In [1]:

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

%matplotlib inline

import warnings
warnings.filterwarnings("ignore")
```

In [3]:

```
df=pd.read_csv("Heart Disease data.csv")
```

In [4]:

```
df.head()
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1025 non-null   int64  
 1   sex       1025 non-null   int64  
 2   cp        1025 non-null   int64  
 3   trestbps  1025 non-null   int64  
 4   chol      1025 non-null   int64  
 5   fbs       1025 non-null   int64  
 6   restecg   1025 non-null   int64  
 7   thalach   1025 non-null   int64  
 8   exang     1025 non-null   int64  
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64  
 11  ca        1025 non-null   int64  
 12  thal      1025 non-null   int64  
 13  target    1025 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
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 [7]:
```
df.describe().T
```

Out[7]:

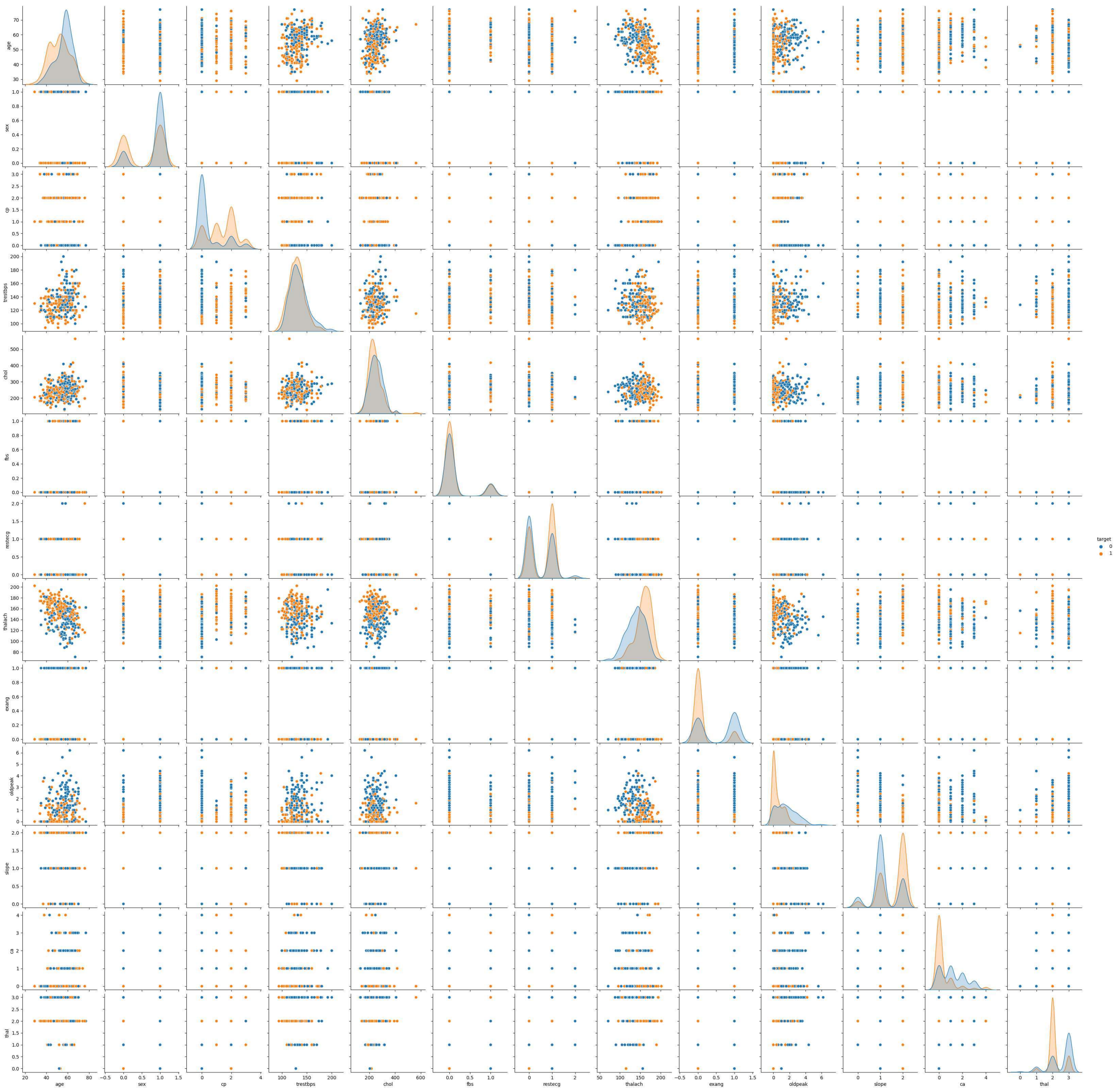
	count	mean	std	min	25%	50%	75%	max
age	1025.0	54.434146	9.072290	29.0	48.0	56.0	61.0	77.0
sex	1025.0	0.695610	0.460373	0.0	0.0	1.0	1.0	1.0
cp	1025.0	0.942439	1.029641	0.0	0.0	1.0	2.0	3.0
trestbps	1025.0	131.611707	17.516718	94.0	120.0	130.0	140.0	200.0
chol	1025.0	246.000000	51.592510	126.0	211.0	240.0	275.0	564.0
fbs	1025.0	0.149268	0.356527	0.0	0.0	0.0	0.0	1.0
restecg	1025.0	0.529756	0.527878	0.0	0.0	1.0	1.0	2.0
thalach	1025.0	149.114146	23.005724	71.0	132.0	152.0	166.0	202.0
exang	1025.0	0.336585	0.472772	0.0	0.0	0.0	1.0	1.0
oldpeak	1025.0	1.071512	1.175053	0.0	0.0	0.8	1.8	6.2
slope	1025.0	1.385366	0.617755	0.0	1.0	1.0	2.0	2.0
ca	1025.0	0.754146	1.030798	0.0	0.0	0.0	1.0	4.0
thal	1025.0	2.323902	0.620660	0.0	2.0	2.0	3.0	3.0
target	1025.0	0.513171	0.500070	0.0	0.0	1.0	1.0	1.0

