program1-chronic

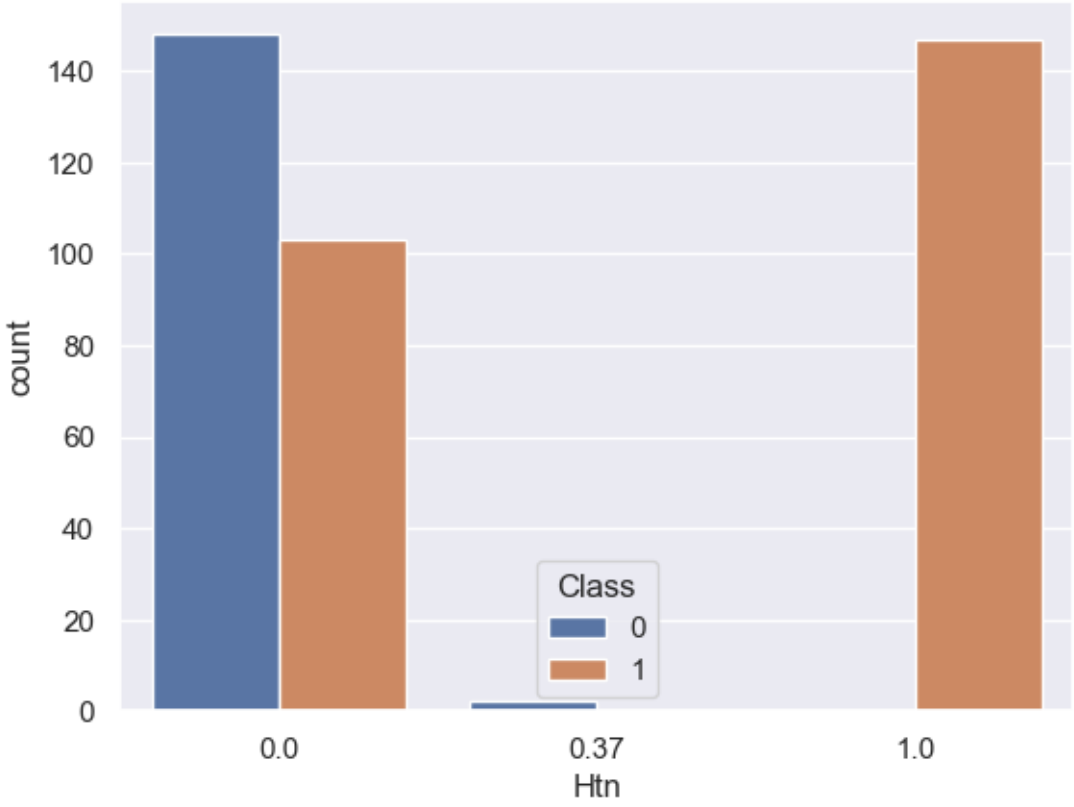
March 29, 2024

[77]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set_theme(color_codes=True)
```

[78]:
```
df=pd.read_csv('ChronicDataset.csv')
```

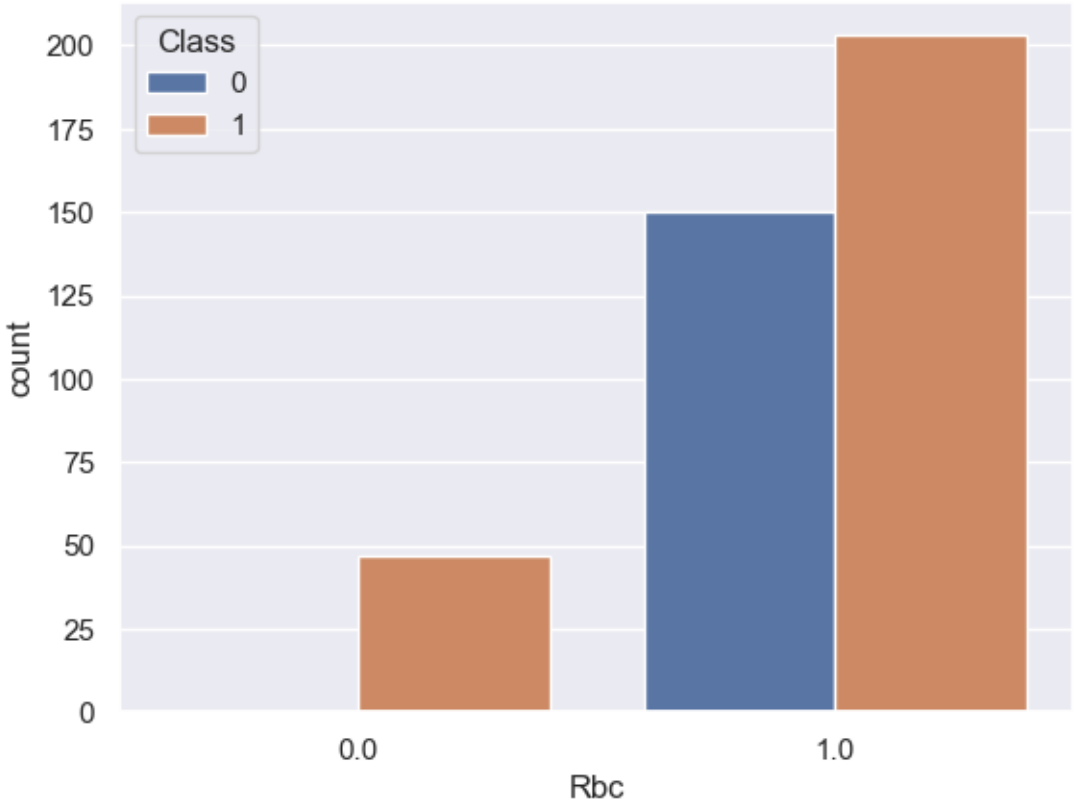
[79]:
```
sns.countplot(data=df, hue="Class", x="Htn")
```

[79]:
```
<Axes: xlabel='Htn', ylabel='count'>
```



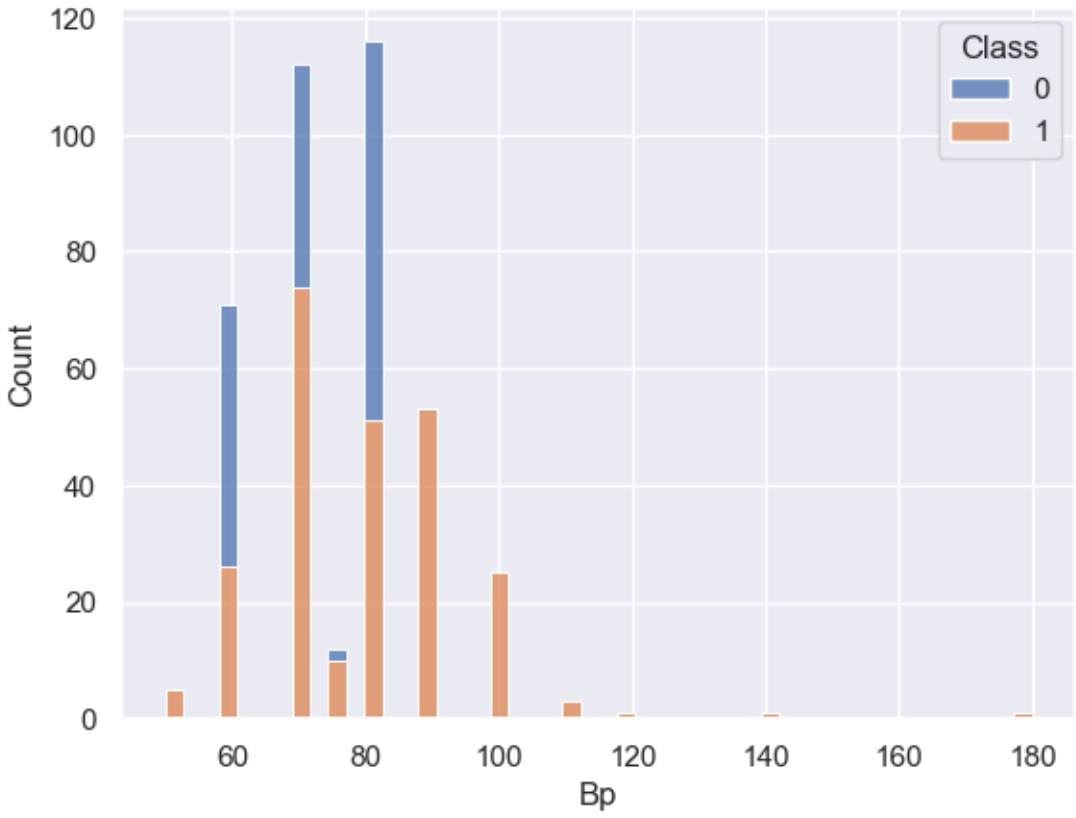
[80]:
```
sns.countplot(data=df , hue="Class", x="Rbc")
```

[80]: <Axes: xlabel='Rbc', ylabel='count'>



[81]:
```
sns.histplot(data=df, hue="Class", x="Bp" ,multiple="stack")
```

[81]: <Axes: xlabel='Bp', ylabel='Count'>



Data Pre-Processing

[82]:
```
df.isnull().sum()
```

[82]:
```
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
```

[83]:
```
df_copy=df.copy(deep=True)
```

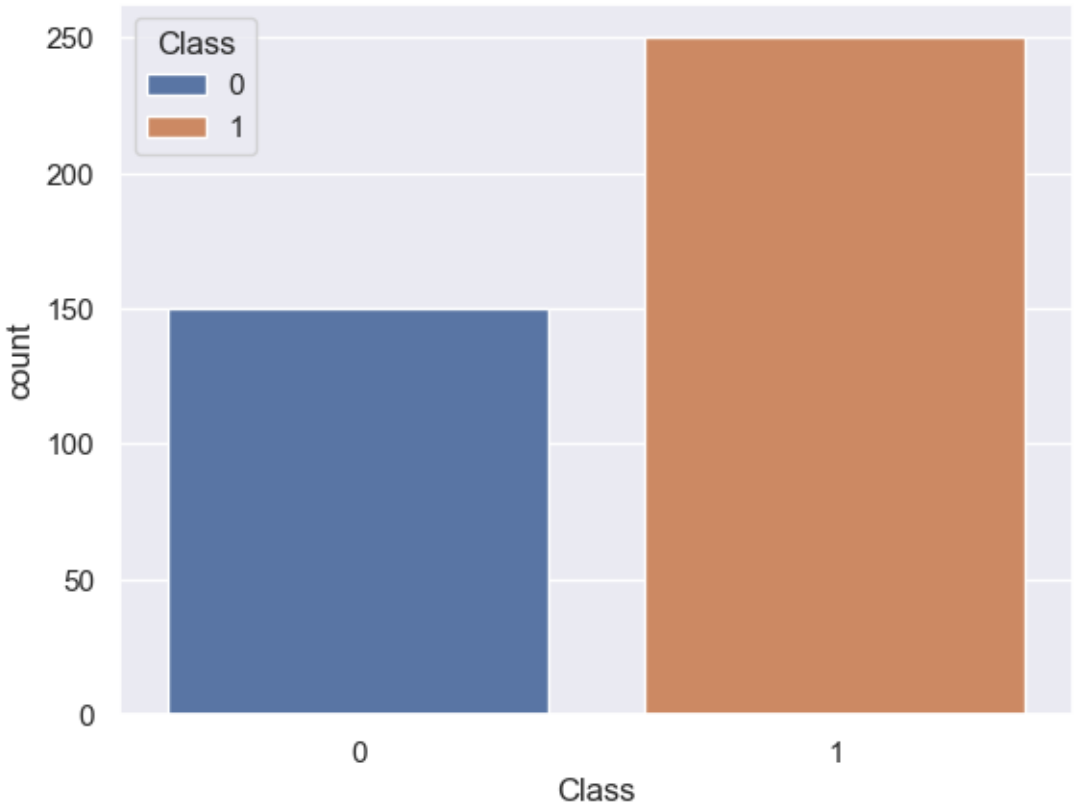
[84]:
```
df_copy[['Bp', 'Sg', 'Bu', 'Sc', 'Sod', 'Pot', 'Hemo', 'Wbcc', 'Rbcc']] =␣
↪df_copy[['Bp', 'Sg', 'Bu', 'Sc', 'Sod', 'Pot', 'Hemo', 'Wbcc', 'Rbcc']].
↪replace(0, np.nan)
```

[85]:
```
print(df_copy.isnull().sum())
```

```
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
```

[86]:
```
sns.countplot(data=df ,x='Class',hue='Class')
print(df.Class.value_counts())
```

