In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.model_selection import KFold, StratifiedKFold, cross_v
from sklearn import linear_model, tree, ensemble
```

In [2]:
```
dataframe=pd.read_csv("/heart - UCI.csv")
dataframe.head(10)
```

Out[2]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	ta
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	
7	44	1	1	120	263	0	1	173	0	0.0	2	0	3	
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	

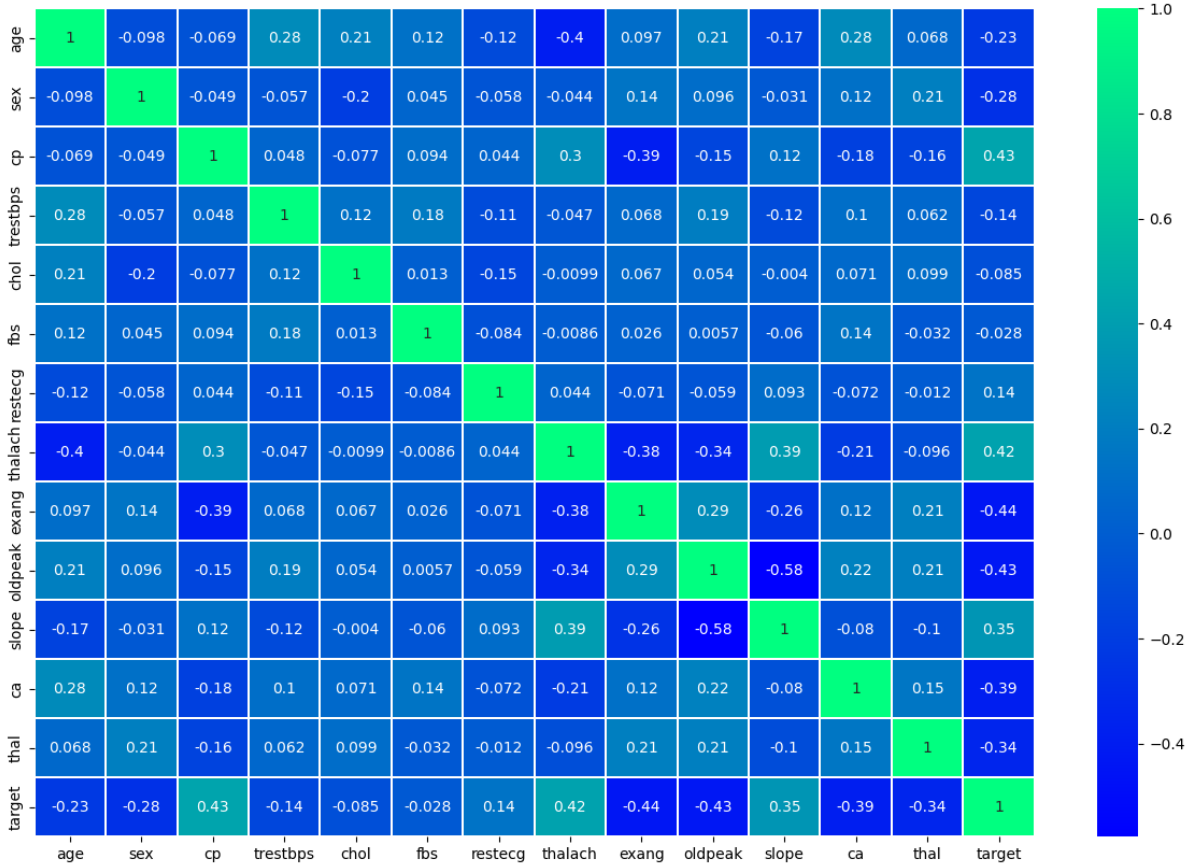
In [3]:
```
dataframe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

