WDBC dataset classification

March 25, 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('wdbc.csv')
```

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

[5]:

	Radius1	Texture1	Perimeter1	Area1	Smoothness1	Compactness1	\
0	17.99	10.38	122.8	1001.0	0.118	0.278	
1	20.57	17.77	132.9	1326.0	0.085	0.079	

	Concavity1	Concave_points1	Symmetry1	Fractal_dimension1	…	Texture3	\
0	0.300	0.147	0.242	0.079	…	17.33	
1	0.087	0.070	0.181	0.057	…	23.41	

	Perimeter3	Area3	Smoothness3	Compactness3	Concavity3	Concave_points3	\
0	184.6	2019.0	0.162	0.666	0.712	0.265	
1	158.8	1956.0	0.124	0.187	0.242	0.186	

	Symmetry3	Fractal_dimension3	Class
0	0.460	0.119	M
1	0.275	0.089	M

[2 rows x 31 columns]

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
```

```
Data columns (total 31 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Radius1              569 non-null    float64
 1   Texture1             569 non-null    float64
 2   Perimeter1           569 non-null    float64
 3   Area1                569 non-null    float64
 4   Smoothness1          569 non-null    float64
 5   Compactness1         569 non-null    float64
 6   Concavity1           569 non-null    float64
 7   Concave_points1      569 non-null    float64
 8   Symmetry1            569 non-null    float64
 9   Fractal_dimension1   569 non-null    float64
 10  Radius2              569 non-null    float64
 11  Texture2             569 non-null    float64
 12  Perimeter2           569 non-null    float64
 13  Area2                569 non-null    float64
 14  Smoothness2          569 non-null    float64
 15  Compactness2         569 non-null    float64
 16  Concavity2           569 non-null    float64
 17  Concave_points2      569 non-null    float64
 18  Symmetry2            569 non-null    float64
 19  Fractal_dimension2   569 non-null    float64
 20  Radius3              569 non-null    float64
 21  Texture3             569 non-null    float64
 22  Perimeter3           569 non-null    float64
 23  Area3                569 non-null    float64
 24  Smoothness3          569 non-null    float64
 25  Compactness3         569 non-null    float64
 26  Concavity3           569 non-null    float64
 27  Concave_points3      569 non-null    float64
 28  Symmetry3            569 non-null    float64
 29  Fractal_dimension3   569 non-null    float64
 30  Class                569 non-null    object
dtypes: float64(30), object(1)
memory usage: 137.9+ KB
```

[7]:
```
# 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

[8]:
```
# Train and evaluate a Logistic Regression model
lr_model = LogisticRegression(random_state=42)
```

[9]:
```
lr_model.fit(X_train, y_train)
lr_preds = lr_model.predict(X_test)
```

[10]:
```
lr_acc = accuracy_score(y_test, lr_preds)
lr_prec = precision_score(y_test, lr_preds, pos_label="M")
lr_rec = recall_score(y_test, lr_preds, pos_label="M")
lr_f1 = f1_score(y_test, lr_preds, pos_label="M")
```

[11]:
```
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: 0.9649122807017544
Logistic Regression Precision: 0.975609756097561
Logistic Regression Recall: 0.9302325581395349
Logistic Regression F1 Score: 0.9523809523809524
```

[12]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import numpy as np
import itertools
```

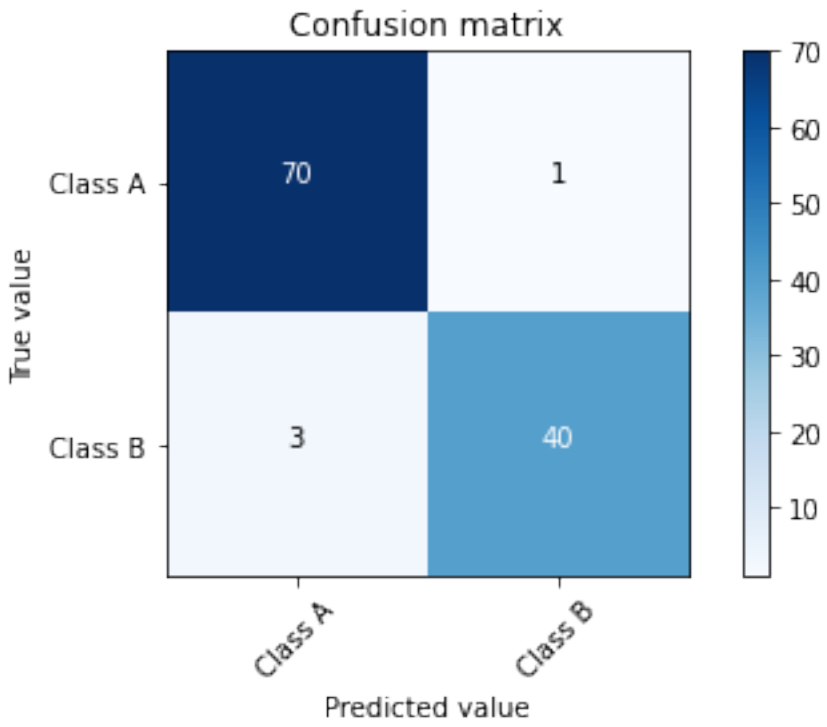
[13]:
```
# Define class labels
classes = ['Class A', 'Class B']
```

[14]:
```
# Compute confusion matrix
cm = confusion_matrix(y_test, lr_preds)
```

[15]:
```
# 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

[16]:
```
# Train and evaluate a Decision Tree model
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
```

[16]:
```
DecisionTreeClassifier(random_state=42)
```

[17]:
```
dt_preds = dt_model.predict(X_test)
dt_acc = accuracy_score(y_test, dt_preds)
dt_prec = precision_score(y_test, dt_preds, pos_label="M")
dt_rec = recall_score(y_test, dt_preds, pos_label="M")
dt_f1 = f1_score(y_test, dt_preds, pos_label="M")
```

[18]:
```
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: 0.9210526315789473
Decision Tree Precision: 0.8863636363636364
Decision Tree Recall: 0.9069767441860465
```

```
Decision Tree F1 Score: 0.896551724137931
```

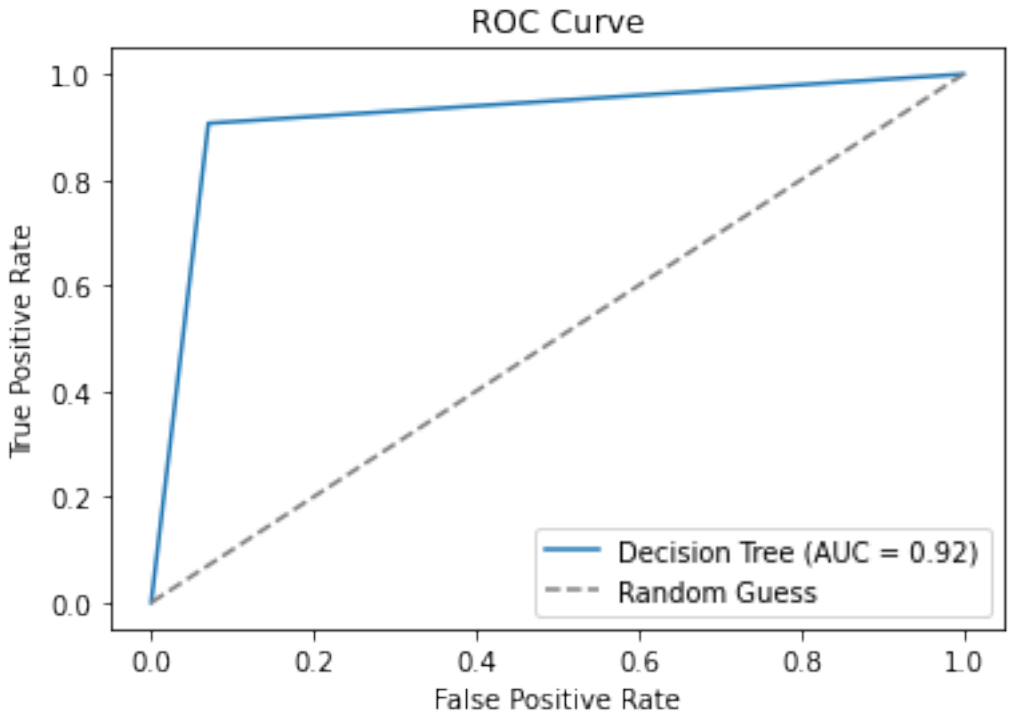
[19]:
```python
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
```

[20]:
```python
# Calculate predicted probabilities for positive class
dt_probs = dt_model.predict_proba(X_test)[:, 1]
```

[21]:
```python
# Calculate FPR, TPR, and thresholds
fpr, tpr, thresholds = roc_curve(y_test, dt_probs, pos_label="M")
```

[22]:
```python
# Calculate AUC
auc_dt = auc(fpr, tpr)
```

[23]:
```python
# Plot ROC curve
plt.plot(fpr, tpr, label='Decision Tree (AUC = {:.2f})'.format(auc_dt))
plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random Guess')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```



0.0.3 Gradient Boosting

[24]:
```
gb_model = GradientBoostingClassifier()
gb_model.fit(X_train, y_train)
```

[24]:
```
GradientBoostingClassifier()
```

[25]:
```
gb_preds = gb_model.predict(X_test)
```

[26]:
```
gb_acc = accuracy_score(y_test, gb_preds)
gb_prec = precision_score(y_test, gb_preds, pos_label="M")
gb_rec = recall_score(y_test, gb_preds, pos_label="M")
gb_f1 = f1_score(y_test, gb_preds, pos_label="M")
```

[27]:
```
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: 0.956140350877193
Gradient Boosting Precision: 0.9523809523809523
Gradient Boosting Recall: 0.9302325581395349
Gradient Boosting F1 Score: 0.9411764705882352
```