Adaboosting

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
dataset = pd.read_csv('C:/Users/HP/Downloads/heart (2).csv')
```

In [3]:
```
dataset.head()
```

Out[3]:

	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 [4]:
```
dataset.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 [5]:
```
 X= dataset.drop( "target",axis=1)
Y= dataset['target']
```

In [6]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.30, random_state
```

In [7]:
```
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("Y_train shape:", y_train.shape)
print("Y_test shape:", y_test.shape)
```

```
X_train shape: (717, 13)
X_test shape: (308, 13)
Y_train shape: (717,)
Y_test shape: (308,)
```

In [8]:
```
from sklearn.ensemble import AdaBoostClassifier
```

In [9]:
```
adb = AdaBoostClassifier(n_estimators=5)
```

In [10]:
```
adb.fit(X_train, y_train)
```

Out[10]:
```
AdaBoostClassifier
AdaBoostClassifier(n_estimators=5)
```

In [11]:
```
y_pred = adb.predict(X_test)
```

In [12]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [13]:
```
confusion_matrix(y_test, y_pred)
```

Out[13]:
```
array([[120,  20],
       [ 33, 135]], dtype=int64)
```

In [15]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, recall_score, precision_sc

conf_mat = confusion_matrix(y_test, y_pred)
TN, FP, FN, TP = conf_mat.ravel()

accuracy = accuracy_score(y_test, y_pred) * 100
sensitivity = recall_score(y_test, y_pred) * 100
specificity = precision_score(y_test, y_pred) * 100

print("True Negative:", TN, "False Negative:", FN, "True Positive:", TP, "False Positive
name = "AdaBoost"

print("--------------Accuracy----------------")
print("Accuracy of", name, " is", round(accuracy, 2))

print("--------------Sensitivity----------------")
print("Sensitivity of", name, " is", round(sensitivity, 2))

print("--------------Specificity----------------")
print("Specificity of", name, " is", round(specificity, 2))
```

```
True Negative: 120 False Negative: 33 True Positive: 135 False Positive: 20
--------------Accuracy----------------
Accuracy of AdaBoost  is 82.79
--------------Sensitivity----------------
Sensitivity of AdaBoost  is 80.36
--------------Specificity----------------
Specificity of AdaBoost  is 87.1
```