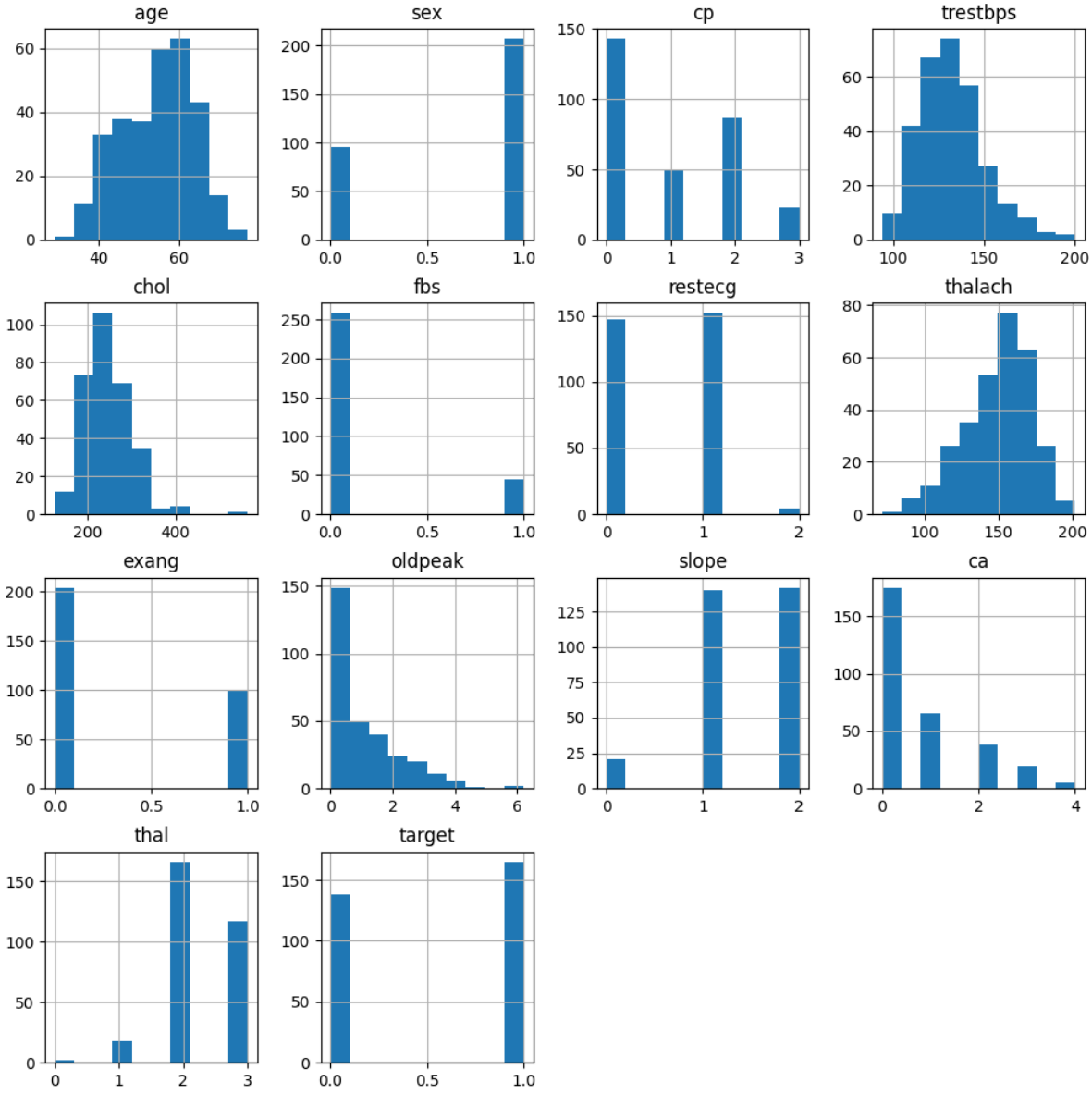
In [4]:
```
dataframe.isna().sum()
```

Out[4]:
```
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 [5]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(15, 10))
sns.heatmap(dataframe.corr(), linewidth=0.01, annot=True, cmap="win
plt.savefig('correlationfigure.png')  # Save the figure before show
plt.show()
```



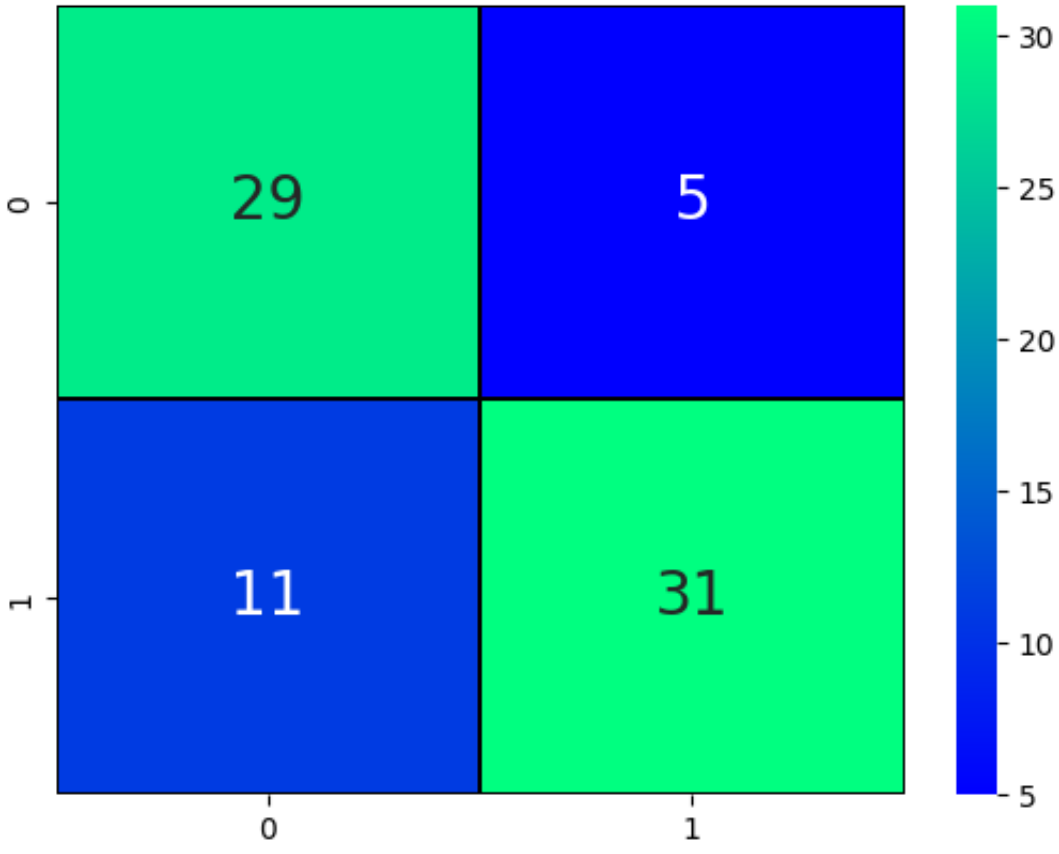
In [6]:
```
import matplotlib.pyplot as plt

dataframe.hist(figsize=(12, 12))
plt.savefig('featuresplot.png')  # Save the figure before showing
plt.show()
```



In [7]:
```
import pandas as pd
from sklearn.model_selection import train_test_split

# Assuming you have a DataFrame named 'dataframe' with the mentione
# Adjust the column names accordingly based on your dataset

# Extract features (X) and target variable (y)
X = dataframe[['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'rest
y = dataframe['target']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
```

In [15]:
```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report

def evaluate_model(model, X_test, y_test):
    prediction = model.predict(X_test)
    cm = confusion_matrix(y_test, prediction)

    sns.heatmap(cm, annot=True, cmap='winter', linewidths=0.3, line
    plt.show()

    print(classification_report(y_test, prediction))

    TP, TN, FN, FP = cm[0, 0], cm[1, 1], cm[1, 0], cm[0, 1]
    accuracy = (TP + TN) / (TP + TN + FN + FP)
    sensitivity = TP / (TP + FN)
    specificity = TN / (TN + FP)
    precision = TP / (TP + FP)

    print('Testing Accuracy:', accuracy)
    print('Testing Sensitivity:', sensitivity)
    print('Testing Specificity:', specificity)
    print('Testing Precision:', precision)

# Create and fit the Decision Tree model
tree_model = DecisionTreeClassifier(max_depth=5, criterion='entropy
tree_model.fit(X_train, y_train)

# Evaluate the model
evaluate_model(tree_model, X_test, y_test)
```

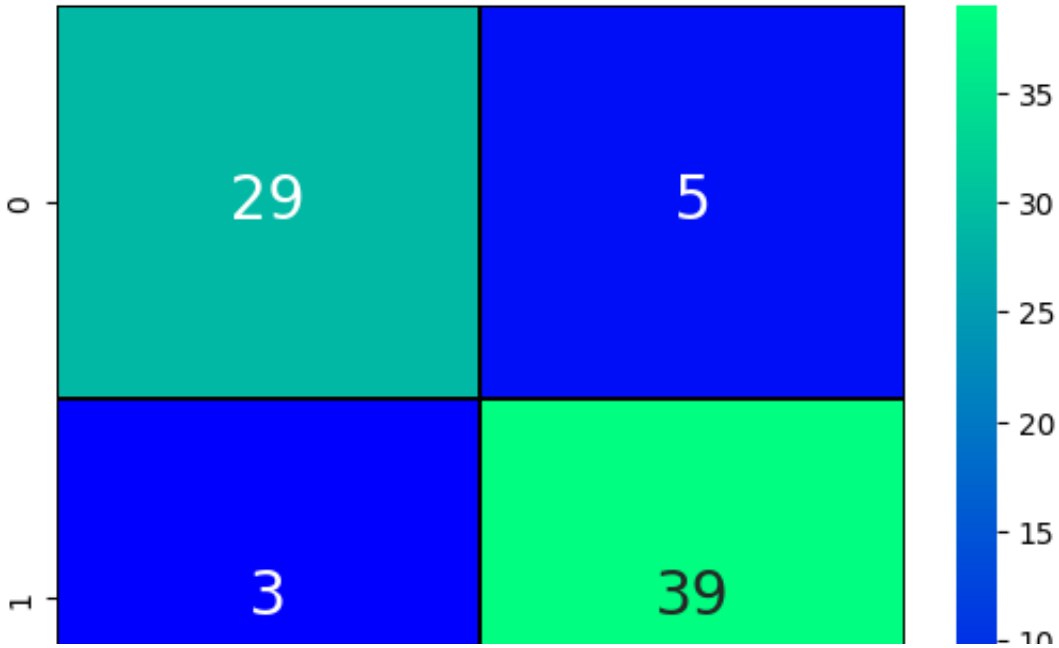


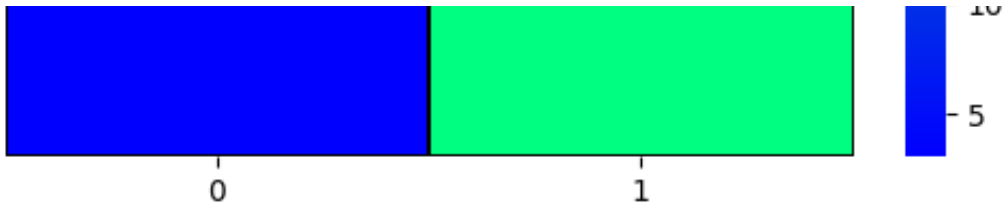

```
              precision    recall  f1-score   support
```

```
              0       0.72      0.85      0.78        34
              1       0.86      0.74      0.79        42

    accuracy                            0.79        76
   macro avg       0.79      0.80      0.79        76
weighted avg       0.80      0.79      0.79        76

Testing Accuracy: 0.7894736842105263
Testing Sensitivity: 0.725
Testing Specificity: 0.8611111111111112
Testing Precision: 0.8529411764705882
```