In [8]:
```
df.columns
```

Out[8]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

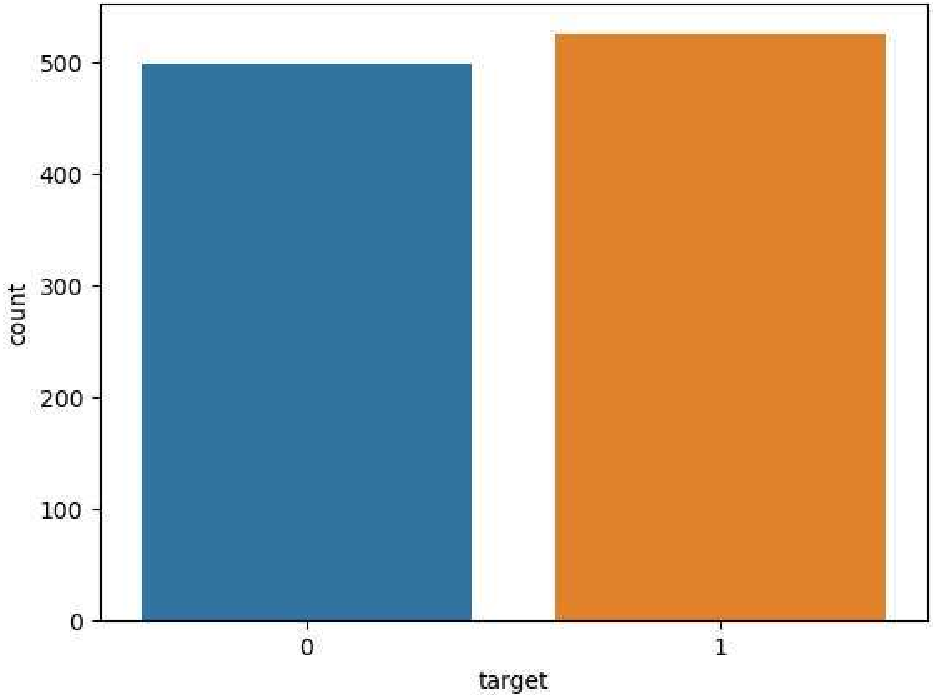
In [9]:
```
sns.pairplot(df,hue='target')
```

Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x1e62053f290>
```



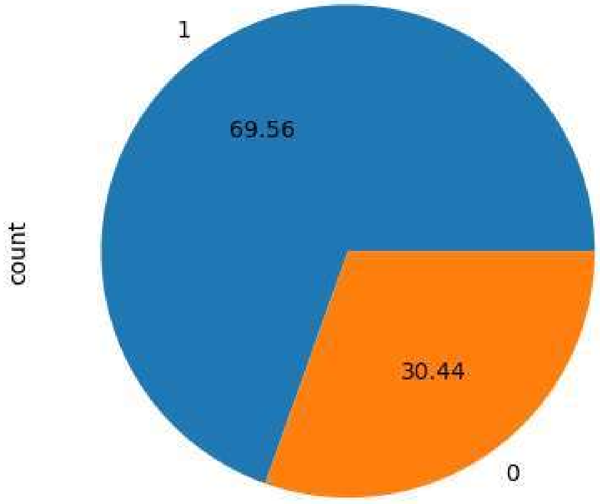
```
In [10]: sns.countplot(x='target', data = df)
```

```
Out[10]: <Axes: xlabel='target', ylabel='count'>
```



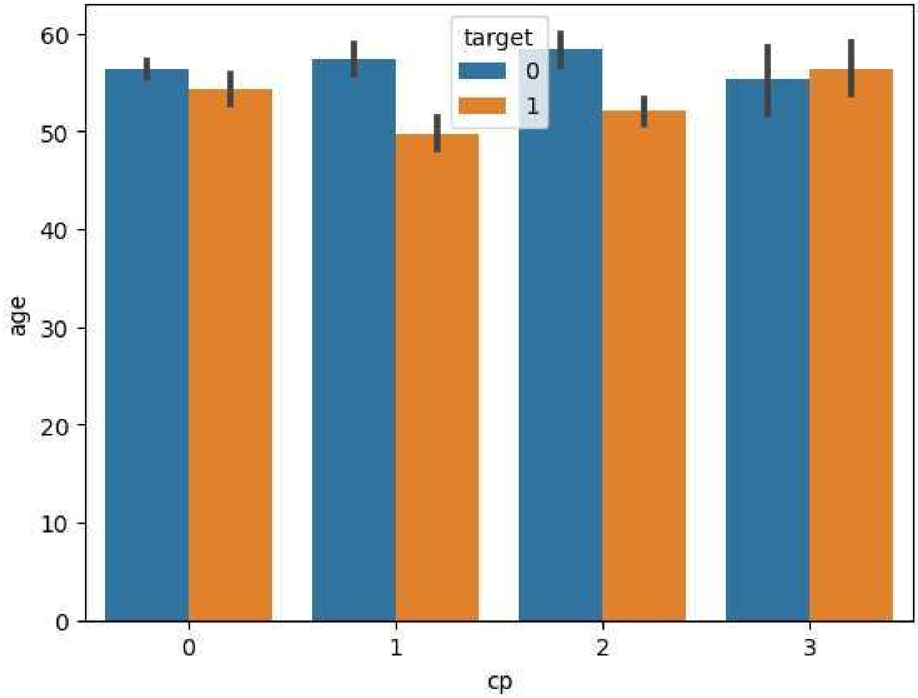
```
In [11]: df['sex'].value_counts().plot.pie(autopct='%.2f')
```

```
Out[11]: <Axes: ylabel='count'>
```



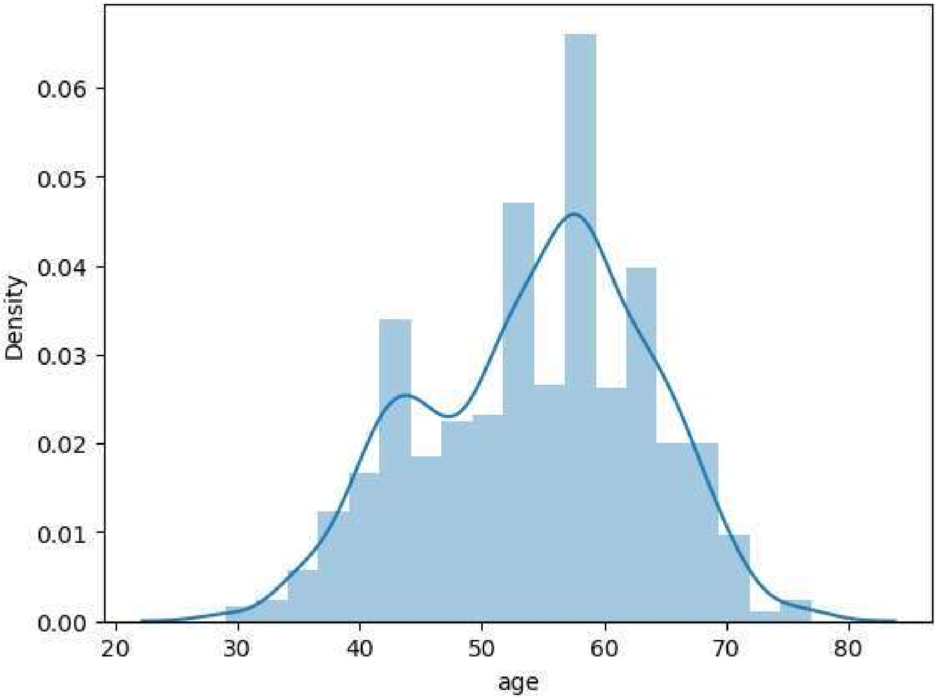
```
In [12]: sns.barplot(y='age',x='cp',data=df,hue='target')
```

```
Out[12]: <Axes: xlabel='cp', ylabel='age'>
```



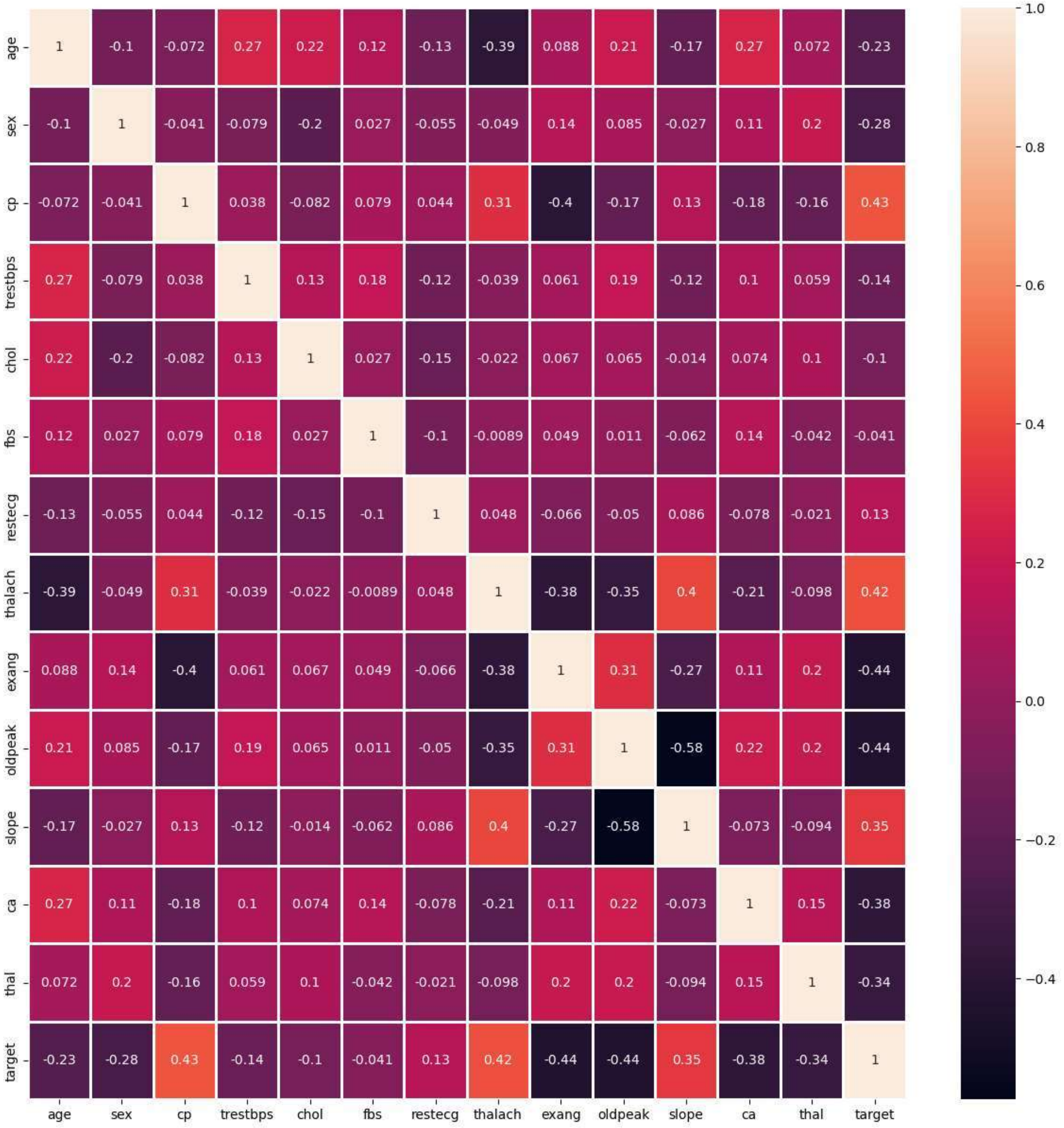
```
In [13]: sns.distplot(df['age'], kde=True)
```

```
Out[13]: <Axes: xlabel='age', ylabel='Density'>
```



In [14]:
```
corr=df.corr()
plt.figure(figsize= (15,15))
sns.heatmap(corr, linewidth=1,annot=True,linecolor='white')
```

Out[14]:
```
<Axes: >
```



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1	-0.1	-0.072	0.27	0.22	0.12	-0.13	-0.39	0.088	0.21	-0.17	0.27	0.072	-0.23
sex	-0.1	1	-0.041	-0.079	-0.2	0.027	-0.055	-0.049	0.14	0.085	-0.027	0.11	0.2	-0.28
cp	-0.072	-0.041	1	0.038	-0.082	0.079	0.044	0.31	-0.4	-0.17	0.13	-0.18	-0.16	0.43
trestbps	0.27	-0.079	0.038	1	0.13	0.18	-0.12	-0.039	0.061	0.19	-0.12	0.1	0.059	-0.14
chol	0.22	-0.2	-0.082	0.13	1	0.027	-0.15	-0.022	0.067	0.065	-0.014	0.074	0.1	-0.1
fbs	0.12	0.027	0.079	0.18	0.027	1	-0.1	-0.0089	0.049	0.011	-0.062	0.14	-0.042	-0.041
restecg	-0.13	-0.055	0.044	-0.12	-0.15	-0.1	1	0.048	-0.066	-0.05	0.086	-0.078	-0.021	0.13
thalach	-0.39	-0.049	0.31	-0.039	-0.022	-0.0089	0.048	1	-0.38	-0.35	0.4	-0.21	-0.098	0.42
exang	0.088	0.14	-0.4	0.061	0.067	0.049	-0.066	-0.38	1	0.31	-0.27	0.11	0.2	-0.44
oldpeak	0.21	0.085	-0.17	0.19	0.065	0.011	-0.05	-0.35	0.31	1	-0.58	0.22	0.2	-0.44
slope	-0.17	-0.027	0.13	-0.12	-0.014	-0.062	0.086	0.4	-0.27	-0.58	1	-0.073	-0.094	0.35
ca	0.27	0.11	-0.18	0.1	0.074	0.14	-0.078	-0.21	0.11	0.22	-0.073	1	0.15	-0.38
thal	0.072	0.2	-0.16	0.059	0.1	-0.042	-0.021	-0.098	0.2	0.2	-0.094	0.15	1	-0.34
target	-0.23	-0.28	0.43	-0.14	-0.1	-0.041	0.13	0.42	-0.44	-0.44	0.35	-0.38	-0.34	1

In [15]:
```
X = df.drop('target', axis=1)
y = df['target']
```

In [16]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [17]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

In [18]:

```
from sklearn.linear_model import  LogisticRegression
```

In [22]:

```
lr=LogisticRegression()
model_lr=lr.fit(X_train, y_train)
pred_lr = model_lr.predict(X_test)
```

In [26]:

```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_scor
cm_lr=confusion_matrix(y_test,pred_lr)
print("Confusion Matrix   :", cm_lr)
accuracy_lr = accuracy_score(y_test, pred_lr)
print("Accuracy   :", accuracy_lr)
precision_lr = precision_score(y_test, pred_lr)
print("Precision :", precision_lr)
recall_lr = recall_score(y_test, pred_lr)
print("Recall    :", recall_lr)
F1_score_lr = f1_score(y_test, pred_lr)
print("F1-score  :", F1_score_lr)
```

```
Confusion Matrix   : [[112  47]
 [ 16 133]]
Accuracy   : 0.7954545454545454
Precision : 0.7388888888888889
Recall    : 0.8926174496644296
F1-score  : 0.8085106382978723
```

In [27]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [29]:

```
rf = RandomForestClassifier(n_estimators=300, random_state=42, max_depth=5)
rf.fit(X_train, y_train)
pred_rf = rf.predict(X_test)
```

In [33]:

```
cm_rf=confusion_matrix(y_test,pred_rf)
print("Confusion Matrix   :", cm_rf)
accuracy_rf = accuracy_score(y_test, pred_rf)
print("Accuracy   :", accuracy_rf)
precision_rf = precision_score(y_test, pred_rf)
print("Precision :", precision_rf)
recall_rf = recall_score(y_test, pred_rf)
print("Recall    :", recall_rf)
F1_score_rf = f1_score(y_test, pred_rf)
print("F1-score  :", F1_score_rf)
```

```
Confusion Matrix   : [[128  31]
 [  8 141]]
Accuracy   : 0.8733766233766234
Precision : 0.8197674418604651
Recall    : 0.9463087248322147
F1-score  : 0.8785046728971962
```

In [34]:

```
from sklearn.svm import SVC
```

In [36]:

```
svc =  SVC(C=2)
svc.fit(X_train, y_train)
pred_svm = svc.predict(X_test)
```

In [38]:

```
cm_svm=confusion_matrix(y_test,pred_svm)
print("Confusion Matrix   :", cm_svm)
accuracy_svm = accuracy_score(y_test, pred_svm)
print("Accuracy   :", accuracy_svm)
precision_svm = precision_score(y_test, pred_svm)
print("Precision :", precision_svm)
recall_svm = recall_score(y_test, pred_svm)
print("Recall    :", recall_svm)
F1_score_svm = f1_score(y_test, pred_svm)
print("F1-score  :", F1_score_svm)
```

```
Confusion Matrix   : [[142  17]
 [  7 142]]
Accuracy   : 0.922077922077922
Precision : 0.8930817610062893
Recall    : 0.9530201342281879
F1-score  : 0.922077922077922
```

In [39]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [41]:

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
pred_knn = knn.predict(X_test)
```

In [43]:

```
cm_knn=confusion_matrix(y_test,pred_knn)
print("Confusion Matrix   :", cm_knn)
accuracy_knn = accuracy_score(y_test, pred_knn)
print("Accuracy   :", accuracy_knn)
precision_knn = precision_score(y_test, pred_knn)
print("Precision :", precision_knn)
recall_knn = recall_score(y_test, pred_knn)
print("Recall    :", recall_knn)
F1_score_knn= f1_score(y_test, pred_knn)
print("F1-score  :", F1_score_knn)
```

```
Confusion Matrix   : [[131  28]
 [ 14 135]]
Accuracy   : 0.8636363636363636
Precision : 0.8282208588957055
Recall    : 0.9060402684563759
F1-score  : 0.8653846153846153
```

In [ ]:

In [ ]:

In [ ]: