Heart failure prediction dataset.

→ 심장 질환의 원인 살펴보기 (다양한 EDA + 예측모델)

· 코드 실행 전에,

① 범주형 / 수치형 변수의 분류 필요하다.

② target 변수: Heart Disease

③ target 변수가 영향있는 변수인지 확인

④ 모델 만들기.

1 필요한 library 모두 import

⇒ optuna, catboost, missingno 는 아나콘다 프롬프트에 pip install ____ 로 먼저 설치.

· df.info() 를 통해 null 값 확인 (null 값 없음)

· df.duplicated().sum() 를 통해 row 마다의 중복값 확인 (중복 없음)

· def missing (df) 를 지정하여 한눈에 null 값 확인 (null 없음)

```
def missing (df):
    missing_number = df.isnull().sum().sort_values(ascending=
False)
    missing_percent = (df.isnull().sum()/df.isnull().count
()).sort_values(ascending=False)
    missing_values = pd.concat([missing_number, missing_perce
nt], axis=1, keys=['Missing_Number', 'Missing_Percent'])
    return missing_values

missing(df)
```

· 범주형 / 수치형 변수 구분.

Numerical : select_dtypes ('number'). columns

Categorical : select_dtypes ('object'). columns

```
numerical= df.drop(['HeartDisease'], axis=1).select_dtypes('n
umber').columns

categorical = df.select_dtypes('object').columns

print(f'Numerical Columns:  {df[numerical].columns}')
print('\n')
print(f'Categorical Columns: {df[categorical].columns}')
```

2 Target 변수 (Heart Disease)

: normalize 를 이용하여 심장병 걸린 사람의 percent 를 구한다.

· plot을 이용하여 심장병 유무 사람의 수를 histogram 으로 보인다.

⇒ balanced data (evaluation metric 으로 accuracy 사용이 가능하다.)

```
y = df['HeartDisease']
print(f'Percentage of patient had a HeartDisease:  {round(y.value_counts(normalize=True)[1]*100,2)} %  --> ({y.value_counts()[1]} patient)\nPercentage of patient did not have a HeartDisease: {round(y.value_counts(normalize=True)[0]*100,2)}  % --> ({y.value_counts()[0]} patient)')
```

3. EDA - 수치형 변수

- df[numerical].iplot(kind='hist') 를 이용하여 feature 별 분포를 1개의 그림에 표시.
- df[numerical].iplot(kind='histogram', subplots=True, bins=50) 을 이용하여 각 피처별로 개별 분포를 그린다.

```
skew_limit = 0.75 # This is our threshold-limit to evaluate skewness. Overall below abs(1) seems acceptable for the linear models.
skew_vals = df[numerical].drop('FastingBS', axis=1).skew()
skew_cols= skew_vals[abs(skew_vals)> skew_limit].sort_values(ascending=False)
skew_cols
```

왜도측정
- skew limit을 0.75로 설정.
- 이 peak 이하인 threshold 넘는 값이 나옴.
- 정규분포와 유사할 수록 더 신뢰가능.

```
Oldpeak    1.022872
dtype: float64
```

- corr() 와 heatmap 을 이용하여 수치형변수와 타겟 변수 간의 상관관계 확인하기.
  → 상관관계 약하다.

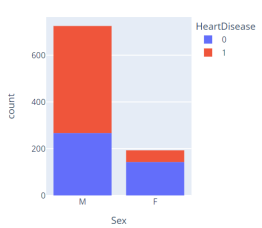
4. EDA - 범주형 변수

- 각 범주별로 심장병 유무를 숫자로 계산, histogram으로 나타낸다.

```
print (f'A female person has a probability of {round(df[df["Sex"]=="F"]["HeartDisease"].mean()*100,2)} % have a HeartDisease')

print()

print (f'A male person has a probability of {round(df[df["Sex"]=="M"]["HeartDisease"].mean()*100,2)} % have a HeartDisease')

print()
```

```
A female person has a probability of 25.91 % have a HeartDisease

A male person has a probability of 63.17 % have a HeartDisease
```

```
fig = px.histogram(df, x="Sex", color="HeartDisease",width=400, height=400)
fig.show()
```



EDA를 통해,,

- Target 변수가 balanced data 임을 확인.
- 수치형 변수는 target 변수와 연관 상관관계를 보인다.
- 범주형 변수 — 범주 별로 확연한 차이가 보인다.
  - 심장병 Male > female
  - Asy > ATA
  - Y > N
  - UP < Flat, Down

5 model selection

1. baseline model → dummy classifier
2. Logistic, linear discriminant, SVC, KNN
3. Ensemble model (Ada boost, gradient boosting, Random forest, Extra Trees)
4. Famous Trio (XGBoost, Light GBM, Catboost)
5. Catboost
6. Catboost tuning (optuna)
7. Feature importance
8. model comparison

① Baseline model (모델 성능 비교의 기준이 되는 모델): 모델 성능에 대한 최소 하한선을 제공.

```
accuracy =[]
model_names =[]


X= df.drop('HeartDisease', axis=1)
y= df['HeartDisease']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

ohe= OneHotEncoder()
ct= make_column_transformer((ohe,categorical),remainder='passthrough')

model = DummyClassifier(strategy='constant', constant=1)
pipe = make_pipeline(ct, model)
pipe.fit(X_train, y_train)
y_pred = pipe.predict(X_test)
accuracy.append(round(accuracy_score(y_test, y_pred),4))
print (f'model : {model} and  accuracy score is : {round(accuracy_score(y_test, y_pred),4)}')

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

→ imbalanced dataset 일 경우, stratified 로.

dummy classifier accuracy: 0.5942

0.5942보다 높아야 어느 정도 설명력을 가짐...

② pipeline (범주형)

: 전처리 단계, 모델 생성, 학습 등 여러 프로세스를 한번에 처리하는 것이 가능.

```
accuracy =[]
model_names =[]


X= df.drop('HeartDisease', axis=1)
y= df['HeartDisease']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

ohe= OneHotEncoder()
ct= make_column_transformer((ohe,categorical),remainder='passthrough')

lr = LogisticRegression(solver='liblinear')
lda= LinearDiscriminantAnalysis()
svm = SVC(gamma='scale')
knn = KNeighborsClassifier()

models = [lr,lda,svm,knn]

for model in models:
    pipe = make_pipeline(ct, model)
    pipe.fit(X_train, y_train)
    y_pred = pipe.predict(X_test)
    accuracy.append(round(accuracy_score(y_test, y_pred),4))
    print (f'model : {model} and  accuracy score is : {round(accuracy_score(y_test, y_pred),
4)}')

model_names = ['Logistic','LinearDiscriminant','SVM','KNeighbors']
result_df1 = pd.DataFrame({'Accuracy':accuracy}, index=model_names)
result_df1
```

(범주형 변수에 원핫 인코딩 적용)
ohe: 변환기 / categorical: 변환기 적용변수 / remainder='passthrough': 적용 요소 시간
ct: 변환기 / model: 사용모델

```
model : LogisticRegression(solver='liblinear') and  accuracy score is : 0.8841
model : LinearDiscriminantAnalysis() and  accuracy score is : 0.8696
model : SVC() and  accuracy score is : 0.7246
model : KNeighborsClassifier() and  accuracy score is : 0.7174
```

	Accuracy
Logistic	0.8841
LinearDiscriminant	0.8696
SVM	0.7246
KNeighbors	0.7174

pipeline (수치형)

```
accuracy =[]
model_names =[]


X= df.drop('HeartDisease', axis=1)
y= df['HeartDisease']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

ohe= OneHotEncoder()
s= StandardScaler()
ct1= make_column_transformer((ohe,categorical),(s,numerical))

lr = LogisticRegression(solver='liblinear')
lda= LinearDiscriminantAnalysis()
svm = SVC(gamma='scale')
knn = KNeighborsClassifier()

models = [lr,lda,svm,knn]

for model in models:
    pipe = make_pipeline(ct1, model)
    pipe.fit(X_train, y_train)
    y_pred = pipe.predict(X_test)
    accuracy.append(round(accuracy_score(y_test, y_pred),4))
    print (f'model : {model} and  accuracy score is : {round(accuracy_score(y_test, y_pred),
4)}')

model_names = ['Logistic_scl','LinearDiscriminant_scl','SVM_scl','KNeighbors_scl']
result_df2 = pd.DataFrame({'Accuracy':accuracy}, index=model_names)
result_df2
```

수치형 변수 추가.

```
model : LogisticRegression(solver='liblinear') and  accuracy score is : 0.8804
model : LinearDiscriminantAnalysis() and  accuracy score is : 0.8696
model : SVC() and  accuracy score is : 0.8841
model : KNeighborsClassifier() and  accuracy score is : 0.8841
```

	Accuracy
Logistic_scl	0.8804
LinearDiscriminant_scl	0.8696
SVM_scl	0.8841
KNeighbors_scl	0.8841

) 향상

+) Ada Boost, Gradient Boosting, Random forest, 에 대해서도 동일하게 진행.

CatBoost (카테고리형 잘 확인)

```
accuracy =[]
model_names =[]


X= df.drop('HeartDisease', axis=1)
y= df['HeartDisease']
categorical_features_indices = np.where(X.dtypes != np.float)[0]

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

model = CatBoostClassifier(verbose=False,random_state=0)

model.fit(X_train, y_train,cat_features=categorical_features_indices,eval_set=(X_test, y_test))
y_pred = model.predict(X_test)
accuracy.append(round(accuracy_score(y_test, y_pred),4))

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

	Accuracy
Catboost_default	0.8804

③ Optuna를 위해 Catboost 최적화

: GridSearchCV와는 달리 파라미터 값을 직접 지정할 필요 없음.

- 범위 지정 , 수행시간, 비교적 빠름 , 시각화 등으로 학습절차 확인 가능

```
def objective(trial):
    X= df.drop('HeartDisease', axis=1)
    y= df['HeartDisease']
    categorical_features_indices = np.where(X.dtypes != np.float)[0]

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

    param = {
        "objective": trial.suggest_categorical("objective", ["Logloss", "CrossEntropy"]),
        "colsample_bylevel": trial.suggest_float("colsample_bylevel", 0.01, 0.1),
        "depth": trial.suggest_int("depth", 1, 12),
        "boosting_type": trial.suggest_categorical("boosting_type", ["Ordered", "Plain"]),
        "bootstrap_type": trial.suggest_categorical(
            "bootstrap_type", ["Bayesian", "Bernoulli", "MVS"]
        ),
        "used_ram_limit": "3gb",
    }

    if param["bootstrap_type"] == "Bayesian":
        param["bagging_temperature"] = trial.suggest_float("bagging_temperature", 0, 10)
    elif param["bootstrap_type"] == "Bernoulli":
        param["subsample"] = trial.suggest_float("subsample", 0.1, 1)

    cat_cls = CatBoostClassifier(**param)

    cat_cls.fit(X_train, y_train, eval_set=[(X_test, y_test)], cat_features=categorical_features
_indices,verbose=0, early_stopping_rounds=100)
    preds = cat_cls.predict(X_test)
    pred_labels = np.rint(preds)
    accuracy = accuracy_score(y_test, pred_labels)
    return accuracy


if __name__ == "__main__":
    study = optuna.create_study(direction="maximize")
    study.optimize(objective, n_trials=50, timeout=600)

    print("Number of finished trials: {}".format(len(study.trials)))

    print("Best trial:")
    trial = study.best_trial

    print("  Value: {}".format(trial.value))

    print("  Params: ")
    for key, value in trial.params.items():
        print("    {}: {}".format(key, value))
```

조절해야하는 값.

maximize / minimize

④ feature importance / model accuracy