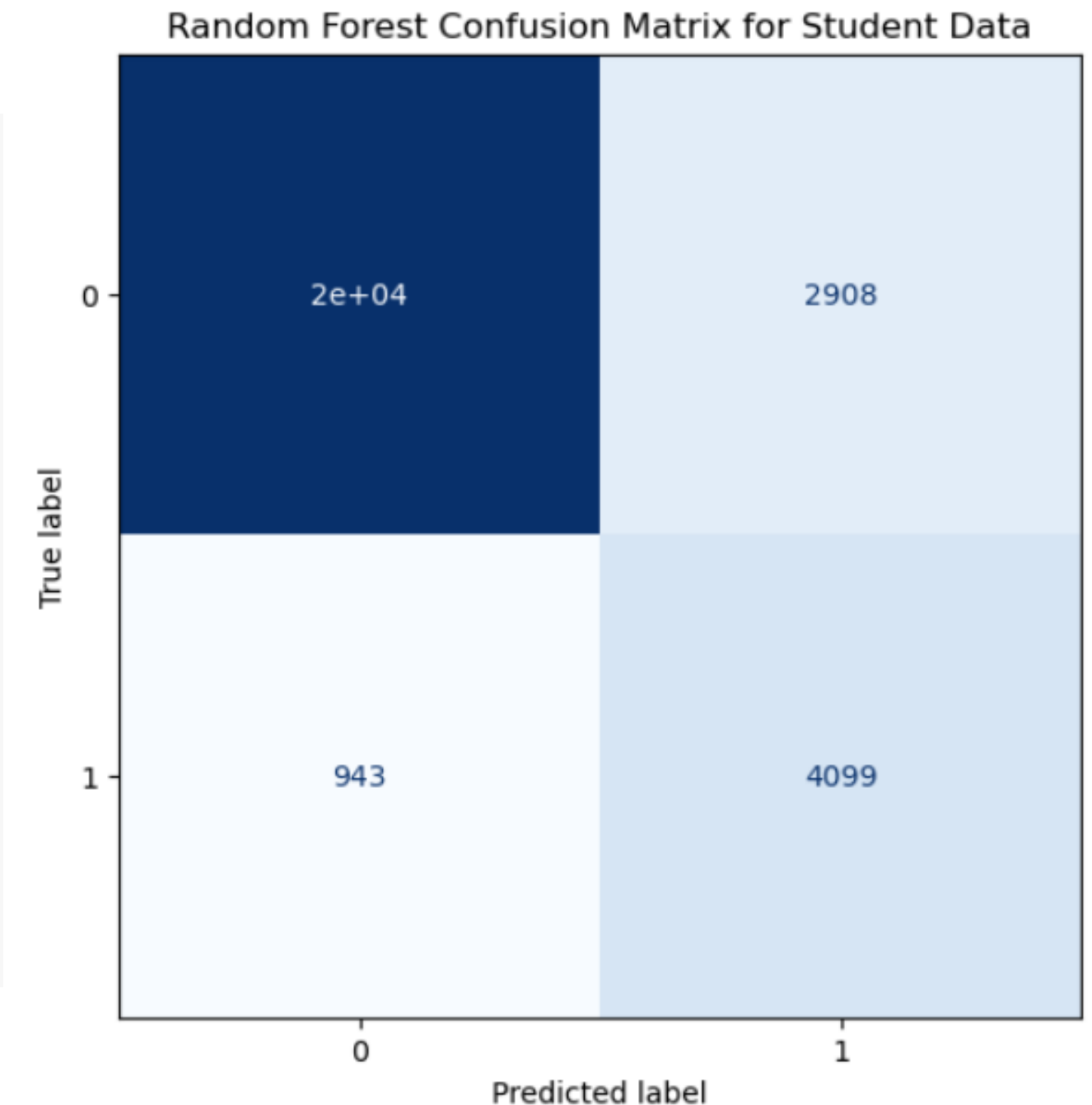
## ● 혼동행렬 학생

```
# Confusion Matrix 통한 우울 증 예측의 시각화 (학생)
# 훈련 데이터와 테스트 데이터의 특성 순서가 동일한지 확인
X_train = X_train[best_model_student.feature_names_in_]
X_test = X_test[best_model_student.feature_names_in_]

# 예측 수행
y_pred = best_model_student.predict(X_test)

# 혼동행렬 시각화
cm = confusion_matrix(y_test, y_pred, labels=best_model_student.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=best_model_student.classes_)

fig, ax = plt.subplots(figsize=(6, 6))
disp.plot(ax=ax, cmap='Blues', colorbar=False)
ax.set_title("Random Forest Confusion Matrix for Student Data")
plt.show()
```



# 랜덤 포레스트

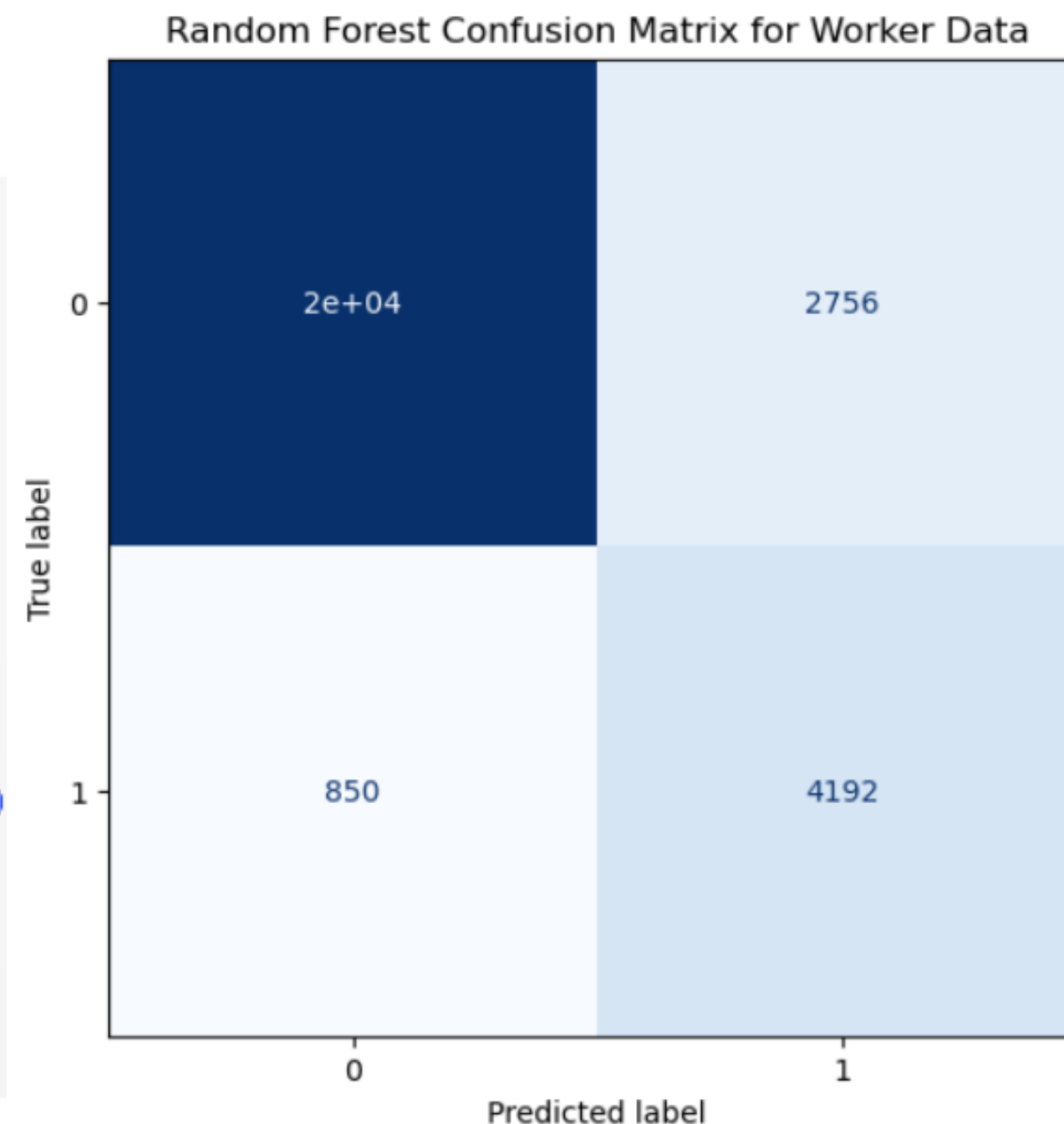
- 혼동행렬 직장인

```
# Confusion Matrix 통한 우울 증 예측의 시각화 (직장인)
# 훈련 데이터와 테스트 데이터의 특성 순서가 동일한지 확인
X_train = X_train[best_model_worker.feature_names_in_]
X_test = X_test[best_model_worker.feature_names_in_]

# 예측 수행
y_pred = best_model_worker.predict(X_test)

# 혼동행렬 시각화
cm = confusion_matrix(y_test, y_pred, labels=best_model_worker.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=best_model_worker.classes_)

fig, ax = plt.subplots(figsize=(6, 6))
disp.plot(ax=ax, cmap='Blues', colorbar=False)
ax.set_title("Random Forest Confusion Matrix for Worker Data")
plt.show()
```





# 학생 직장인에 따른 우울증 상관분석 및 예측 모델 구축

“

결론

”

# 마무리

감사합니다