```
Class
1    250
0    150
Name: count, dtype: int64
```

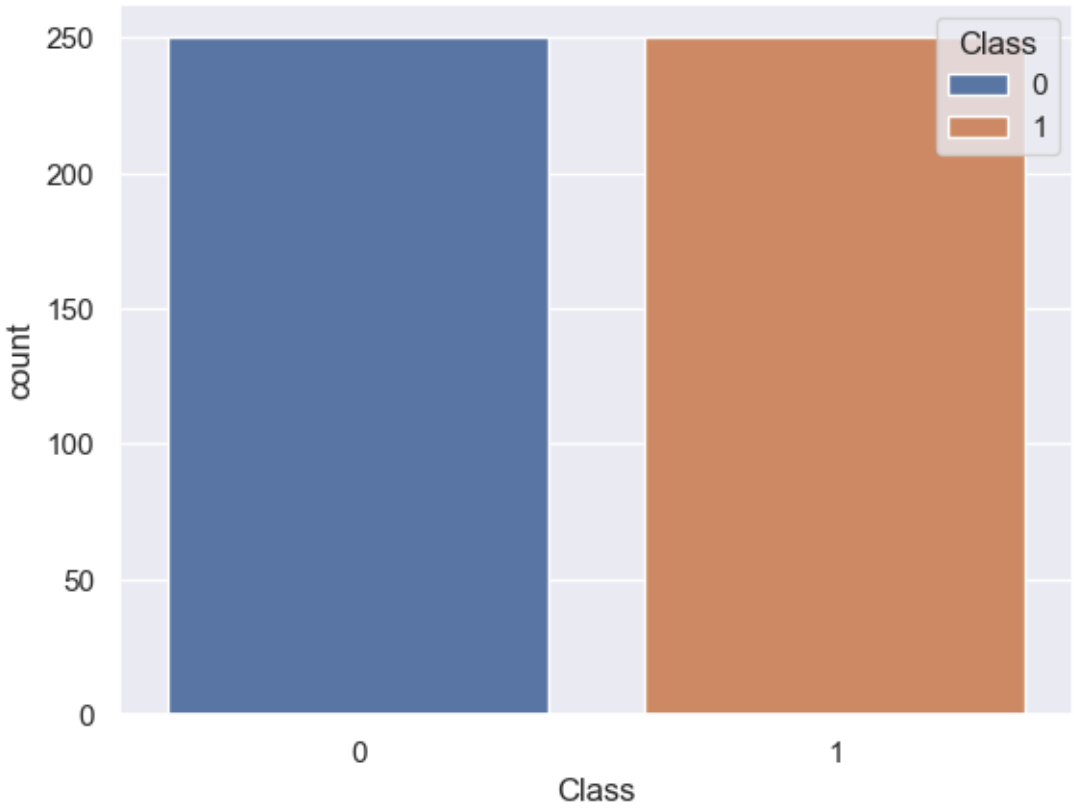


Balancing the data

[87]:
```
from sklearn.utils import resample
df_maj=df[(df['Class']==1)]
df_min=df[(df['Class']==0)]
df_min_up=resample(df_min,
                   n_samples=250,
                   random_state=0)
df2=pd.concat([df_min_up,df_maj])
```

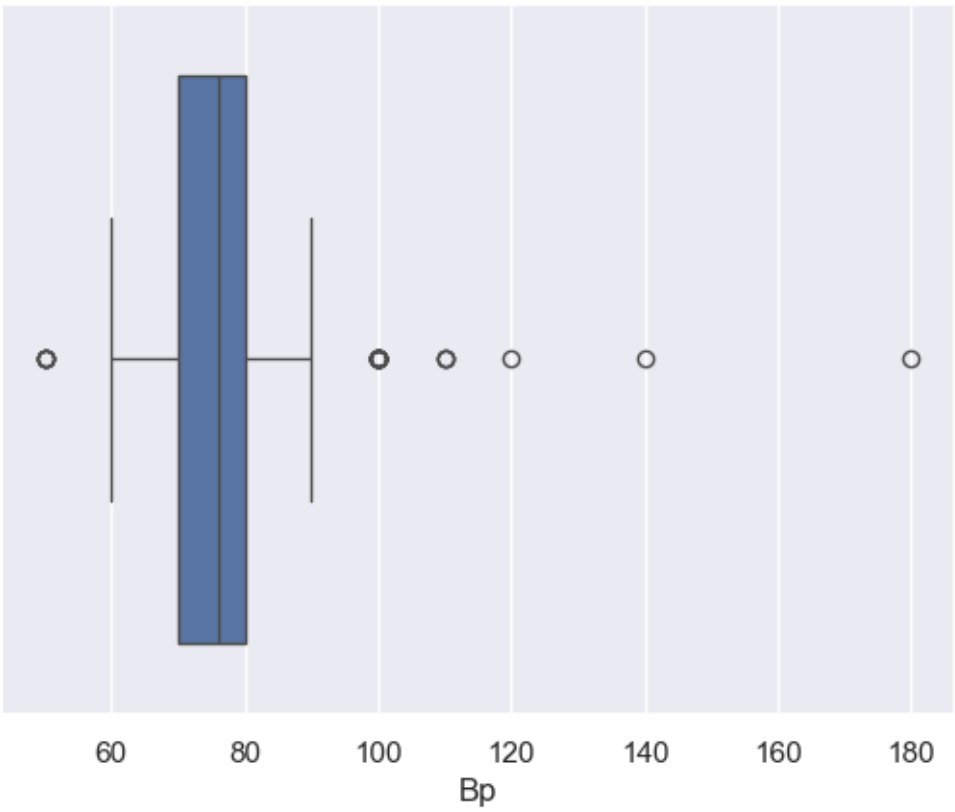
[88]:
```
sns.countplot(data=df2, x="Class", hue="Class")
print(df2.Class.value_counts())
```

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