In [16]:
```
from sklearn.metrics import classification_report
```

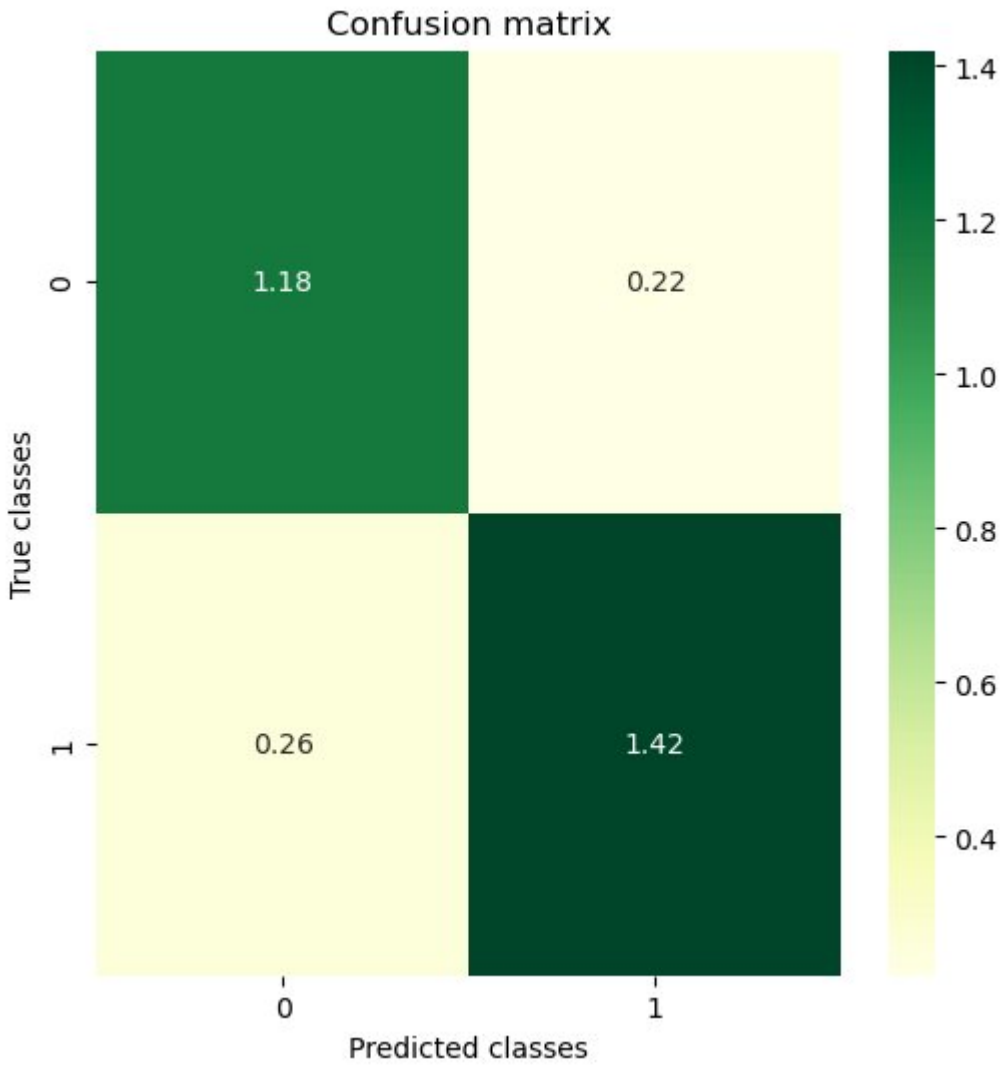
In [17]:
```
report = classification_report(y_test, y_pred)

print(report)
```

```
              precision    recall  f1-score   support
```

```
              0       0.78      0.86      0.82       140
              1       0.87      0.80      0.84       168

    accuracy                            0.83       308
   macro avg       0.83      0.83      0.83       308
weighted avg       0.83      0.83      0.83       308
```

In [18]:

```python
import seaborn as sns
import matplotlib.pyplot as plt

adb = AdaBoostClassifier(n_estimators=10)
adb.fit(X_train, y_train)

y_pred = adb.predict(X_test)

conf_mat = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 6))
sns.heatmap(conf_mat / 100, annot=True, cmap="YlGn", fmt='.2f')
plt.title('Confusion matrix')
plt.xlabel('Predicted classes')
plt.ylabel('True classes')
plt.show()

report = classification_report(y_test, y_pred)

print(report)
```



```
              precision    recall  f1-score   support

           0       0.82      0.84      0.83       140
           1       0.87      0.85      0.86       168

    accuracy                           0.84       308
   macro avg       0.84      0.84      0.84       308
weighted avg       0.84      0.84      0.84       308
```

Random Forest

In [19]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [20]:
```
dataset = pd.read_csv('C:/Users/HP/Downloads/heart (2).csv')
```

In [21]:
```
dataset.head()
```

Out[21]:

	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 [22]:
```
dataset.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 [23]:
```
X = dataset.drop( "target",axis=1)
```

```
Y= dataset['target']
```

In [24]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.30, random_state
```

In [25]:
```
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("Y_train shape:", y_train.shape)
print("Y_test shape:", y_test.shape)
```

```
X_train shape: (717, 13)
X_test shape: (308, 13)
Y_train shape: (717,)
Y_test shape: (308,)
```

In [26]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [27]:
```
rf = RandomForestClassifier(n_estimators=5)
```

In [28]:
```
rf.fit(X_train, y_train)
```

Out[28]:
```
RandomForestClassifier
RandomForestClassifier(n_estimators=5)
```

In [29]:
```
y_pred = rf.predict(X_test)
```

In [30]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [31]:
```
confusion_matrix(y_test, y_pred)
```

Out[31]:
```
array([[140,   0],
       [  4, 164]], dtype=int64)
```

In [32]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, recall_score, precision_sc

conf_mat = confusion_matrix(y_test, y_pred)
TN, FP, FN, TP = conf_mat.ravel()

accuracy = accuracy_score(y_test, y_pred) * 100
sensitivity = recall_score(y_test, y_pred) * 100
specificity = precision_score(y_test, y_pred) * 100

print("True Negative:", TN, "False Negative:", FN, "True Positive:", TP, "False Positive
name = "Random Forest"

print("--------------Accuracy----------------")
print("Accuracy of", name, " is", round(accuracy, 2))

print("--------------Sensitivity----------------")
print("Sensitivity of", name, " is", round(sensitivity, 2))

print("--------------Specificity----------------")
print("Specificity of", name, " is", round(specificity, 2))
```

```
True Negative: 140 False Negative: 4 True Positive: 164 False Positive: 0
--------------Accuracy----------------
Accuracy of Random Forest  is 98.7
--------------Sensitivity----------------
Sensitivity of Random Forest  is 97.62
```

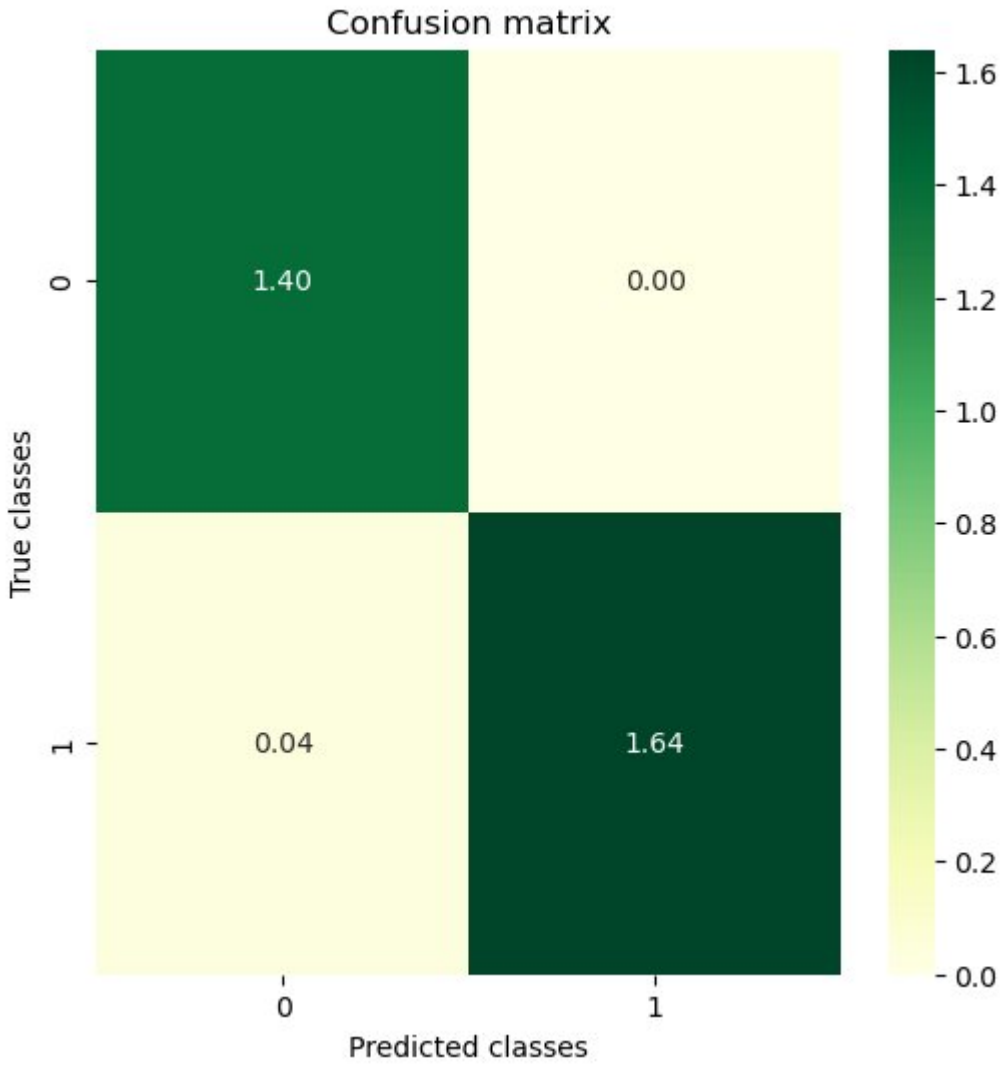
```
--------------Specificity----------------
Specificity of Random Forest  is 100.0
```

In [33]:
```
from sklearn.metrics import classification_report
```

In [34]:
```
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.99       140
           1       1.00      0.98      0.99       168

    accuracy                           0.99       308
   macro avg       0.99      0.99      0.99       308
weighted avg       0.99      0.99      0.99       308
```

In [35]:
```
import seaborn as sns
import matplotlib.pyplot as plt

conf_mat = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 6))
sns.heatmap(conf_mat / 100, annot=True, cmap="YlGn", fmt='.2f')
plt.title('Confusion matrix')
plt.xlabel('Predicted classes')
plt.ylabel('True classes')
plt.show()
```



Support Vector Machine

In [36]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [37]:
```
dataset = pd.read_csv('C:/Users/HP/Downloads/heart (2).csv')
```

In [38]:
```
dataset.head()
```

Out[38]:

	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 [39]:
```
dataset.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 [40]:
```
 X= dataset.drop( "target",axis=1)
Y= dataset['target']
```

In [41]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.30, random_state
```

In [42]:
```
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("Y_train shape:", y_train.shape)
print("Y_test shape:", y_test.shape)
```

```
X_train shape: (717, 13)
X_test shape: (308, 13)
Y_train shape: (717,)
Y_test shape: (308,)
```

In [43]:
```
from sklearn.svm import SVC
```

In [44]:
```
svc = SVC(kernel='linear', C=5)
```

In [45]:
```
svc.fit(X_train, y_train)
```

Out[45]:
```
SVC(C=5, kernel='linear')
```

In [46]:
```
y_pred = svc.predict(X_test)
```

In [47]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [48]:
```
confusion_matrix(y_test, y_pred)
```

Out[48]:
```
array([[112,  28],
       [ 12, 156]], dtype=int64)
```

In [51]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, recall_score, precision_sc

from sklearn.svm import SVC

svc = SVC(kernel='linear', C=1.0)

svc.fit(X_train, y_train)

y_pred = svc.predict(X_test)

conf_mat = confusion_matrix(y_test, y_pred)
TN, FP, FN, TP = conf_mat.ravel()

accuracy = accuracy_score(y_test, y_pred) * 100
sensitivity = recall_score(y_test, y_pred) * 100
specificity = precision_score(y_test, y_pred) * 100

print("True Negative:", TN, "False Negative:", FN, "True Positive:", TP, "False Positive
name = "SVM"

print("--------------Accuracy----------------")
print("Accuracy of", name, " is", round(accuracy, 2))

print("--------------Sensitivity----------------")
print("Sensitivity of", name, " is", round(sensitivity, 2))

print("--------------Specificity----------------")
print("Specificity of", name, " is", round(specificity, 2))
```

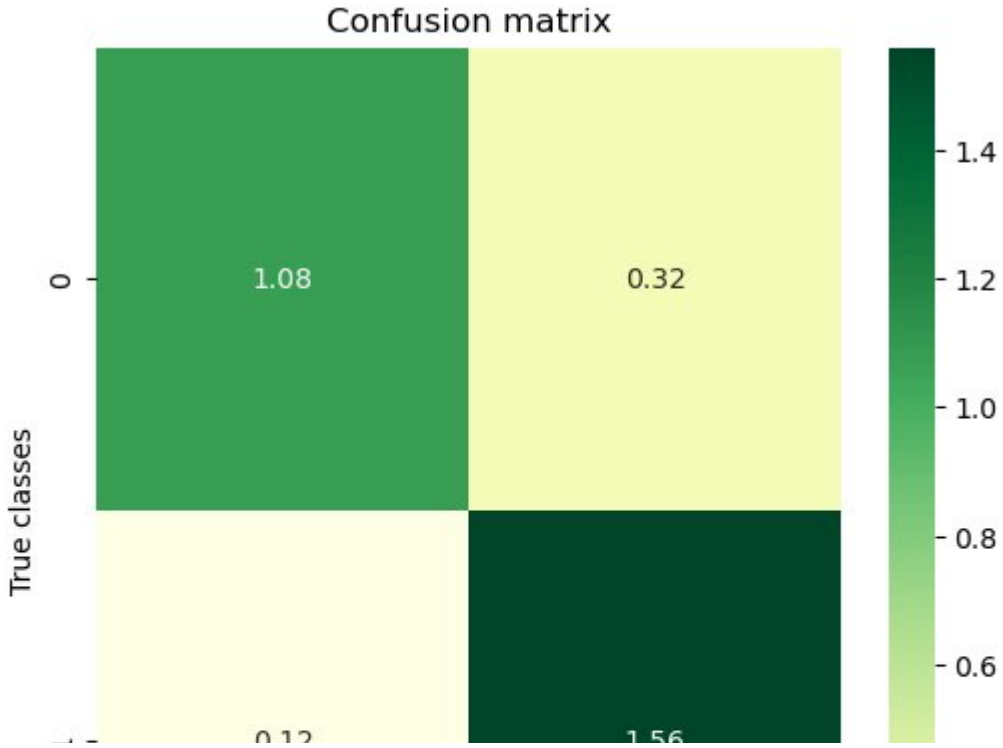
```
True Negative: 108 False Negative: 12 True Positive: 156 False Positive: 32
--------------Accuracy----------------
Accuracy of SVM  is 85.71
--------------Sensitivity----------------
Sensitivity of SVM  is 92.86
--------------Specificity----------------
Specificity of SVM  is 82.98
```

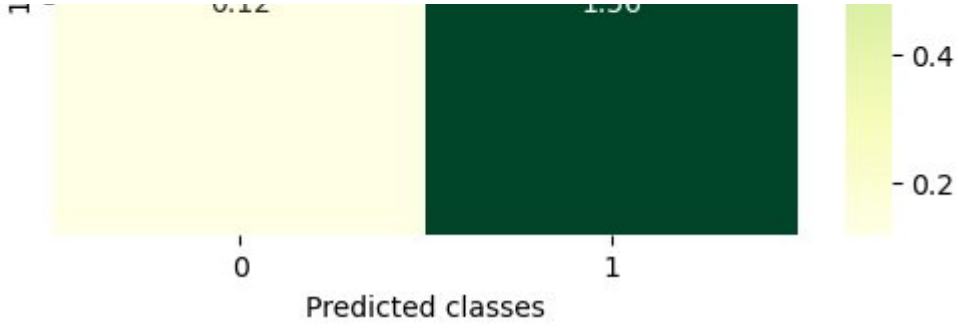
In [52]:
```
from sklearn.metrics import classification_report
```

In [53]:
```
report = classification_report(y_test, y_pred)

print(report)
```

```
              precision    recall  f1-score   support

           0       0.90      0.77      0.83       140
           1       0.83      0.93      0.88       168

    accuracy                           0.86       308
   macro avg       0.86      0.85      0.85       308
weighted avg       0.86      0.86      0.86       308
```

In [54]:
```
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.svm import SVC

svc = SVC(kernel='linear', C=1.0)

svc.fit(X_train, y_train)

y_pred = svc.predict(X_test)

conf_mat = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 6))
sns.heatmap(conf_mat / 100, annot=True, cmap="YlGn", fmt='.2f')
plt.title('Confusion matrix')
plt.xlabel('Predicted classes')
plt.ylabel('True classes')
plt.show()

report = classification_report(y_test, y_pred)

print(report)
```





	precision	recall	f1-score	support
0	0.90	0.77	0.83	140
1	0.83	0.93	0.88	168
accuracy			0.86	308
macro avg	0.86	0.85	0.85	308
weighted avg	0.86	0.86	0.86	308

Multilayer Perception

In [55]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [56]:
```
dataset = pd.read_csv('C:/Users/HP/Downloads/heart (2).csv')
```

In [57]:
```
dataset.head()
```

Out[57]:

	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 [58]:
```
dataset.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 [59]:
```
 X= dataset.drop( "target",axis=1)
Y= dataset['target']
```

In [60]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.30, random_state
```

In [61]:
```
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("Y_train shape:", y_train.shape)
print("Y_test shape:", y_test.shape)
```

```
X_train shape: (717, 13)
X_test shape: (308, 13)
Y_train shape: (717,)
Y_test shape: (308,)
```

In [66]:
```
from sklearn.neural_network import MLPClassifier
```

In [72]:
```
hidden_layer_sizes = 5
```

In [73]:
```
mlp = MLPClassifier(hidden_layer_sizes=hidden_layer_sizes,
                    activation='relu',
                    solver='adam',
                    max_iter=200)
```

In [74]:
```
mlp.fit(X_train, y_train)
```

Out[74]:
```
MLPClassifier
MLPClassifier(hidden_layer_sizes=5)
```

In [75]:
```
y_pred = mlp.predict(X_test)
```

In [76]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [77]:
```
confusion_matrix(y_test, y_test)
```

Out[77]:
```
array([[140,   0],
       [  0, 168]], dtype=int64)
```

In [78]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, recall_score, precision_sc

conf_mat = confusion_matrix(y_test, y_pred)
TN, FP, FN, TP = conf_mat.ravel()

accuracy = accuracy_score(y_test, y_pred) * 100
sensitivity = recall_score(y_test, y_pred) * 100
specificity = precision_score(y_test, y_pred) * 100

print("True Negative:", TN, "False Negative:", FN, "True Positive:", TP, "False Positive
name = "MLP"
```

```
print("--------------Accuracy----------------")
print("Accuracy of", name, " is", round(accuracy, 2))

print("--------------Sensitivity----------------")
print("Sensitivity of", name, " is", round(sensitivity, 2))

print("--------------Specificity----------------")
print("Specificity of", name, " is", round(specificity, 2))
```

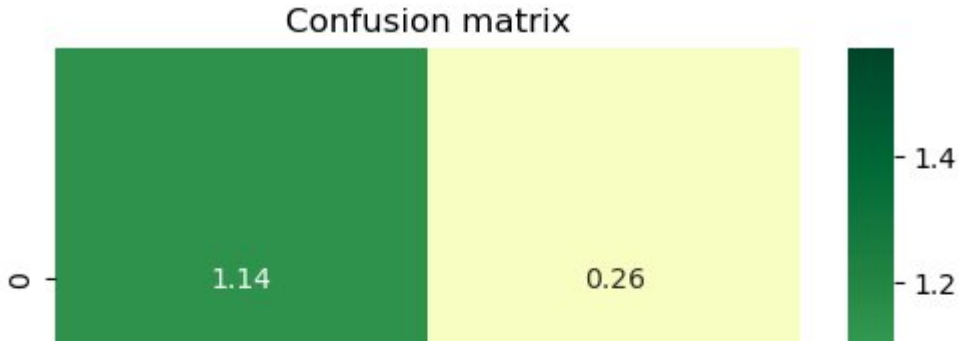
```
True Negative: 0 False Negative: 0 True Positive: 168 False Positive: 140
--------------Accuracy----------------
Accuracy of MLP  is 54.55
--------------Sensitivity----------------
Sensitivity of MLP  is 100.0
--------------Specificity----------------
Specificity of MLP  is 54.55
```

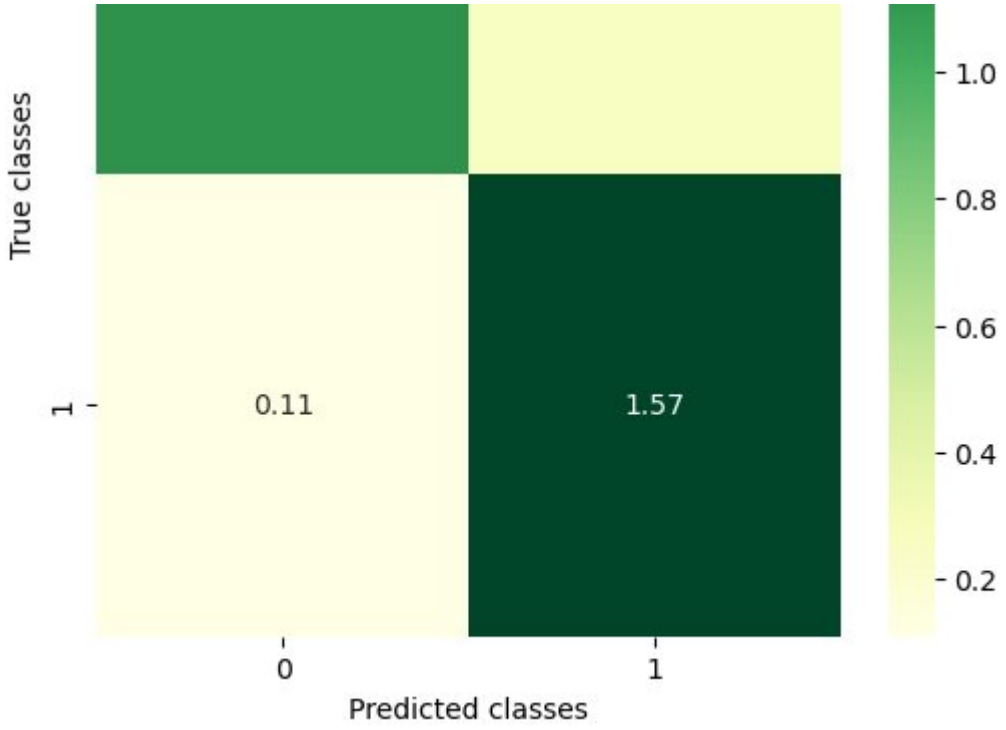
In [81]:
```
from sklearn.metrics import classification_report
```

In [82]:
```
report = classification_report(y_test, y_pred)

print(report)
```

```
              precision    recall  f1-score   support

           0       0.91      0.84      0.88       140
           1       0.88      0.93      0.90       168

    accuracy                           0.89       308
   macro avg       0.90      0.89      0.89       308
weighted avg       0.89      0.89      0.89       308
```

In [83]:
```
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier(hidden_layer_sizes=(100, ), activation='relu', solver='adam', max_it
mlp.fit(X_test, y_test)

y_pred = mlp.predict(X_test)

conf_mat = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 6))
sns.heatmap(conf_mat / 100, annot=True, cmap="YlGn", fmt='.2f')
plt.title('Confusion matrix')
plt.xlabel('Predicted classes')
plt.ylabel('True classes')
plt.show()
```





In [ ]: