어프렌티스 프로젝트

프로젝트 중간평가

2023254006
이선경

CONTENS

01. 데이터 선정

1. 데이터 선정
- 신체 신호가 포함된 흡연 및 음주 데이터세트 _한국 국민건강보험공단

2. 목표
- 생체 인식으로 흡연 예측

3. 비즈니스 문제 이해
- 건강 관리 및 예방 : 흡연 관련된 생체 신호를 분석하여, 개별 개체의 건강 상태를 예측하고 이를 토대로 건강 상태 개선을 위한 조언을 제공하는 것이 목표

02. 데이터 이해 및 시각화

1. 데이터 이해

```
import pandas as pd
import matplotlib.pyplot as plt

# CSV 파일 불러오기
data = pd.read_csv('smoking_driking_dataset_Ver01.csv')

# 데이터 확인
print(data.head())
```

```
    sex  age  height  weight  waistline  sight_left  sight_right  hear_left  ₩
0  Male   35     170      75       90.0         1.0          1.0        1.0
1  Male   30     180      80       89.0         0.9          1.2        1.0
2  Male   40     165      75       91.0         1.2          1.5        1.0
3  Male   50     175      80       91.0         1.5          1.2        1.0
4  Male   50     165      60       80.0         1.0          1.2        1.0

   hear_right    SBP  ...  LDL_chole  triglyceride  hemoglobin  urine_protein  ₩
0         1.0  120.0  ...      126.0          92.0        17.1            1.0
1         1.0  130.0  ...      148.0         121.0        15.8            1.0
2         1.0  120.0  ...       74.0         104.0        15.8            1.0
3         1.0  145.0  ...      104.0         106.0        17.6            1.0
4         1.0  138.0  ...      117.0         104.0        13.8            1.0

   serum_creatinine  SGOT_AST  SGOT_ALT  gamma_GTP  SMK_stat_type_cd  DRK_YN
0               1.0      21.0      35.0       40.0               1.0       Y
1               0.9      20.0      36.0       27.0               3.0       N
2               0.9      47.0      32.0       68.0               1.0       N
3               1.1      29.0      34.0       18.0               1.0       N
4               0.8      19.0      12.0       25.0               1.0       N

[5 rows x 24 columns]
```

02. 데이터 이해 및 시각화

1. 데이터 이해

```
print(data.shape)
data.head(11).T
```

```
(991346, 24)
```

	0	1	2	3	4	5	6	7	8	9	10
sex	Male	Male	Male	Male	Male	Male	Female	Male	Male	Male	Male
age	35	30	40	50	50	50	45	35	55	40	45
height	170	180	165	175	165	165	150	175	170	175	155
weight	75	80	75	80	60	55	55	65	75	75	55
waistline	90.0	89.0	91.0	91.0	80.0	75.0	69.0	84.2	84.0	82.0	79.2
sight_left	1.0	0.9	1.2	1.5	1.0	1.2	0.5	1.2	1.2	1.5	1.0
sight_right	1.0	1.2	1.5	1.2	1.2	1.5	0.4	1.0	0.9	1.5	1.0
hear_left	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
hear_right	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
SBP	120.0	130.0	120.0	145.0	138.0	142.0	101.0	132.0	145.0	132.0	118.0
DBP	80.0	82.0	70.0	87.0	82.0	92.0	58.0	80.0	85.0	105.0	70.0
BLDS	99.0	106.0	98.0	95.0	101.0	99.0	89.0	94.0	104.0	100.0	90.0
tot_chole	193.0	228.0	136.0	201.0	199.0	218.0	196.0	185.0	217.0	195.0	183.0
HDL_chole	48.0	55.0	41.0	76.0	61.0	77.0	66.0	58.0	56.0	60.0	42.0
LDL_chole	126.0	148.0	74.0	104.0	117.0	95.0	115.0	107.0	141.0	118.0	130.0
triglyceride	92.0	121.0	104.0	106.0	104.0	232.0	75.0	101.0	100.0	83.0	55.0
hemoglobin	17.1	15.8	15.8	17.6	13.8	13.8	12.3	14.4	15.1	13.9	12.9
urine_protein	1.0	1.0	1.0	1.0	1.0	3.0	1.0	1.0	1.0	1.0	1.0
serum_creatinine	1.0	0.9	0.9	1.1	0.8	0.8	0.8	0.8	0.8	0.9	0.8
SGOT_AST	21.0	20.0	47.0	29.0	19.0	29.0	19.0	18.0	32.0	21.0	19.0
SGOT_ALT	35.0	36.0	32.0	34.0	12.0	40.0	12.0	18.0	23.0	38.0	14.0
gamma_GTP	40.0	27.0	68.0	18.0	25.0	37.0	12.0	35.0	26.0	16.0	19.0
SMK_stat_type_cd	1.0	3.0	1.0	1.0	1.0	3.0	1.0	3.0	1.0	2.0	1.0
DRK_YN	Y	N	N	N	N	Y	N	Y	Y	Y	N

sex : 성별 **age** : 나이
height : 키 **weight** : 몸무게
waistline : 허리 둘레 **sight**_left : eyesight(left)
hear_left : hearing left, 1(normal), 2(abnormal) **SBP** : 수축기 혈압
DBP : 이완기 혈압 **BLDS** : 공복혈당
tot_chole : 총 콜레스테롤 **HDL_chole** : HDL 콜레스테롤
LDL_chole : LDL 콜레스테롤 **triglyceride** : 중성지방
hemoglobin : 헤모글로빈 (g/dL)
urine_protein : 1(-), 2(+/-), 3(+1), 4(+2), 5(+3), 6(+4) 요단백
serum_creatinine : 혈청 크레아티닌 **SGOT_AST** : 혈청지오티 AST
SGOT_ALT : 혈청지오티 ALT **gamma_GTP** : 감마지티피
SMK_stat_type_cd : 흡연상태, 1(never), 2(used to smoke but quit), 3(still smoke)
DRK_YN : 음주여부

02. 데이터 이해 및 시각화

1. 데이터 이해

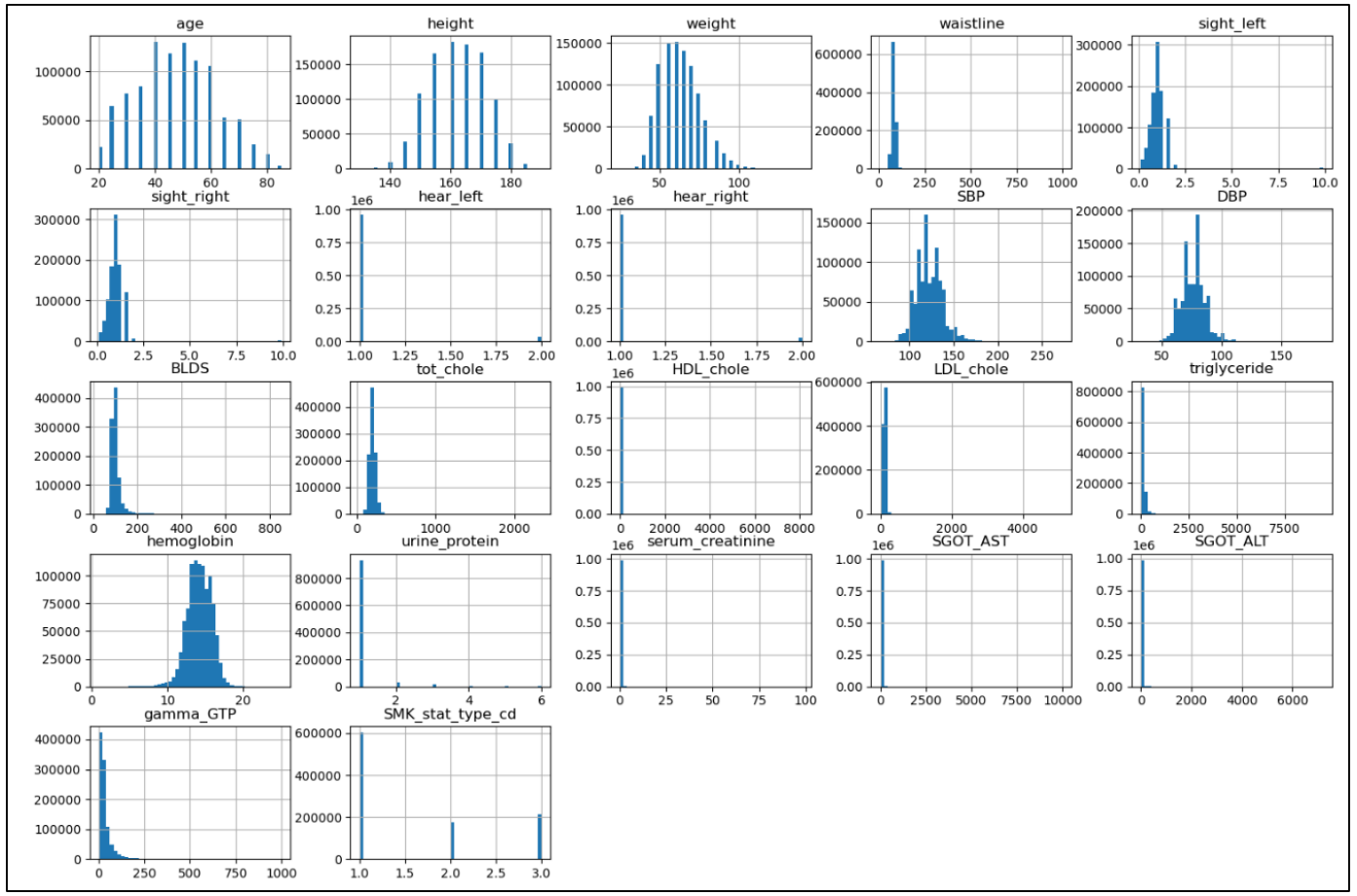
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 991346 entries, 0 to 991345
Data columns (total 24 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   sex               991346 non-null  object
 1   age               991346 non-null  int64
 2   height            991346 non-null  int64
 3   weight            991346 non-null  int64
 4   waistline         991346 non-null  float64
 5   sight_left        991346 non-null  float64
 6   sight_right       991346 non-null  float64
 7   hear_left         991346 non-null  float64
 8   hear_right        991346 non-null  float64
 9   SBP               991346 non-null  float64
 10  DBP               991346 non-null  float64
 11  BLDS              991346 non-null  float64
 12  tot_chole         991346 non-null  float64
 13  HDL_chole         991346 non-null  float64
 14  LDL_chole         991346 non-null  float64
 15  triglyceride      991346 non-null  float64
 16  hemoglobin        991346 non-null  float64
 17  urine_protein     991346 non-null  float64
 18  serum_creatinine  991346 non-null  float64
 19  SGOT_AST          991346 non-null  float64
 20  SGOT_ALT          991346 non-null  float64
 21  gamma_GTP         991346 non-null  float64
 22  SMK_stat_type_cd  991346 non-null  float64
 23  DRK_YN            991346 non-null  object
dtypes: float64(19), int64(3), object(2)
memory usage: 181.5+ MB
```

02. 데이터 이해 및 시각화

2. 데이터 시각화

```
data.hist(bins=50, figsize=(18, 12))
```



02. 데이터 이해 및 시각화

2. 데이터 시각화

```
correlation_matrix = data.corr()
correlation_with_smoking = correlation_matrix['SMK_stat_type_cd']
correlation_df = pd.DataFrame({'Correlation with SMK_stat_type_cd': correlation_with_smoking})
print(correlation_df)
```

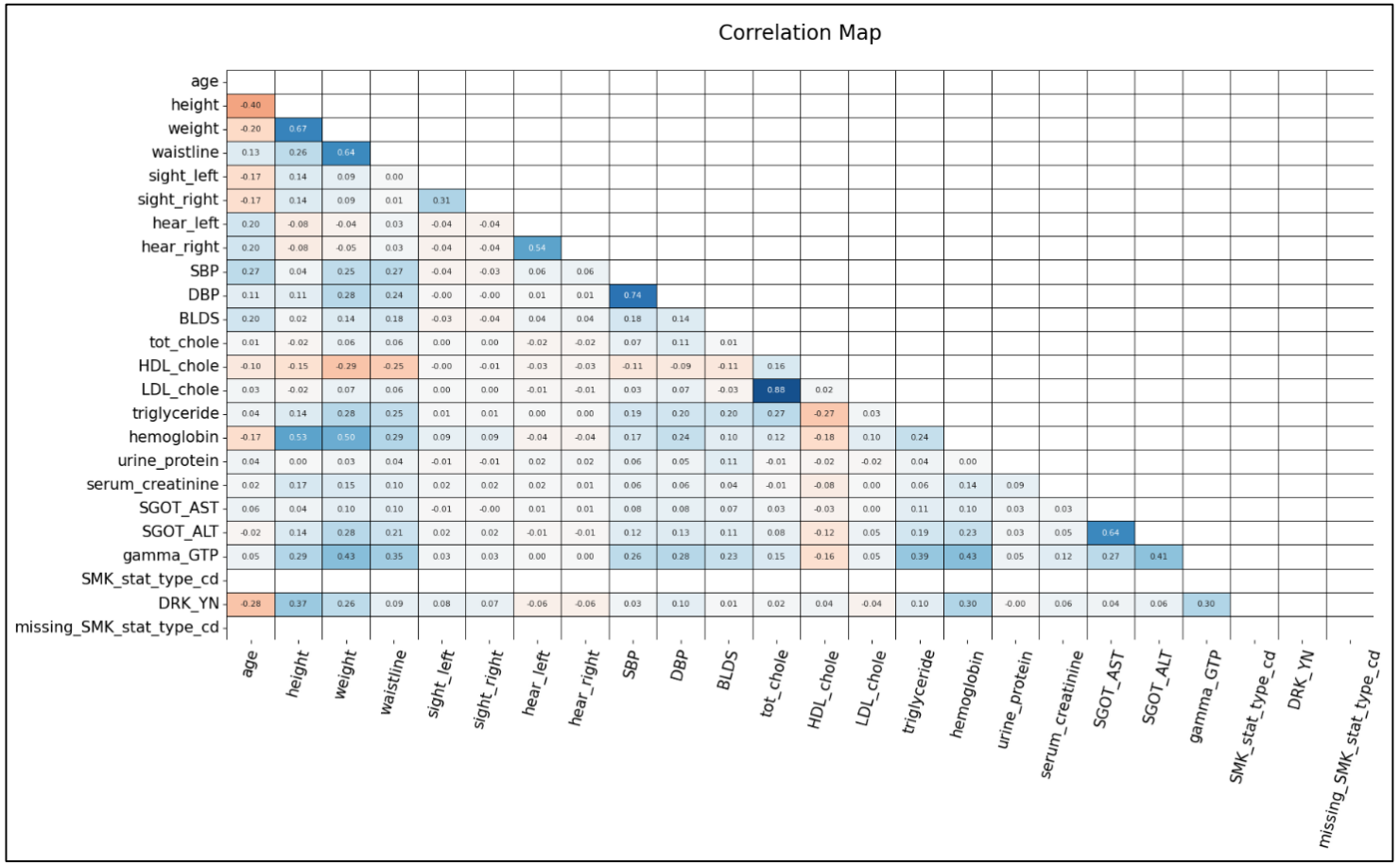
	Correlation with SMK_stat_type_cd
age	-0.126031
height	0.467317
weight	0.366499
waistline	0.205485
sight_left	0.064194
sight_right	0.065051
hear_left	-0.021245
hear_right	-0.022330
SBP	0.084918
DBP	0.126849
BLDS	0.086490
tot_chole	0.011833
HDL_chole	-0.161031
LDL_chole	-0.012315
triglyceride	0.216026
hemoglobin	0.453385
urine_protein	0.015263
serum_creatinine	0.129679
SGOT_AST	0.062437
SGOT_ALT	0.131569
gamma_GTP	0.243576
SMK_stat_type_cd	1.000000

02. 데이터 이해 및 시각화

2. 데이터 시각화

```
import seaborn as sns
def corr_map(data, width=14, height=6, annot_kws=15):
    mtx = np.triu(data.corr())
    f, ax = plt.subplots(figsize = (width,height))
    sns.heatmap(data.corr(),
                annot= True,
                fmt = ".2f",
                ax=ax,
                vmin = -1,
                vmax = 1,
                cmap = "RdBu",
                mask = mtx,
                linewidth = 0.4,
                linecolor = "black",
                cbar=False,
                annot_kws={"size": annot_kws})
    plt.yticks(rotation=0,size=15)
    plt.xticks(rotation=75,size=15)
    plt.title('\nCorrelation Map\n', size = 20)
    plt.show();

corr_map(data, width=20, height=10, annot_kws=8)
```



03. 데이터 전처리

1. 결측치 확인

```
#결측치 없는지 확인
missing_values = data.isnull().sum()

if missing_values.sum() == 0:
    print("결측치 없음")
else :
    print("결측치 있음")
```

```
결측치 없음
```

결측치 없음

2. 임의 결측치

```
column_with_missing_values = 'SMK_stat_type_cd'
sampled_indices = data[column_with_missing_values].sample(frac=0.1, random_state=42).index

# 해당 행의 값을 결측치로 설정
data.loc[sampled_indices, 'missing_' + column_with_missing_values] = np.nan
data.head()
```

SMK_stat_type_cd	DRK_YN	missing_SMK_stat_type_cd
1.0	1	NaN
3.0	0	NaN
1.0	0	NaN
1.0	0	NaN
1.0	0	NaN

04. 텍스트, 범주형 특성

1. OneHotEncoder

```
from sklearn.preprocessing import OneHotEncoder

# 'DRK_YN' 'Y' 1로, 'N' 0
data['DRK_YN'] = data['DRK_YN'].map({'Y': 1, 'N': 0})

# OneHotEncoder 객체 생성
encoder = OneHotEncoder(sparse=False)  # sparse=False로 설정하여 밀집 배열로 얻음

# "DRK_YN" 열을 2D 배열로 변환
DRK_YN_encoded = encoder.fit_transform(data[['DRK_YN']].values.reshape(-1, 1))

print(DRK_YN_encoded)
```

```
[[0. 1.]
 [1. 0.]
 [1. 0.]
 ...
 [0. 1.]
 [1. 0.]
 [0. 1.]]
```

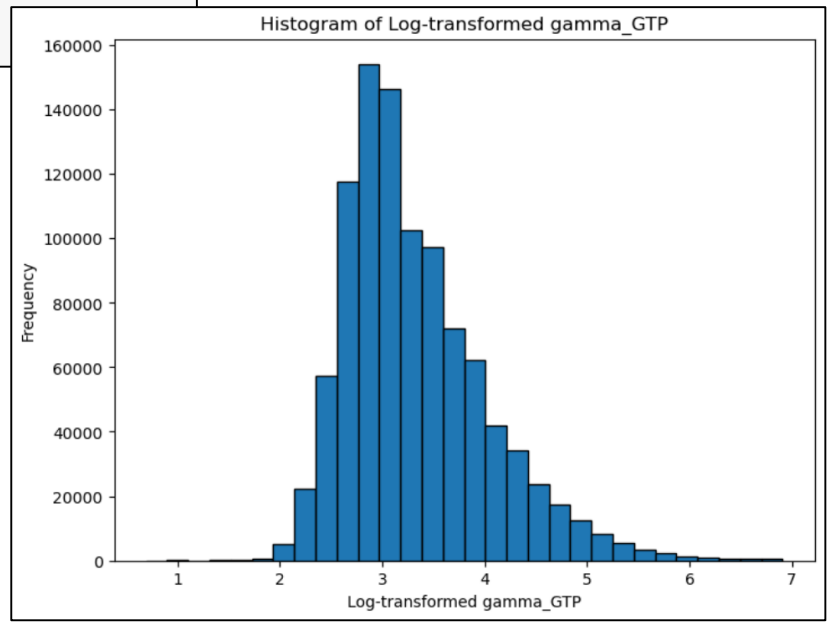
1) 범주형 변수를 독립적인 이진 변수로 변환
2) 순서 정보 고려 X

05. 특성 스케일링 및 변환

1. 특성 스케일링

```
# gamma_GTP 열에 로그 변환 적용
data['gamma_GTP'] = np.log1p(data['gamma_GTP'])

plt.figure(figsize=(8, 6))
data['gamma_GTP'].plot(kind='hist', bins=30, edgecolor='k')
plt.title('Histogram of Log-transformed gamma_GTP')
plt.xlabel('Log-transformed gamma_GTP')
plt.ylabel('Frequency')
plt.show()
```



6. 사용자 정의 변환기

1. 사용자 정의 변환기

```
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer

class SmokingPredictionTransformer(BaseEstimator, TransformerMixin):
    def __init__(self):
        self.preprocessor = ColumnTransformer(
            transformers=[
                ('num', StandardScaler(), ['age', 'height', 'weight', 'waistline', 'SBP', 'DBP',
                                           'BLDS', 'tot_chole', 'HDL_chole', 'LDL_chole',
                                           'triglyceride', 'hemoglobin', 'serum_creatinine',
                                           'SGOT_AST', 'SGOT_ALT', 'gamma_GTP']),
                ('cat', OneHotEncoder(), ['sex', 'urine_protein'])
            ])

    def fit(self, X, y=None):
        self.preprocessor.fit(X)
        return self

    def transform(self, X):
        X_transformed = self.preprocessor.transform(X)
        return X_transformed
```

07. 변환 파이프라인

1. 변환 파이프라인 구성

```
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier

preprocessing_pipeline = Pipeline([
    ('preprocess', SmokingPredictionTransformer()),
])

model_pipeline = Pipeline([
    ('model', RandomForestClassifier()),
])
```

사용자 정의 변환기, 모델 포함한 변환 파이프라인 구성

08. 모델 선택, 훈련

1. 모델 선택, 훈련

```python
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report

# 특성(X), 목표 변수(y)
X = data.drop(columns=['SMK_stat_type_cd'])
y = data['SMK_stat_type_cd']

# 데이터를 훈련용, 테스트용 세트로 분류
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 전처리 파이프라인을 훈련 데이터에 적용
preprocessing_pipeline.fit(X_train)

# 전처리 파이프라인을 사용하여 훈련 데이터와 테스트 데이터를 변환
X_train_transformed = preprocessing_pipeline.transform(X_train)
X_test_transformed = preprocessing_pipeline.transform(X_test)

# 기계 학습 모델 (RandomForestClassifier)을 생성, 훈련
model = RandomForestClassifier()
model.fit(X_train_transformed, y_train)

# 테스트 세트에 대한 예측 수행
y_pred = model.predict(X_test_transformed)

# 모델 평가
accuracy = accuracy_score(y_test, y_pred)# 정확도 계산
classification_rep = classification_report(y_test, y_pred)# 분류 보고서

print("Accuracy:", accuracy)
print("Classification Report:₩n", classification_rep)
```

```
Accuracy: 0.6932213648055682
Classification Report:
              precision    recall  f1-score   support

         1.0       0.83      0.83      0.83    120582
         2.0       0.43      0.38      0.40     34919
         3.0       0.51      0.58      0.54     42769

    accuracy                           0.69    198270
   macro avg       0.59      0.59      0.59    198270
weighted avg       0.69      0.69      0.69    198270
```

다중 클래스 분류 – RandomForestClassifier 사용

07. 모델 세부튜닝

1. 모델 세부튜닝

```
from sklearn.model_selection import GridSearchCV

# 모델과 조정할 하이퍼파라미터 그리드 정의
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt'],
    'bootstrap': [True, False]
}

# 그리드 서치 객체 생성
grid_search = GridSearchCV(RandomForestClassifier(), param_grid, cv=5, n_jobs=-1, verbose=2)

# 그리드 서치를 사용하여 모델 훈련
grid_search.fit(X_train_transformed, y_train)

# 최적의 모델 및 하이퍼파라미터 출력
best_model = grid_search.best_estimator_
best_params = grid_search.best_params_
print("최적의 모델:", best_model)
print("최적의 하이퍼파라미터:", best_params)

# 최적의 모델로 예측 및 평가
y_pred = best_model.predict(X_test_transformed)
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
print("최적 모델의 정확도:", accuracy)
print("최적 모델의 분류 보고서:₩n", classification_rep)
```

```
Fitting 5 folds for each of 432 candidates, totalling 2160 fits
```