

```
In [27]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv("heart_disease_data.csv")
data
```

```
Out[27]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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
...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

```
In [29]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv("heart_disease_data.csv")
data.shape
```

```
Out[29]: (303, 14)
```

```
In [30]: data.duplicated().sum()
```

```
Out[30]: 1
```

```
In [31]: data.isnull().sum()
```

```
Out[31]: age      0
sex      0
cp       0
trestbps 0
chol     0
fbs      0
restecg  0
thalach  0
exang    0
oldpeak  0
slope    0
ca       0
thal     0
target   0
dtype: int64
```

```
In [32]: data.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   trestbps    303 non-null   int64
4   chol        303 non-null   int64
5   fbs         303 non-null   int64
6   restecg     303 non-null   int64
7   thalach     303 non-null   int64
8   exang       303 non-null   int64
9   oldpeak     303 non-null   float64
10  slope       303 non-null   int64
11  ca          303 non-null   int64
12  thal        303 non-null   int64
13  target      303 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

```
In [33]: data.describe()
```

Out[33]:	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000

```
In [34]: from sklearn import preprocessing
le = preprocessing.LabelEncoder()
data["age"]=le.fit_transform(data["age"])
```

```
In [35]: data
```

Out[35]:	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	29	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	3	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	7	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	22	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	23	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...
298	23	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	11	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	34	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	23	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	23	0	1	130	236	0	0	174	0	0.0	1	1	2	0

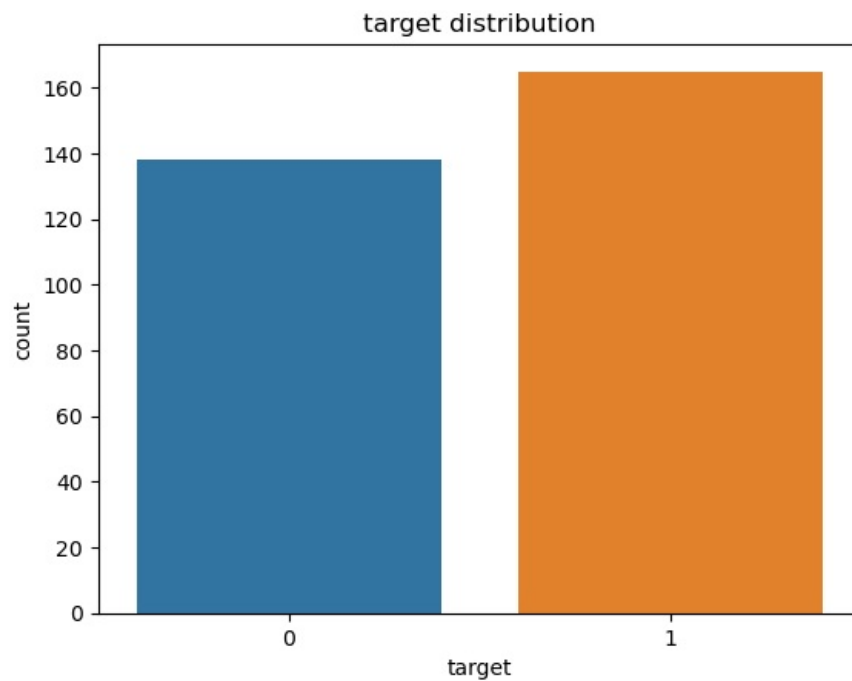
303 rows × 14 columns

```
In [36]: data.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   trestbps    303 non-null    int64
 4   chol        303 non-null    int64
 5   fbs         303 non-null    int64
 6   restecg     303 non-null    int64
 7   thalach     303 non-null    int64
 8   exang       303 non-null    int64
 9   oldpeak     303 non-null    float64
10   slope       303 non-null    int64
11   ca          303 non-null    int64
12   thal        303 non-null    int64
13   target      303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

```
In [37]: #distribution of Target Variable
import matplotlib.pyplot as plt
sns.countplot(x="target", data=data)
plt.title("target distribution")
```

```
Out[37]: Text(0.5, 1.0, 'target distribution')
```



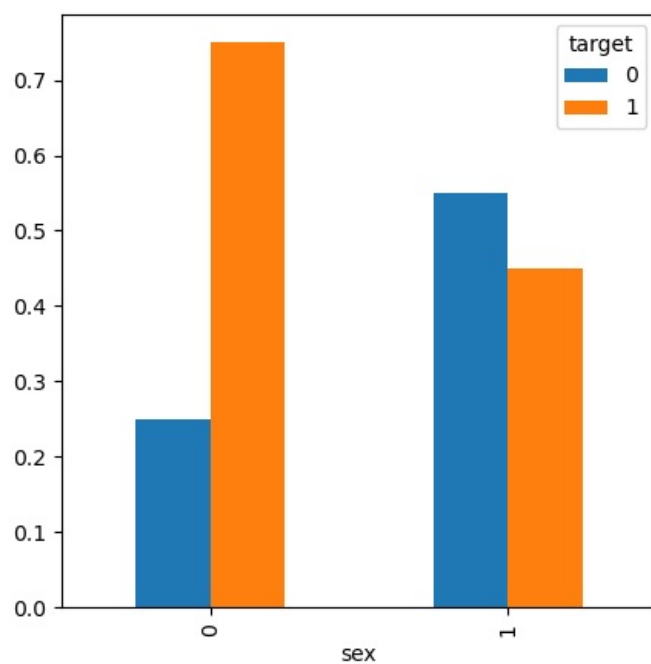
```
In [38]: data["target"].value_counts()
```

```
Out[38]: 1    165  
0     138  
Name: target, dtype: int64
```

```
In [39]: #function for plotting  
def plot(col, data=data):  
    return data.groupby(col)["target"].value_counts(normalize=True).unstack().plot(kind="bar", figsize=(5,5))
```

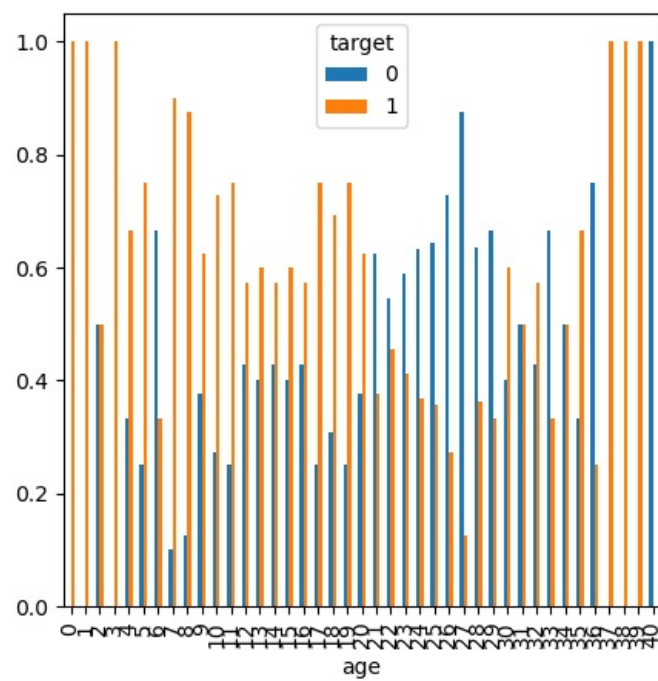
```
In [40]: plot("sex")
```

```
Out[40]: <AxesSubplot:xlabel='sex'>
```



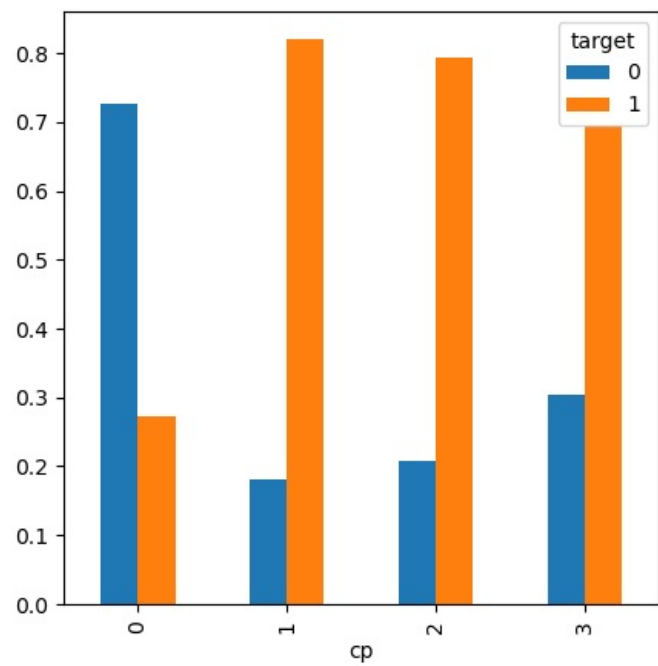
```
In [41]: plot("age")
```

```
Out[41]: <AxesSubplot:xlabel='age'>
```



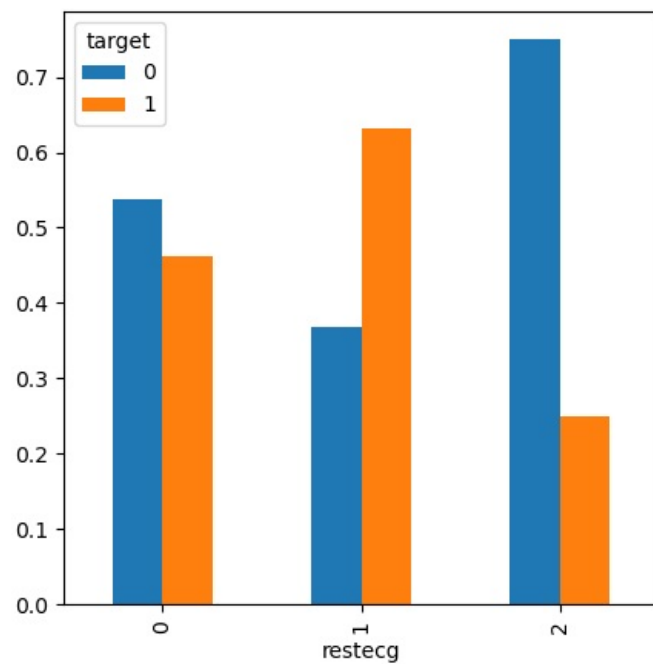
In [42]: `plot("cp")`

Out[42]: `<AxesSubplot:xlabel='cp'>`



In [43]: `plot("restecg")`

Out[43]: `<AxesSubplot:xlabel='restecg'>`



```
In [44]: df_new=data.drop(columns=["sex","age","cp"])
df_new
```

```
Out[44]:
```

	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	145	233	1	0	150	0	2.3	0	0	1	1
1	130	250	0	1	187	0	3.5	0	0	2	1
2	130	204	0	0	172	0	1.4	2	0	2	1
3	120	236	0	1	178	0	0.8	2	0	2	1
4	120	354	0	1	163	1	0.6	2	0	2	1
...
298	140	241	0	1	123	1	0.2	1	0	3	0
299	110	264	0	1	132	0	1.2	1	0	3	0
300	144	193	1	1	141	0	3.4	1	2	3	0
301	130	131	0	1	115	1	1.2	1	1	3	0
302	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 11 columns

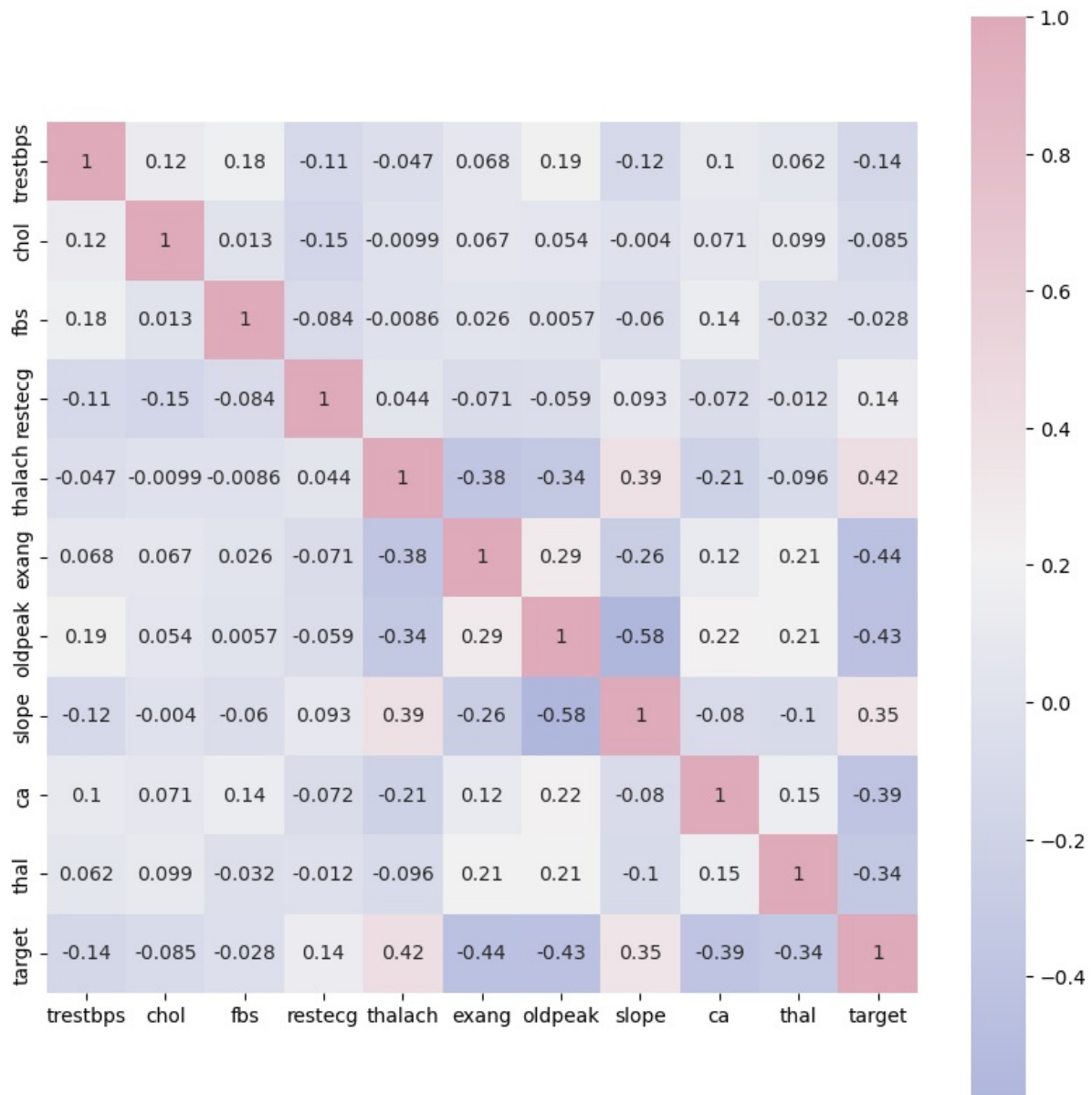
```
In [45]: #Finding correlation
cn=df_new.corr()
cn
```

Out[45]:

	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
trestbps	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.067616	0.193216	-0.121475	0.101389	0.062210	-0.144931
chol	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.067023	0.053952	-0.004038	0.070511	0.098803	-0.085239
fbs	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	0.005747	-0.059894	0.137979	-0.032019	-0.028046
restecg	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.070733	-0.058770	0.093045	-0.072042	-0.011981	0.137230
thalach	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.378812	-0.344187	0.386784	-0.213177	-0.096439	0.421741
exang	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.000000	0.288223	-0.257748	0.115739	0.206754	-0.436757
oldpeak	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.288223	1.000000	-0.577537	0.222682	0.210244	-0.430696
slope	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.257748	-0.577537	1.000000	-0.080155	-0.104764	0.345877
ca	0.101389	0.070511	0.137979	-0.072042	-0.213177	0.115739	0.222682	-0.080155	1.000000	0.151832	-0.391724
thal	0.062210	0.098803	-0.032019	-0.011981	-0.096439	0.206754	0.210244	-0.104764	0.151832	1.000000	-0.344029
target	-0.144931	-0.085239	-0.028046	0.137230	0.421741	-0.436757	-0.430696	0.345877	-0.391724	-0.344029	1.000000

In [46]:

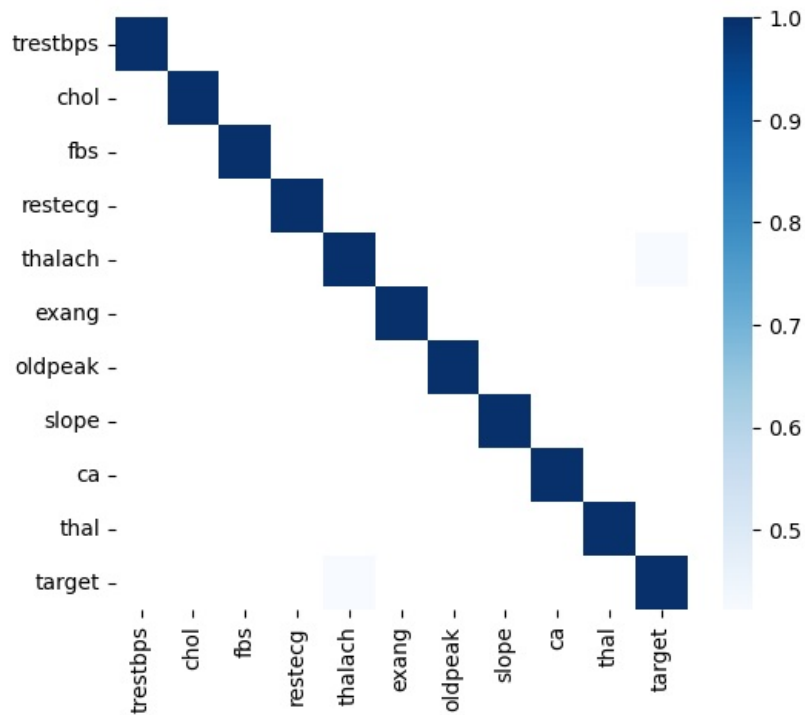
```
#correlation
cmap = sns.diverging_palette(260,-10, s=50, l=75, n=6, as_cmap=True)
plt.subplots(figsize=(10,10))
sns.heatmap(cn,cmap=cmap,annot=True,square=True)
plt.show()
```



In [47]:

```
df = cn[cn>=.40]
plt.figure(figsize=(6,5))
sns.heatmap(df, cmap="Blues")
```

Out[47]: <AxesSubplot:>



```
In [48]: df_new["choltrestbps"]=df_new["chol"]*df_new["trestbps"]
df_new
```

```
Out[48]:
```

	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target	choltrestbps
0	145	233	1	0	150	0	2.3	0	0	1	1	33785
1	130	250	0	1	187	0	3.5	0	0	2	1	32500
2	130	204	0	0	172	0	1.4	2	0	2	1	26520
3	120	236	0	1	178	0	0.8	2	0	2	1	28320
4	120	354	0	1	163	1	0.6	2	0	2	1	42480
...
298	140	241	0	1	123	1	0.2	1	0	3	0	33740
299	110	264	0	1	132	0	1.2	1	0	3	0	29040
300	144	193	1	1	141	0	3.4	1	2	3	0	27792
301	130	131	0	1	115	1	1.2	1	1	3	0	17030
302	130	236	0	0	174	0	0.0	1	1	2	0	30680

303 rows × 12 columns

```
In [52]: x = df_new.drop("target", axis = 1)
y = df_new["target"]
```

```
In [62]: from imblearn.over_sampling import ADASYN
adasyn = ADASYN(random_state=42)
x,y = adasyn.fit_resample(x,y)
```

```
-----
ModuleNotFoundError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_436\2895492658.py in <module>
----> 1 from imblearn.over_sampling import ADASYN
      2 adasyn = ADASYN(random_state=42)
      3 x,y = adasyn.fit_resample(x,y)

ModuleNotFoundError: No module named 'imblearn'
```

```
In [54]: len(x)
```

```
Out[54]: 303
```

```
In [55]: #splitting data for training and testing
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest=train_test_split(x,y,test_size=0.25, random_state=0)
```

```
In [56]: #fitting training data to the model
```

```
from sklearn.linear_model import LogisticRegression
lr_model=LogisticRegression(random_state=0)
lr_model.fit(xtrain, ytrain)
```

Out[56]: LogisticRegression(random_state=0)

In [57]: *#predicting result using testing data*
y_pred = lr_model.predict(xtest)
y_pred

Out[57]: array([0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1,
0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0,
0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 0, 0, 1], dtype=int64)

In [58]: *#model accuracy*
from sklearn.metrics import classification_report, accuracy_score, f1_score
lr_cr=classification_report(ytest,y_pred)
print(lr_cr)

	precision	recall	f1-score	support
0	0.81	0.67	0.73	33
1	0.78	0.88	0.83	43
accuracy			0.79	76
macro avg	0.80	0.78	0.78	76
weighted avg	0.79	0.79	0.79	76

In [59]: *#Decision Tree*
from sklearn.tree import DecisionTreeClassifier
dt_model = DecisionTreeClassifier(criterion = "entropy", random_state=0)
dt_model.fit(xtrain,ytrain)

Out[59]: DecisionTreeClassifier(criterion='entropy', random_state=0)

In [60]: y_dt_pred = dt_model.predict(xtest)
y_dt_pred

Out[60]: array([0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1,
0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0,
0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1,
0, 0, 1, 1, 1, 1, 1, 0, 0, 1], dtype=int64)

In [61]: *#model accuracy*
dt_cr = classification_report(ytest,y_dt_pred)
print(dt_cr)

	precision	recall	f1-score	support
0	0.69	0.73	0.71	33
1	0.78	0.74	0.76	43
accuracy			0.74	76
macro avg	0.73	0.74	0.73	76
weighted avg	0.74	0.74	0.74	76