In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import warnings
df=pd.read_csv(r"C:\Users\saira\OneDrive\Desktop\heart.csv")
df.head()
```

Out[1]:

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal
0	63	1	typical	145	233	1	2	150	0	2.3	3	0	fixed
1	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3	normal
2	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2	reversable
3	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0	normal
4	41	0	nontypical	130	204	0	2	172	0	1.4	1	0	normal

In [2]:
```

print(df.isnull().sum())
```

```
Age          0
Sex          0
ChestPain    0
RestBP       0
Chol         0
Fbs          0
RestECG      0
MaxHR        0
ExAng        0
Oldpeak      0
Slope        0
Ca           0
Thal         2
Target       0
dtype: int64
```

In [3]:
```
df=df.dropna()
```

In [4]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 301 entries, 0 to 302
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Age        301 non-null    int64  
 1   Sex        301 non-null    int64  
 2   ChestPain  301 non-null    object 
 3   RestBP     301 non-null    int64  
 4   Chol       301 non-null    int64  
 5   Fbs        301 non-null    int64  
 6   RestECG    301 non-null    int64  
 7   MaxHR      301 non-null    int64  
 8   ExAng      301 non-null    int64  
 9   Oldpeak    301 non-null    float64
 10  Slope      301 non-null    int64  
 11  Ca         301 non-null    int64  
 12  Thal       301 non-null    object 
 13  Target     301 non-null    int64  
dtypes: float64(1), int64(11), object(2)
memory usage: 35.3+ KB
```

In [5]:
```
1 df.describe()
```

Out[5]:

	Age	Sex	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	C
count	301.000000	301.000000	301.000000	301.000000	301.000000	301.000000	301.000000	301.000000	301
mean	54.451827	0.681063	131.714286	246.936877	0.146179	0.990033	149.700997	0.325581	1
std	9.067258	0.466841	17.655729	51.859869	0.353874	0.994937	22.860817	0.469372	1
min	29.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0
25%	48.000000	0.000000	120.000000	211.000000	0.000000	0.000000	134.000000	0.000000	0
50%	56.000000	1.000000	130.000000	242.000000	0.000000	1.000000	153.000000	0.000000	0
75%	61.000000	1.000000	140.000000	275.000000	0.000000	2.000000	166.000000	1.000000	1
max	77.000000	1.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6

In [6]:
```
1 import pandas_profiling
2 pandas_profiling.ProfileReport(df)
```

```
C:\Users\saira\AppData\Local\Temp\ipykernel_4792\1118840070.py:1: DeprecationWarning:
`import pandas_profiling` is going to be deprecated by April 1st. Please use `import y
data_profiling` instead.
  import pandas_profiling
```

Summarize dataset: 48/48 [00:06<00:00, 4.04it/s, 100% Completed]

Generate report structure: 100% 1/1 [00:06<00:00, 6.90s/it]

Render HTML: 100% 1/1 [00:01<00:00, 1.34s/it]

Overview

Dataset statistics

Number of variables	14
Number of observations	301
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	35.3 KiB
Average record size in memory	120.0 B

Variable types

Numeric	5
Categorical	9

Alerts

`ChestPain` is highly overall correlated with `Target`	**High correlation**
`Target` is highly overall correlated with `ChestPain` and 1 other fields (ChestPain, Thal)	**High correlation**
`Thal` is highly overall correlated with `Target`	**High correlation**

Out[6]:

In [7]:
```
d=df['Target'].value_counts()
print(d)
```

```
0    163
1    138
Name: Target, dtype: int64
```

In [8]:
```
import seaborn as sns
def plotTarget():
    sns.countplot(x='Target', data=df, ax=ax)
    for i, p in enumerate(ax.patches):
        count=df['Target'].value_counts().values[i]
        x=p.get_x()+ p.get_width() /2.
        y=p.get_height() + 3
        label='{:1.2f}'.format(count / float(df.shape[0]))
        ax.text(x, y,label, ha='center')

fig_target,ax=plt.subplots(nrows=1, ncols=1, figsize=(5, 2))
plotTarget()
```

In [9]:
```
df.corr()
```

```
C:\Users\saira\AppData\Local\Temp\ipykernel_4792\1134722465.py:1: FutureWarning: The d
efault value of numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  df.corr()
```

Out[9]:

	Age	Sex	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	
Age	1.000000	-0.098138	0.284734	0.208287	0.121670	0.149037	-0.395982	0.092985	0.203604	0
Sex	-0.098138	1.000000	-0.065284	-0.202126	0.041025	0.029016	-0.057065	0.140802	0.098482	0
RestBP	0.284734	-0.065284	1.000000	0.129371	0.178498	0.147089	-0.046402	0.065564	0.188801	0
Chol	0.208287	-0.202126	0.129371	1.000000	0.015762	0.171185	-0.005690	0.064250	0.044836	-0
Fbs	0.121670	0.041025	0.178498	0.015762	1.000000	0.079892	-0.012297	0.013534	0.004855	0
RestECG	0.149037	0.029016	0.147089	0.171185	0.079892	1.000000	-0.077950	0.092626	0.117580	0
MaxHR	-0.395982	-0.057065	-0.046402	-0.005690	-0.012297	-0.077950	1.000000	-0.386043	-0.349391	-0
ExAng	0.092985	0.140802	0.065564	0.064250	0.013534	0.092626	-0.386043	1.000000	0.287926	0
Oldpeak	0.203604	0.098482	0.188801	0.044836	0.004855	0.117580	-0.349391	0.287926	1.000000	0
Slope	0.162228	0.031571	0.117437	-0.004228	0.054079	0.140144	-0.393527	0.254076	0.576795	1
Ca	0.331939	0.100345	0.100535	0.106125	0.164689	0.127330	-0.256365	0.151731	0.274451	0
Target	0.224394	0.272006	0.151471	0.086762	0.015613	0.177049	-0.425870	0.427860	0.423894	0

In [10]:
```
x=df['ChestPain']
x.value_counts()
```

Out[10]:
```
asymptomatic    143
nonanginal       85
nontypical       50
typical          23
Name: ChestPain, dtype: int64
```

In [11]:
```
x=df['Thal']
x.value_counts()
```

Out[11]:
```
normal        166
reversable    117
fixed          18
Name: Thal, dtype: int64
```

In [12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 301 entries, 0 to 302
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Age        301 non-null    int64  
 1   Sex        301 non-null    int64  
 2   ChestPain  301 non-null    object 
 3   RestBP     301 non-null    int64  
 4   Chol       301 non-null    int64  
 5   Fbs        301 non-null    int64  
 6   RestECG    301 non-null    int64  
 7   MaxHR      301 non-null    int64  
 8   ExAng      301 non-null    int64  
 9   Oldpeak    301 non-null    float64
 10  Slope      301 non-null    int64  
 11  Ca         301 non-null    int64  
 12  Thal       301 non-null    object 
 13  Target     301 non-null    int64  
dtypes: float64(1), int64(11), object(2)
memory usage: 43.4+ KB
```

In [13]:
```
df_X= df.loc[:, df.columns != 'Target']
df_y= df.loc[:, df.columns == 'Target']
```

In [14]:
```
from sklearn.compose import make_column_selector as selector

numerical_columns_selector = selector(dtype_exclude=object)
categorical_columns_selector = selector(dtype_include=object)

numerical_columns = numerical_columns_selector(df_X)
categorical_columns = categorical_columns_selector(df_X)
```

In [18]:
```
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

In [26]:
```
categorical_preprocessor = Pipeline(steps=[('label_encoder', LabelEncoder())])
numerical_preprocessor = StandardScaler()
```

In [27]:
```
categorical_columns,numerical_columns
```

Out[27]:
```
(['ChestPain', 'Thal'],
 ['Age',
  'Sex',
  'RestBP',
  'Chol',
  'Fbs',
  'RestECG',
  'MaxHR',
  'ExAng',
  'Oldpeak',
  'Slope',
  'Ca'])
```

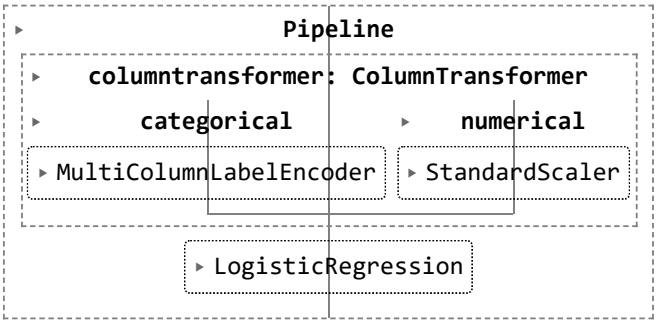
In [34]:
```
from sklearn.base import TransformerMixin
from sklearn.preprocessing import LabelEncoder
from sklearn.utils.validation import check_is_fitted
from sklearn.compose import ColumnTransformer
class MultiColumnLabelEncoder(TransformerMixin):
    def __init__(self):
        self.encoders = {}

    def fit(self, X, y=None):
        for col in X.columns:
            self.encoders[col] = LabelEncoder().fit(X[col])
        return self

    def transform(self, X):
        check_is_fitted(self, 'encoders')
        output = X.copy()
        for col, encoder in self.encoders.items():
            output[col] = encoder.transform(X[col])
        return output

    def get_params(self, deep=True):
        return {}

categorical_preprocessor = MultiColumnLabelEncoder()
numerical_preprocessor = StandardScaler()

preprocessor = ColumnTransformer(
    transformers=[
        ("categorical", categorical_preprocessor, categorical_columns),
        ("numerical", numerical_preprocessor, numerical_columns),
    ]
)
```

In [35]:
```
from sklearn.model_selection import train_test_split
X_train,X_test, y_train, y_test=train_test_split(df_X,df_y, test_size = 0.25, rando
columns = X_train.columns
```

In [36]:
```
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
pipeline = Pipeline(steps=[('columntransformer', preprocessor), ('LogisticRegressio
pipeline.fit(X_train, y_train)
```

```
C:\Users\saira\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataConve
rsionWarning: A column-vector y was passed when a 1d array was expected. Please change
the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[36]:



In [37]:
```
pre=pipeline.predict(X_test)
```

In [38]:

```
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(pre,y_test))
print(confusion_matrix(pre,y_test))
```

```
              precision    recall  f1-score   support

           0       0.90      0.82      0.86        44
           1       0.78      0.88      0.82        32

    accuracy                           0.84        76
   macro avg       0.84      0.85      0.84        76
weighted avg       0.85      0.84      0.84        76

[[36  8]
 [ 4 28]]
```

In [39]:

```
import lightgbm as lgb
clf = lgb.LGBMClassifier()
pipeline = Pipeline(steps=[('columntransformer', preprocessor), ('lgbmclassifier',
pipeline.fit(X_train, y_train)
```

```
[LightGBM] [Info] Number of positive: 102, number of negative: 123
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing wa
s 0.000185 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 234
[LightGBM] [Info] Number of data points in the train set: 225, number of used featur
es: 13
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.453333 -> initscore=-0.187212
[LightGBM] [Info] Start training from score -0.187212
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

In [40]:

```
y_pred = pipeline.predict(X_test)
```

In [41]:

```
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_pred,y_test))
print(confusion_matrix(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.82      0.85      0.84        39
           1       0.83      0.81      0.82        37

    accuracy                           0.83        76
   macro avg       0.83      0.83      0.83        76
weighted avg       0.83      0.83      0.83        76

[[33  6]
 [ 7 30]]
```

In [ ]:
```
""
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
import lightgbm as lgb

# Define the pipeline with hyperparameter tuning
param_grid = {
    'lgbmclassifier__num_leaves': [31, 62, 127],
    'lgbmclassifier__learning_rate': [ 0.2,0.119,0.21],
    'lgbmclassifier__n_estimators': [91,92],
    'lgbmclassifier__max_depth': [3, 5, 7],
    'lgbmclassifier__min_child_samples': [2, 5, 10],
    'lgbmclassifier__min_child_weight': [0.001, 0.1, 1.0],
    'lgbmclassifier__subsample': [0.5, 0.8, 1.0],
    'lgbmclassifier__colsample_bytree': [0.5, 0.8, 1.0],
    'lgbmclassifier__reg_alpha': [0.0, 0.1, 1.0],
    'lgbmclassifier__reg_lambda': [0.0, 0.1, 1.0],
    'lgbmclassifier__feature_fraction': [0.5, 0.8, 1.0],
    'lgbmclassifier__bagging_fraction': [0.5, 0.8, 1.0],
    'lgbmclassifier__bagging_freq': [1, 3, 5],
    'lgbmclassifier__early_stopping_round': [5, 10, 20]
}

pipeline = Pipeline(steps=[('columntransformer', preprocessor),
                           ('lgbmclassifier', lgb.LGBMClassifier())])

grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='f1_macro')
grid_search.fit(X_train, y_train)

print("Best parameters:", grid_search.best_params_)
print("Best score:", grid_search.best_score_)
```

```
C:\Users\saira\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.py:97: DataC
onversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\saira\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.py:132: Data
ConversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, dtype=self.classes_.dtype, warn=True)
C:\Users\saira\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.py:97: DataC
onversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\saira\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.py:132: Data
ConversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, dtype=self.classes_.dtype, warn=True)
C:\Users\saira\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.py:97: DataC
onversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples, ), for example using ravel().
```

In [ ]:
```
y_pred = grid_search.predict(X_test)
```

In [ ]:
```
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_pred,y_test))
print(confusion_matrix(y_pred,y_test))
```

In [ ]:
```
1
```