ЛАБОРАТОРНА РОБОТА № 5

ДОСЛІДЖЕННЯ МЕТОДІВ АНСАМБЛЕВОГО НАВЧАННЯ

Мета роботи: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити методи ансамблів у машинному навчанні.

Посилання на гіт: https://github.com/IPZ213mmv/Lab5

Завдання 2.1. Створення класифікаторів на основі випадкових та гранично випадкових лісів

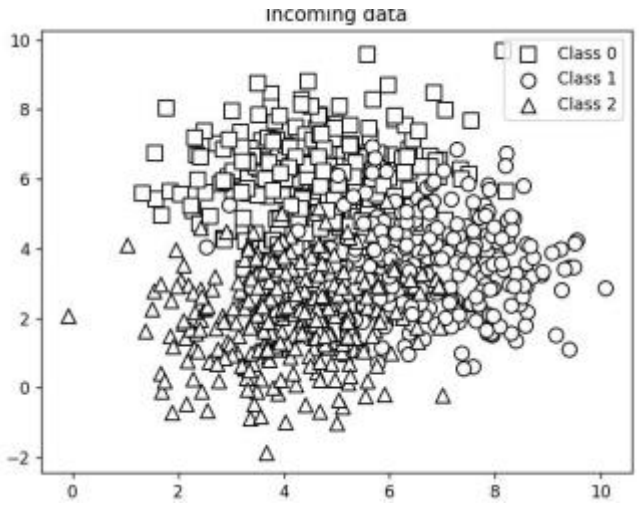
Лістинг програми:

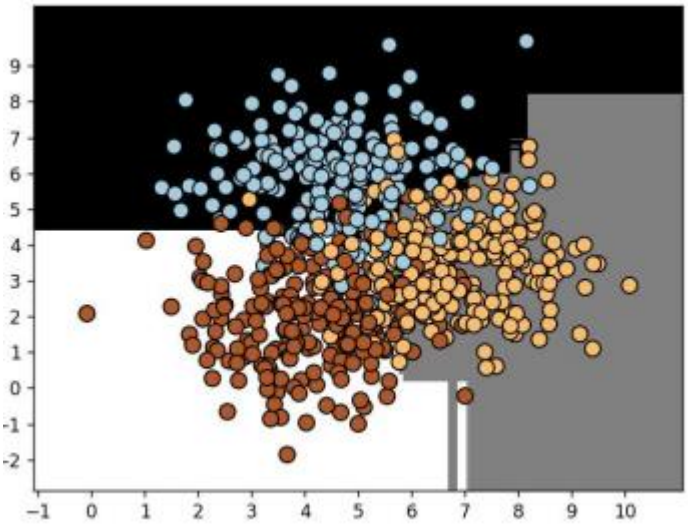
```
import argparse
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from utilities import visualize_classifier


def build_arg_parser():
    parser = argparse.ArgumentParser(description="Classify data using Ensemble
Learning techniques")
    parser.add_argument('--classifier-type', dest='classifier_type',required=True,
choices=['rf', 'erf'], help="Type of classifier to use; can be either 'rf' or
'erf'")
    return parser

args = build_arg_parser().parse_args()
classifier_type = args.classifier_type

input_file = 'data_random_forests.txt'
data = np.loadtxt(input_file, delimiter=',')
x, y = data[:, :-1], data[:, -1]
class_0 = np.array(x[y == 0])
class_1 = np.array(x[y == 1])
class_2 = np.array(x[y == 2])

plt.figure()

plt.scatter(class_0[:, 0], class_0[:, 1], s=75,
facecolors='white',edgecolors="black", linewidth=1, marker='s', label='Class 0')

plt.scatter(class_1[:, 0], class_1[:, 1], s=75,
facecolors='white',edgecolors="black", linewidth=1, marker='o', label='Class 1')

plt.scatter(class_2[:, 0], class_2[:, 1], s=75,
facecolors='white',edgecolors='black', linewidth=1, marker='^', label='Class 2')

plt.title('Incoming data')
plt.legend()

X_train, X_test, y_train, y_test = train_test_split(x, y,
test_size=0.25,random_state=5)
```

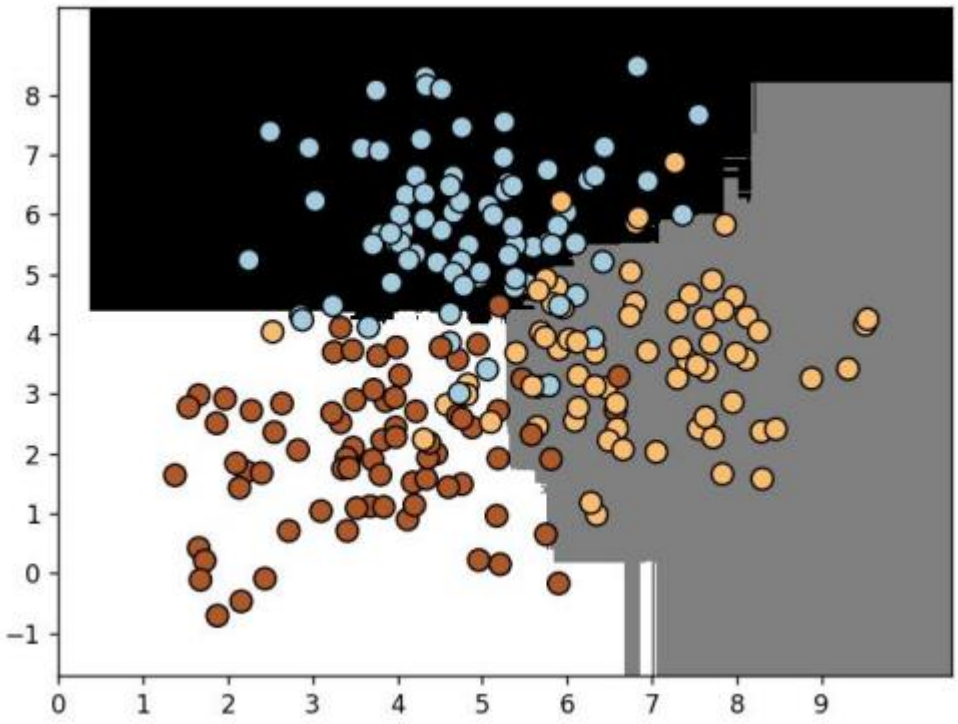
```
params = {'n_estimators': 100, 'max_depth': 4, 'random_state': 0}

if classifier_type == 'rf':
    classifier = RandomForestClassifier(**params)
else:
    classifier = ExtraTreesClassifier(**params)

classifier.fit(X_train, y_train)
visualize_classifier(classifier, X_train, y_train)

y_test_pred = classifier.predict(X_test)
visualize_classifier(classifier, X_test, y_test)

class_names = ['Class-0', 'Class-1', 'Class-2']
print("\nClassifier performance on training dataset\n")
print(classification_report(y_train, classifier.predict(X_train),
target_names=class_names))
print("#" * 40 + "\n")

print("\nClassifier performance on test dataset\n")
print(classification_report(y_test, y_test_pred, target_names=class_names))
print("#" * 40 + "\n")

plt.show()
```

Результат:

```
Classifier performance on training dataset

              precision    recall  f1-score   support

     Class-0       0.91      0.86      0.88       221
     Class-1       0.84      0.87      0.86       230
   macro avg       0.87      0.87      0.87       225
weighted avg       0.87      0.87      0.87       225

########################################
```







Висновок порівняння: Обидва типи класифікаторів (rf та erf) показують дуже схожі результати на цьому навчальному наборі даних. Обидва мають високу точність та повноту для всіх класів, і середні значення також однакові. Основна різниця між ними полягає в тому, що erf має меншу схильність до перенавчання через більшу випадковість у виборі розділувачів на кожному вузлі дерева рішень. Таким чином, erf може бути вигідним варіантом, коли потрібно зменшити перенавчання на навчальному наборі даних.

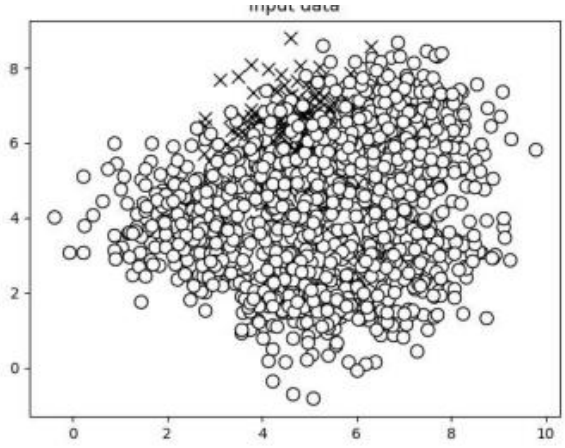
Завдання 2.2. Обробка дисбалансу класів

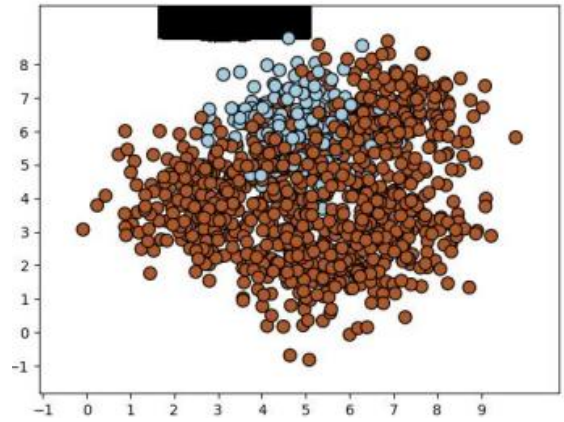
Лістинг програми:

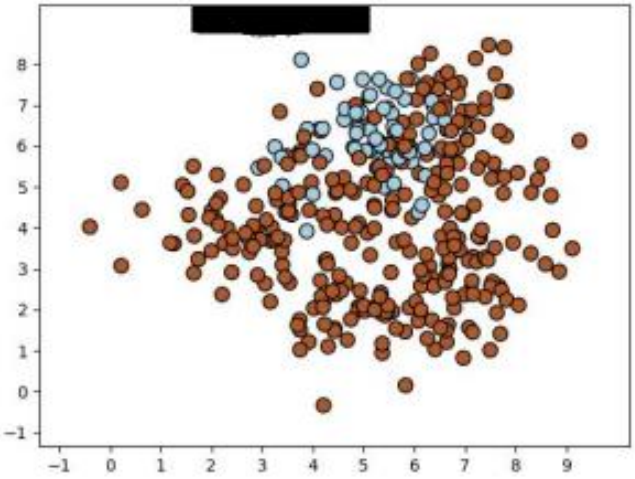
```
import sys
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import ExtraTreesClassifier
from sklearn import model_selection
from sklearn.metrics import classification_report
from utilities import visualize_classifier

input_file = 'data_imbalance.txt'
data = np.loadtxt(input_file, delimiter=',')
x, y = data[:, :-1], data[:, -1]

class_0 = np.array(x[y == 0])
class_1 = np.array(x[y == 1])

plt.figure()
plt.scatter(class_0[:, 0], class_0[:, 1], s=75,facecolors='black',
edgecolors='black', linewidth=1, marker='x')
plt.scatter(class_1[:, 0], class_1[:, 1], s=75,facecolors='white',
edgecolors='black', linewidth=1, marker='o')
plt.title('Input data')

X_train, X_test, y_train, y_test = model_selection.train_test_split(x, y,
test_size=0.25, random_state=5)
params = {'n_estimators': 100, 'max_depth': 4, 'random_state': 0}

if len(sys.argv) > 1 and sys.argv[1] == "balance":
    params['class_weight'] = 'balanced'
else:
    print("Use the 'balance' flag to account for class imbalance")

classifier = ExtraTreesClassifier(**params)

classifier.fit(X_train, y_train)

visualize_classifier(classifier, X_train, y_train)

y_test_pred = classifier.predict(X_test)
visualize_classifier(classifier, X_test, y_test)

class_names = ['Class-0', 'Class-1']
print("\n" + "#" * 40)
print("\nClassifier performance on training dataset\n")
print(classification_report(y_train, classifier.predict(X_train),
target_names=class_names))
print("#" * 40 + "\n")
print("#" * 40)
print("\nClassifier performance on test dataset\n")
print(classification_report(y_test, y_test_pred, target_names=class_names))
print("#" * 40 + "\n")

plt.show()
```

Результат:

	precision	recall	f1-score	support
Class-0	0.00	0.00	0.00	69
Class-1	0.82	1.00	0.90	306
accuracy			0.82	375
macro avg	0.41	0.50	0.45	375
weighted avg	0.67	0.82	0.73	375







Завдання 2.3. Знаходження оптимальних навчальних параметрів за допомогою сіткового пошуку

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import ExtraTreesClassifier
from utilities import visualize_classifier

input_file = 'data_random_forests.txt'
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]

class_0 = np.array(X[y == 0])
class_1 = np.array(X[y == 1])
class_2 = np.array(X[y == 2])

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25,random_state=5)

parameter_grid = {
    'n_estimators': [100],
    'max_depth': [2, 4, 7, 12, 16]
}

metrics = ['precision_weighted', 'recall_weighted']

for metric in metrics:
    print("\n##### Searching optimal parameters for", metric)
    classifier = GridSearchCV(
        ExtraTreesClassifier(random_state=0),
        parameter_grid,
        cv=5,
        scoring=metric
    )
    classifier.fit(X_train, y_train)
    print("\nGrid scores for the parameter grid:")
    for params, avg_score in zip(classifier.cv_results_['params'],
classifier.cv_results_['mean_test_score']):
        print(params, '-->', round(avg_score, 3))
    print("\nBest parameters:", classifier.best_params_)

    y_pred = classifier.predict(X_test)
    print("\nPerformance report for", metric, ":\n")
    print(classification_report(y_test, y_pred))
```

Результат:

```
##### Searching optimal parameters for precision_weighted

Grid scores for the parameter grid:
{'max_depth': 2, 'n_estimators': 100} --> 0.85
{'max_depth': 4, 'n_estimators': 100} --> 0.841
{'max_depth': 7, 'n_estimators': 100} --> 0.844
{'max_depth': 12, 'n_estimators': 100} --> 0.832
{'max_depth': 16, 'n_estimators': 100} --> 0.816

Best parameters: {'max_depth': 2, 'n_estimators': 100}

Performance report for precision_weighted :

              precision    recall  f1-score   support

         0.0       0.94      0.81      0.87        79
         1.0       0.81      0.86      0.83        70
         2.0       0.83      0.91      0.87        76

    accuracy                           0.86       225
   macro avg       0.86      0.86      0.86       225
weighted avg       0.86      0.86      0.86       225
```

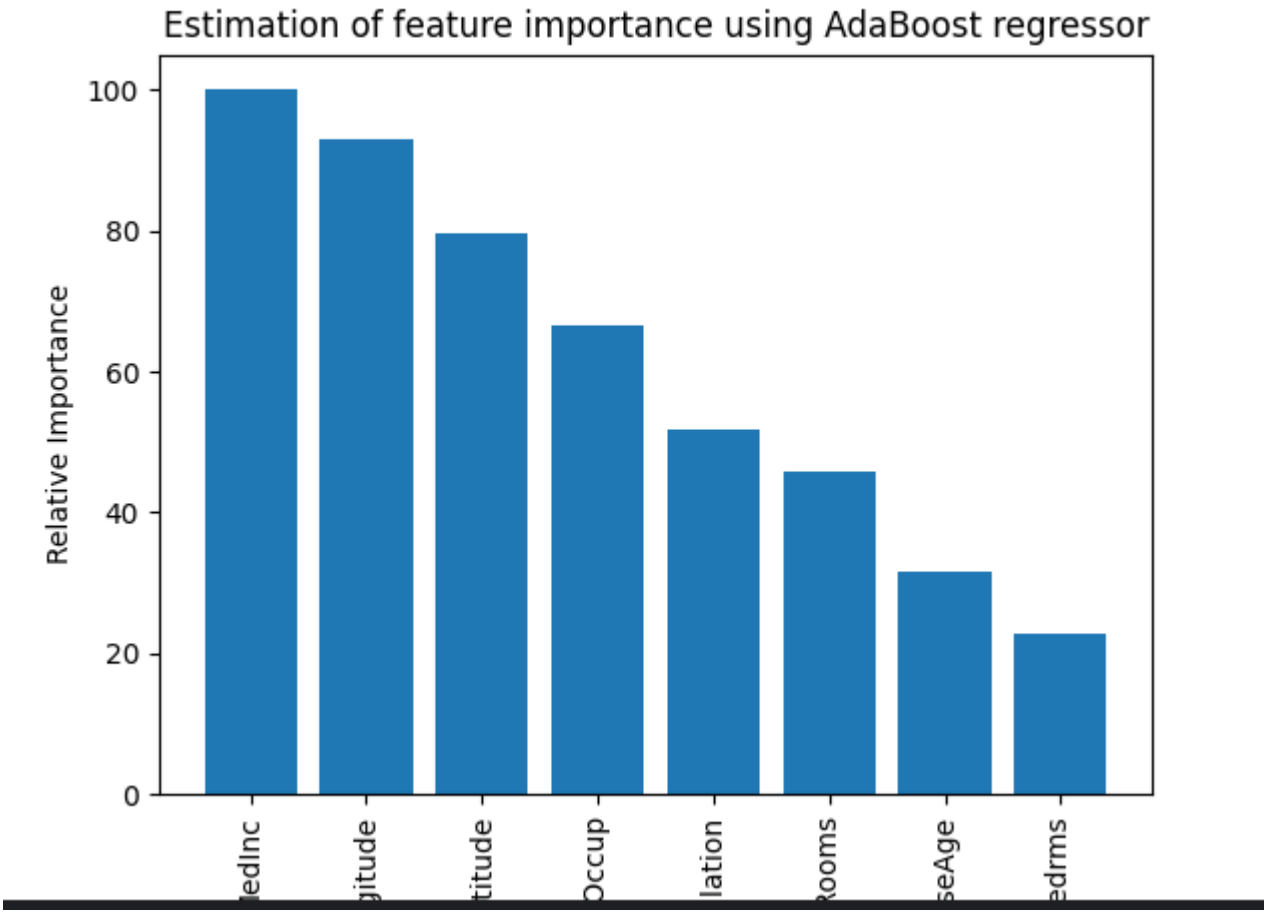
```
##### Searching optimal parameters for recall_weighted

Grid scores for the parameter grid:
{'max_depth': 2, 'n_estimators': 100} --> 0.843
{'max_depth': 4, 'n_estimators': 100} --> 0.837
{'max_depth': 7, 'n_estimators': 100} --> 0.841
{'max_depth': 12, 'n_estimators': 100} --> 0.83
{'max_depth': 16, 'n_estimators': 100} --> 0.815

Best parameters: {'max_depth': 2, 'n_estimators': 100}

Performance report for recall_weighted :

              precision    recall  f1-score   support

         0.0       0.94      0.81      0.87        79
         1.0       0.81      0.86      0.83        70
         2.0       0.83      0.91      0.87        76

    accuracy                           0.86       225
   macro avg       0.86      0.86      0.86       225
weighted avg       0.86      0.86      0.86       225
```

Завдання 2.4. Обчислення відносної важливості ознак

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.datasets import fetch_california_housing
from sklearn.metrics import mean_squared_error, explained_variance_score
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle

housing_data = fetch_california_housing()

X, y = shuffle(housing_data.data, housing_data.target, random_state=7)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2,random_state=7)

regressor = AdaBoostRegressor(DecisionTreeRegressor(max_depth=4),n_estimators=400,
random_state=7)
regressor.fit(X_train, y_train)

y_pred = regressor.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
evs = explained_variance_score(y_test, y_pred)

print("\nADABOOST REGRESSOR")
print("Mean squared error =", round(mse, 2))
print("Explained variance score =", round(evs, 2))

feature_importances = regressor.feature_importances_
feature_names = housing_data.feature_names

feature_importances = 100.0 * (feature_importances /
max(feature_importances))

index_sorted = np.flipud(np.argsort(feature_importances))

pos = np.arange(len(index_sorted)) + 0.5

plt.figure()
plt.bar(pos, feature_importances[index_sorted], align="center")
plt.xticks(pos, [feature_names[i] for i in index_sorted], rotation=90)
plt.ylabel('Relative Importance')
plt.title("Estimation of feature importance using AdaBoost regressor")
plt.show()
```

Результат:




```
ADABOOST REGRESSOR
Mean squared error = 1.18
Explained variance score = 0.47
```

Висновок: використовуючи спеціалізовані бібліотеки та мову програмування Python я дослідив методи ансамблів у машинному навчанні.