```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from itertools import product
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report


df = pd.read_csv('/content/heart.csv')
df
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
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
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

303 rows × 14 columns

```
df.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
```

```
df.shape
```

```
(303, 14)
```

```
df.columns
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

```
null_values = df.isnull().sum()
print("Null values in each column:\n", null_values)
```

```
Null values in each column:
 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
```

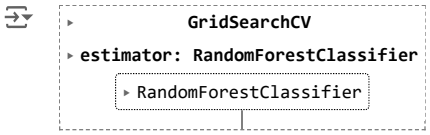
```
X = df.drop("target", axis=1)
y = df["target"].apply(lambda x: 1 if x > 0 else 0)  # Binarize the target
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Hyperparameters: n_estimators, max_depth, min_samples_split, and max_leaf_nodes.

```
# Define hyperparameters and their values
hyperparameters = {
    'n_estimators': [100, 500],
    'max_depth': [10, 20],
    'min_samples_split': [2, 10],
    'max_leaf_nodes': [10, 20]
}
```

```
# Hyperparater Optimazation using GridSearch
model = RandomForestClassifier()
model_gs = GridSearchCV(estimator=model, param_grid=hyperparameters)
model_gs.fit(X_train, y_train)
```




```
# Get the best parameters
model_gs.best_params_
```

```
{'max_depth': 20,
 'max_leaf_nodes': 10,
 'min_samples_split': 2,
 'n_estimators': 100}
```

```
# Get the best score
model_gs.best_score_
```

```
0.8304421768707482
```

Grid Search

```
# Random Forest with Grid Search
y_pred_gs = model_gs.predict(X_test)

print("\nRandom Forest Grid Search Performance:")
print("Accuracy:", accuracy_score(y_pred_gs, y_test))
print("\nClassification Report:")
print(classification_report(y_pred_gs, y_test))
```

```
Random Forest Grid Search Performance:
Accuracy: 0.8852459016393442

Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.89      0.88        28
           1       0.91      0.88      0.89        33
```

```
          accuracy                           0.89        61
         macro avg       0.88      0.89      0.88        61
      weighted avg       0.89      0.89      0.89        61
```

Random Search

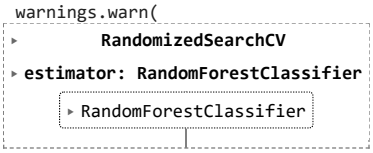
```
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV


# Initialize the RandomForestClassifier
rf = RandomForestClassifier()

# Initialize RandomizedSearchCV
random_search = RandomizedSearchCV(
    estimator=rf,
    param_distributions=hyperparameters,
    n_iter=100,  # Number of parameter settings that are sampled
    cv=5,  # 5-fold cross-validation
    verbose=2,
    random_state=42,
    n_jobs=-1  # Use all available cores
)

# Fit RandomizedSearchCV to the data
random_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:305: UserWarr
  warnings.warn(
```




```
from sklearn.metrics import precision_score, recall_score, f1_score, confusion_matrix, accuracy_score

# Print the best parameters and the corresponding score
print(f"Best parameters found: {random_search.best_params_}")
print(f"Best cross-validation score: {random_search.best_score_}")

# Predict with the best estimator
best_rf = random_search.best_estimator_
y_pred = best_rf.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Test set accuracy: {accuracy}")

# Calculate and print precision, recall, f1 score, and support
precision = precision_score(y_test, y_pred, average=None)
recall = recall_score(y_test, y_pred, average=None)
f1 = f1_score(y_test, y_pred, average=None)
cm = confusion_matrix(y_test, y_pred)

# Calculate support from confusion matrix
support = cm.sum(axis=1)

# Print the classification report manually
print("\nClassification Report:")
print(f"{'Class':<10}{'Precision':<10}{'Recall':<10}{'F1-Score':<10}{'Support':<10}")
for i in range(len(precision)):
    print(f"{i:<10}{precision[i]:<10.2f}{recall[i]:<10.2f}{f1[i]:<10.2f}{support[i]:<10}")
```

```
Best parameters found: {'n_estimators': 100, 'min_samples_split': 10, 'max_leaf_nodes': 10, 'max_depth': 10}
Best cross-validation score: 0.8263605442176871
Test set accuracy: 0.8852459016393442

Classification Report:
Class     Precision Recall    F1-Score  Support
```

```
0          0.89      0.86      0.88      29
1          0.88      0.91      0.89      32
```