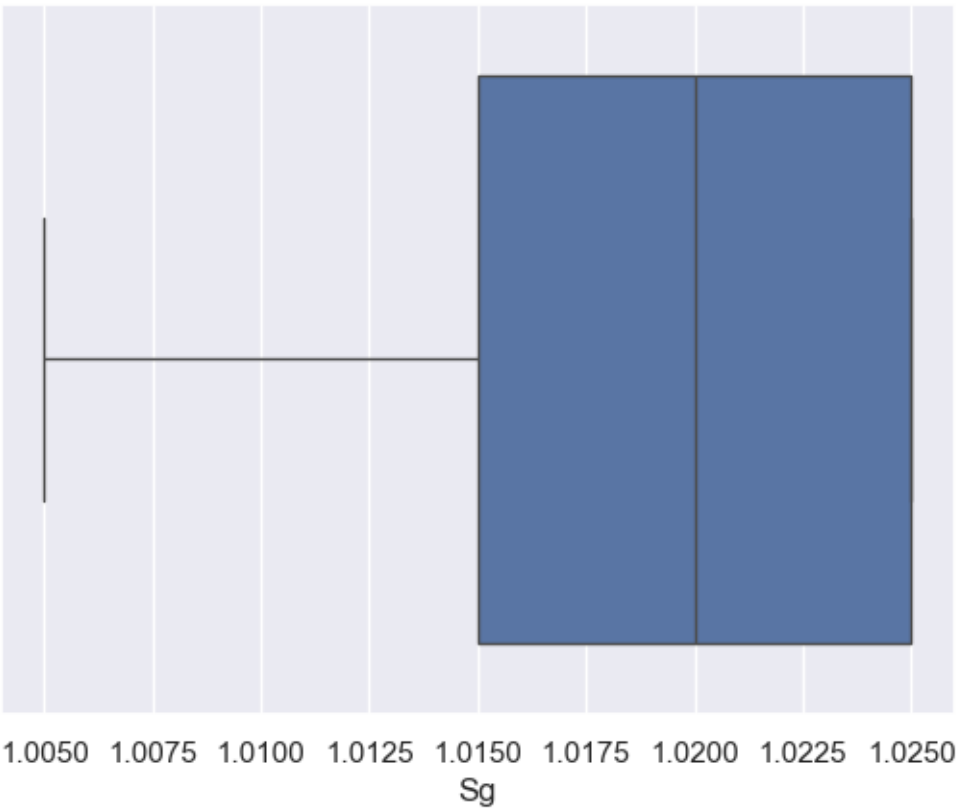
[89]:
```
sns.boxplot(x=df2["Bp"])
```

[89]: <Axes: xlabel='Bp'>



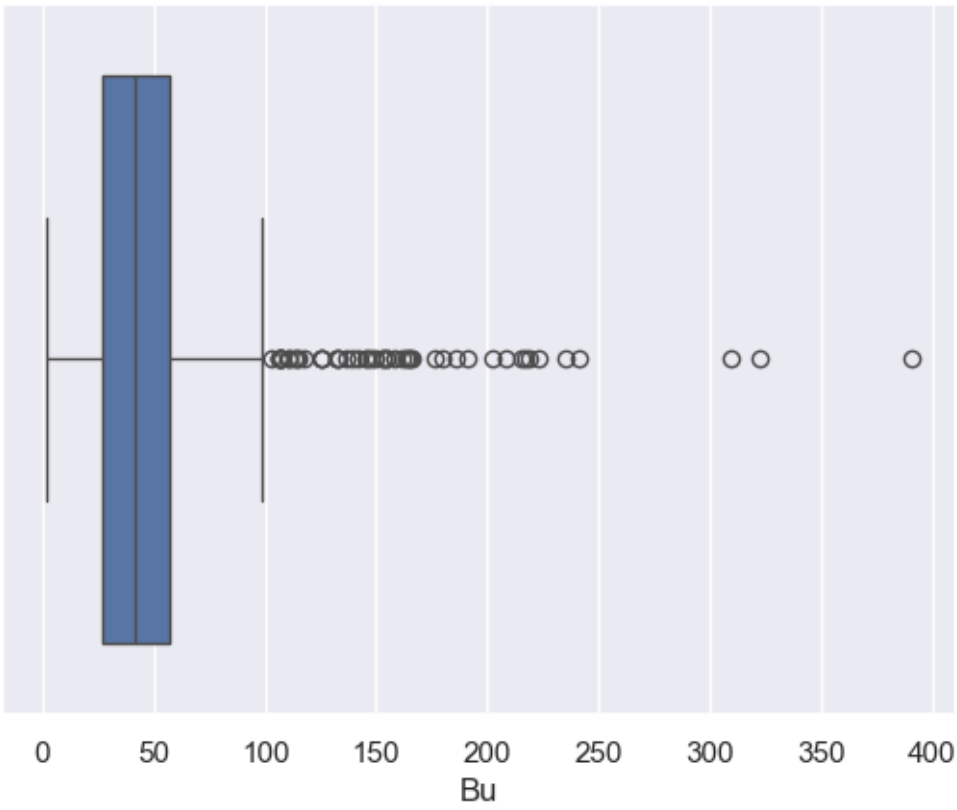
[90]:
```
sns.boxplot(x=df2["Sg"])
```

[90]:
```
<Axes: xlabel='Sg'>
```



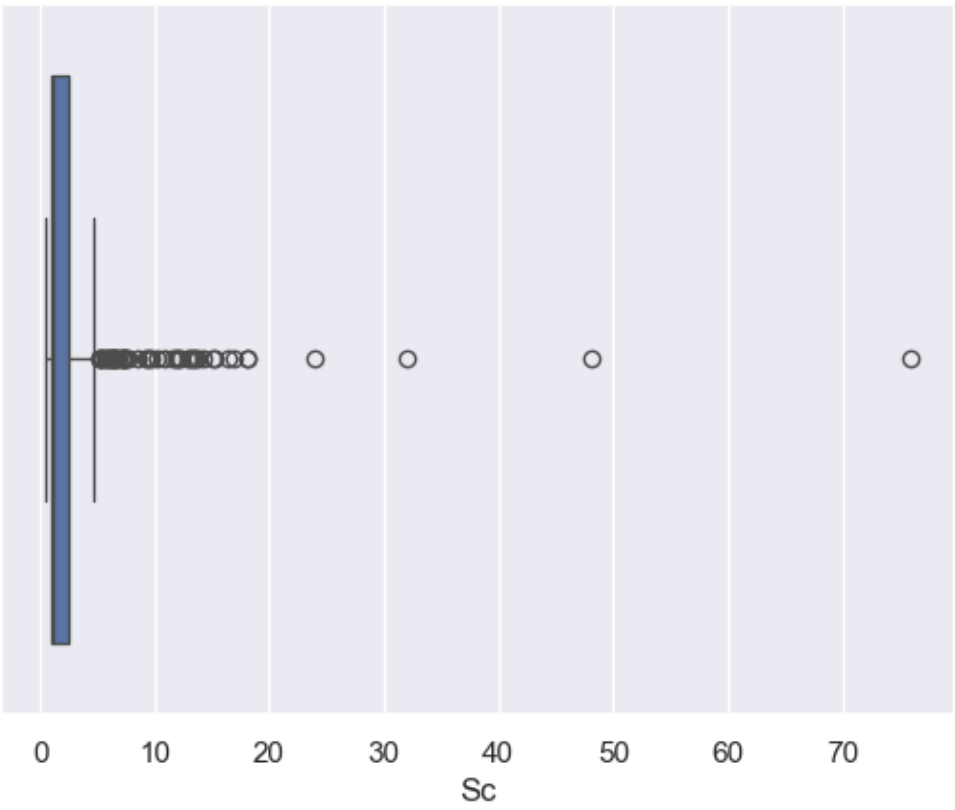
[91]:
```
sns.boxplot(x=df2["Bu"])
```

[91]:
```
<Axes: xlabel='Bu'>
```



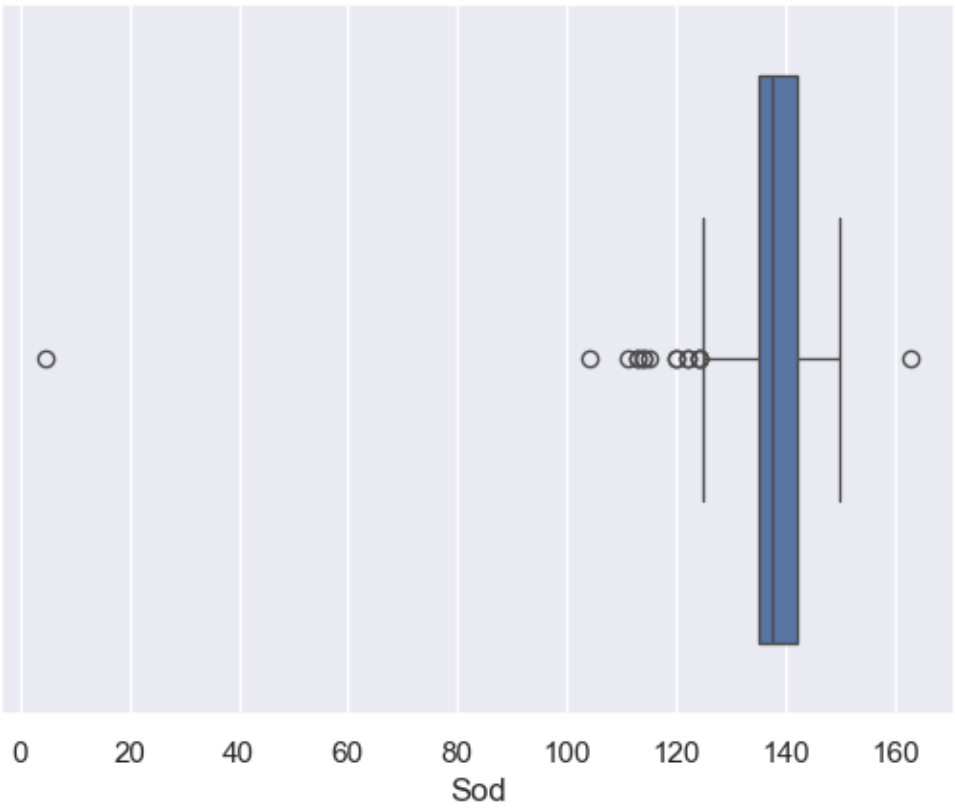
[92]:
```
sns.boxplot(x=df2["Sc"])
```

[92]:
```
<Axes: xlabel='Sc'>
```



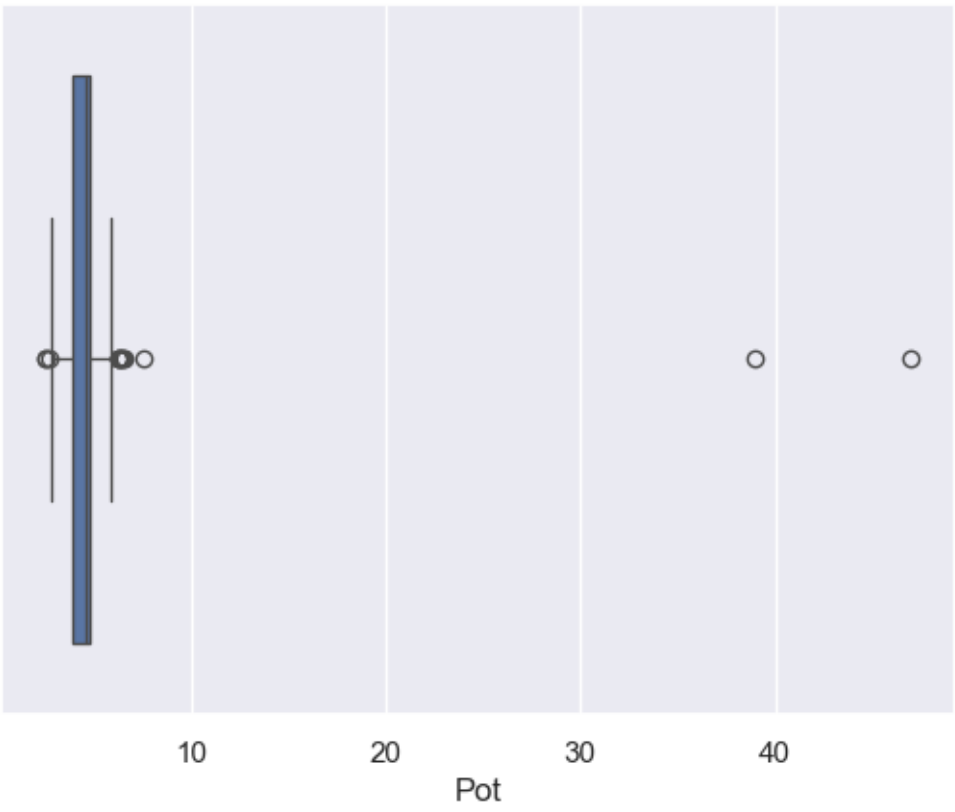
[93]:
```
sns.boxplot(x=df2["Sod"])
```

[93]: <Axes: xlabel='Sod'>



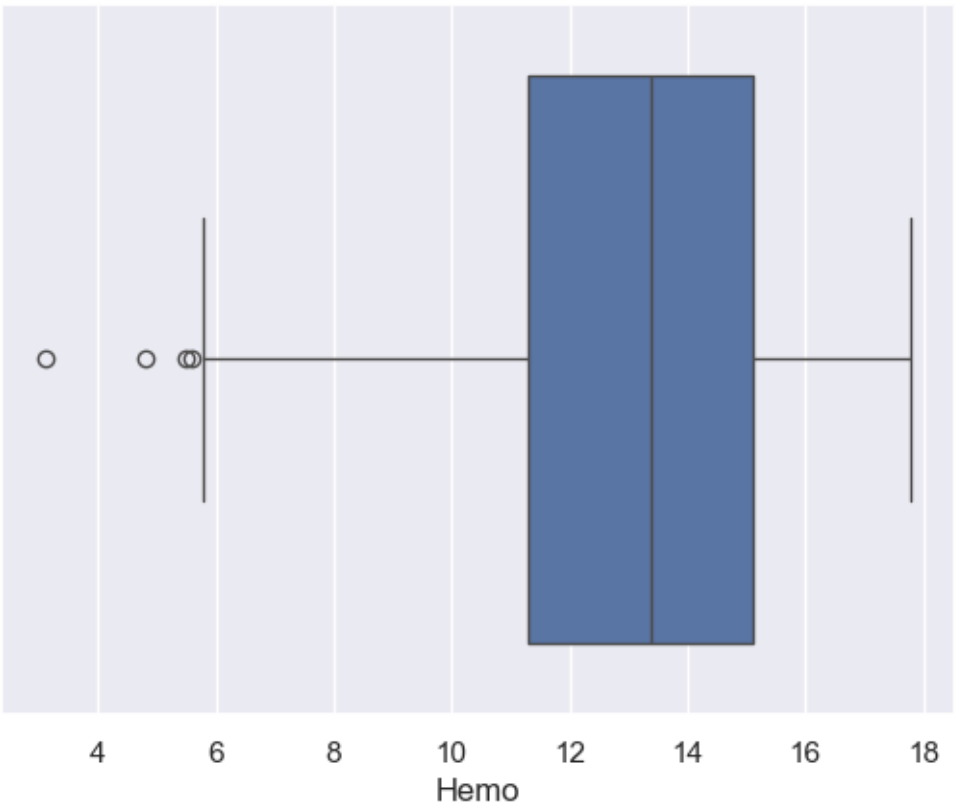
[94]:
```
sns.boxplot(x=df2["Pot"])
```

[94]:
```
<Axes: xlabel='Pot'>
```



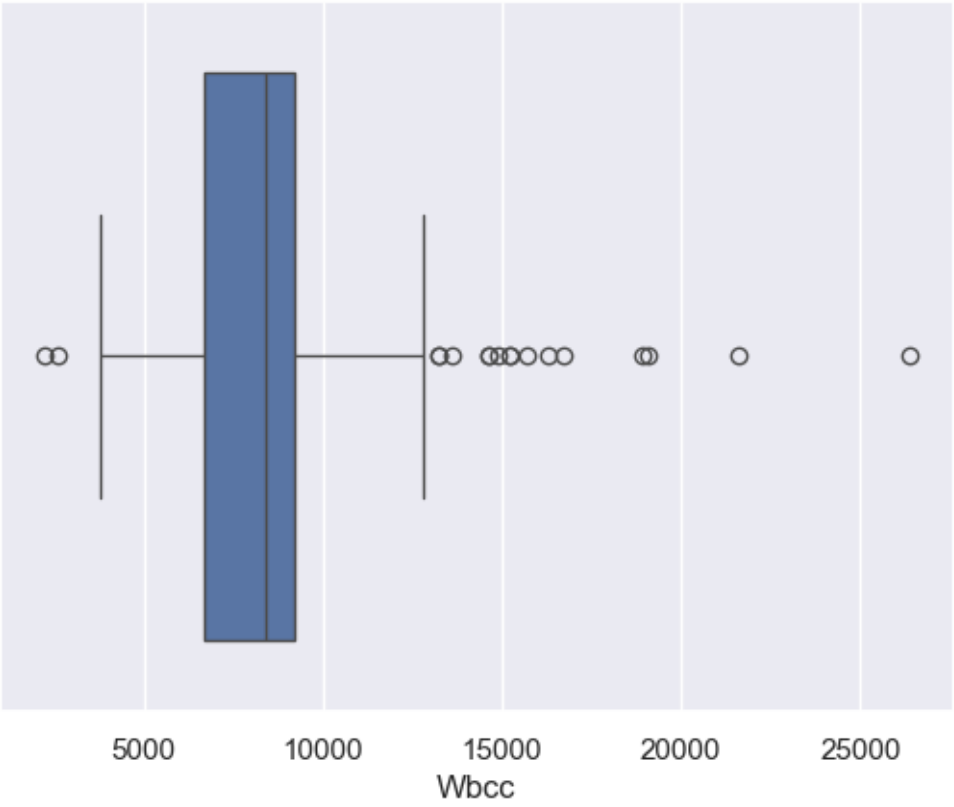
[95]:
```
sns.boxplot(x=df2["Hemo"])
```

[95]:
```
<Axes: xlabel='Hemo'>
```



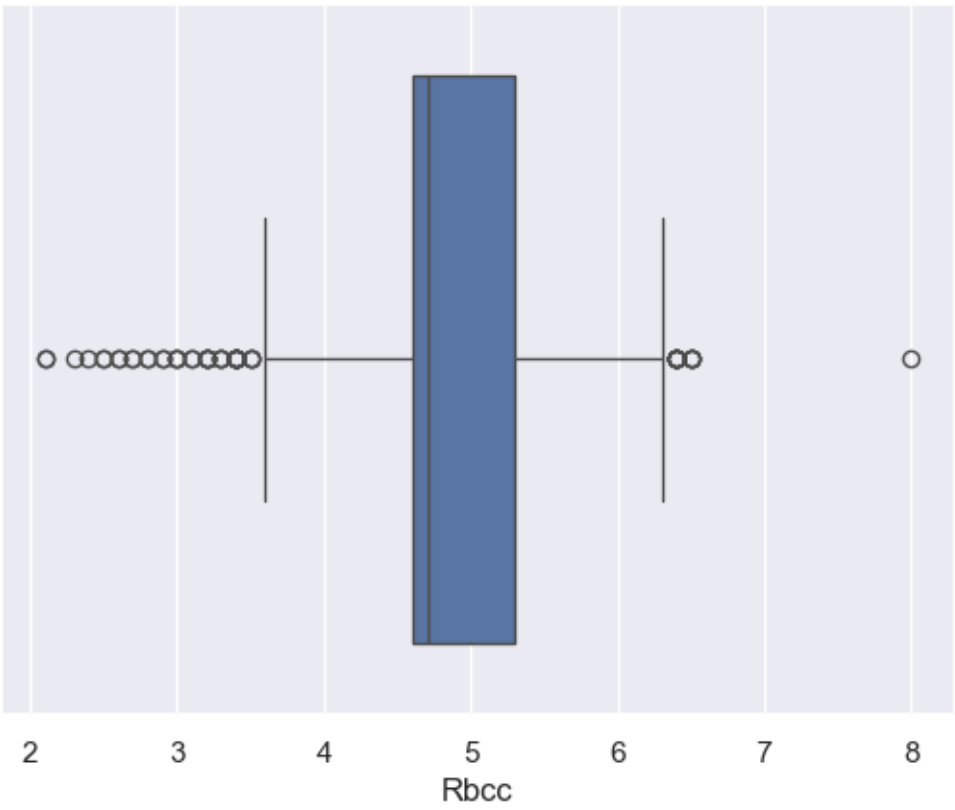
[96]:
```
sns.boxplot(x=df2["Wbcc"])
```

[96]: <Axes: xlabel='Wbcc'>



[97]:
```
sns.boxplot(x=df2["Rbcc"])
```

[97]:
```
<Axes: xlabel='Rbcc'>
```

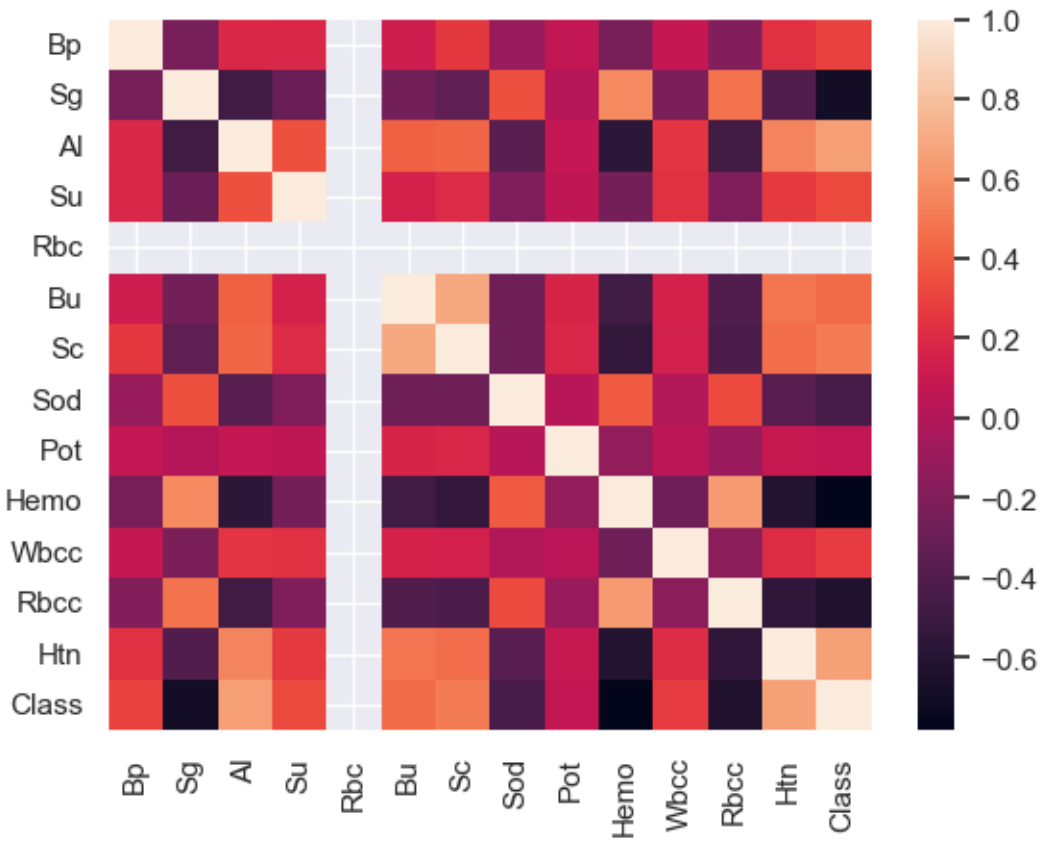


[98]:
```
import scipy.stats as stats
z = np.abs(stats.zscore(df2))
data_clean = df2[(z<3).all(axis = 1)]
data_clean.shape
```

[98]: (420, 14)

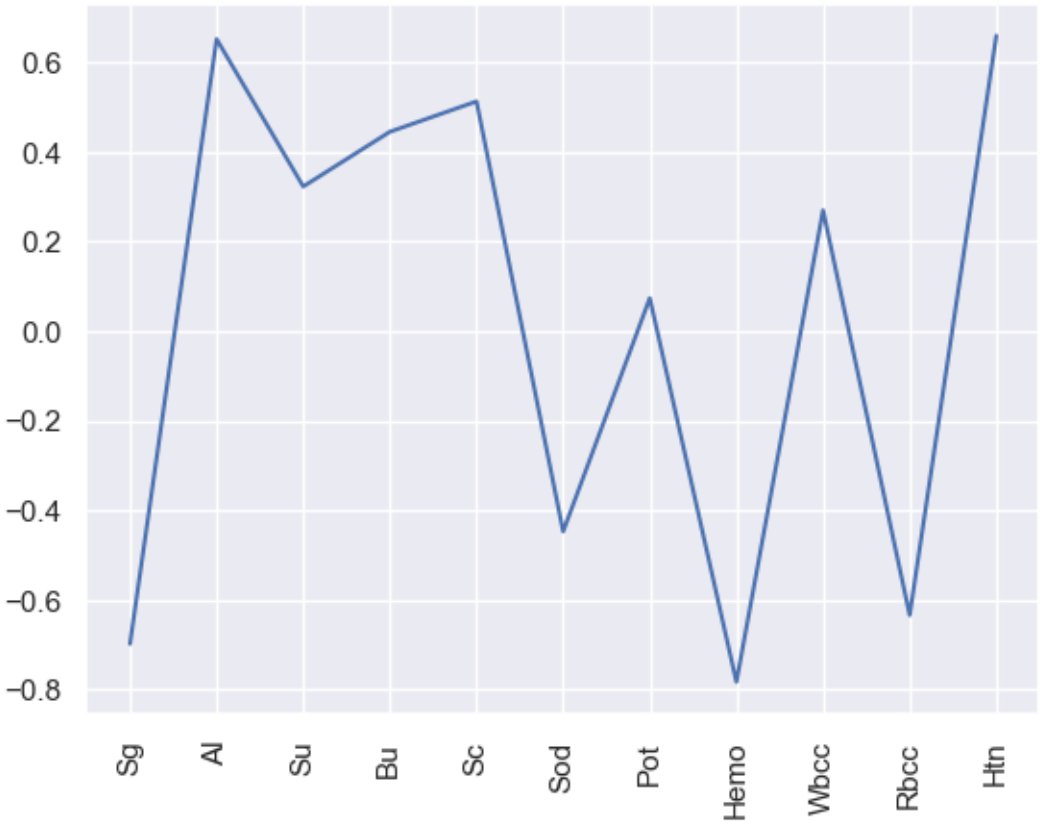
[99]:
```
sns.heatmap(data_clean.corr(), fmt='.2g')
```

[99]: <Axes: >



[100]:
```
#Rbc attribute is irrlevant, so we have to remove it
data_clean2 = data_clean.drop(columns=['Rbc'])
```

[101]:
```
corr = data_clean2[data_clean2.columns[1:]].corr()['Class'][:-1]
plt.plot(corr)
plt.xticks(rotation=90)
plt.show()
```



[102]:
```
X = data_clean2.drop('Class', axis=1)
y = data_clean2['Class']
```

[103]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.
 ↪2,random_state=0)
```

[104]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(random_state=0)
rfc.fit(X_train, y_train)
```

[104]:
```
RandomForestClassifier(random_state=0)
```

[105]:
```
y_pred = rfc.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

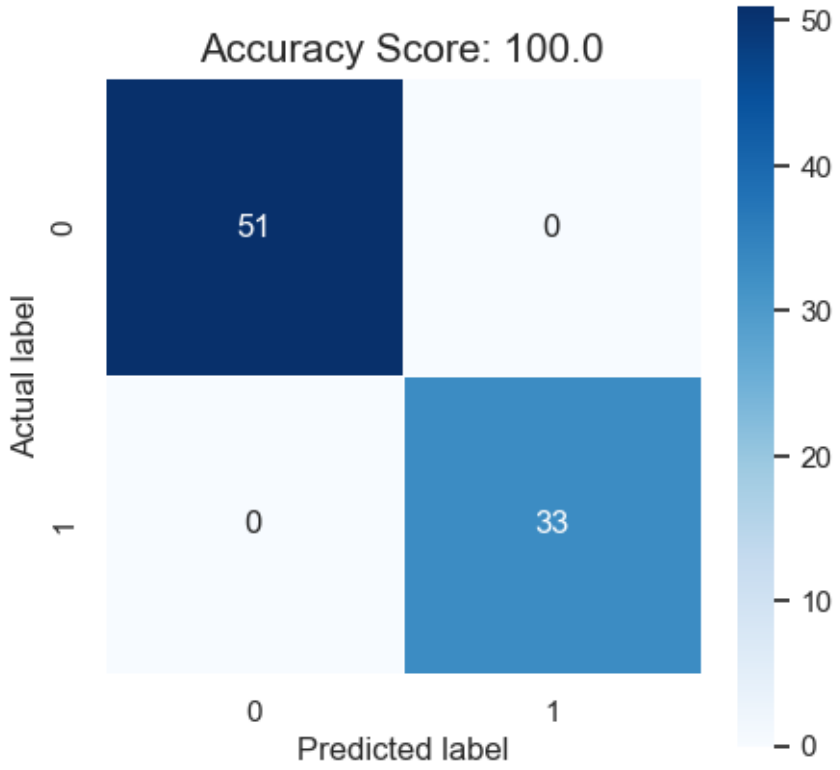
```
Accuracy Score : 100.0 %
```

[106]:
```
from sklearn.metrics import accuracy_score, f1_score, precision_score,␣
 ↪recall_score
print('F-1 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)))
```

```
F-1 Score :  1.0
Precision Score :  1.0
Recall Score :  1.0
```

[107]:
```
from sklearn.metrics import classification_report, confusion_matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=.5, annot=True,square = True, cmap = 'Blues')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Accuracy Score: {0}'.format(rfc.score(X_test, y_test)*100)
plt.title(all_sample_title, size = 15)
```

[107]: Text(0.5, 1.0, 'Accuracy Score: 100.0')



[108]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
```

[108]:
```
KNeighborsClassifier()
```

[109]:
```
y_pred = knn.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

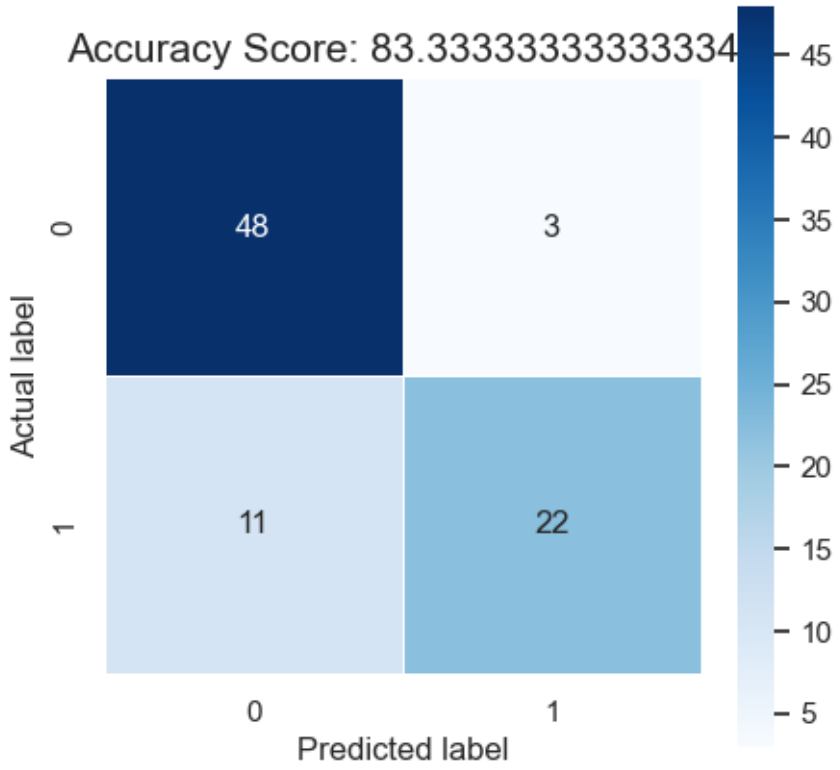
```
Accuracy Score : 83.33 %
```

[110]:
```
from sklearn.metrics import accuracy_score, f1_score, precision_score,␣
 ↪recall_score
print('F-1 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)))
```

```
F-1 Score :  0.7586206896551724
Precision Score :  0.88
Recall Score :  0.6666666666666666
```

[111]:
```
from sklearn.metrics import classification_report, confusion_matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=.5, annot=True,square = True, cmap = 'Blues')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Accuracy Score: {0}'.format(knn.score(X_test, y_test)*100)
plt.title(all_sample_title, size = 15)
```

[111]:
```
Text(0.5, 1.0, 'Accuracy Score: 83.33333333333334')
```



[112]:
```
from sklearn.ensemble import AdaBoostClassifier
ada = AdaBoostClassifier(random_state=0)
ada.fit(X_train, y_train)
```

```
C:\Users\dell\AppData\Local\Programs\Python\Python312\Lib\site-
packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(
```

[112]:
```
AdaBoostClassifier(random_state=0)
```

[113]:
```
y_pred = ada.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

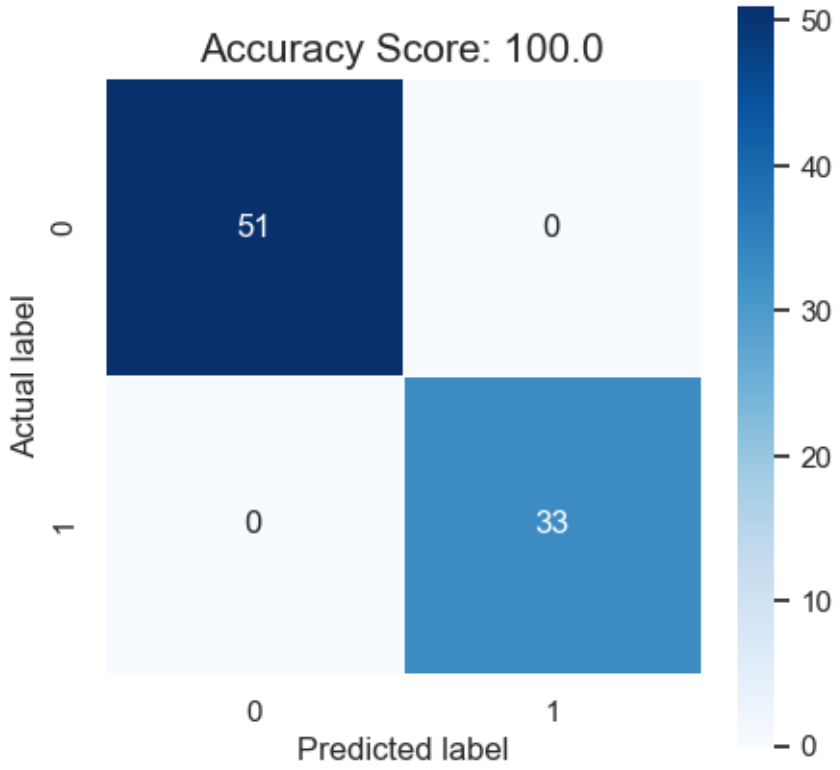
```
Accuracy Score : 100.0 %
```

[114]:
```
from sklearn.metrics import accuracy_score, f1_score, precision_score,␣
 ↪recall_score
print('F-1 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)))
```

```
F-1 Score :  1.0
Precision Score :  1.0
Recall Score :  1.0
```

[115]:
```
from sklearn.metrics import classification_report, confusion_matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=.5, annot=True,square = True, cmap = 'Blues')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Accuracy Score: {0}'.format(ada.score(X_test, y_test)*100)
plt.title(all_sample_title, size = 15)
```

[115]: Text(0.5, 1.0, 'Accuracy Score: 100.0')



[116]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(random_state = 0)
lr.fit(X_train, y_train)
```

```
C:\Users\dell\AppData\Local\Programs\Python\Python312\Lib\site-
packages\sklearn\linear_model\_logistic.py:469: 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(
```

[116]:
```
LogisticRegression(random_state=0)
```

[117]:
```
y_pred = lr.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

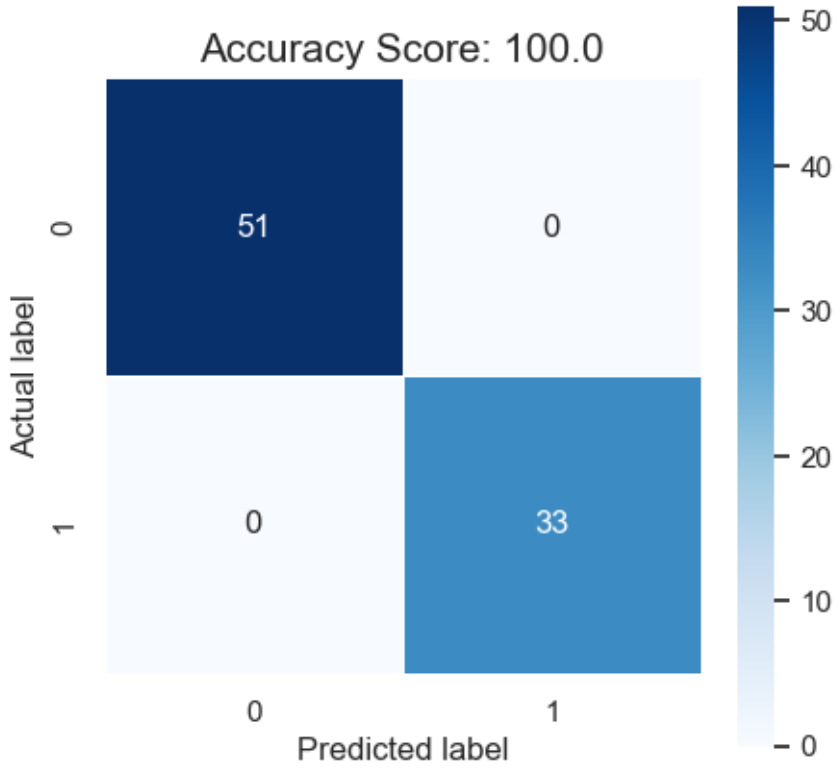
```
Accuracy Score : 100.0 %
```

[118]:
```
from sklearn.metrics import accuracy_score, f1_score, precision_score,␣
 ↪recall_score
print('F-1 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)))
```

```
F-1 Score :  1.0
Precision Score :  1.0
Recall Score :  1.0
```

[119]:
```
from sklearn.metrics import classification_report, confusion_matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=.5, annot=True,square = True, cmap = 'Blues')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Accuracy Score: {0}'.format(lr.score(X_test, y_test)*100)
plt.title(all_sample_title, size = 15)
```

[119]:
```
Text(0.5, 1.0, 'Accuracy Score: 100.0')
```


Accuracy Score: 100.0
51
0
0
33
Actual label
0
1
0
1
Predicted label
50
40
30
20
10
0

[ ]: