In [59]:

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

In [60]:

```
df=pd.read_csv('data.csv')
df.head()
```

Out[60]:

	Bp	Sg	Al	Su	Rbc	Bu	Sc	Sod	Pot	Hemo	Wbcc	Rbcc	Htn	Class
0	80.0	1.020	1.0	0.0	1.0	36.0	1.2	137.53	4.63	15.4	7800.0	5.20	1.0	1
1	50.0	1.020	4.0	0.0	1.0	18.0	0.8	137.53	4.63	11.3	6000.0	4.71	0.0	1
2	80.0	1.010	2.0	3.0	1.0	53.0	1.8	137.53	4.63	9.6	7500.0	4.71	0.0	1
3	70.0	1.005	4.0	0.0	1.0	56.0	3.8	111.00	2.50	11.2	6700.0	3.90	1.0	1
4	80.0	1.010	2.0	0.0	1.0	26.0	1.4	137.53	4.63	11.6	7300.0	4.60	0.0	1

In [61]:

```
df.info()
```

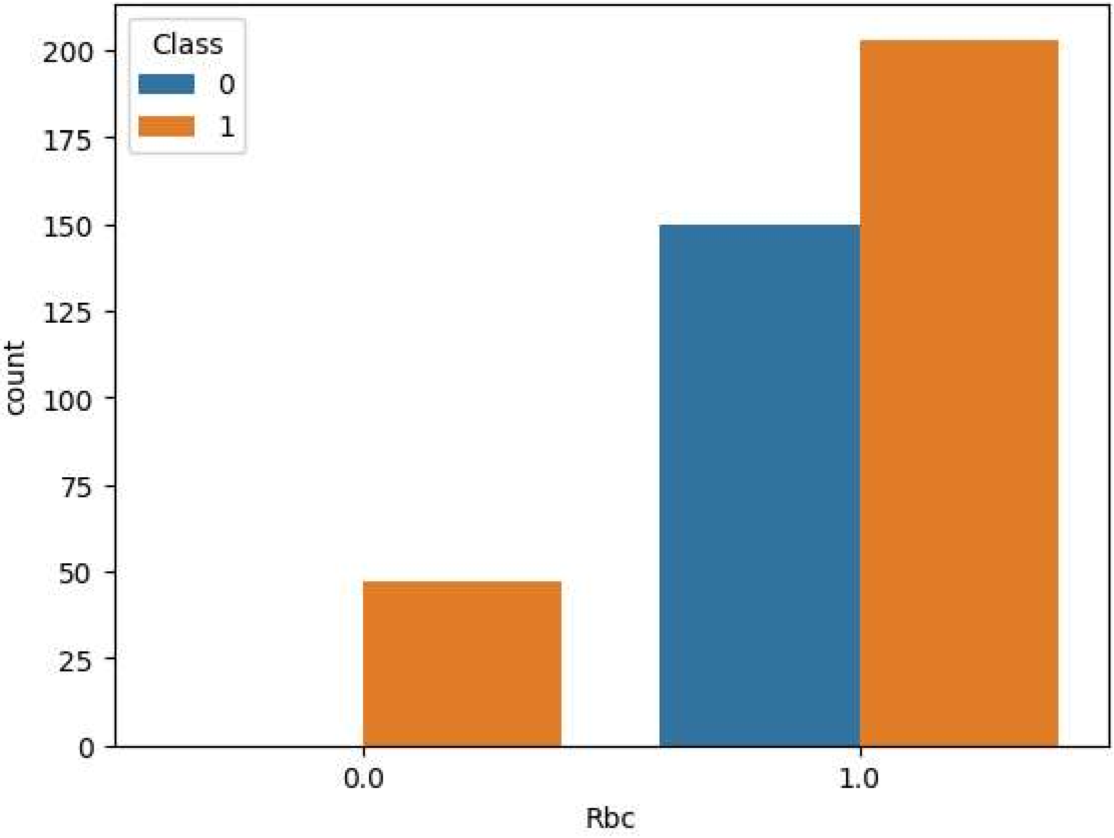
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 14 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Bp      400 non-null    float64
 1   Sg      400 non-null    float64
 2   Al      400 non-null    float64
 3   Su      400 non-null    float64
 4   Rbc     400 non-null    float64
 5   Bu      400 non-null    float64
 6   Sc      400 non-null    float64
 7   Sod     400 non-null    float64
 8   Pot     400 non-null    float64
 9   Hemo    400 non-null    float64
 10  Wbcc    400 non-null    float64
 11  Rbcc    400 non-null    float64
 12  Htn     400 non-null    float64
 13  Class   400 non-null    int64  
dtypes: float64(13), int64(1)
memory usage: 43.9 KB
```

In [62]:

```
sns.countplot(data=df,x="Rbc",hue="Class")
```

Out[62]:

```
<Axes: xlabel='Rbc', ylabel='count'>
```

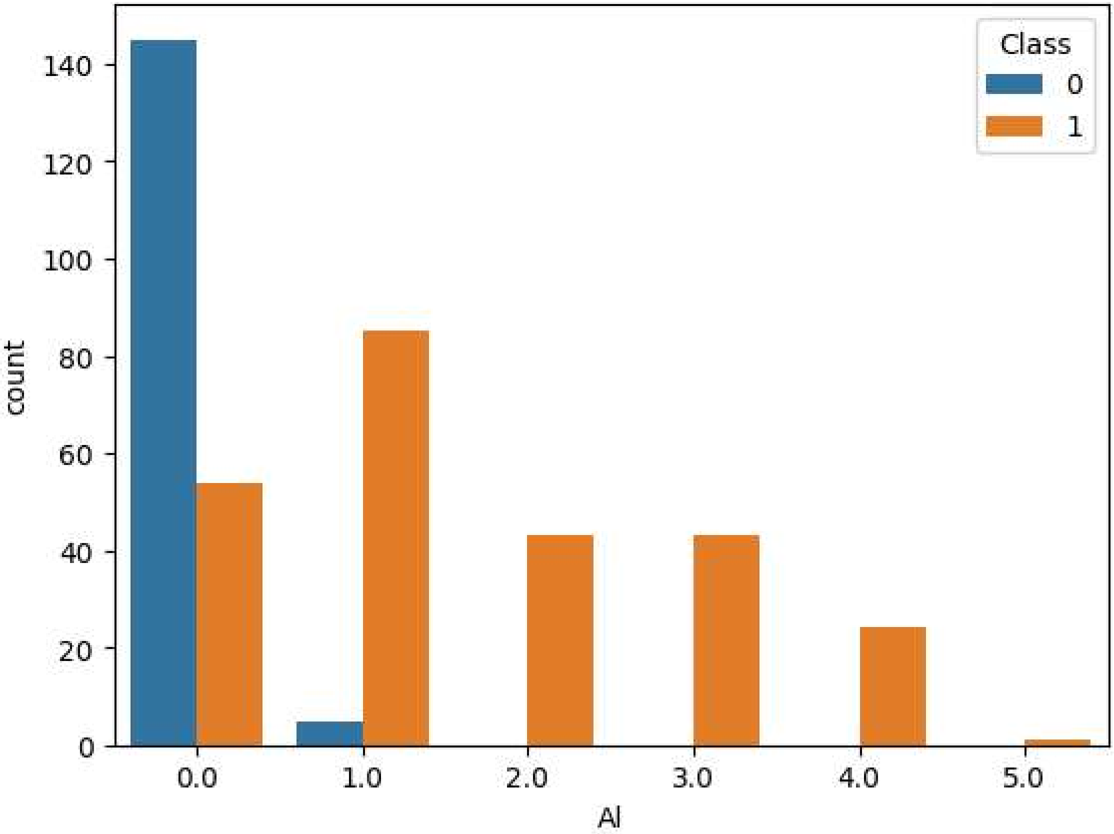


In [63]:

```
sns.countplot(data=df,x="Al",hue="Class")
```

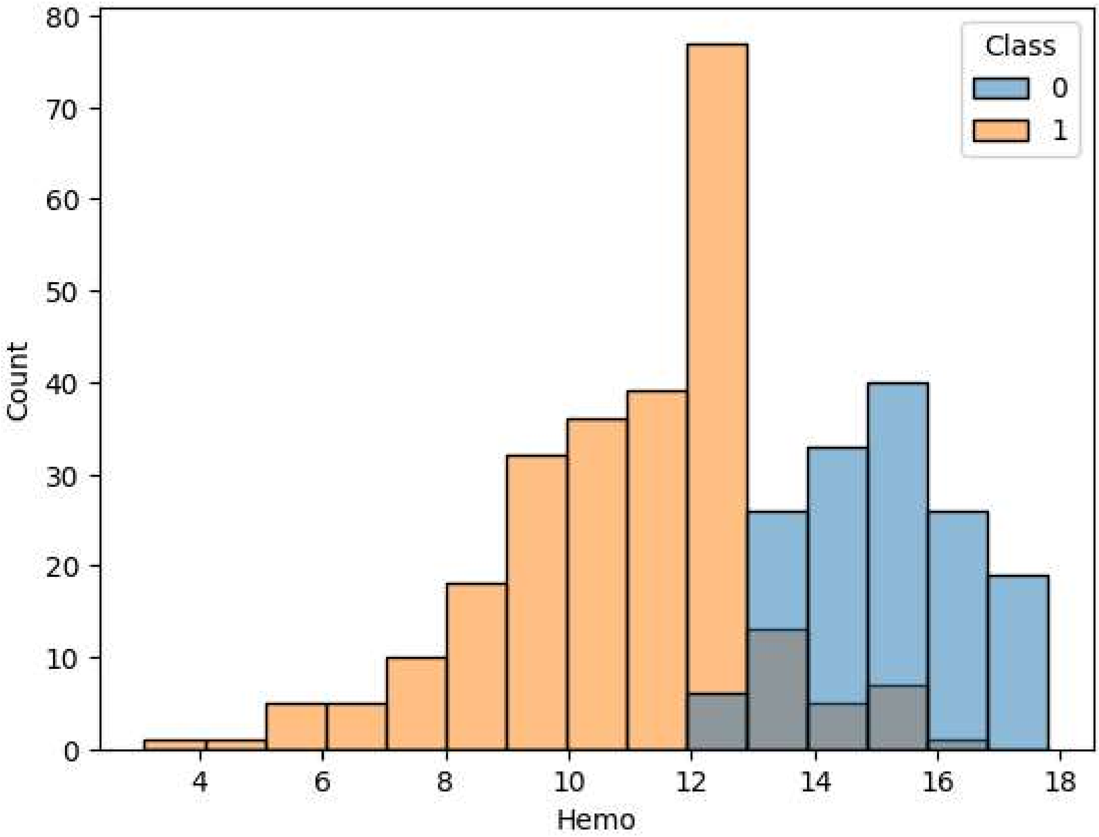
Out[63]:

```
<Axes: xlabel='Al', ylabel='count'>
```



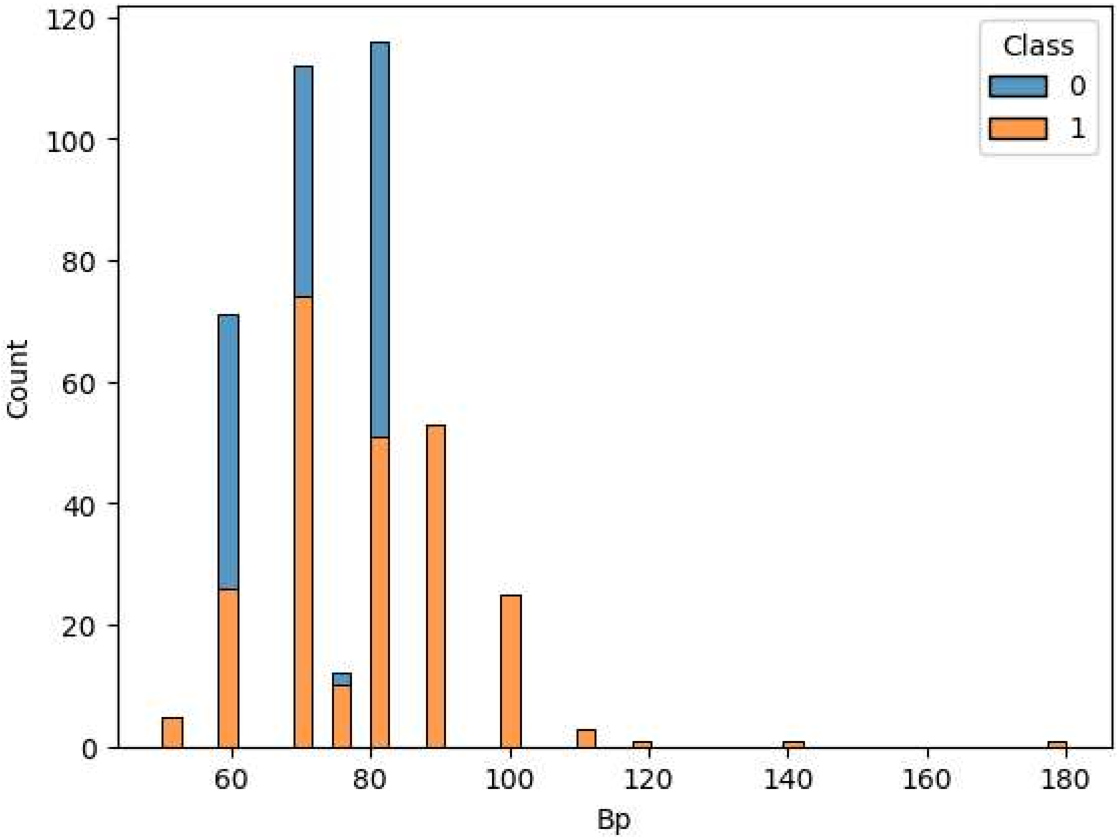
```
In [64]: sns.histplot(data=df,x="Hemo",hue="Class")
```

```
Out[64]: <Axes: xlabel='Hemo', ylabel='Count'>
```



```
In [65]: sns.histplot(data=df,x="Bp",hue="Class",multiple="stack")  # multiple=stack to have
```

```
Out[65]: <Axes: xlabel='Bp', ylabel='Count'>
```



Data Preprocessing

In [66]:
```
# Checking for Nan Values in the dataset
df.isnull().sum()
```

Out[66]:
```
Bp       0
Sg       0
Al       0
Su       0
Rbc      0
Bu       0
Sc       0
Sod      0
Pot      0
Hemo     0
Wbcc     0
Rbcc     0
Htn      0
Class    0
dtype: int64
```

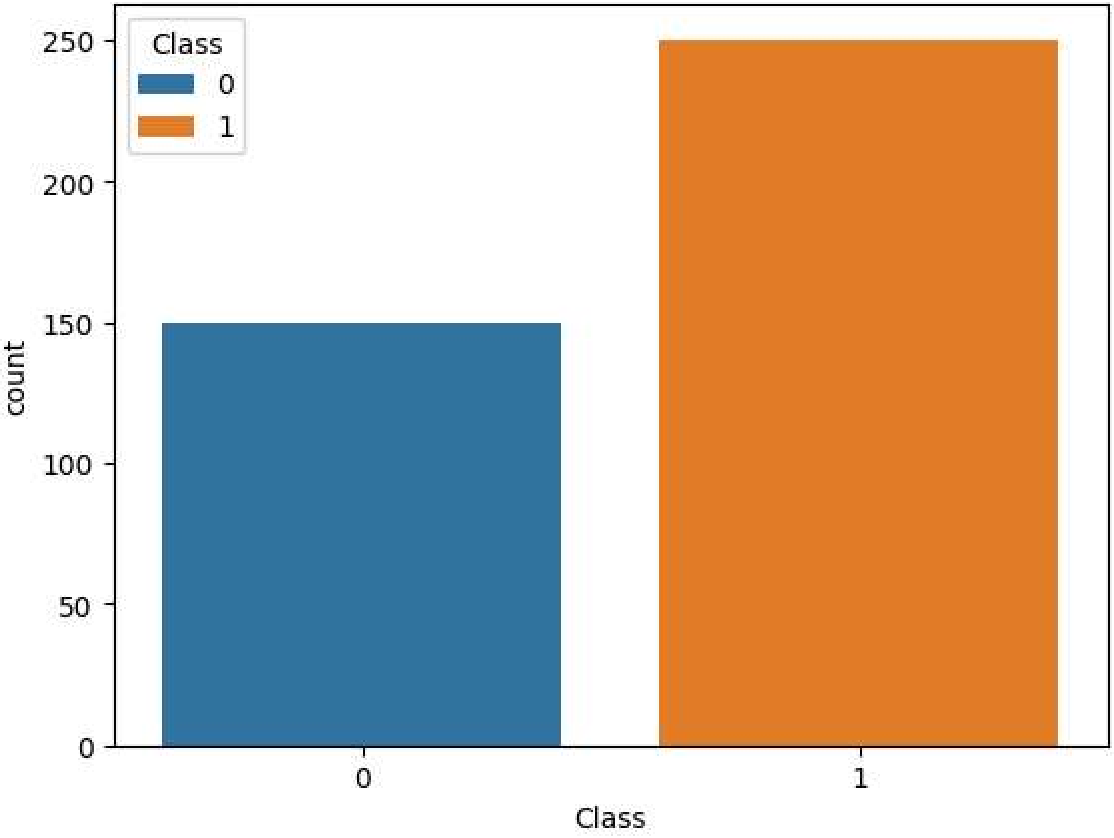
In [67]:
```
df = df.fillna(0)
df.isnull().sum()
```

```
Out[67]: Bp       0
         Sg       0
         Al       0
         Su       0
         Rbc      0
         Bu       0
         Sc       0
         Sod      0
         Pot      0
         Hemo     0
         Wbcc     0
         Rbcc     0
         Htn      0
         Class    0
         dtype: int64
```

Checking for balance in the class

```
In [68]: sns.countplot(data=df,x="Class",hue="Class")
         df['Class'].value_counts()
```

```
Out[68]: Class
         1    250
         0    150
         Name: count, dtype: int64
```



```
In [69]: # Using oversampling algorithm to balance the class
         from sklearn.utils import resample
         df_majority_count=df[(df['Class'])==1]
```

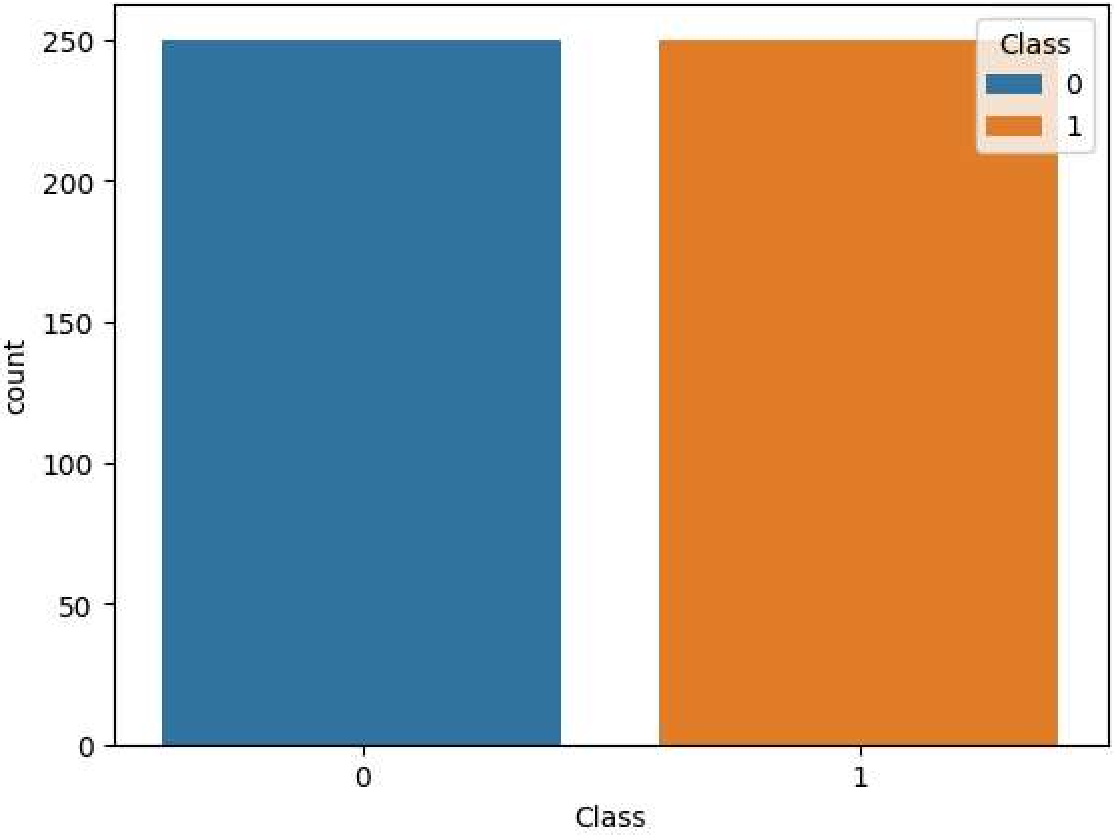
```
df_minority_count=df[(df['Class'])==0]

df_minor_upsampled=resample(df_minority_count,n_samples=250,random_state=0)
df=pd.concat([df_minor_upsampled,df_majority_count])
```

In [70]:

```
sns.countplot(data=df,x="Class",hue="Class")
df['Class'].value_counts()
```

Out[70]:

```
Class
0    250
1    250
Name: count, dtype: int64
```



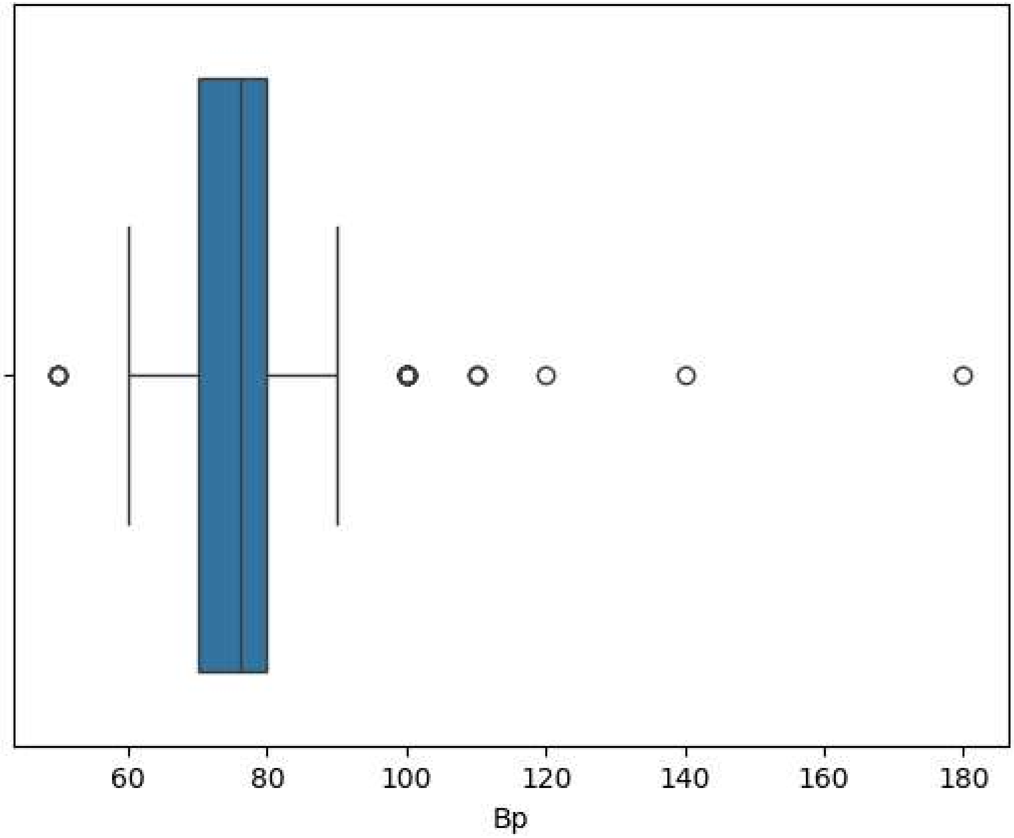
Boxplot - To display outliers

In [71]:

```
sns.boxplot(data=df,x="Bp")
```

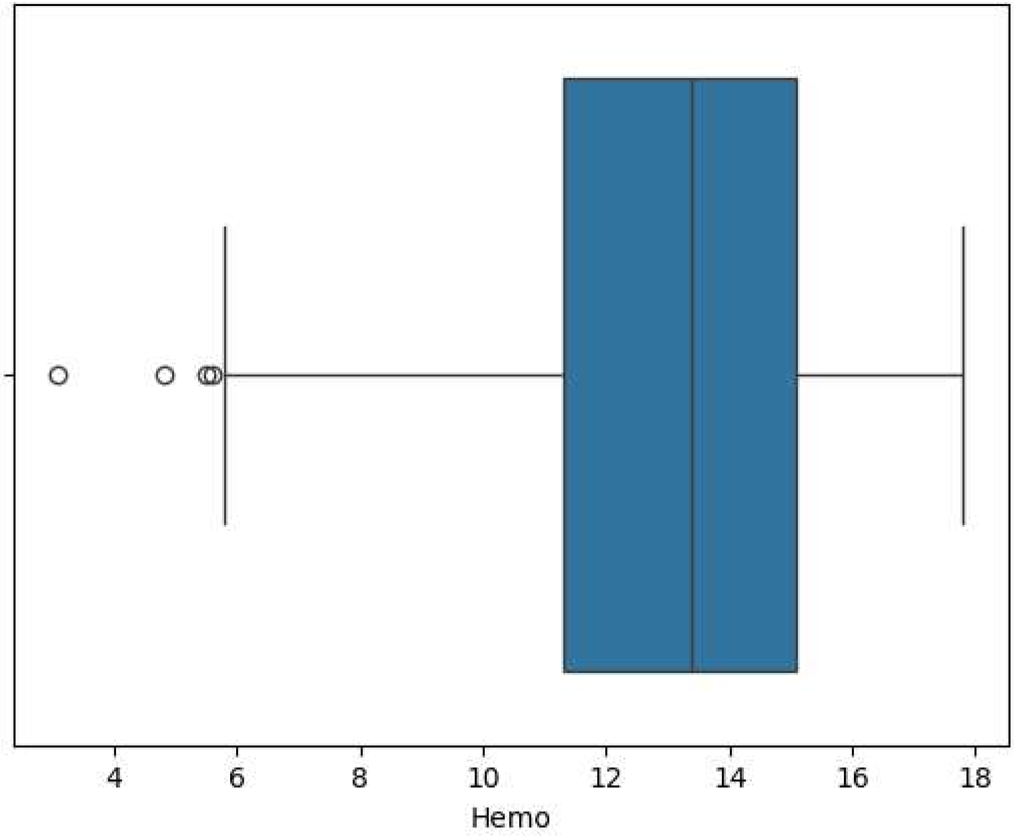
Out[71]:

```
<Axes: xlabel='Bp'>
```



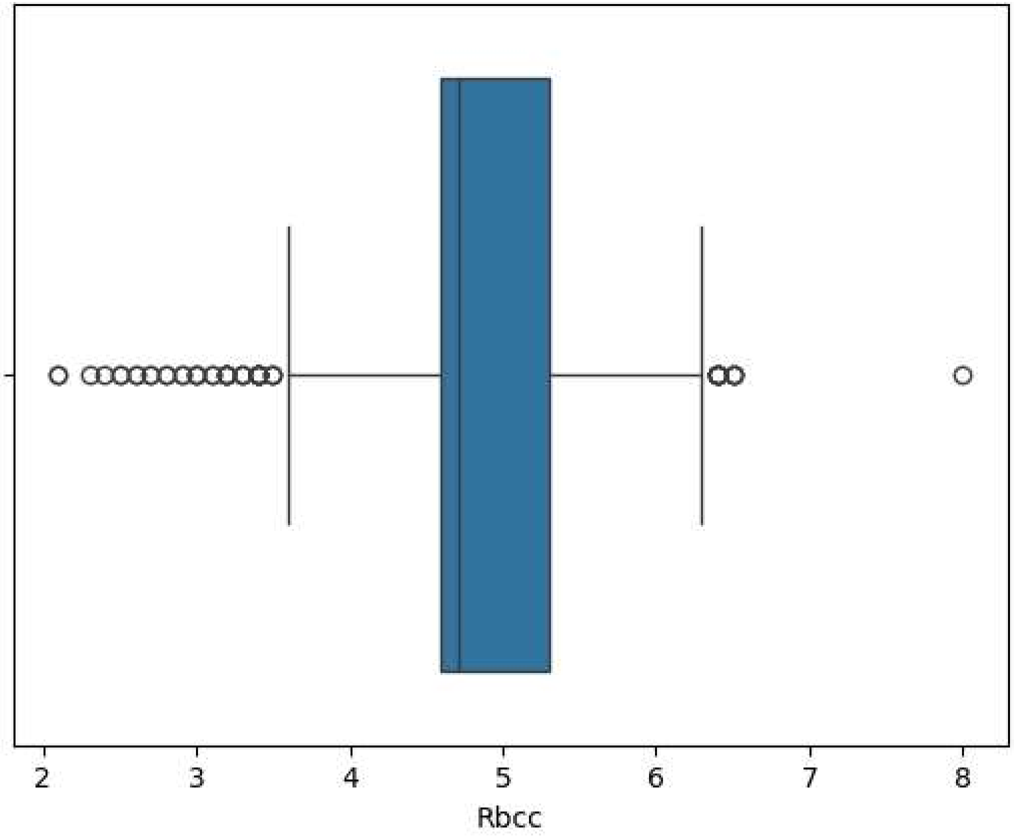
```
In [72]: sns.boxplot(data=df,x="Hemo")
```

```
Out[72]: <Axes: xlabel='Hemo'>
```



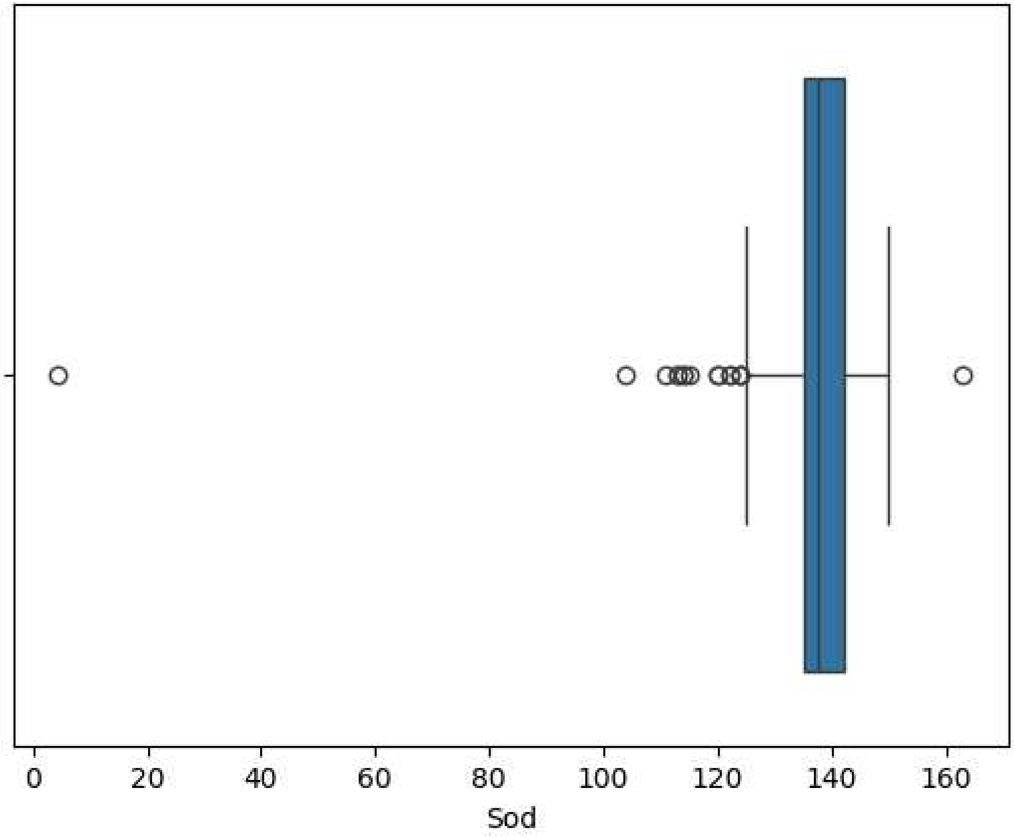
```
In [73]: sns.boxplot(data=df,x="Rbcc")
```

```
Out[73]: <Axes: xlabel='Rbcc'>
```



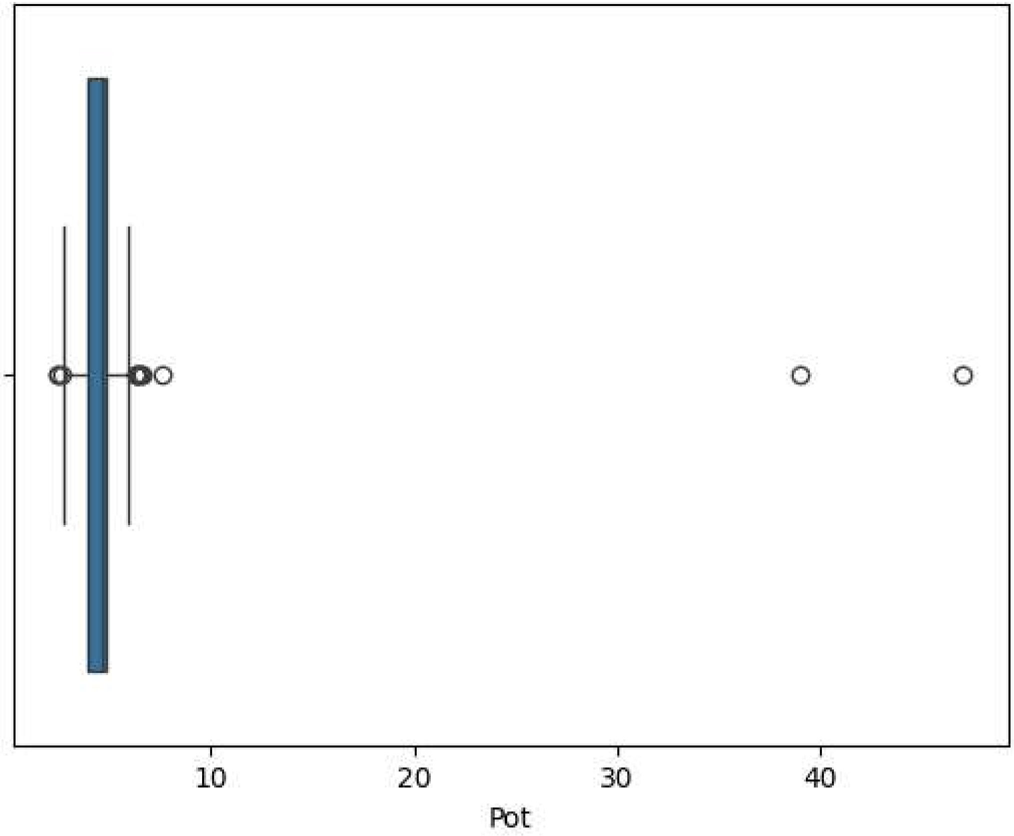
```
In [74]: sns.boxplot(data=df,x="Sod")
```

```
Out[74]: <Axes: xlabel='Sod'>
```



In [75]:
```
sns.boxplot(data=df,x="Pot")
```

Out[75]:
```
<Axes: xlabel='Pot'>
```



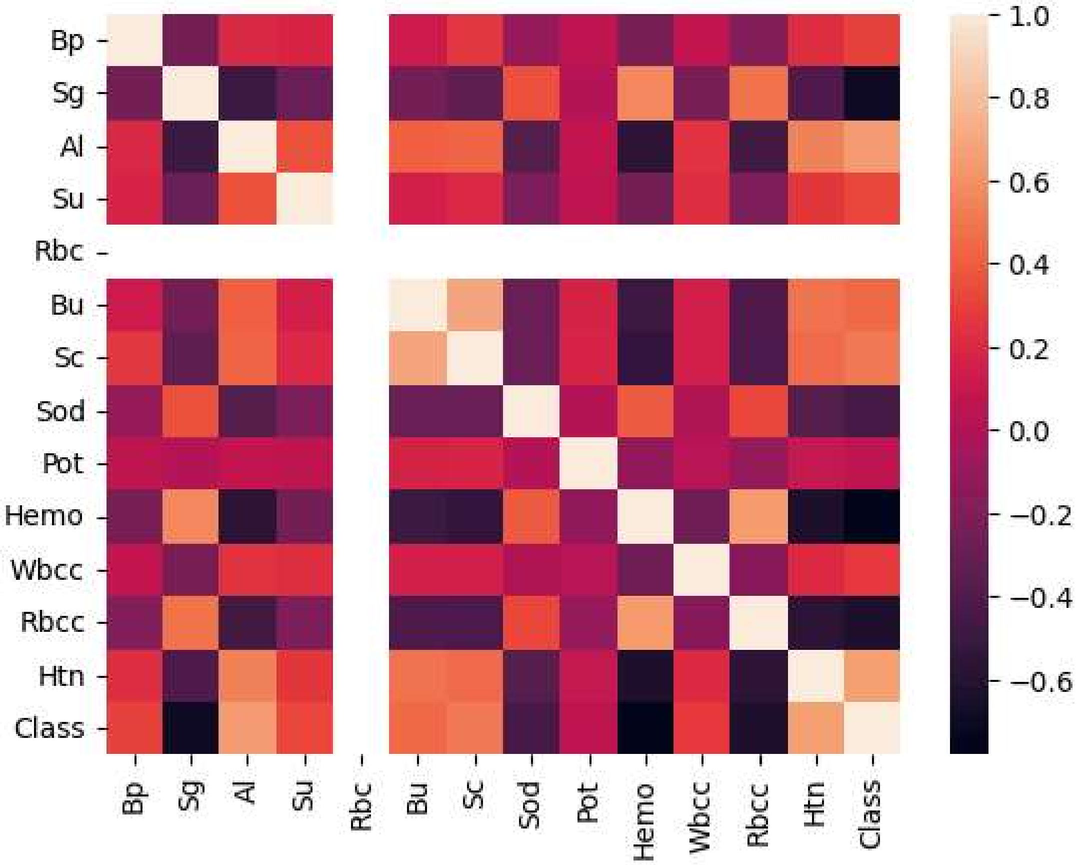
Dealing with outliers using Z-score method

In [76]:
```
import scipy.stats as stats

z=np.abs(stats.zscore(df))
df=df[(z<3).all(axis=1)]
df.shape
```

Out[76]:
```
(420, 14)
```

In [77]:
```
sns.heatmap(df.corr())
```

Out[77]:
```
<Axes: >
```



Dropping RBC Column since it doesn't contain any information/data

In [78]:
```
df = df.drop(columns=['Rbc'])
```

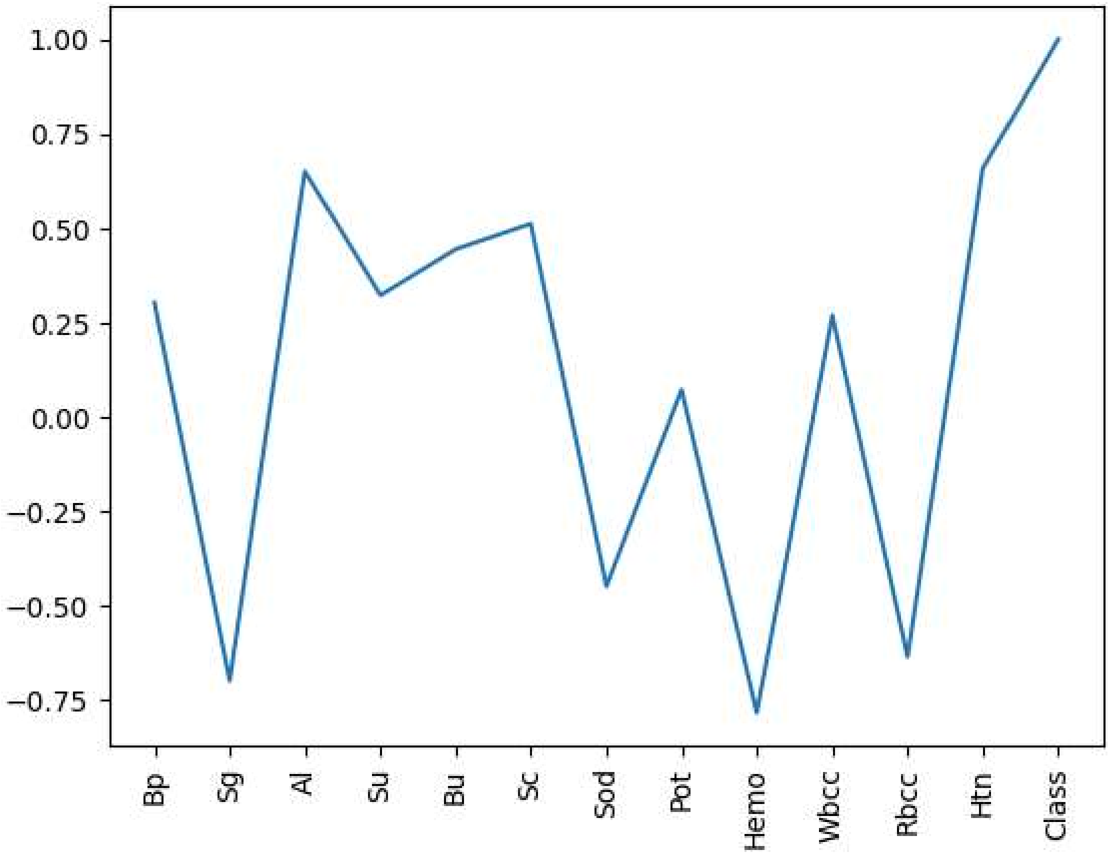
In [79]:
```
df.head()
```

Out[79]:

	Bp	Sg	Al	Su	Bu	Sc	Sod	Pot	Hemo	Wbcc	Rbcc	Htn	Class
297	60.0	1.025	0.0	0.0	26.0	1.0	146.0	4.9	15.8	7700.0	5.2	0.37	0
367	60.0	1.025	0.0	0.0	41.0	1.1	139.0	3.8	17.4	6700.0	6.1	0.00	0
317	70.0	1.020	0.0	0.0	48.0	1.2	139.0	4.3	15.0	8100.0	4.9	0.00	0
353	60.0	1.020	0.0	0.0	37.0	0.6	150.0	5.0	13.6	5800.0	4.5	0.00	0
259	80.0	1.020	0.0	0.0	31.0	1.2	135.0	5.0	16.1	4300.0	5.2	0.00	0

In [80]:

```
corr=df.corr()["Class"]
plt.plot(corr)
plt.xticks(rotation=90)
plt.show()
```



In [81]:

```
from sklearn.model_selection import train_test_split

x=df.drop('Class',axis=1)
y=df['Class']

x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0)
```

In [82]:

```
from sklearn.metrics import accuracy_score,f1_score,confusion_matrix,classification
```

In [83]:

```
def model_fit(model):
    model.fit(x_train,y_train)
    y_pred=model.predict(x_test)
    model_acc=round(accuracy_score(y_test,y_pred)*100,2)
    print(f"The accuracy of the model is:  {model_acc}%")
    print("")
    print("f1_score: ",f1_score(y_test,y_pred))
    print("precision_score: ",precision_score(y_test,y_pred))
    print("recall_score: ",recall_score(y_test,y_pred))
    print("")
    print("Confusion Matrix: ")
    sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,square=True,cmap='coolwa
    plt.ylabel("Actual Label")
    plt.xlabel("Predicted Label")
    plt.title(f"comfusion matrix")

    print("")
    print("Classification Report: ")
    print(classification_report(y_test,y_pred))
```

Applying algorithms

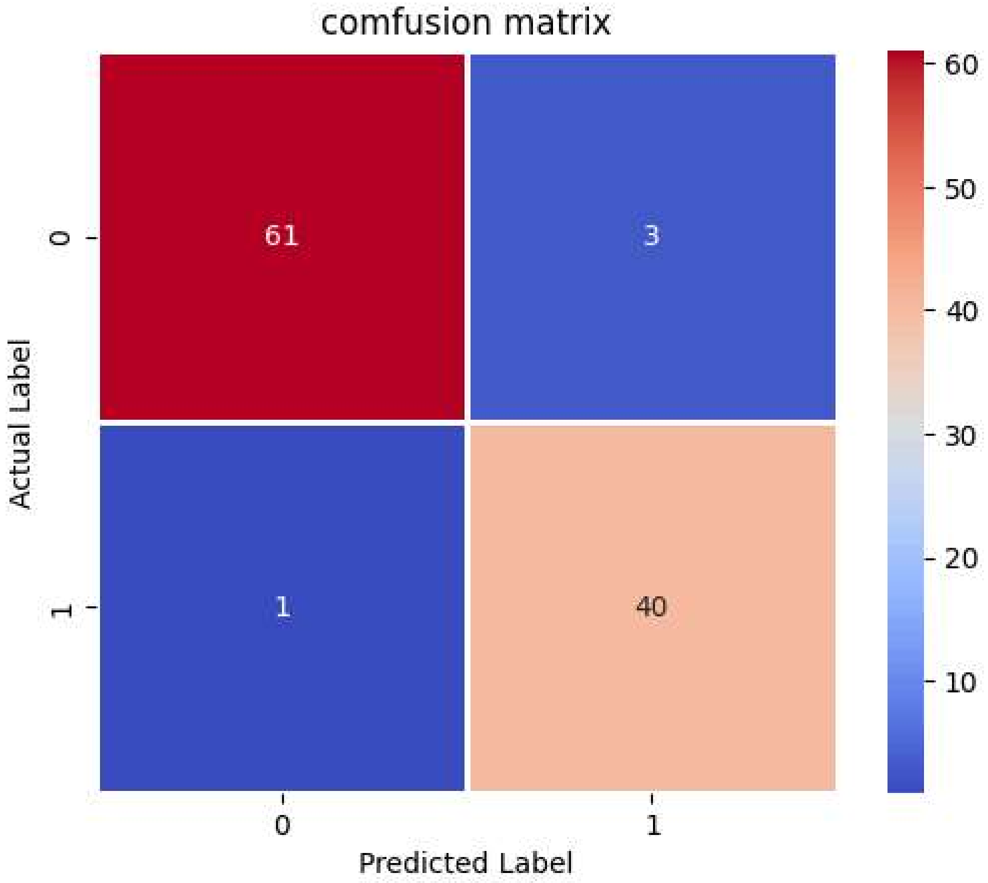
Logistic Regression

In [84]:

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model_fit(model)
```

```
The accuracy of the model is:  96.19%

f1_score:  0.9523809523809523
precision_score:  0.9302325581395349
recall_score:  0.975609756097561

Confusion Matrix: 

Classification Report: 
              precision    recall  f1-score   support

           0       0.98      0.95      0.97        64
           1       0.93      0.98      0.95        41

    accuracy                           0.96       105
   macro avg       0.96      0.96      0.96       105
weighted avg       0.96      0.96      0.96       105
```

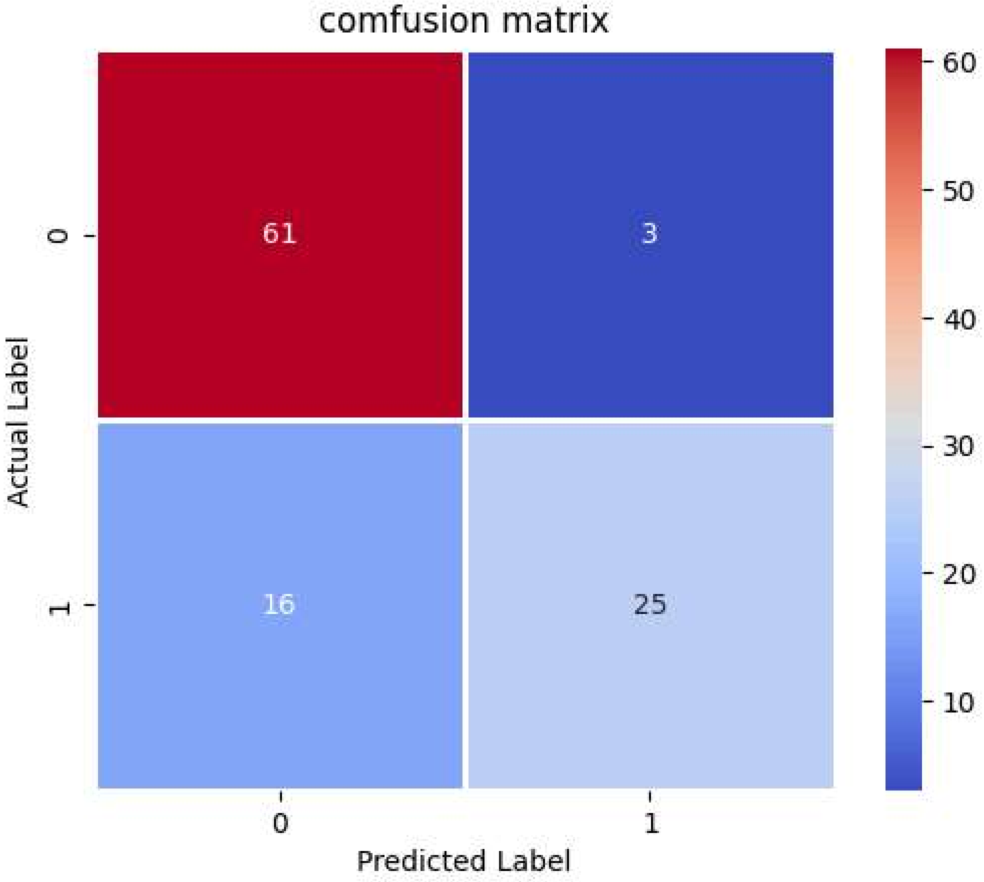
```
c:\Users\karth\Documents\CSE\3rd year\6th sem\ML\lab\venv\Lib\site-packages\sklearn
\linear_model\_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```



KNN

In [85]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=7)
model_fit(knn)
```

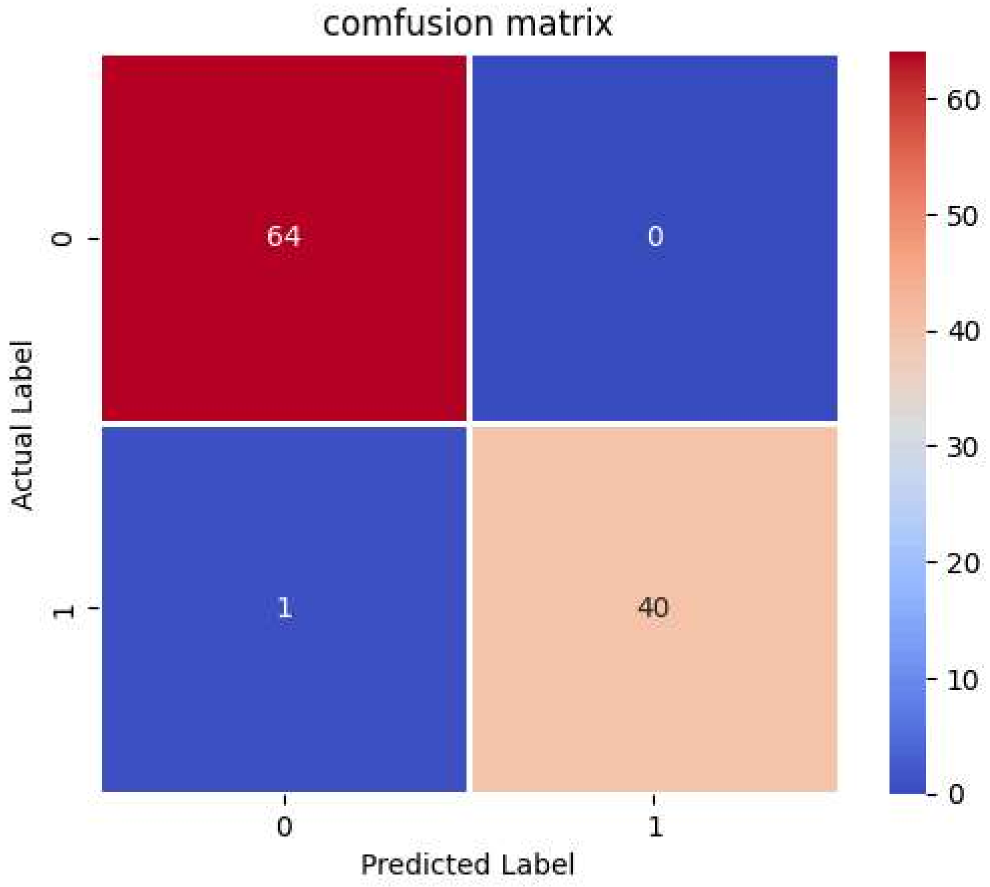
```
The accuracy of the model is:  81.9%

f1_score:  0.7246376811594203
precision_score:  0.8928571428571429
recall_score:  0.6097560975609756

Confusion Matrix:

Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.95      0.87        64
           1       0.89      0.61      0.72        41

    accuracy                           0.82       105
   macro avg       0.84      0.78      0.79       105
weighted avg       0.83      0.82      0.81       105
```



Random Forest Algorithm

In [86]:
```
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators= 10, criterion="entropy")
model_fit(classifier)
```

```
The accuracy of the model is:  99.05%

f1_score:  0.9876543209876543
precision_score:  1.0
recall_score:  0.975609756097561

Confusion Matrix:

Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        64
           1       1.00      0.98      0.99        41

    accuracy                           0.99       105
   macro avg       0.99      0.99      0.99       105
weighted avg       0.99      0.99      0.99       105
```

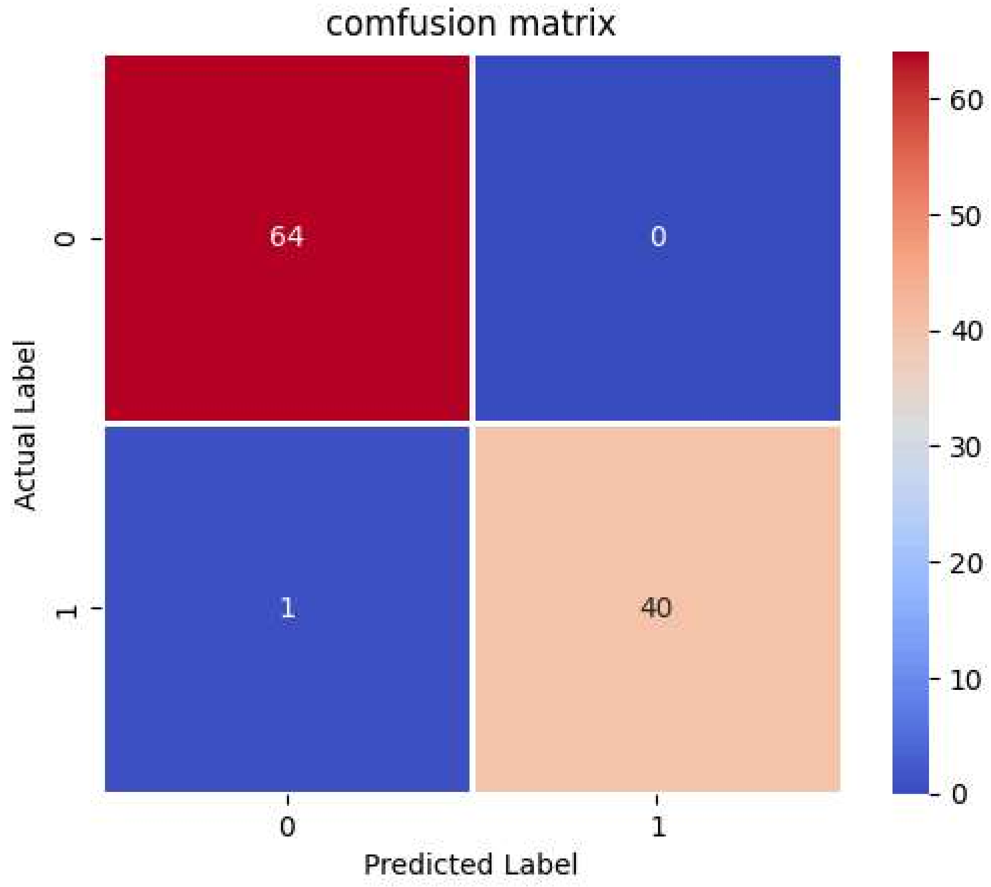


Ada Boost Algorithm

In [87]:

```
from sklearn.ensemble import AdaBoostClassifier
abc = AdaBoostClassifier(n_estimators=50,learning_rate=1)
model_fit(abc)
```

```
The accuracy of the model is:  99.05%

f1_score:  0.9876543209876543
precision_score:  1.0
recall_score:  0.975609756097561

Confusion Matrix:

Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        64
           1       1.00      0.98      0.99        41

    accuracy                           0.99       105
   macro avg       0.99      0.99      0.99       105
weighted avg       0.99      0.99      0.99       105
```


comfusion matrix
Actual Label
Predicted Label