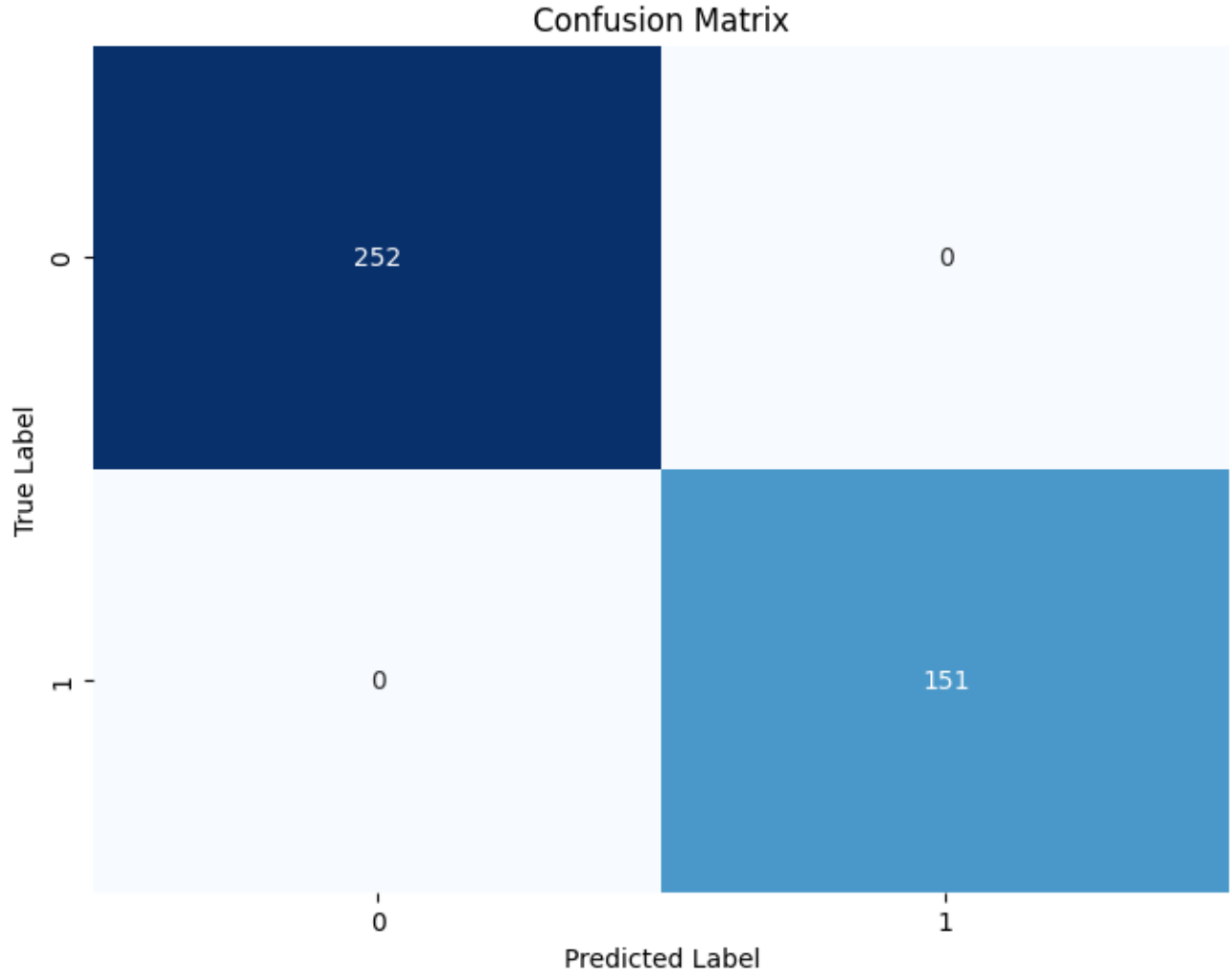
```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, classification_report

# Importing the dataset
df = pd.read_csv("/content/water_potability (1).csv")

# Dropping rows with missing values
df = df.dropna()

# Dropping unnecessary columns
columns_to_drop = ['ph']
X = df.drop(columns=columns_to_drop)
y = df['Potability']

# Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_

# Training the Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=0)
rf_classifier.fit(X_train, y_train)

# Predicting the test set results
y_pred = rf_classifier.predict(X_test)

# Evaluating the model
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

# Plotting the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d', cbar=False)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()

# Displaying the classification report
print(class_report)
```

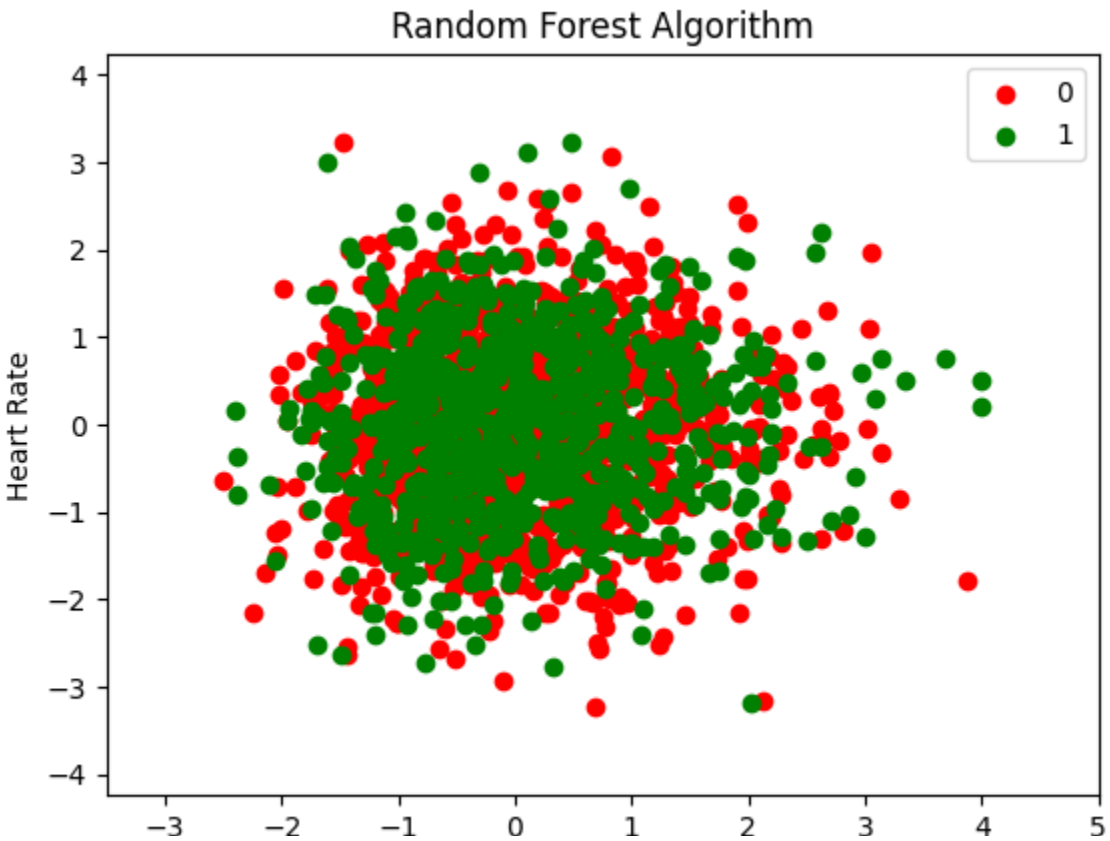


	precision	recall	f1-score	support
0	1.00	1.00	1.00	252
1	1.00	1.00	1.00	151
accuracy			1.00	403
macro avg	1.00	1.00	1.00	403
weighted avg	1.00	1.00	1.00	403

```
# Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from matplotlib.colors import ListedColormap


# Feature Scaling
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Fitting Random Forest classifier to the dataset
classifier = RandomForestClassifier(n_estimators=10, criterion="entropy", random_state
classifier.fit(X_scaled, y)

# Visualizing the dataset with the Random Forest algorithm
X1, X2 = np.meshgrid(np.arange(start=X_scaled[:, 0].min() - 1, stop=X_scaled[:, 0].max
                     np.arange(start=X_scaled[:, 1].min() - 1, stop=X_scaled[:, 1].max
#plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape
             #alpha=0.75, cmap=ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y)):
    plt.scatter(X_scaled[y == j, 0], X_scaled[y == j, 1],
                c=ListedColormap(('red', 'green'))(i), label=j)
y = [95,95,95]
plt.title('Random Forest Algorithm')
plt.xlabel('SPO2')
plt.ylabel('Heart Rate')
plt.legend()
plt.show()
```

```
<ipython-input-46-93093bd54a2e>:26: UserWarning: *c* argument looks like a single
  plt.scatter(X_scaled[y == j, 0], X_scaled[y == j, 1],
```



SPO2

```
sheet_id = '1EyQm3BIQ21r86rHPmyNQC_j2jU9GQREMZbNamJL02Cc'
data = pd.read_csv(f"https://docs.google.com/spreadsheets/d/{sheet_id}/export?format=c
X = data[['Pulse data', 'Temperature', 'SPO2']]  # Features


# Splitting the dataset into training and testing sets


# Initializing the Random Forest classifier
classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Fitting the classifier to the training data
classifier.fit(X_train, y_train)

# Predicting on the test data
y_pred = classifier.predict(X_test)

# Evaluating the model
a="Healthy"
b="Sick"

if(y_pred.any==0):
  print(b\n)
else:
  print(a\n)
  print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
a

Confusion Matrix:
[[252   0]
 [  0 151]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       252
           1       1.00      1.00      1.00       151

    accuracy                           1.00       403
   macro avg       1.00      1.00      1.00       403
weighted avg       1.00      1.00      1.00       403
```