심장마비 분석 및 예측

인공지능

컴퓨터공학과

2018108273 이재찬

목차

개요

HOME > 의원·병원

'한국형 의료 빅데이터'로 심장마비 예측

이정환 기자 leejh91@doctorsnews.co.kr | 승인 2015.10.01 16:53 | 댓글 0

수백만 임상·유전체 데이터 분석...만성질환 관리부터 암 치료까지
서울아산병원, 한국전자통신연구원과 헬스케어 빅데이터 분석 기술 개발

고혈압을 앓고 있는 50대 직장인 김 모씨. 아침에 일어나 스마트폰의 지문 인식 센서를 통해 맥박과 혈압을 측정한다. 측정 결과가 기존 건강 데이터 등이 연동된 헬스케어 빅 데이터 플랫폼을 통해 분석된다. 스마트폰에 경고 메시지가 뜬다. 급성 심근경색의 위험성이 매우 높다는 것. 김 모씨는 바로 병원을 찾아 정밀검사를 받고 혈관조영술을 시행해 이미 심장 혈관이 반쯤 막혀 언제 일어날지 모르는 심장 마비의 가능성을 찾게 됐다.

거대 정보로부터 가치를 창출하고 결과를 분석하는 기술, 이른바 빅데이터를 활용한다면 이처럼 심장 마비를 예측하는 의료 서비스가 가능해진다. 수많은 사람들의 임상, 유전체, 일상생활 등의 건강 데이터를 분석 및 연계해 개인의 미래 건강에 대해서도 가장 적합한 솔루션을 제공할 수 있는 것.

서울아산병원은 최근 한국전자통신연구원(ETRI, 원장 김흥남)과 헬스케어 빅데이터 분석 플랫폼 기술개발을 위한 공동 연구에 합의했다.

이른바 한국형 '왓슨 컴퓨터'인 의료 빅데이터 분석 플랫폼을 개발해 수백만 의료 데이터를 개개인에게 적용 가능하게 하는 맞춤형 통합 의료 서비스를 시행한다는 계획.

공동 연구팀은 한국형 빅데이터 분석 플랫폼을 통해 개인 건강의 지속적 관리는 물론 암, 심혈관질환 등 중증질환에 대한 예측 및 최적의 진단·치료 가이드를 제시하는 차별화된 맞춤형 보건의료 시스템을 마련한다.

특히 이번 공동 연구는 기존 빅데이터를 바탕으로 의사가 작성한 진료 기록을 분석, 최적의 치료법을 제안하는 미국 '왓슨 컴퓨터'의 임상 의사결정 지원시스템과는 또 다른 서비스를 제시하는 데 그 목표를 둔다고 밝혔다.

수백만 건강정보를 통합 분석한 의료 빅데이터에 특정 대상자에 대한 임상 기록과 함께 유전체 데이터, 기후 및 환경 데이터, 기술의 발전으로 인해 측정 및 저장이 가능해진 일상생활 데이터를 결합해, 만성질환 관리부터 암 치료까지 한 개인의 질환에 대한 더욱 체계적인 분석이 가능해 지는 것이다.

개요

심장마비관련 유전자 발견

입력 2003-12-01 00:00:00 수정 2003.12.01 00:00:00 정문재 기자

심장마비와 직접 연관된 유전자가 처음으로 발견됐다. 미국 클리블랜드 클리닉의 에릭 토폴 박사는 과학전문지 `사이언스` 최근호에 발표한 연구보고서에서 대대로 심장마비 등 관상동맥질환을 겪어온 아이오와주의 한 가계(家系)를 대상으로 실시한 유전자검사 결과 MEF2A로 불리는 유전자가 변이돼 있음을 알아냈다고 밝혔다. MEF2A는 혈류를 막아 심장마비에 이르게 하는 플라크 퇴적물(지방덩어리)로부터 동맥벽을 보호하는 역할을 하는 심장마비 차단 유전자다. 토폴 박사는 대대로 심장마비와 심장병 환자가 빈발하는 이 가계 구성원 약 100명의 유전자를 분석한 결과, 심장마비나 심장병에 걸린 사람은 모두 MEF2A 유전자에서 DNA의 핵심가닥 일부가 결여돼 있음을 발견했다고 밝혔다. 그는 "이러한 결함이 동맥을 경화시켜 막아버리는 것으로 생각된다"며 "이 가계 이외의 사람들에게서도 MEF2A 변이유전자가 심장병을 유발하는 지는 더 연구가 필요하다. 이 유전자가 생산하는 단백질은 일부 다른 유전자들의 활동을 조절하기 때문에 다른 유전자들도 심장병과 연관이 있는 지도 분석할 필요가 있다"고 덧붙였다. <정문재기자 timothy@sed.co.kr>

개요

-심장마비는 즉각적인 응급 처치가 필요한 상황으로, 신속한 치료 없이는 생명을 위협할 수 있다.
-심장마비에 대한 인공지능 분석은 이러한 상황에서 의료 전문가에게 중요한 정보를 제공할 것이다.
-인공지능을 통해 심장마비에 더 걸리기 쉬운 환경 혹은 잠재적 위험 환자를 찾아낼 수 있을 것이다.

시각화 - 모듈 import

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

시각화 - 헤더

```
df = pd.read_csv("/kaggle/input/heart-attack-analysis-prediction-dataset/heart.csv")

df.head()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

시각화 - 데이터 세트

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trtbps    303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalachh  303 non-null    int64
 8   exng      303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slp       303 non-null    int64
 11  caa       303 non-null    int64
 12  thall     303 non-null    int64
 13  output    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

```
print("Total rows in the dataset is:",df.shape[0],"Rows")
print("Total columns in the dataset is:",df.shape[1],"Columns")
```

```
Total rows in the dataset is: 303 Rows
Total columns in the dataset is: 14 Columns
```

열 데이터는 14개의 항목이 있다.
데이터 세트에는 총 303개의 항목이 있다.

시각화-데이터 세트 정보

- age : 환자의 연령
- sex : 환자의 성별(1: 남성, 0: 여성)
- cp : 흉통 종류(0: 전형적인 협심증, 1: 비정형 협심증, 2: 비협심증 통증, 3: 무증상)
- trtbps : 안정시 혈압(mmHg)
- chol : BMI 센서를 통해 가져온 콜레스테롤(mg/dl)
- fbs : (공복 혈당 > 120 mg/dl) (1: 참, 0: 거짓)
- restecg : 휴식 중인 심전도 결과(0: 정상, 1: ST-T파 이상이 있음, 2: Estes 기준에 따라 좌심실 비대 가능성이 있거나 확실하게 나타남)
- thalachh : 도달한 최대 심박수
- exang: 운동 유발 협심증(1: 예, 0: 아니요)
- oldpeak: 휴식에 비해 운동으로 인해 유발된 ST 우울증
- slp: 최대 운동 ST 세그먼트의 기울기(0: 경사가 없음, 1: 플랫, 2: 하향 경사)
- caa: 주요 혈관 수(0-3)
- thall : 지중해 빈혈(0: 널, 1: 수정된 결함, 2: 정상, 3: 되돌릴 수 있는 결함)
- output: 심장질환 진단(0: 심장병에 걸릴 가능성이 적음 1: 심장병에 걸릴 확률이 더 높음

시각화-데이터 세트 정보

1- 협심증: 심장 근육으로의 혈류 감소로 인한 흉통. 협심증에는 안정형 협심증, 불안정형 협심증, 변이형 협심증의 3가지 유형이 있다. 1)

2- 콜레스테롤: 스테롤(스테로이드와 알코올의 조합)의 하나로서 모든 동물 세포의 세포막에서 발견되는 지질이며 혈액을 통해 운반된다. "좋은 콜레스테롤"로 알려진 고밀도 지질단백질(HDL), "나쁜 콜레스테롤"로 알려진 저밀도 지질단백질(LDL)이 있다. 콜레스테롤 기준치는 정상 성인의 경우 200 mg/dL이며, 240 mg/dL 이상이면 위험하다. HDL의 정상 기준치는 60 mg/dl이상 이고 LDL의 정상 기준치는 130 mg/dl이하가 적당하다. 2)

3- ECG: 심전도(electrocardiogram)의 약어로, 정해진 시간에 심장의 전기적 활동을 해석하는 것. 3)

4- ST 우울증: ST 세그먼트 이상의 일종. ST 세그먼트는 ECG의 평평한 등전위 부분이며 심실 탈분극과 재분극 사이의 간격을 나타낸다. 4)

5- 지중해빈혈: 헤모글로빈 이상에 의해 빈혈이 생기는 유전병. 5)

시각화-결측값 확인

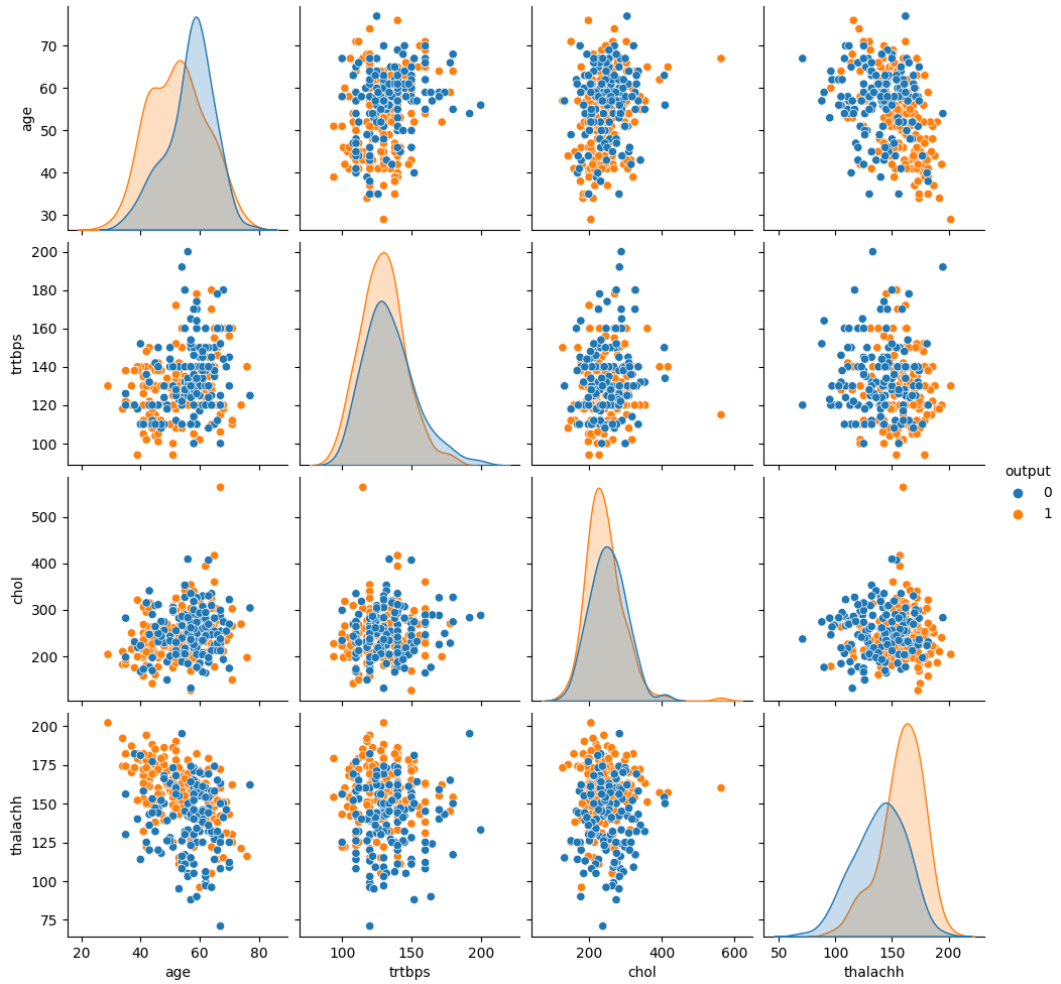
df.isnull().sum()

```
age         0
sex         0
cp          0
trtbps      0
chol        0
fbs         0
restecg     0
thalachh    0
exng        0
oldpeak     0
slp         0
caa         0
thall       0
output      0
dtype: int64
```

데이터에 null값은 없다.

시각화 - 관계그래프(Pairplot)

```
sns.pairplot(df[["age","trtbps","chol","thalachh","output"]],hue="output")
```



나이, 안정혈압, 콜레스테롤, 최대 심박수에 따른 심장 마비 확률

최대 심박수가 높을 수록 심장마비 확률이 높다.

시각화 - 히트맵(Heatmap)

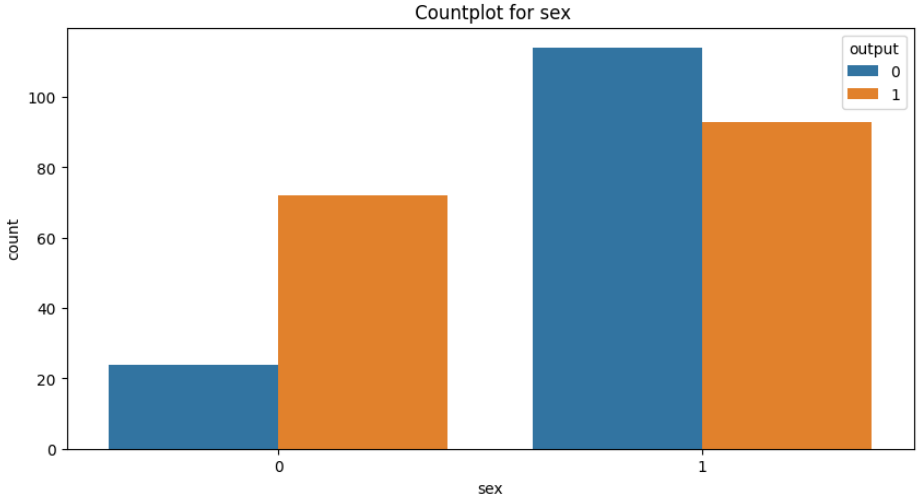
```
plt.figure(figsize=(10,7))
sns.heatmap(df.corr(),annot=True)
```

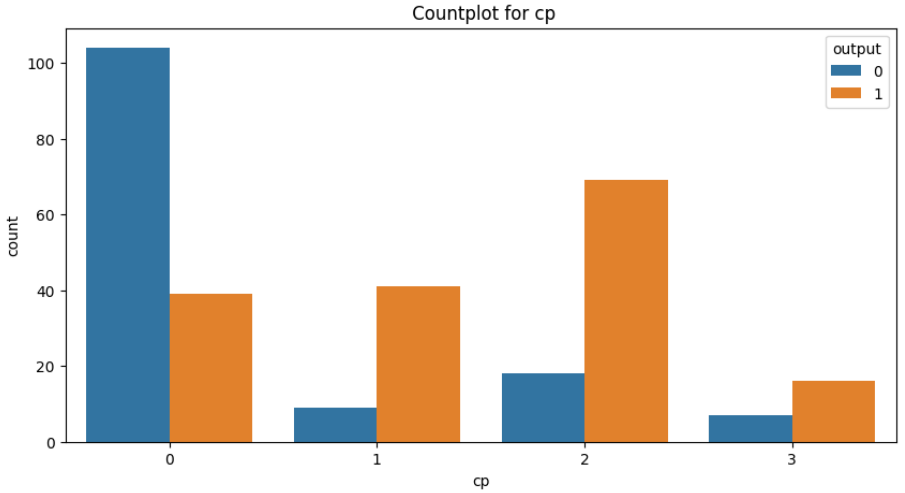
시각화 - Countplot

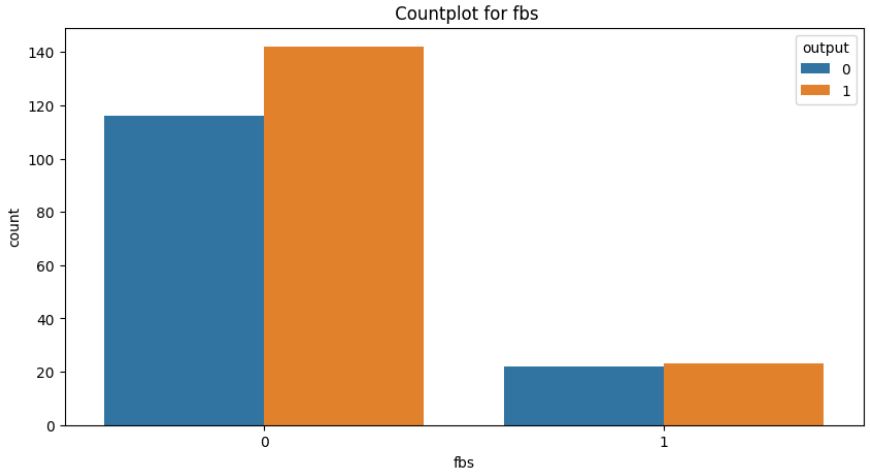
```
cat_col = ["sex","cp","fbs","restecg","exng","slp","caa","thall","output"]

for i in cat_col:
    plt.figure(figsize=(10,5))
    sns.countplot(data=df,x=i,hue="output")
    plt.title(f" Countplot for {i}")
    plt.xlabel(i)
    plt.show()
```

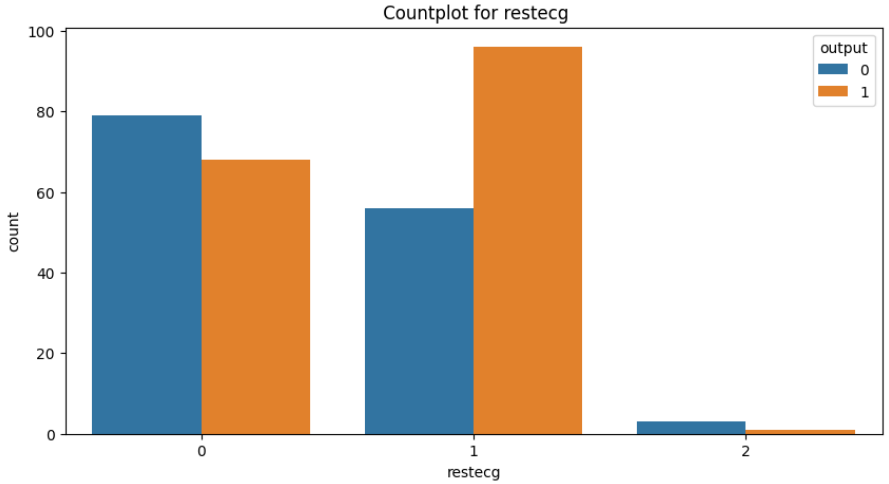
각 카테고리별 output 시각화

시각화 - Countplot

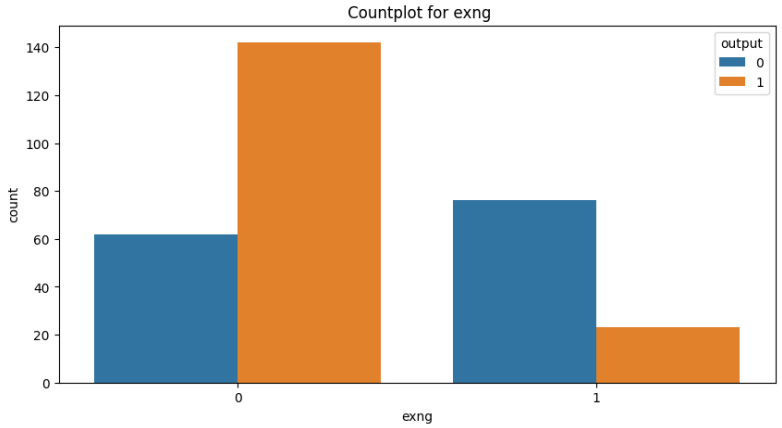


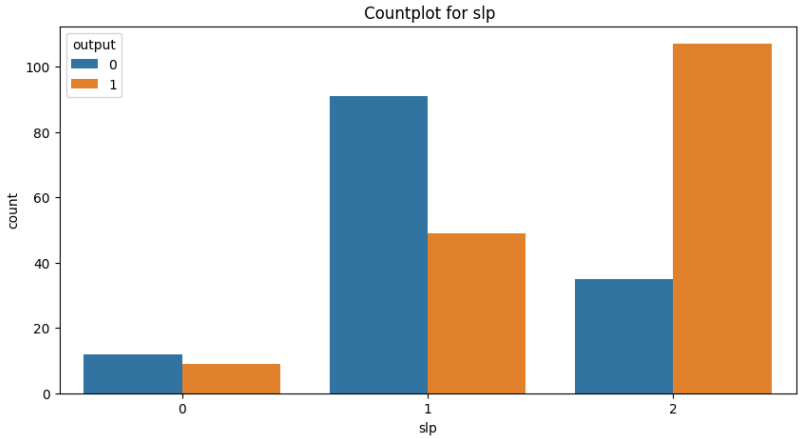


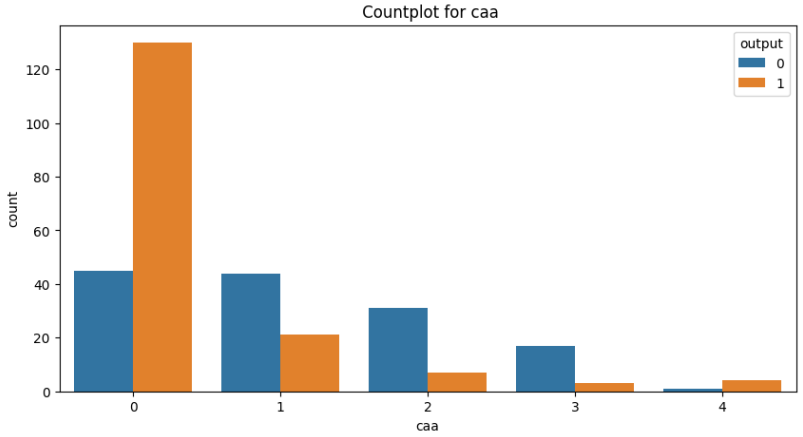


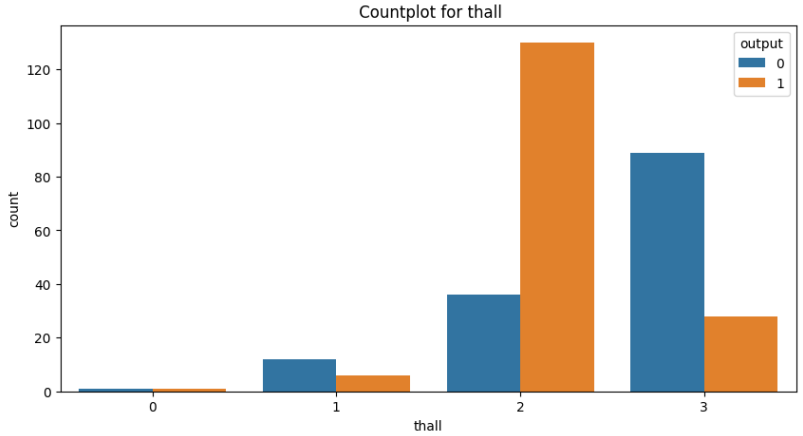


시각화 - Countplot

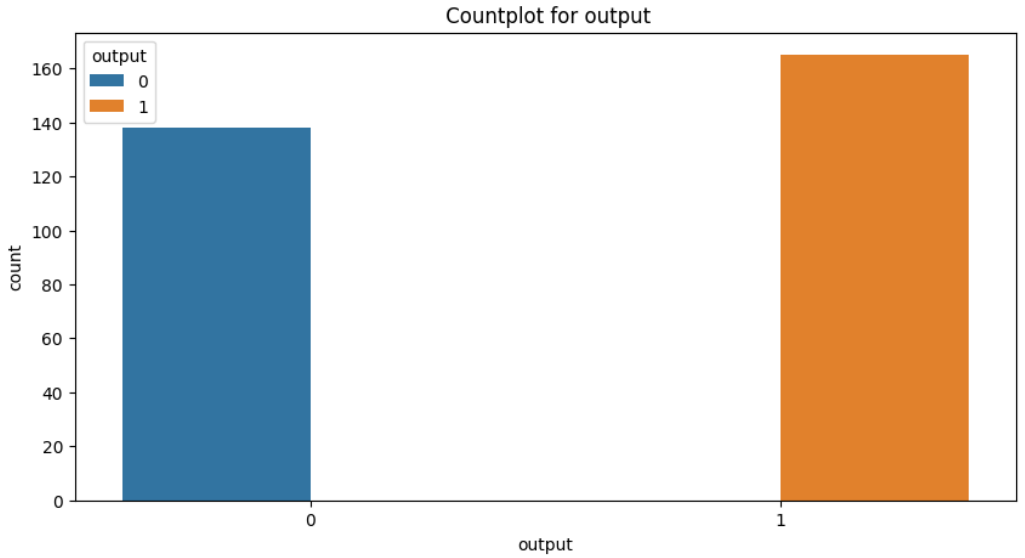








시각화 - Countplot

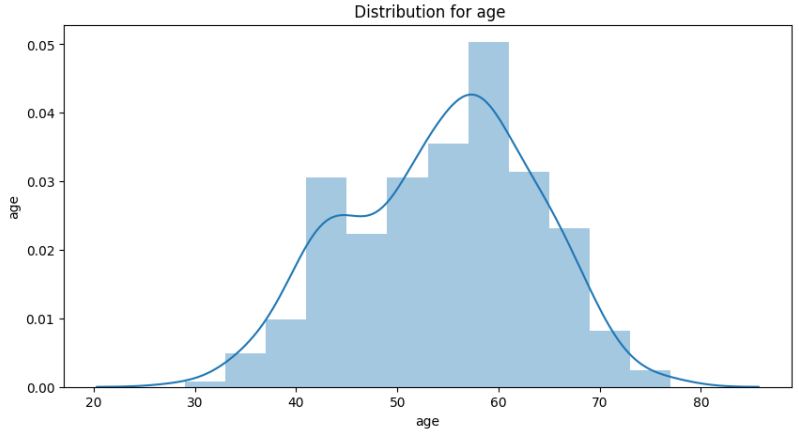


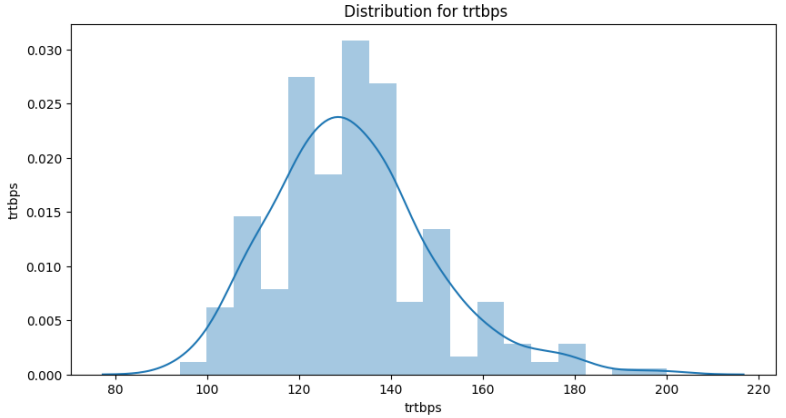
시각화 - Distplot

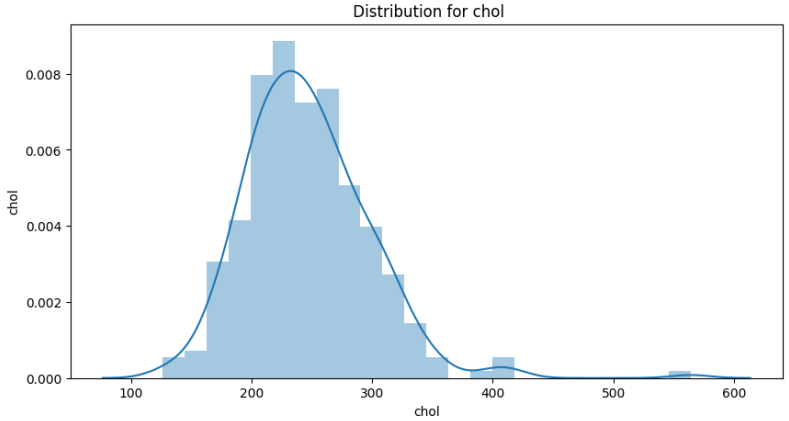
```
cont_col = ["age","trtbps","chol","thalachh","oldpeak"]

for i in cont_col:
    plt.figure(figsize=(10,5))
    sns.distplot(df[i])
    plt.title(f" Distribution for {i}")
    plt.ylabel(i)
    plt.show()
```

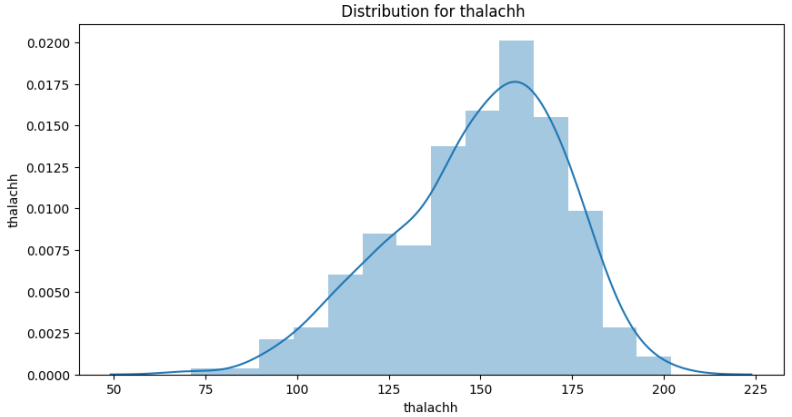
연속 데이터 변수의 전체 분포 표현

시각화 - Distplot

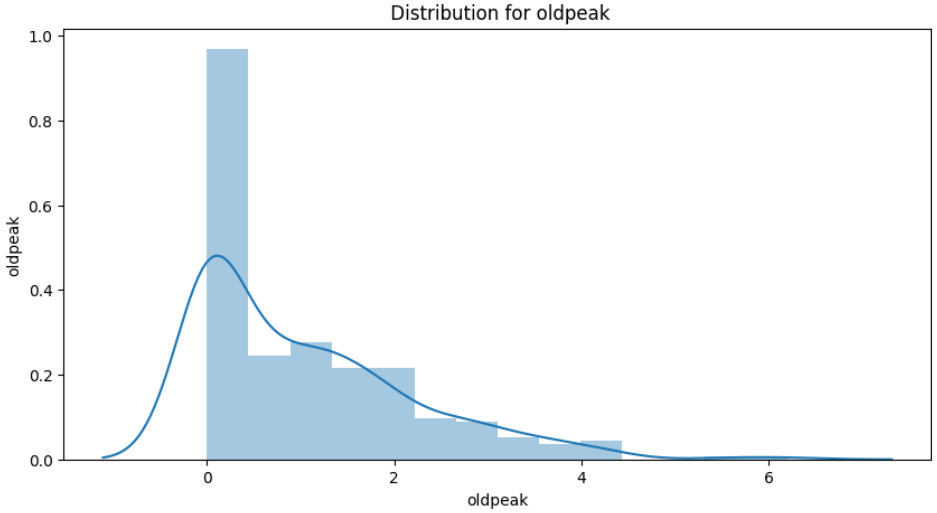








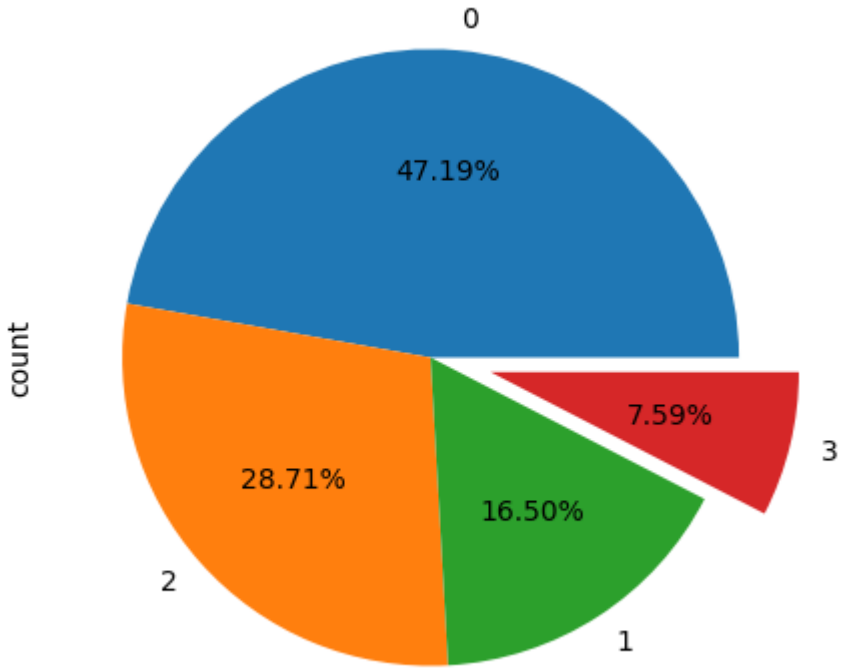
시각화 - Distplot



시각화 - Pie Chart

```
explode=[0,0,0,0.2]
df["cp"].value_counts().plot(kind="pie",autopct="%.2f%%",figsize=(5,5),explode=explode)
```

데이터를 비교하고 데이터 크기 분석



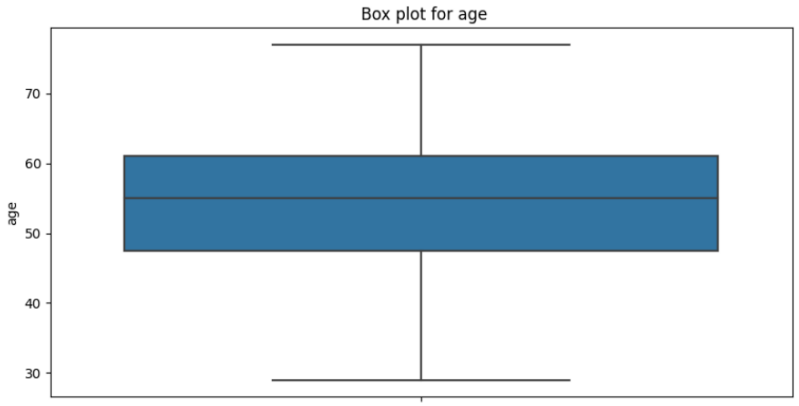
0: 전형적인 협심증 - 47.19%
1: 비정형 협심증 - 16.50%
2: 비협심증 통증 - 28.71%
3: 무증상 - 7.59%

전처리

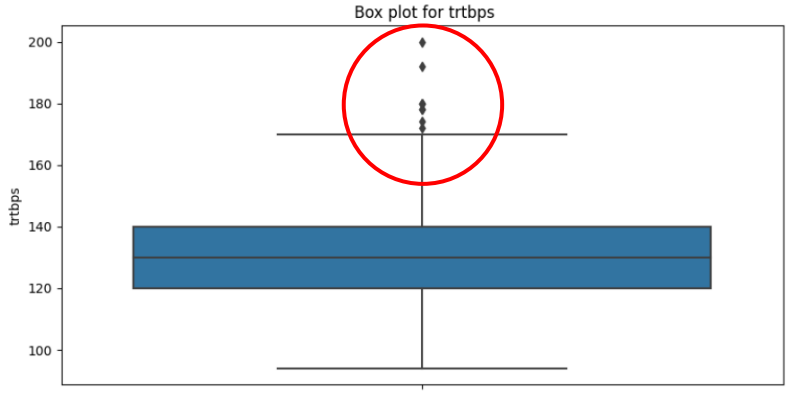
```
for i in cont_col:
    plt.figure(figsize=(10,5))
    sns.boxplot(data=df,y=i)
    plt.title(f" Box plot for {i}")
    plt.ylabel(i)
    plt.show()
```

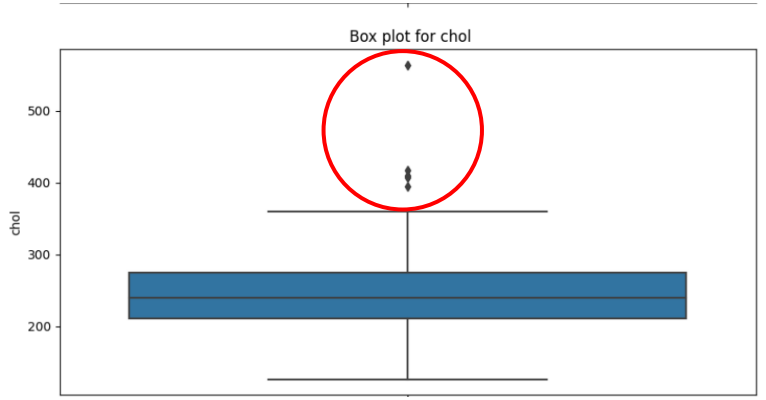
["age","trtbps","chol","thalachh","oldpeak"]

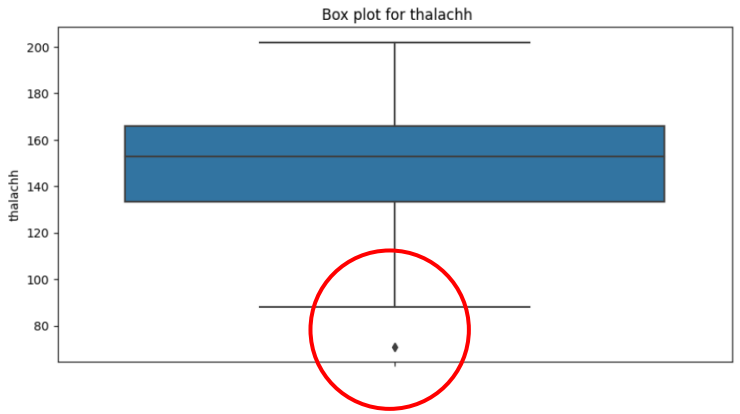
나이

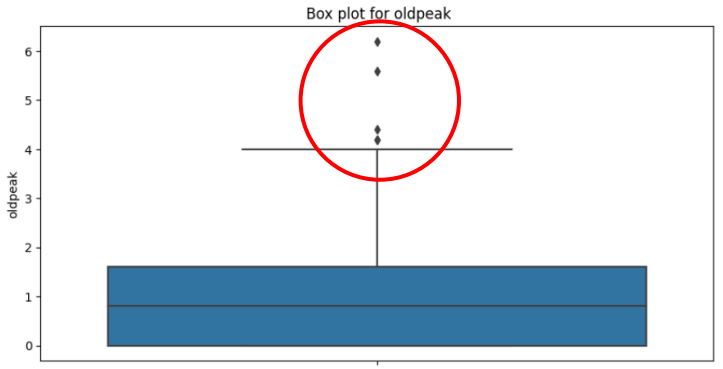


전처리









상자 그림을 살펴보면 몇 가지 특이치 값(이상값)이 있음을 알 수 있다.
데이터세트에서 이상값을 삭제하기로 결정.
열의 행에서 이상값을 찾아서 삭제하여 데이터를 정리.

전처리

```
features = df.iloc[:,:-1]
target = df.iloc[:,-1]
```

데이터 세트에서 output 분리

```
features[(features["trtbps"]>(165))].index

features.drop([8, 101, 110, 152, 195, 203, 223, 228, 241, 248, 260, 266,
292],axis=0,inplace=True)
target.drop([8, 101, 110, 152, 195, 203, 223, 228, 241, 248, 260, 266,
292],axis=0,inplace=True)
```

이상값 (165 초과) 삭제.

```
features[(features["chol"]>(350))].index

features.drop([4, 28, 39, 85, 96, 180, 220, 246],axis=0,inplace=True)
target.drop([4, 28, 39, 85, 96, 180, 220, 246],axis=0,inplace=True)
```

이상값 (350 초과) 삭제.

전처리

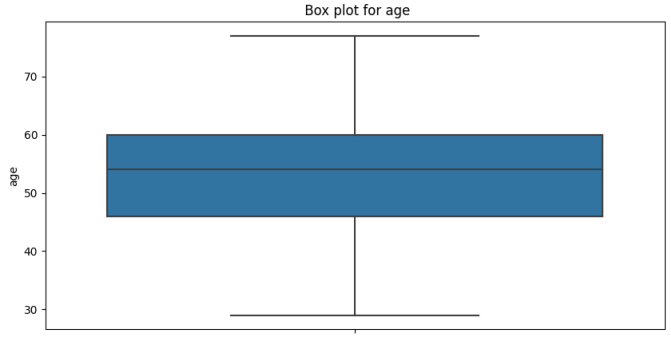
```
features[(features["thalachh"]<(80))].index

features.drop(272,axis=0,inplace=True)
target.drop(272,axis=0,inplace=True)

features[(features["oldpeak"]>(4))].index

features.drop([204,221,250,291],axis=0,inplace=True)
target.drop([204,221,250,291],axis=0,inplace=True)
```

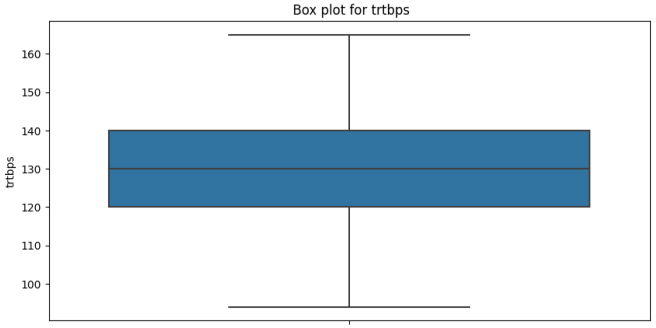
이상값
(80 미만) 삭제.

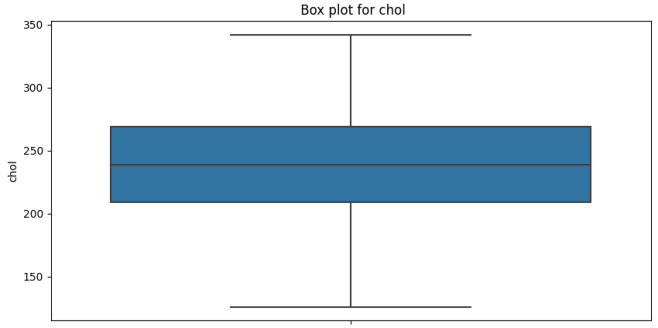
이상값
(4 초과) 삭제.

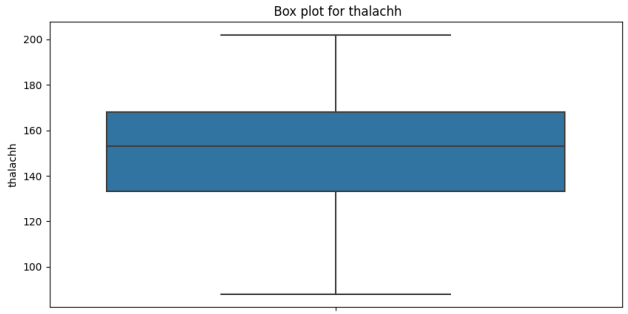
전처리 후

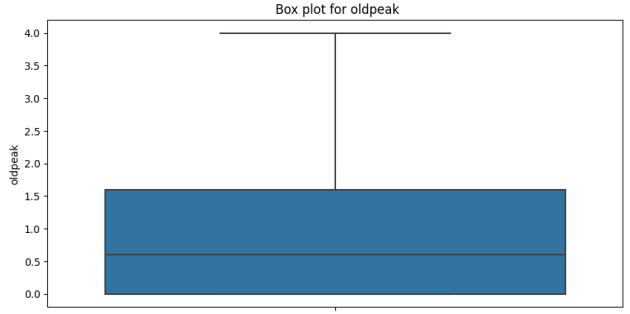
```
out_re = ["age","trtbps","chol","thalachh","oldpeak"]

for i in out_re:
    plt.figure(figsize=(10,5))
    sns.boxplot(data=features,y=i)
    plt.title(f" Box plot for {i}")
    plt.show()
```











전처리 후- 데이터 세트

features.info()

```
<class 'pandas.core.frame.DataFrame'>
Index: 277 entries, 0 to 302
Data columns (total 13 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       277 non-null    int64
 1   sex       277 non-null    int64
 2   cp        277 non-null    int64
 3   trtbps    277 non-null    int64
 4   chol      277 non-null    int64
 5   fbs       277 non-null    int64
 6   restecg   277 non-null    int64
 7   thalachh  277 non-null    int64
 8   exng      277 non-null    int64
 9   oldpeak   277 non-null    float64
 10  slp       277 non-null    int64
 11  caa       277 non-null    int64
 12  thall     277 non-null    int64
dtypes: float64(1), int64(12)
memory usage: 30.3 KB
```

target.info()

```
<class 'pandas.core.series.Series'>
Index: 277 entries, 0 to 302
Series name: output
Non-Null Count  Dtype
--------------  -----
277 non-null    int64
dtypes: int64(1)
memory usage: 4.3 KB
```

데이터세트 데이터 이상값 정리 후.
데이터세트에 **277**개의 항목 존재.

전처리 - 데이터 스케일링

수치 값을 공통 범위로 가져오기 위한 표준화.
데이터를 확장하면 모든 변수가 거리 계산에 미치는 영향의 균형을 맞추고 알고리즘 성능을 향상시키는 데 도움이 될 수 있다.
전처리에는 **Standard Scaler** 알고리즘을 사용.

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
features[out_re[:-1]] = ss.fit_transform(features[out_re[:-1]])
features.head()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall
0	0.994230	1	3	1.076424	-0.173356	1	0	-0.002524	0	2.3	0	0	1
1	-1.831517	1	2	0.051082	0.219868	0	1	1.614558	0	3.5	0	0	2
2	-1.396787	0	1	0.051082	-0.844151	0	0	0.958984	0	1.4	2	0	2
3	0.233452	1	1	-0.632479	-0.103964	0	1	1.221213	0	0.8	2	0	2
5	0.342134	1	0	0.734643	-1.121722	0	1	-0.089934	0	0.4	1	0	1

테스트 분할

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(features,target,test_size=0.30,random_state=100)

xtrain.shape, xtest.shape, ytrain.shape, ytest.shape
```

```
((193, 13), (84, 13), (193,), (84,))
```

데이터는 다음과 같이 분할.
학습 데이터 : 193 rows
테스트 데이터 : 84 rows

학습 및 테스트 결과 - 준비

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier,
GradientBoostingClassifier
from sklearn.svm import SVC

from sklearn.metrics import classification_report
```

분류 보고서 가져오기

```
def my_model(model):
  model.fit(xtrain,ytrain)
  ypred = model.predict(xtest)
  cr = classification_report(ytest,ypred)
  print(cr)
  accuracy = round(metrics.accuracy_score(ytest, ypred) * 100,1)
  return model, accuracy
```

정답지와 예측값

학습 및 테스트 결과 - 로지스틱 회귀

my_model(LogisticRegression())

	precision	recall	f1-score	support
0	0.86	0.66	0.75	38
1	0.76	0.91	0.83	46
accuracy			0.80	84
macro avg	0.81	0.79	0.79	84
weighted avg	0.81	0.80	0.79	84

(LogisticRegression(), 79.8)

Precision: 정밀도
Recall: 재현율
Accuracy: 정확도
F1-score: precision과 **recall**의 조화평균
Support: 각 라벨의 실제 개수
Macro avg: 단순평균**(**샘플 불균형 고려 **X)**
Weighted avg: 가중편귱**(**샘플 불균형 고려**)**

학습 및 테스트 결과 - 의사결정트리

my_model(DecisionTreeClassifier())

	precision	recall	f1-score	support
0	0.77	0.71	0.74	38
1	0.78	0.83	0.80	46
accuracy			0.77	84
macro avg	0.77	0.77	0.77	84
weighted avg	0.77	0.77	0.77	84

(DecisionTreeClassifier(), 77.4)

학습 및 테스트 결과 - 랜덤 포레스트

my_model(RandomForestClassifier())

	precision	recall	f1-score	support
0	0.86	0.66	0.75	38
1	0.76	0.91	0.83	46
accuracy			0.80	84
macro avg	0.81	0.79	0.79	84
weighted avg	0.81	0.80	0.79	84

(RandomForestClassifier(), 79.8)

학습 및 테스트 결과 - AdaBoostClassifier

my_model(AdaBoostClassifier())

	precision	recall	f1-score	support
0	0.81	0.76	0.78	38
1	0.81	0.85	0.83	46
accuracy			0.81	84
macro avg	0.81	0.81	0.81	84
weighted avg	0.81	0.81	0.81	84

(AdaBoostClassifier(), 81.0)

학습 및 테스트 결과 - GradientBoostingClassifier

my_model(GradientBoostingClassifier())

	precision	recall	f1-score	support
0	0.90	0.68	0.78	38
1	0.78	0.93	0.85	46
accuracy			0.82	84
macro avg	0.84	0.81	0.81	84
weighted avg	0.83	0.82	0.82	84

(GradientBoostingClassifier(), 82.1)

학습 및 테스트 결과 - SVM

my_model(SVC())

	precision	recall	f1-score	support
0	0.88	0.61	0.72	38
1	0.74	0.93	0.83	46
accuracy			0.79	84
macro avg	0.81	0.77	0.77	84
weighted avg	0.81	0.79	0.78	84

(SVC(), 78.6)

학습 및 테스트 결과 - 비교

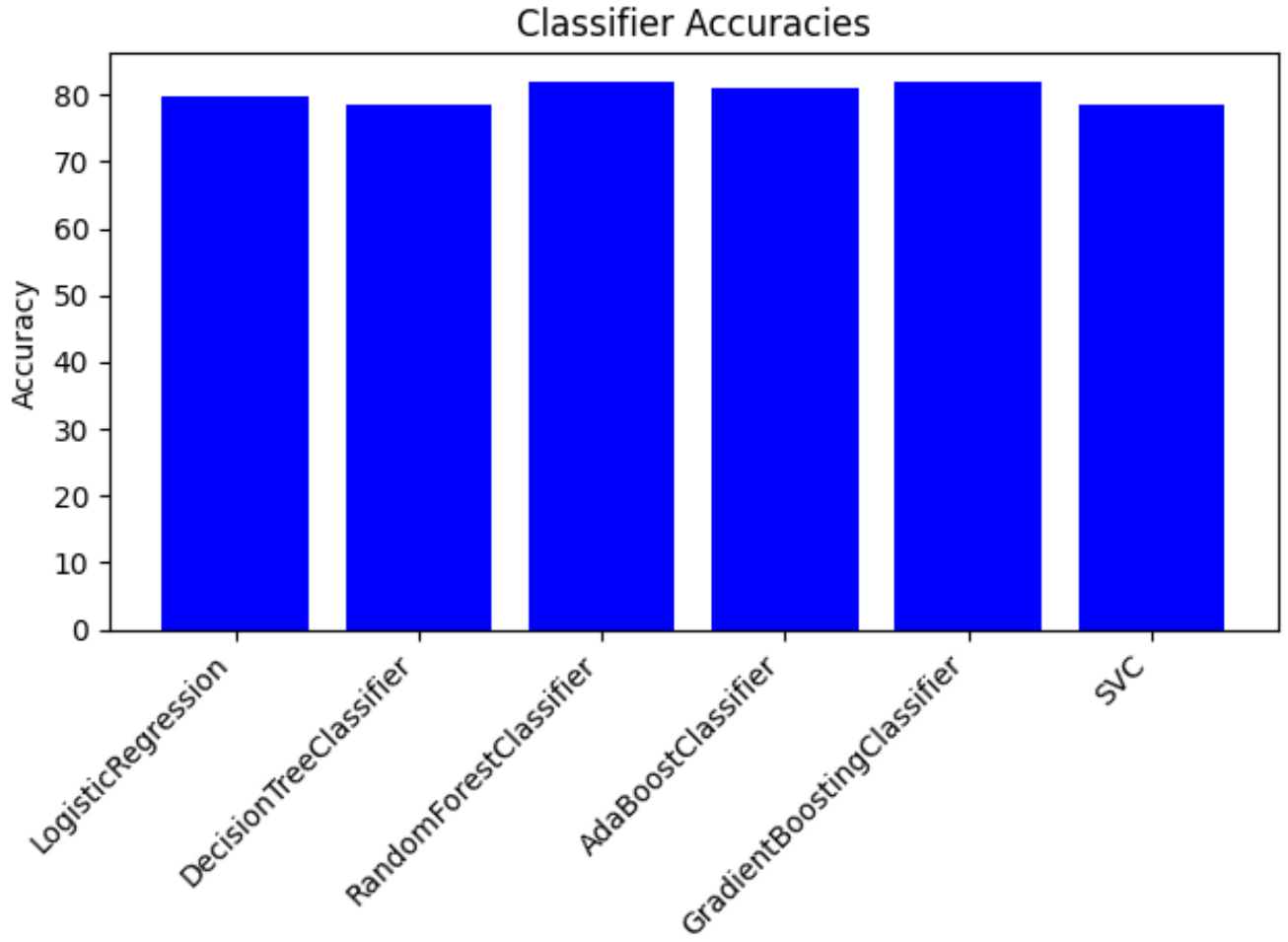
```
from sklearn.metrics import accuracy_score
from sklearn import metrics

def my_model_2(model):
  model.fit(xtrain,ytrain)
  ypred = model.predict(xtest)
  cr = classification_report(ytest,ypred)
  accuracy = round(metrics.accuracy_score(ytest, ypred)
* 100,1)
  return model, accuracy
```

학습 및 테스트 결과 - 비교

```
classifiers = [
    LogisticRegression(),
    DecisionTreeClassifier(),
    RandomForestClassifier(),
    AdaBoostClassifier(),
    GradientBoostingClassifier(),
    SVC()
]

# Store accuracies and trained models
accuracies = []
trained_models = []
```

```
# Train and evaluate each classifier
for clf in classifiers:
    model, accuracy = my_model_2(clf)
    trained_models.append(model)
    accuracies.append(accuracy)

# Plot the accuracies
fig, ax = plt.subplots()
classifiers_names = [type(clf).__name__ for clf in
classifiers]
ax.bar(classifiers_names, accuracies, color='blue')
ax.set_ylabel('Accuracy')
ax.set_title('Classifier Accuracies')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

학습 및 테스트 결과 - 비교


Classifier Accuracies
Accuracy
0
10
20
30
40
50
60
70
80
LogisticRegression
DecisionTreeClassifier
RandomForestClassifier
AdaBoostClassifier
GradientBoostingClassifier
SVC

결론

-인공지능을 다루면서 인공지능에 대해 보다 깊게 이해할 수 있었다.

-심장마비에 대한 분석을 진행하면서 인공지능이 생각한 것보다 사회에 깊게 연관되어 있다는 것을 알았다.

-인공지능 분석을 통해 환자의 심전도, 혈압, 호흡 속도 등을 분석하여 심장마비 가능성을 빠르게 판단할 수 있을 것이다.

-심장마비에 대한 인공지능 분석은 향후 응급 상황에서 의료 전문가들을 지원하는 중요한 수단이 될 것으로 기대된다.

참고문헌

1)https://www.nhs.uk/conditions/angina/#:~:text=Angina%20is%20chest%20pain%20caused,of%20these%20more%20serious%20problems
2)https://ko.wikipedia.org/wiki/%EC%BD%9C%EB%A0%88%EC%8A%A4%ED%85%8C%EB%A1%A4
3)https://ko.wikipedia.org/wiki/%EC%8B%AC%EC%A0%84%EB%8F%84
4)https://litfl.com/st-segment-ecg-library/
5)https://ko.wikipedia.org/wiki/%EC%A7%80%EC%A4%91%ED%95%B4%EB%B9%88%ED%98%88
6)https://www.doctorsnews.co.kr/news/articleView.html?idxno=106119
7)https://www.sedaily.com/NewsView/1HOZOZAKHT