ESSENTIALS OF MACHINE LEARNING (EML)

PBL- ACTIVITY

TITLE : HEART DISEASE

TEAM:
23EG107F30- YASANI SHREYAS REDDY
23EG107F35- PRANATHI KATAPALLY
23EG107F39 - KATTA AKASH REDDY
23EG107F52 - G.SRI SAHASRA
23EG107F53 - G.SRI SAHAJA

DATA SET

Data set link : #!/bin/bashkaggle datasets download shrutikubade/heart-dieases-dataset

- ❖ Number of records : 303 records
- ❖ Number of columns :14
- ❖ Number of classes: 2

Class(0): No heart disease

Class (1): Heart disease present

The Dependent features or the target variable includes: No heart disease and heart disease presence i.e 0 and 1. Dependent variable is ‘target’.

The independent features includes: ‘age’,’sex’,’cp’,’trestbps’,’chol’,’fbs’, ‘restecg’,’thalach’,’exang’,’oldpeak’,’slope’,’ca’,’thal’.

ATTRIBUTE INFORMATION

1.age : Age of the patient (in years)

2.sex: Sex of the patient (1 = male, 0 = female)

3. cp: Chest pain type (categorical: 0–3)

4. trestbps: Resting blood pressure

5. chol:Serum cholesterol

6. **fbs** – Fasting blood sugar >120 mg/dl (1 = true, 0 = false)

7. restecg: Resting electrocardiographic results

- 0: Normal
- 1: Having ST-T wave abnormality
- 2: Showing probable/definite left ventricular hypertrophy

8. thalach: Maximum heart rate achieved

9. exang: Exercise-induced angina (1 = yes, 0 = no)

10. oldpeak: ST depression induced by exercise relative to rest (float value)

11. slope: The slope of the peak exercise ST segment

- 0: Upsloping
- 1: Flat
- 2: Downsloping

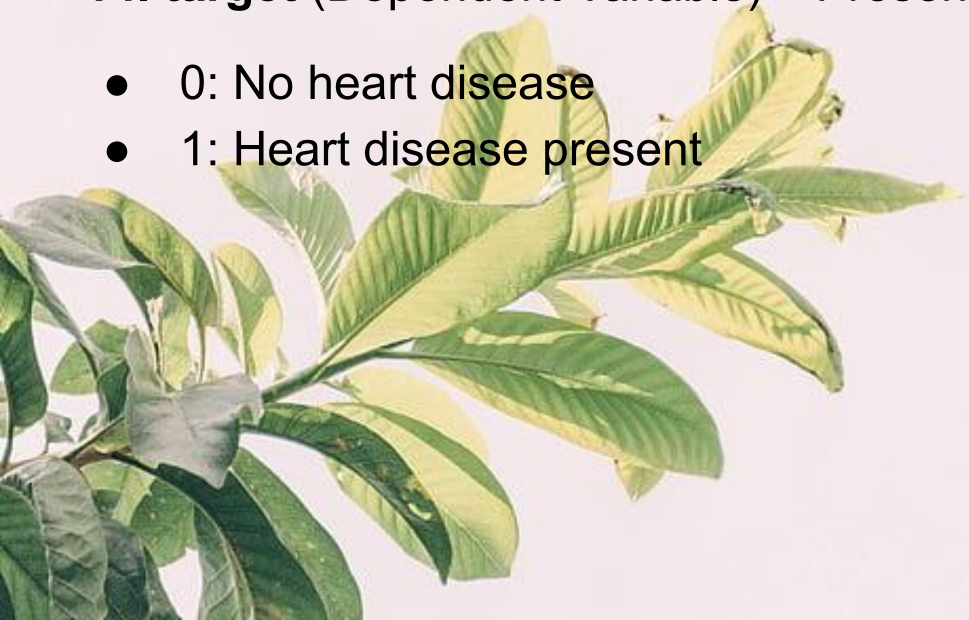
12.ca – Number of major vessels (0–3) colored by fluoroscopy

13.thal – Thalassemia status

- 1: Normal
- 2: Fixed defect
- 3: Reversible defect

14. target (Dependent Variable) – Presence of heart disease

- 0: No heart disease
- 1: Heart disease present



IMPORTING THE PACKAGES

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

IMPORTING THE DATASET

```
df=pd.read_csv("/content/Day9_Heart_Disease_Data.csv")
df.head()
```

	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

MISSING VALUES INFORMATION

- ★ There are **no missing values** in any of the 14 columns.
- ★ All 303 records are **complete**.
- ★ No imputation
- ★ Dataset is already clean and ready for preprocessing (encoding, scaling, etc.).

