heart classification

March 24, 2023

[1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
```

[2]:
```
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, f1_score, precision_score,␣
 ↪recall_score
```

[3]:
```
import warnings
warnings.filterwarnings("ignore")
```

[4]:
```
data = pd.read_csv('heart.csv')
```

[5]:
```
data.head(2)
```

[5]:

	@inputs Age	Sex	ChestPainType	RestBloodPressure	SerumCholestoral	FastingBloodSugar	ResElectrocardiographic	MaxHeartRate	ExerciseInduced	Oldpeak	Slope	MajorVessels	Thal	Class
0	70	1	4	130	322	0	2	109	0	24	2	3	3	2
1	67	0	3	115	564	0	2	160	0	16	2	0	7	1

[6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 270 entries, 0 to 269
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   @inputs Age             270 non-null    int64
 1   Sex                     270 non-null    int64
 2   ChestPainType           270 non-null    int64
```

```
 3   RestBloodPressure         270 non-null    int64
 4   SerumCholestoral          270 non-null    int64
 5   FastingBloodSugar         270 non-null    int64
 6   ResElectrocardiographic   270 non-null    int64
 7   MaxHeartRate              270 non-null    int64
 8   ExerciseInduced           270 non-null    int64
 9   Oldpeak                   270 non-null    int64
 10  Slope                     270 non-null    int64
 11  MajorVessels              270 non-null    int64
 12  Thal                      270 non-null    int64
 13  Class                     270 non-null    int64
dtypes: int64(14)
memory usage: 29.7 KB
```

[7]:
```
data = data.rename(columns={"@inputs Age": "Age"})
```

[8]:
```
data1 = data
```

[9]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 270 entries, 0 to 269
Data columns (total 14 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       270 non-null    int64
 1   Sex                       270 non-null    int64
 2   ChestPainType             270 non-null    int64
 3   RestBloodPressure         270 non-null    int64
 4   SerumCholestoral          270 non-null    int64
 5   FastingBloodSugar         270 non-null    int64
 6   ResElectrocardiographic   270 non-null    int64
 7   MaxHeartRate              270 non-null    int64
 8   ExerciseInduced           270 non-null    int64
 9   Oldpeak                   270 non-null    int64
 10  Slope                     270 non-null    int64
 11  MajorVessels              270 non-null    int64
 12  Thal                      270 non-null    int64
 13  Class                     270 non-null    int64
dtypes: int64(14)
memory usage: 29.7 KB
```

[19]:
```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data.iloc[:, :-1], data.
 ↪iloc[:, -1], test_size=0.2, random_state=42)
```

0.0.1 Logistic Regression

[20]:
```
# Train and evaluate a Logistic Regression model
lr_model = LogisticRegression(random_state=42)
```

[21]:
```
lr_model.fit(X_train, y_train)
lr_preds = lr_model.predict(X_test)
```

[22]:
```
lr_acc = accuracy_score(y_test, lr_preds)
lr_prec = precision_score(y_test, lr_preds)
lr_rec = recall_score(y_test, lr_preds)
lr_f1 = f1_score(y_test, lr_preds)
```

[23]:
```
print("Logistic Regression Accuracy:", lr_acc)
print("Logistic Regression Precision:", lr_prec)
print("Logistic Regression Recall:", lr_rec)
print("Logistic Regression F1 Score:", lr_f1)
```

```
Logistic Regression Accuracy: 0.8888888888888888
Logistic Regression Precision: 0.8857142857142857
Logistic Regression Recall: 0.9393939393939394
Logistic Regression F1 Score: 0.9117647058823529
```

[24]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import numpy as np
import itertools
```

[25]:
```
# Define class labels
classes = ['Class A', 'Class B']
```

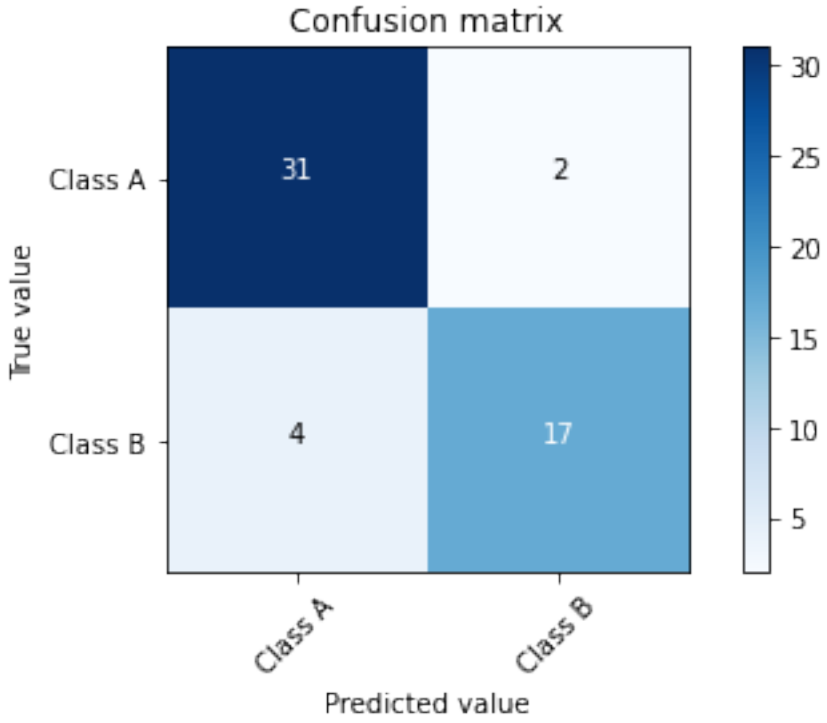
[26]:
```
# Compute confusion matrix
cm = confusion_matrix(y_test, lr_preds)
```

[27]:
```
# Plot confusion matrix
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confusion matrix')
plt.colorbar()
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation=45)
plt.yticks(tick_marks, classes)
plt.xlabel('Predicted value')
plt.ylabel('True value')

# Add text to each cell
thresh = cm.max() / 2.
for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, format(cm[i, j], 'd'),
```

```
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")

plt.tight_layout()
plt.show()
```



0.0.2 Decision Tree

[28]:
```
# Train and evaluate a Decision Tree model
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
```

[28]:
```
DecisionTreeClassifier(random_state=42)
```

[29]:
```
dt_preds = dt_model.predict(X_test)
dt_acc = accuracy_score(y_test, dt_preds)
dt_prec = precision_score(y_test, dt_preds)
dt_rec = recall_score(y_test, dt_preds)
dt_f1 = f1_score(y_test, dt_preds)
```

[30]:
```
print("Results of Decision Tree:")
print("Decision Tree Accuracy:", dt_acc)
print("Decision Tree Precision:", dt_prec)
```

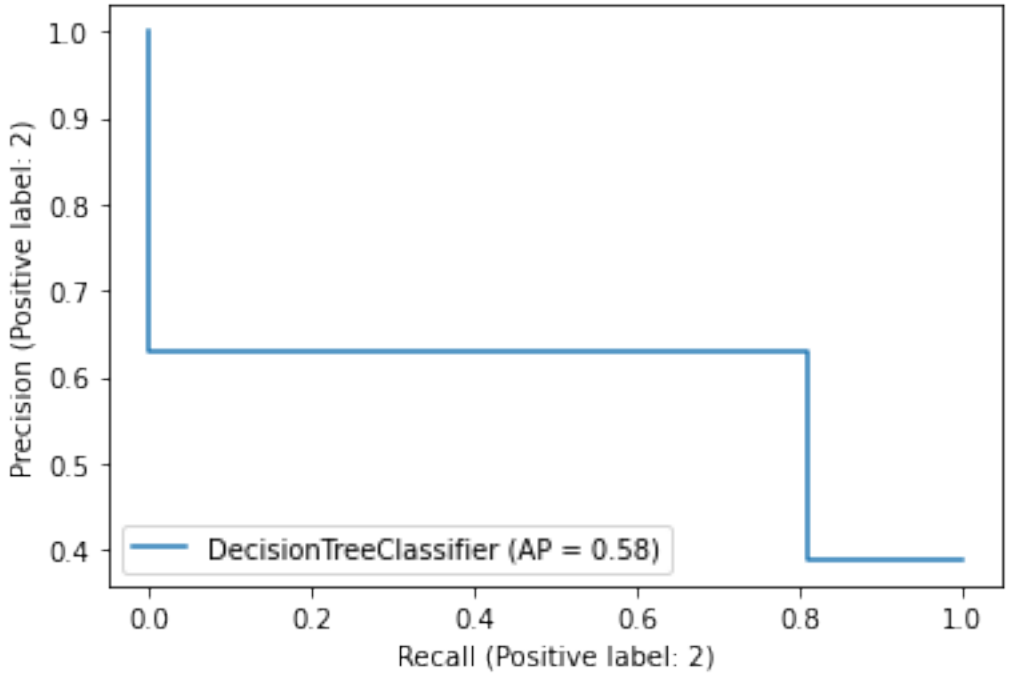
```
print("Decision Tree Recall:", dt_rec)
print("Decision Tree F1 Score:", dt_f1)
```

```
Results of Decision Tree:
Decision Tree Accuracy: 0.7407407407407407
Decision Tree Precision: 0.8518518518518519
Decision Tree Recall: 0.696969696969697
Decision Tree F1 Score: 0.7666666666666667
```

[31]:
```
from sklearn.metrics import plot_precision_recall_curve
import matplotlib.pyplot as plt
```

[32]:
```
plot_precision_recall_curve(dt_model, X_test, y_test)
```

[32]:
```
<sklearn.metrics._plot.precision_recall_curve.PrecisionRecallDisplay at
0x1ee4699fbb0>
```



0.0.3 Gradient Boosting

[33]:
```
gb_model = GradientBoostingClassifier(random_state=42)
gb_model.fit(X_train, y_train)
```

[33]:
```
GradientBoostingClassifier(random_state=42)
```

[34]:
```
gb_preds = gb_model.predict(X_test)
```

[35]:
```
gb_acc = accuracy_score(y_test, gb_preds)
gb_prec = precision_score(y_test, gb_preds)
gb_rec = recall_score(y_test, gb_preds)
gb_f1 = f1_score(y_test, gb_preds)
```

[36]:
```
print("Results of Gradient Boosting")
print("Gradient Boosting Accuracy:", gb_acc)
print("Gradient Boosting Precision:", gb_prec)
print("Gradient Boosting Recall:", gb_rec)
print("Gradient Boosting F1 Score:", gb_f1)
```

```
Results of Gradient Boosting
Gradient Boosting Accuracy: 0.7592592592592593
Gradient Boosting Precision: 0.7777777777777778
Gradient Boosting Recall: 0.8484848484848485
Gradient Boosting F1 Score: 0.8115942028985507
```