In [29]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression

# Define the parameter grid with valid combinations of solver and p
param_grid_lr = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'penalty': ['l2'],
    'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']
}

# Create GridSearchCV object
grid_search_lr = GridSearchCV(estimator=LogisticRegression(class_we

# Fit the GridSearchCV object
grid_search_lr.fit(X_train, y_train)

# Get the best parameters
best_params_lr = grid_search_lr.best_params_

# Train the model with the best parameters
best_lr_model = LogisticRegression(class_weight='balanced', max_ite
best_lr_model.fit(X_train, y_train)

# Evaluate the model
evaluate_model(best_lr_model, X_test, y_test)
```



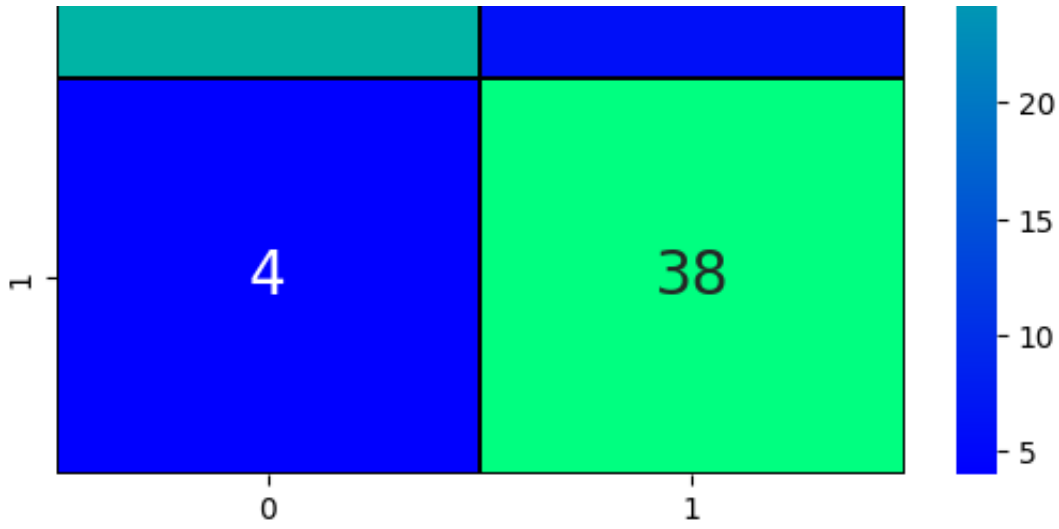


	precision	recall	f1-score	support
0	0.91	0.85	0.88	34
1	0.89	0.93	0.91	42
accuracy			0.89	76
macro avg	0.90	0.89	0.89	76
weighted avg	0.90	0.89	0.89	76

```
Testing Accuracy: 0.8947368421052632
Testing Sensitivity: 0.90625
Testing Specificity: 0.8863636363636364
Testing Precision: 0.8529411764705882
```

In [27]:

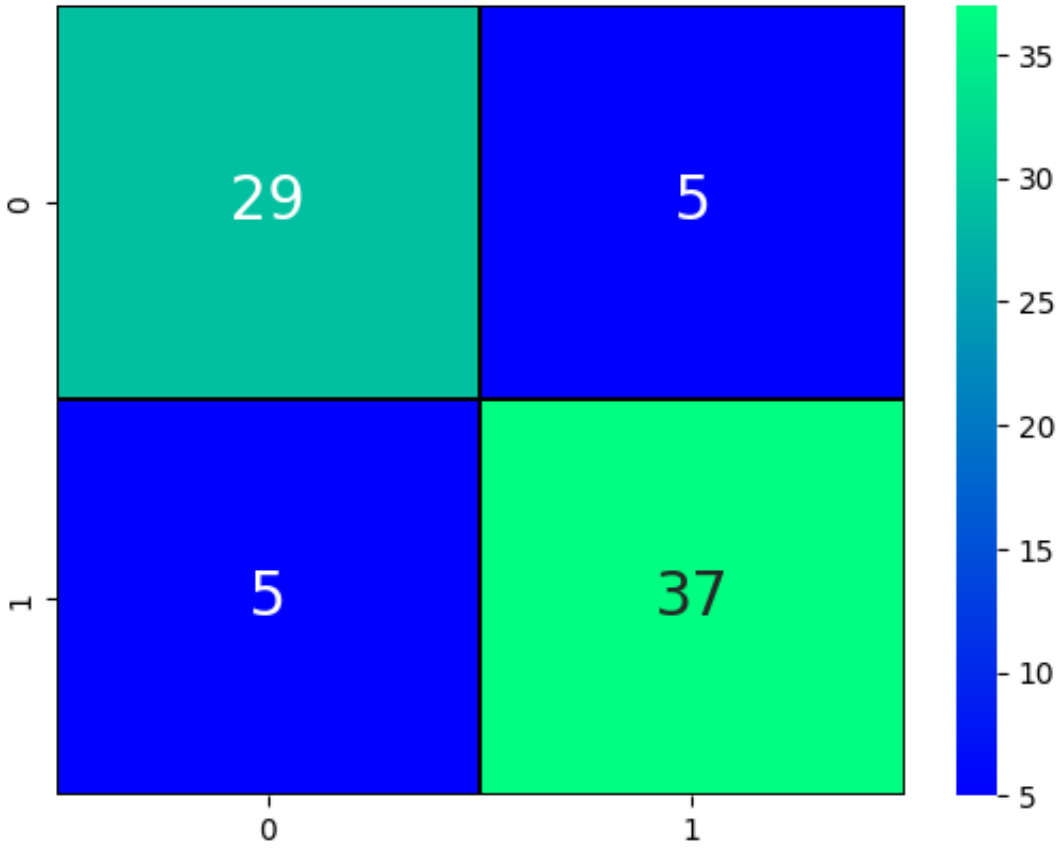
```python
from sklearn.model_selection import GridSearchCV

# Define the parameter grid
param_grid_rf = {
    'n_estimators': [100, 200, 300, 500],
    'max_depth': [5, 8, 10, 15, None],
    'min_samples_split': [2, 5, 10],
    'criterion': ['gini', 'entropy']
}

# Create GridSearchCV object
grid_search_rf = GridSearchCV(estimator=RandomForestClassifier(), p

# Fit the GridSearchCV object
grid_search_rf.fit(X_train, y_train)

# Get the best parameters
best_params_rf = grid_search_rf.best_params_

# Train the model with the best parameters
best_rf_model = RandomForestClassifier(**best_params_rf)
best_rf_model.fit(X_train, y_train)

# Evaluate the model
evaluate_model(best_rf_model, X_test, y_test)
```



	precision	recall	f1-score	support
0	0.88	0.82	0.85	34
1	0.86	0.90	0.88	42
accuracy			0.87	76
macro avg	0.87	0.86	0.87	76
weighted avg	0.87	0.87	0.87	76

Testing Accuracy: 0.868421052631579
Testing Sensitivity: 0.875
Testing Specificity: 0.8636363636363636
Testing Precision: 0.8235294117647058

In [24]:
```
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

svm = SVC(C=12, kernel='linear')
model4 = svm.fit(X_train, y_train)
prediction4 = model4.predict(X_test)
cm4 = confusion_matrix(y_test, prediction4)

sns.heatmap(cm4, annot=True, cmap='winter', linewidths=0.3, linecol
plt.show()

print(classification_report(y_test, prediction4))
```




```
              precision    recall  f1-score   support

           0       0.85      0.85      0.85        34
           1       0.88      0.88      0.88        42

    accuracy                           0.87        76
   macro avg       0.87      0.87      0.87        76
weighted avg       0.87      0.87      0.87        76
```