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

	0
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

UNDERSTANDING THE DATASET

```
columns=['cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal','age', 'trestbps', 'chol', 'thalach', 'oldpeak']
unique_values={col:df[col].unique() for col in columns}
unique_values
```

The features sex, cp, fbs, restecg, exang, slope, ca, thal are Categorical Feature and age, trestbps, chol, thalach, oldpeak are Numerical Features.

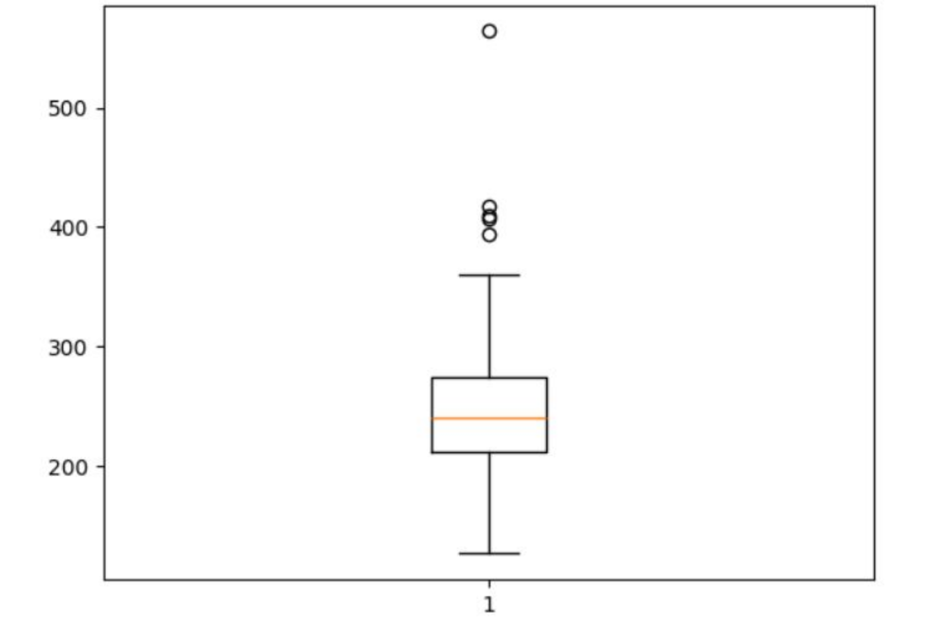
```
{'cp': array([3, 2, 1, 0]),
 'fbs': array([1, 0]),
 'restecg': array([0, 1, 2]),
 'exang': array([0, 1]),
 'slope': array([0, 2, 1]),
 'ca': array([0, 2, 1, 3, 4]),
 'thal': array([1, 2, 3, 0]),
 'age': array([63, 37, 41, 56, 57, 44, 52, 54, 48, 49, 64, 58, 50, 66, 43, 69, 59,
        42, 61, 40, 71, 51, 65, 53, 46, 45, 39, 47, 62, 34, 35, 29, 55, 60,
        67, 68, 74, 76, 70, 38, 77]),
 'trestbps': array([145, 130, 120, 140, 172, 150, 110, 135, 160, 105, 125, 142, 155,
        104, 138, 128, 108, 134, 122, 115, 118, 100, 124,  94, 112, 102,
        152, 101, 132, 148, 178, 129, 180, 136, 126, 106, 156, 170, 146,
        117, 200, 165, 174, 192, 144, 123, 154, 114, 164]),
 'chol': array([233, 250, 204, 236, 354, 192, 294, 263, 199, 168, 239, 275, 266,
        211, 283, 219, 340, 226, 247, 234, 243, 302, 212, 175, 417, 197,
        198, 177, 273, 213, 304, 232, 269, 360, 308, 245, 208, 264, 321,
        325, 235, 257, 216, 256, 231, 141, 252, 201, 222, 260, 182, 303,
        265, 309, 186, 203, 183, 220, 209, 258, 227, 261, 221, 205, 240,
        318, 298, 564, 277, 214, 248, 255, 207, 223, 288, 160, 394, 315,
        246, 244, 270, 195, 196, 254, 126, 313, 262, 215, 193, 271, 268,
        267, 210, 295, 306, 178, 242, 180, 228, 149, 278, 253, 342, 157,
        286, 229, 284, 224, 206, 167, 230, 335, 276, 353, 225, 330, 290,
        172, 305, 188, 282, 185, 326, 274, 164, 307, 249, 341, 407, 217,
        174, 281, 289, 322, 299, 300, 293, 184, 409, 259, 200, 327, 237,
        218, 319, 166, 311, 169, 187, 176, 241, 131]),
```

```
 'thalach': array([150, 187, 172, 178, 163, 148, 153, 173, 162, 174, 160, 139, 171,
        144, 158, 114, 151, 161, 179, 137, 157, 123, 152, 168, 140, 188,
        125, 170, 165, 142, 180, 143, 182, 156, 115, 149, 146, 175, 186,
        185, 159, 130, 190, 132, 147, 154, 202, 166, 164, 184, 122, 169,
        138, 111, 145, 194, 131, 133, 155, 167, 192, 121,  96, 126, 105,
        181, 116, 108, 129, 120, 112, 128, 109, 113,  99, 177, 141, 136,
         97, 127, 103, 124,  88, 195, 106,  95, 117,  71, 118, 134,  90]),
 'oldpeak': array([2.3, 3.5, 1.4, 0.8, 0.6, 0.4, 1.3, 0. , 0.5, 1.6, 1.2, 0.2, 1.8,
        1. , 2.6, 1.5, 3. , 2.4, 0.1, 1.9, 4.2, 1.1, 2. , 0.7, 0.3, 0.9,
        3.6, 3.1, 3.2, 2.5, 2.2, 2.8, 3.4, 6.2, 4. , 5.6, 2.9, 2.1, 3.8,
        4.4])}
```

UNDERSTANDING THE DATASET

There are many outliers in Cholesterol (chol) has the most outliers (values > 400). So we can not impute mean.

```
[25] import matplotlib.pyplot as plt
     plt.boxplot(df['chol'])
```



INFORMATION ABOUT THE DATASET

Now we need to convert all the categorical variables into numerical using encoding.

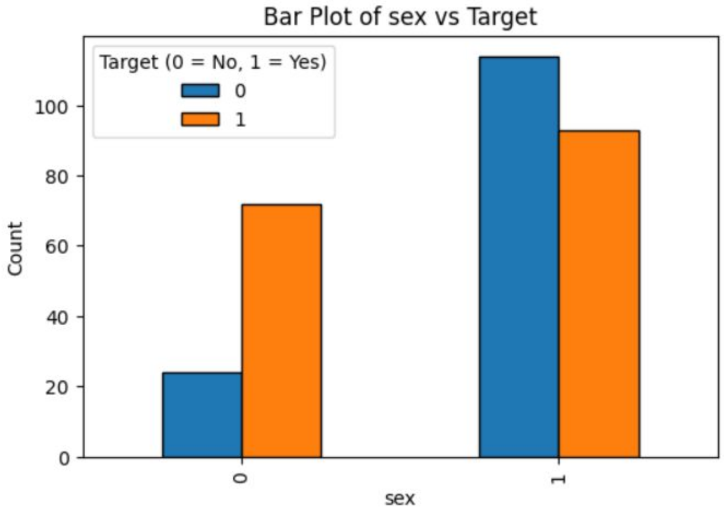
```
0s  [ ]  df.info()
```

```
<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   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
```

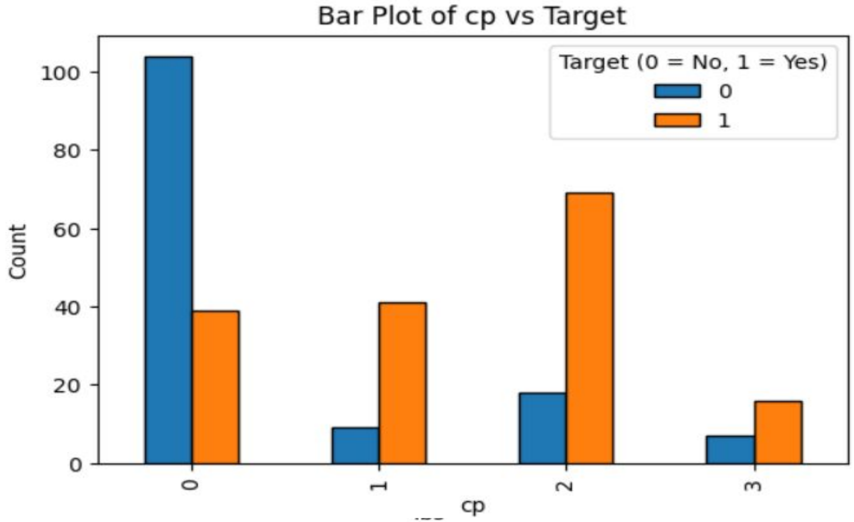
BAR PLOTS (Categorical variables vs target variables)

```
import pandas as pd
import matplotlib.pyplot as plt
cat_cols = ["sex", "cp", "fbs", "restecg", "exang", "slope", "ca", "thal"]
for col in cat_cols:
    plt.figure(figsize=(6,4))
    pd.crosstab(df[col], df["target"]).plot(kind="bar", figsize=(6,4), edgecolor="black")
    plt.title(f"Bar Plot of {col} vs Target")
    plt.xlabel(col)
    plt.ylabel("Count")
    plt.legend(title="Target (0 = No, 1 = Yes)")
    plt.show()
```

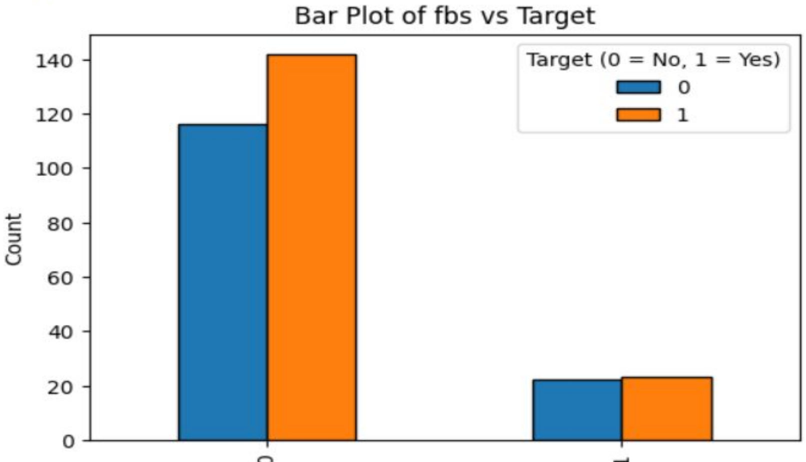
<Figure size 600x400 with 0 Axes>



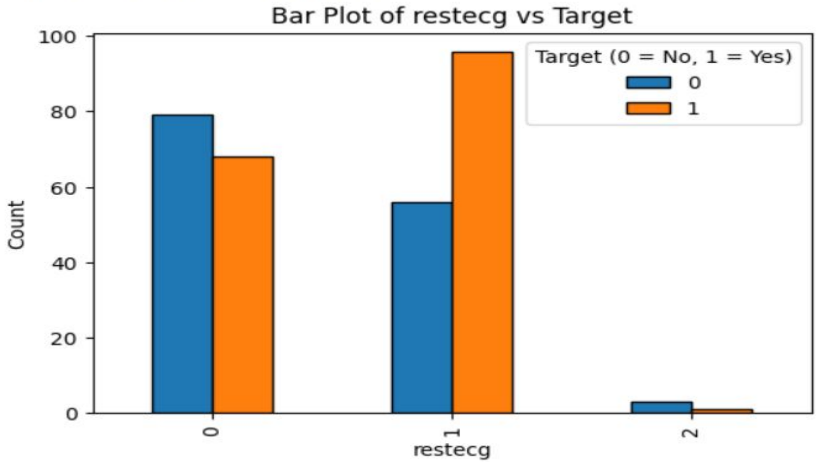
<Figure size 600x400 with 0 Axes>



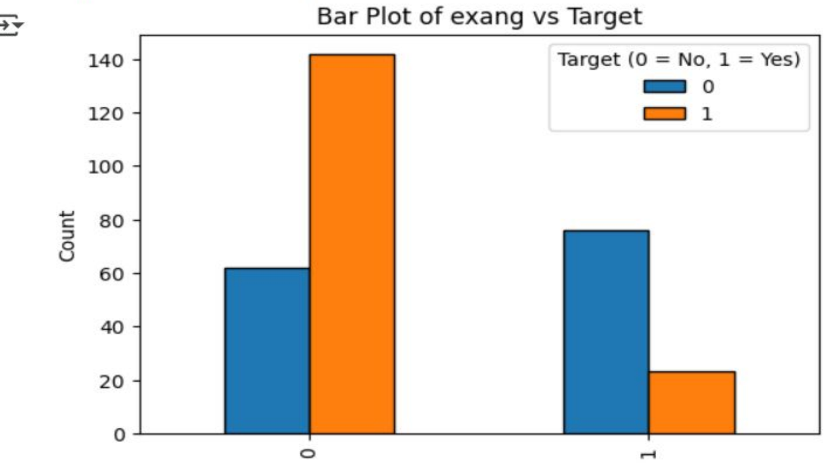
<Figure size 600x400 with 0 Axes>



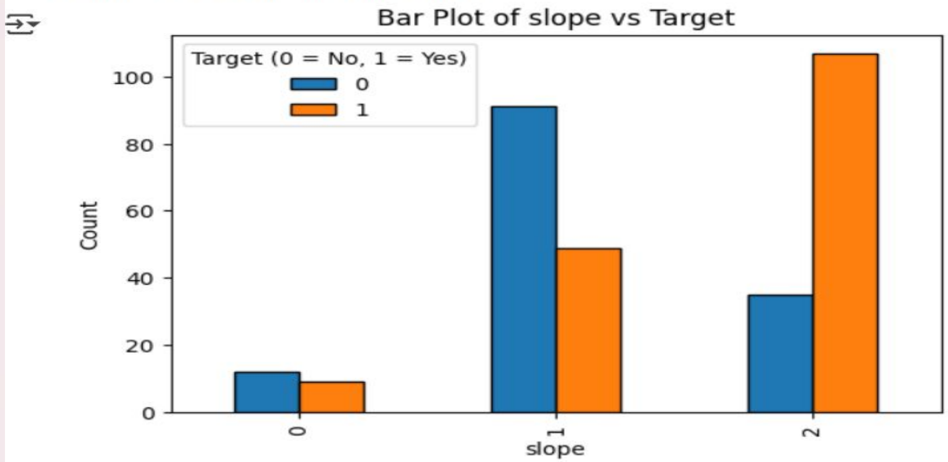
<Figure size 600x400 with 0 Axes>



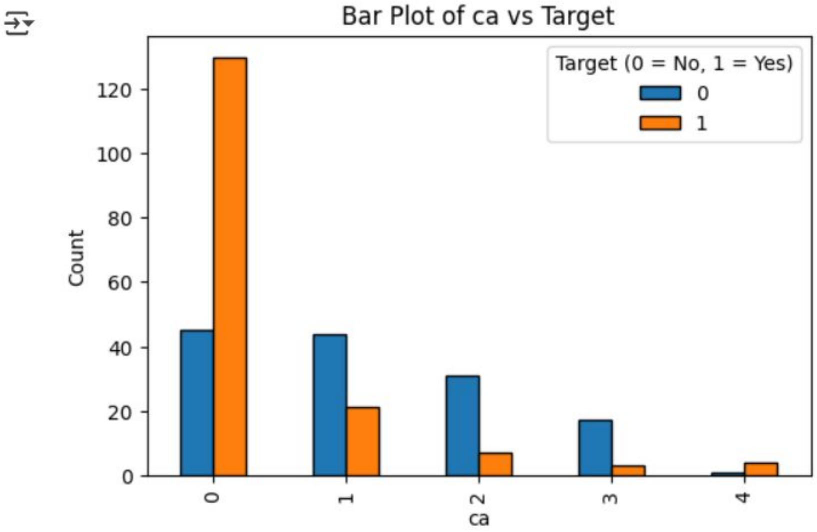
<Figure size 600x400 with 0 Axes>



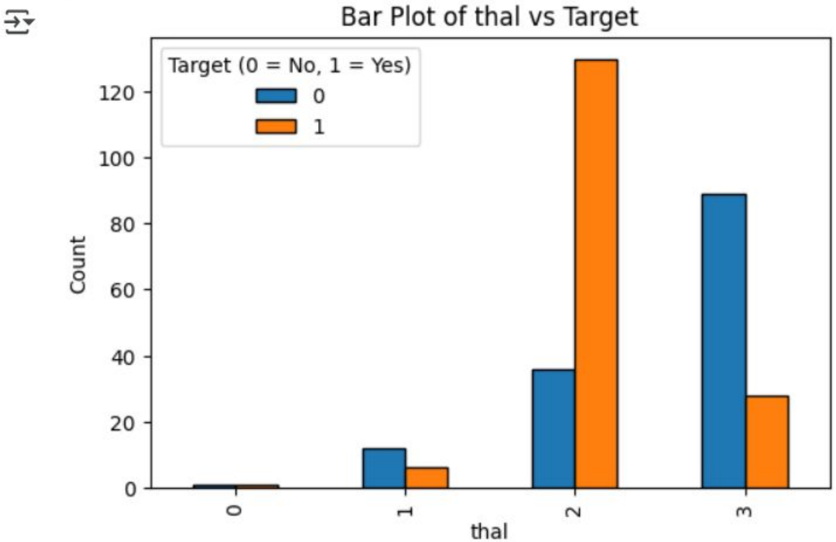
<Figure size 600x400 with 0 Axes>



<Figure size 600x400 with 0 Axes>



<Figure size 600x400 with 0 Axes>



EXPLORATORY DATA ANALYSIS

```
df.describe()
```

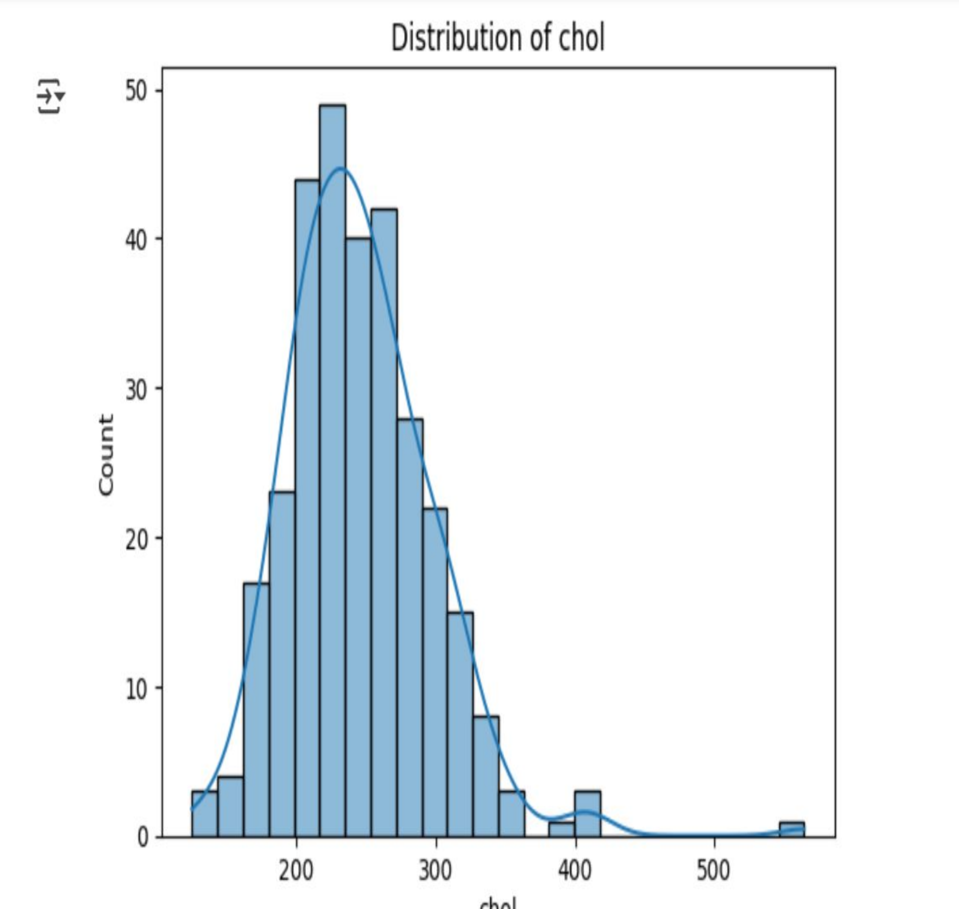
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604
std	9.082101	0.466011	1.032052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

oldpeak	slope	ca	thal	target
3.000000	303.000000	303.000000	303.000000	303.000000
1.039604	1.399340	0.729373	2.313531	0.544554
1.161075	0.616226	1.022606	0.612277	0.498835
0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	1.000000	0.000000	2.000000	0.000000
0.800000	1.000000	0.000000	2.000000	1.000000
1.600000	2.000000	1.000000	3.000000	1.000000
6.200000	2.000000	4.000000	3.000000	1.000000

This data provides summary of mean,count,min,standard deviation.

UNIVARIATE ANALYSIS

```python
import seaborn as sns
sns.histplot(df['chol'],kde=True)
plt.title('Distribution of chol')
plt.show()
```



BI-VARIATE ANALYSIS

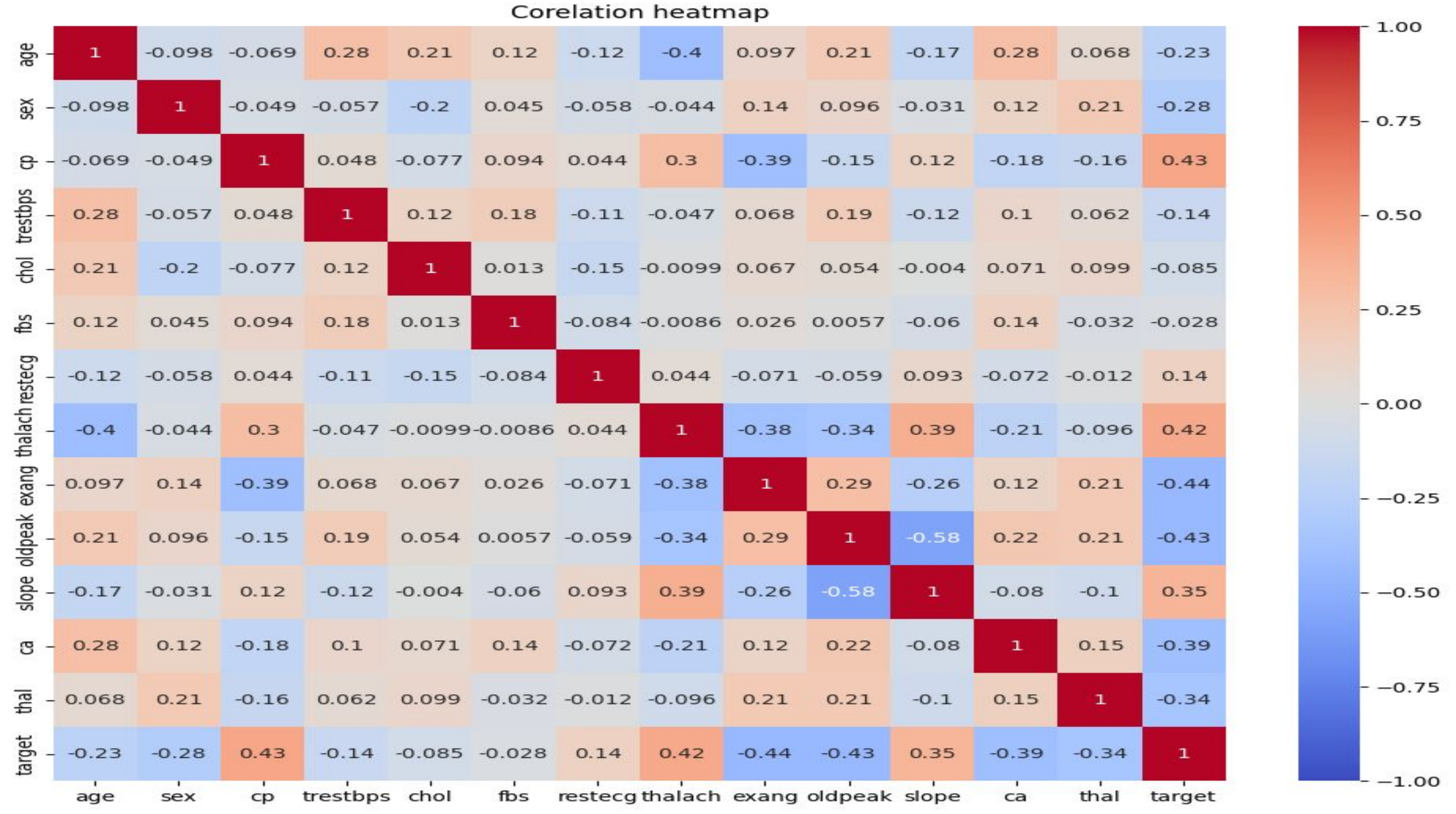
Understanding the correlation between different features.

```
[56] df.corr()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.096801	0.210013	-0.168814
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	0.141664	0.096093	-0.030711
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-0.394280	-0.149230	0.119717
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.067616	0.193216	-0.121475
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.067023	0.053952	-0.004038
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	0.005747	-0.059894
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.070733	-0.058770	0.093045
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.378812	-0.344187	0.386784
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.000000	0.288223	-0.257748
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.288223	1.000000	-0.577537
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.257748	-0.577537	1.000000

CORRELATION HEATMAP

```
plt.figure(figsize=(12,10))
sns.heatmap(df.corr(),annot=True,cmap='coolwarm',vmin=-1,vmax=1)
plt.title('Corelation heatmap')
plt.show()
```



Corelation heatmap
age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

PAIR PLOT

```
[61] sns.pairplot(df,hue='chol')
```

DIVIDING THE DATASET INTO DEPENDENT AND INDEPENDENT FEATURES

[66]

```python
import pandas as pd
# Dependent feature (Target)
y = df.loc[:, 'target']   # target column only

# Independent features (all columns except target)
X = df.iloc[:, df.columns != 'target']

print("Dependent Feature (y):")
print(y.head())

print("\n Independent Features (X):")
print(X.head())
```

DEPENDENT FEATURES

```
Dependent Feature (y):
0    1
1    1
2    1
3    1
4    1
Name: target, dtype: int64
```

INDEPENDENT FEATURES

```
Independent Features (X):
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   63    1   3       145   233    1        0      150      0      2.3      0
1   37    1   2       130   250    0        1      187      0      3.5      0
2   41    0   1       130   204    0        0      172      0      1.4      2
3   56    1   1       120   236    0        1      178      0      0.8      2
4   57    0   0       120   354    0        1      163      1      0.6      2

   ca  thal
0   0     1
1   0     2
2   0     2
3   0     2
4   0     2
```

SPLITTING THE DATASET INTO TRAINING AND TESTING DATA

```python
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

```
((242, 13), (61, 13), (242,), (61,))
```

STANDARDIZATION

```
[82] from sklearn.preprocessing import StandardScaler
     scaler=StandardScaler()
     X_train=scaler.fit_transform(X_train)
     X_train
```

```
array([[-1.35679832,  0.72250438,  0.00809909, ...,  0.95390513,
        -0.68970073, -0.50904773],
       [ 0.38508599,  0.72250438, -0.97189094, ...,  0.95390513,
        -0.68970073,  1.17848036],
       [-0.92132724,  0.72250438,  0.98808912, ..., -0.69498803,
        -0.68970073, -0.50904773],
       ...,
       [ 1.58263146,  0.72250438,  1.96807914, ..., -0.69498803,
         0.32186034, -0.50904773],
       [-0.92132724,  0.72250438, -0.97189094, ...,  0.95390513,
        -0.68970073,  1.17848036],
       [ 0.92942484, -1.38407465,  0.00809909, ...,  0.95390513,
         1.33342142, -0.50904773]])
```

```python
X_test=scaler.transform(X_test)
X_test
```

```
 -1.04610909e+00, -2.29340266e-01,  1.47790748e+00,
 -1.93787048e-01, -6.94988026e-01,  3.21860343e-01,
  1.17848036e+00],
[-7.03591701e-01,  7.22504380e-01, -9.71890936e-01,
 -2.14066346e-02,  1.73802865e-01,  2.60891771e+00,
 -1.04610909e+00, -5.18701733e-03,  1.47790748e+00,
 -9.20864033e-01,  9.53905134e-01,  1.33342142e+00,
  1.17848036e+00],
[-1.03019501e+00,  7.22504380e-01, -9.71890936e-01,
  6.93132066e-01,  1.17975754e+00, -3.83300706e-01,
 -1.04610909e+00, -1.39678967e-01,  1.47790748e+00,
 -9.20864033e-01, -6.94988026e-01,  2.34498250e+00,
  1.17848036e+00],
[-1.46566609e+00,  7.22504380e-01,  8.09909113e-03,
  2.76317824e-01, -8.32151805e-01, -3.83300706e-01,
  8.43132697e-01, -8.12138713e-01, -6.76632341e-01,
 -9.20864033e-01, -6.94988026e-01, -6.89700735e-01,
 -2.19657581e+00],
[ 6.02821534e-01, -1.38407465e+00,  9.88089118e-01,
 -1.68866360e+00,  1.35058003e+00, -3.83300706e-01,
  8.43132697e-01,  4.43119480e-01, -6.76632341e-01,
 -9.20864033e-01,  9.53905134e-01,  3.21860343e-01,
 -5.09047728e-01],
[ 3.85085995e-01,  7.22504380e-01, -9.71890936e-01,
 -3.19131093e-01,  1.00893504e+00, -3.83300706e-01,
 -1.04610909e+00,  9.36256628e-01, -6.76632341e-01,
 -9.20864033e-01,  9.53905134e-01,  1.33342142e+00,
```

FITTING THE MODEL

```
[87] from sklearn.linear_model import LinearRegression
     regression=LinearRegression()
     regression.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

PREDICTING THE OUTPUT

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

```
array([ 0.20461609,  0.62932079,  0.71803669,  0.04394647,  0.93859572,
        0.82919673,  0.55248505, -0.31123783, -0.12686594,  0.50190798,
        0.67329639,  0.22858348,  0.83576074,  0.10160178,  1.1070706 ,
        0.90376346,  1.08827421,  0.21347573, -0.14247183, -0.02892816,
        0.60715419, -0.05665451,  0.35685167,  0.68434807,  0.88713565,
        0.60700687,  0.81935794,  0.53123747, -0.10013016,  0.8970951 ,
        0.05884203,  0.07332101, -0.14826886,  0.24195448,  0.69307817,
        0.19714348,  0.66820272,  0.77828265,  0.68233201,  0.75469071,
        0.49240619,  0.64012072,  0.75759582,  0.67302635,  0.74556992,
       -0.1916457 ,  0.66356747,  0.91773248,  0.22766617, -0.00575935,
        0.15521194, -0.13222102,  0.78206754,  1.04130301,  0.33798087,
       -0.2796243 ,  0.11850097,  0.94519132,  0.01120021, -0.218084  ,
        0.15742582])
```

CONFUSION MATRIX

```python
from sklearn.metrics import confusion_matrix
cm_logistic = confusion_matrix(y_test, y_pred_logistic)
print("Confusion Matrix:")
display(cm_logistic)
```

```
Confusion Matrix:
array([[25,  4],
       [ 5, 27]])
```

CLASSIFICATION REPORT

```
[102] from sklearn.metrics import  classification_report
      print(classification_report(y_test, y_pred_logistic))
```

```
              precision    recall  f1-score   support

           0       0.83      0.86      0.85        29
           1       0.87      0.84      0.86        32

    accuracy                           0.85        61
   macro avg       0.85      0.85      0.85        61
weighted avg       0.85      0.85      0.85        61
```


Thank
you