5주차 예습 과제

- plt.subplots(): figure(전체그림)와 axes(실제 그래프 영역)을 동시에 생성
  - `f,ax=plt.subplots(figsize=(5,5))` : `f`는 figure 객체, `ax`는 축(ax) 객체
- sns.heatmap(): 2차원 데이터를 인자로 받아 히트맵으로 시각화
  - annot=True: 각 셀 안에 값 표시
  - fmt: 숫자 표시 형식

```
import seaborn as sns

f,ax=plt.subplots(figsize=(5,5))
sns.heatmap(data.corr(numeric_only=True),annot=True,linewidths=0.5,line
color='black',fmt='.4f',ax=ax)

# 변수 간 관계 시각화
# hue='Drug' → 'Drug' 컬럼에 따라 색상을 다르게 표시
sns.pairplot(data,hue='Drug')
```

```
import plotly.express as px

# Plotly로 막대그래프(bar chart) 생성
fig=px.bar(DataAge,x='Age',y='Number')
fig.show()

fig=px.bar(x=['HIGH','LOW','NORMAL'],y=[77,64,59])
fig.show()

# 히스토그램 생성
fig=px.histogram(x=['HIGH','NORMAL'],y=[103,97])
fig.show()

# 산점도 생성
# hover_data=['Na_to_K'] → hover시 'Na_to_K' 표시
fig=px.scatter(data,x='Na_to_K',y='Age',color='Drug',size='Age',hover_data
```

```
=['Na_to_K'])
fig.show()
```

```
import plotly.graph_objects as go

# 파이차트 그리기
colors=['gold','mediumturquoise']
fig=go.Figure(data=[go.Pie(labels=['M','F'],values=[104,96])])
# 그래프 요소 속성 수정
fig.update_traces(hoverinfo='label+percent',textinfo='value',textfont_size=
20,
                  marker=dict(colors=colors,line=dict(color='#000000',width=
2)))
fig.show()


fig=go.Figure(data=[go.Pie(labels=['DrugY','DrugX','DrugA','DrugC','Drug
B'],values=[91,54,23,16,16])])

fig.update_traces(hoverinfo='label+percent',textinfo='value',textfont_size=
20,
                  marker=dict(colors=px.colors.sequential.RdBu,line=dict(color='#
000000',width=2)))
fig.show()
```

```
# DataFrame 출력 시 최대 컬럼, 행, 길이 설정
pd.set_option('display.max_columns',100)
pd.set_option('display.max_rows',900)
pd.set_option('display.max_colwidth',200)
```

- df.astype({'col' : type}) : 칼럼 타입 변환
- df.select_dtypes('__') : 특정데이터 타입의 컬럼 선택
- df.nunique(): 컬럼별 고유값 개수 확인

- pd.concat() : 여러 데이터 프레임 합침
  - keys → 계층적 인덱스 만들어 구분해줌
- df.value_counts(normalize=True): 전체 데이터 중 비율 계산
- df.skew(): 칼럼 별 왜도 계산

```
# 상삼각행렬 생성
matrix = np.triu(df[numerical1].corr())
fig, ax = plt.subplots(figsize=(14,10))
sns.heatmap(df[numerical1].corr(), annot=True, fmt= '.2f',
            vmin=-1, vmax=1, center=0, cmap='coolwarm',mask=matrix, ax=ax)
# annot=True → 값 표시
# fmt= '.2f' → 상관계수 표시 형식:소수점 2자리
# vmin=-1, vmax=1 → 색상 범위 설정
# center=0 → 색상 기준점 0 기준으로 양/음 색 구분
# mask → True인 부분은 히트맵에 표시하지 않음
```

[Catboost]

- 범주형 데이터를 자동 처리해줌

```
accuracy =[]
model_names =[]

X= df.drop('HeartDisease', axis=1)
y= df['HeartDisease']

# 범주형 컬럼의 인덱스 찾기
categorical_features_indices = np.where(X.dtypes != np.float64)[0]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_s
tate=42)

# 모델 학습
# cat_features → 범주형 컬럼 인덱스 지정
# eval_set → 검증용 데이터 지정
model = CatBoostClassifier(verbose=False,random_state=0)
```

```
model.fit(X_train, y_train,cat_features=categorical_features_indices,eval_set=(X_test, y_test))
# 정확도 계산
y_pred = model.predict(X_test)
# 정확도 리스트에 저장
accuracy.append(round(accuracy_score(y_test, y_pred),4))

model_names = ['Catboost_default']
result_df5 = pd.DataFrame({'Accuracy':accuracy}, index=model_names)
```

파라미터:

- Objective: 학습 목적과 평가 지표 지정
- colsample_bylevel: 각 트리 레벨에서 사용하는 컬럼 비율 → 학습 속도 향상 O, 모델 성능에는 영향 X
- depht : 트리 최대 깊이
- boosting_type : 부스팅 방식 선택 → 학습 속도 및 과적합 제어
- bootstrap_type : 샘플링 방식 지정