diab

April 12, 2024

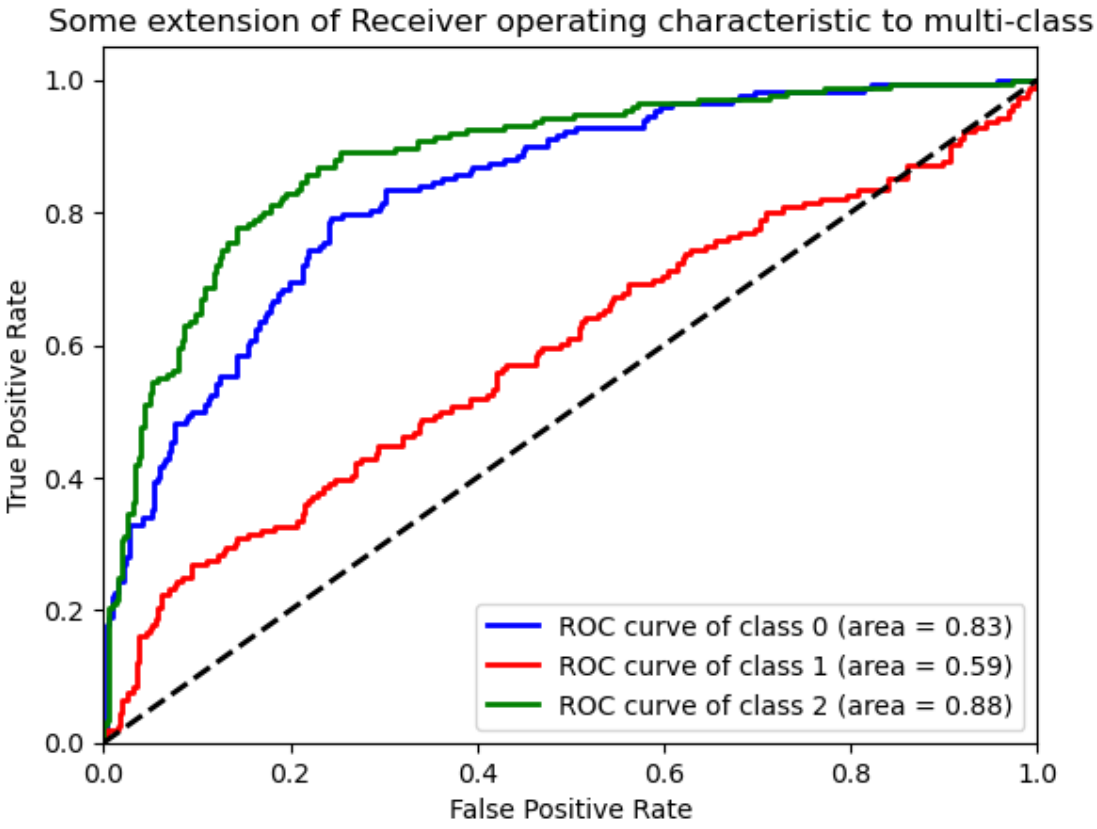
[5]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize
from sklearn.datasets import make_classification
from itertools import cycle

#
diabetes = load_diabetes()
X = diabetes.data
y = diabetes.target

#
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

#
model_lr = LinearRegression()
model_lr.fit(X_train, y_train)
y_pred_lr = model_lr.predict(X_test)

#
model_dt = DecisionTreeRegressor(max_depth=4)
model_dt.fit(X_train, y_train)
y_pred_dt = model_dt.predict(X_test)

#
print("          :")
print(f"RMSE: {mean_squared_error(y_test, y_pred_lr, squared=False):.2f}")
print(f"R^2: {r2_score(y_test, y_pred_lr):.2f}\n")
```

```
print("         :")
print(f"RMSE: {mean_squared_error(y_test, y_pred_dt, squared=False):.2f}")
print(f"R^2: {r2_score(y_test, y_pred_dt):.2f}")

#          ROC-     (                      )
X_class, y_class = make_classification(n_samples=1000, n_features=20,␣
 ↪n_informative=3, n_redundant=0, n_classes=3, random_state=42)
y_class_bin = label_binarize(y_class, classes=[0, 1, 2])
n_classes = y_class_bin.shape[1]

#
X_train_class, X_test_class, y_train_class, y_test_class =␣
 ↪train_test_split(X_class, y_class_bin, test_size=0.5, random_state=0)

#
classifier = LogisticRegression(solver='lbfgs', max_iter=10000,␣
 ↪multi_class='ovr')
classifier.fit(X_train_class, y_train_class.argmax(axis=1))
y_score_class = classifier.predict_proba(X_test_class)

#        ROC-      AUC
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_class[:, i], y_score_class[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

#
plt.figure()
colors = cycle(['blue', 'red', 'green'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=2,
             label='ROC curve of class {0} (area = {1:0.2f})'.format(i,␣
 ↪roc_auc[i]))
plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Some extension of Receiver operating characteristic to multi-class')
plt.legend(loc="lower right")
plt.show()
```

```
          :
RMSE: 53.85
R^2: 0.45
```

:

```
RMSE: 61.78
R^2: 0.28
```



[ ]: