**ЛАБОРАТОРНА РОБОТА № 5**

**Дослідження методів ансамблевого навчання**

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

**Хід роботи:**

**Завдання 1:**

Лістинг коду:

import argparse  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier  
from sklearn.metrics import classification\_report  
from utilities import visualize\_classifier  
  
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'",  
)  
args = 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",  
)  
plt.scatter(  
 class\_1[:, 0],  
 class\_1[:, 1],  
 s=75,  
 facecolors="white",  
 edgecolors="black",  
 linewidth=1,  
 marker="o",  
)  
plt.scatter(  
 class\_2[:, 0],  
 class\_2[:, 1],  
 s=75,  
 facecolors="white",  
 edgecolors="black",  
 linewidth=1,  
 marker="^",  
)  
plt.title("Input data")  
  
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, "Training dataset")  
y\_test\_pred = classifier.predict(X\_test)  
visualize\_classifier(classifier, X\_test, y\_test, "Test dataset")  
class\_names = ["Class-0", "Class-1", "Class-2"]  
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")  
test\_datapoints = np.array([[5, 5], [3, 6], [6, 4], [7, 2], [4, 4], [5, 2]])  
print("\nConfidence measure:")  
for datapoint in test\_datapoints:  
 probabilities = classifier.predict\_proba([datapoint])[0]  
 predicted\_class = "Class-" + str(np.argmax(probabilities))  
 print("\nDatapoint:", datapoint)  
 print("Predicted class:", predicted\_class)  
visualize\_classifier(  
 classifier, test\_datapoints, [0] \* len(test\_datapoints), "Test datapoints"  
)  
plt.show()

Результати виконання програми з прапорцем rf:

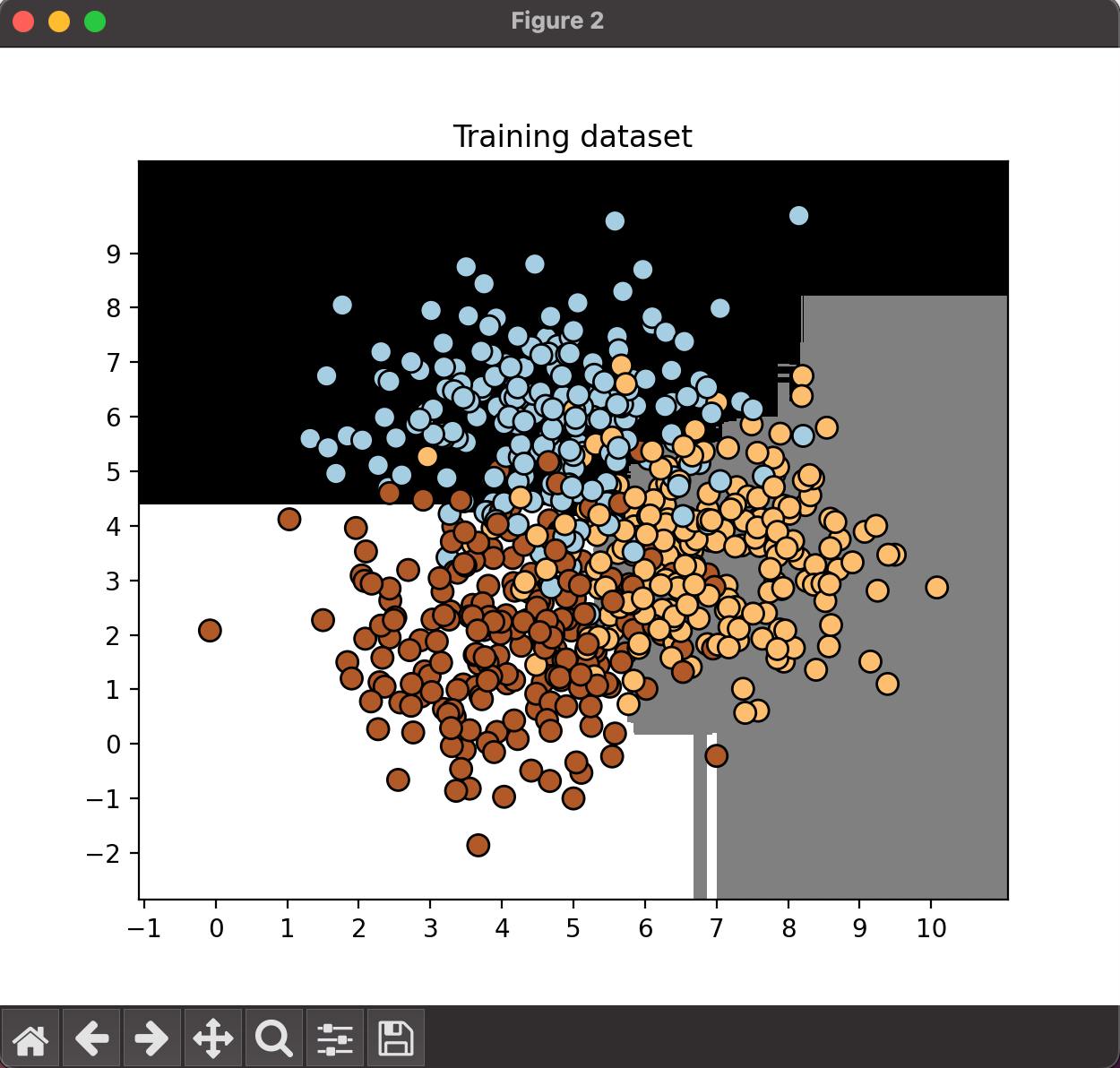


Рис. 1. Результат виконання програми

A screenshot of a computer screen

Description automatically generated

Рис. 2. Результат виконання програми

A screenshot of a computer screen

Description automatically generated

Рис. 3. Результат виконання програми

A screenshot of a graph

Description automatically generated

Рис. 4. Результат виконання програми

A screen shot of a computer

Description automatically generated

Рис. 5. Результат виконання програми

Результати виконання програми з прапорцем erf:

A screenshot of a computer screen

Description automatically generated

Рис. 6. Результат виконання програми

A screen shot of a computer screen

Description automatically generated

Рис. 7. Результат виконання програми

A screenshot of a computer screen

Description automatically generated

Рис. 8. Результат виконання програми

A screenshot of a computer

Description automatically generated

Рис. 9. Результат виконання програми

A screen shot of a computer

Description automatically generated

Рис. 10. Результат виконання програми

Моделі демонструють загалом схожі високі результати, але rf трохи перевершує інші на навчальному наборі.

**Завдання 2:**

Лістинг коду:

import sys  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.ensemble import ExtraTreesClassifier  
from sklearn.model\_selection import train\_test\_split  
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\_0[:, 0],  
 class\_0[:, 1],  
 s=75,  
 facecolors="white",  
 edgecolors="black",  
 linewidth=1,  
 marker="o",  
)  
plt.title("Input Data")  
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 len(sys.argv) > 1:  
 if sys.argv[1] == "balance":  
 params = {  
 "n\_estimators": 100,  
 "max\_depth": 4,  
 "random\_state": 0,  
 "class\_weight": "balanced",  
 }  
 else:  
 raise TypeError("Invalid input argument; should be 'balance'")  
classifier = ExtraTreesClassifier(\*\*params)  
classifier.fit(X\_train, y\_train)  
visualize\_classifier(classifier, X\_train, y\_train, "Training dataset")  
y\_test\_pred = classifier.predict(X\_test)  
visualize\_classifier(classifier, X\_test, y\_test, "Test dataset")  
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()

Результати виконання програми без врахування дисбалансу:

A screenshot of a computer screen

Description automatically generated

Рис. 11. Результат виконання програми

A screen shot of a computer screen

Description automatically generated

Рис. 12. Результат виконання програми

A screenshot of a computer screen

Description automatically generated

Рис. 13. Результат виконання програми

A screenshot of a computer screen

Description automatically generated

Рис. 14. Результат виконання програми

Результати виконання програми з врахуванням дисбалансу:

A screenshot of a computer screen

Description automatically generated

Рис. 15. Результат виконання програми

A screen shot of a computer screen

Description automatically generated

Рис. 16. Результат виконання програми

A screenshot of a computer screen

Description automatically generated

Рис. 17. Результат виконання програми

A screenshot of a computer

Description automatically generated

Рис. 18. Результат виконання програми

Врахування дисбалансу класів підвищило точність для менш численного класу (Class-0), що свідчить про збалансованість моделі.

**Завдання 3:**

Лістинг коду:

import numpy as np  
from sklearn.ensemble import ExtraTreesClassifier  
from sklearn.model\_selection import train\_test\_split, GridSearchCV  
from sklearn.metrics import classification\_report  
  
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  
)  
params\_grid = [  
 {"n\_estimators": [100], "max\_depth": [2, 4, 7, 12, 16]},  
 {"n\_estimators": [25, 50, 100, 250], "max\_depth": [4]},  
]  
metrics = ["precision\_weighted", "recall\_weighted"]  
for m in metrics:  
 print(f"\n##### Searching optimal parameters for {m}")  
 classifier = GridSearchCV(  
 ExtraTreesClassifier(random\_state=0), params\_grid, cv=5, scoring=m  
 )  
 classifier.fit(X\_train, y\_train)  
 print("\nGrid scores for the parameter grid:")  
 for i, params in enumerate(classifier.cv\_results\_["params"]):  
 avg\_score = classifier.cv\_results\_["mean\_test\_score"][i]  
 print(params, "-->", round(avg\_score, 3))  
 print("\nBest parameters:", classifier.best\_params\_)  
 y\_pred = classifier.predict(X\_test)  
 print("\nPerformance report:\n")  
 print(classification\_report(y\_test, y\_pred))

A screen shot of a computer

Description automatically generated

Рис. 19. Результат виконання програми

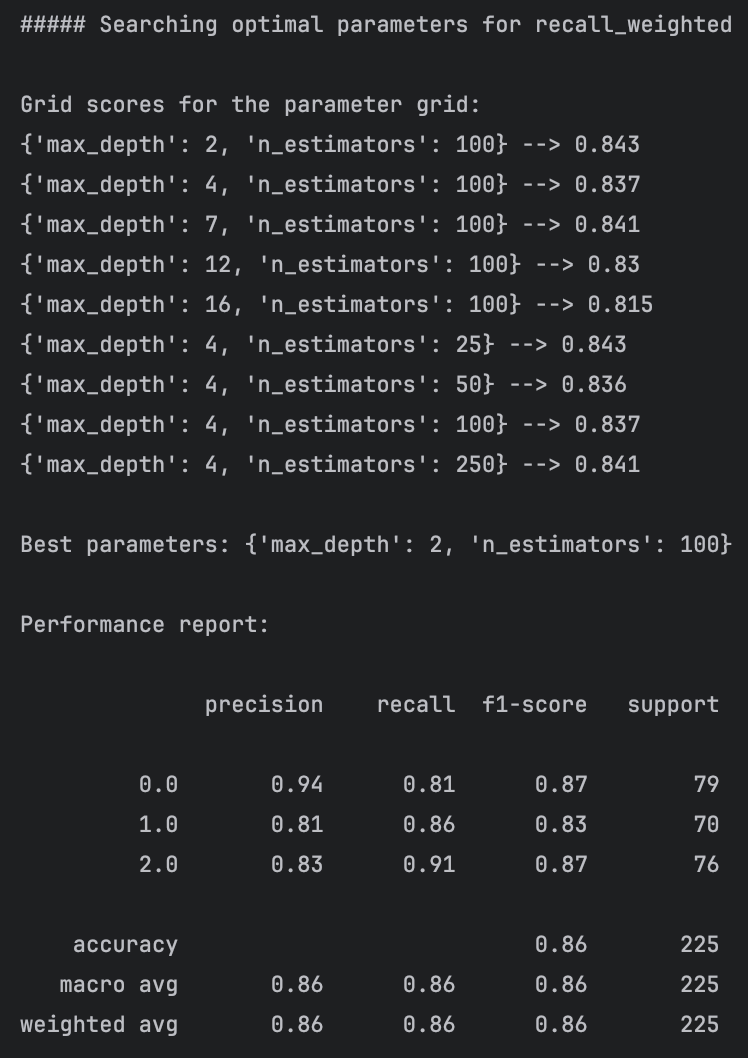


Рис. 20. Результат виконання програми

Найкращі результати моделі досягаються за параметрів max\_depth=2 та n\_estimators=100.

**Завдання 4:**

Лістинг коду:

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.tree import DecisionTreeRegressor  
from sklearn.ensemble import AdaBoostRegressor  
from sklearn import datasets  
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 = (  
 datasets.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 = np.array(housing\_data.feature\_names)  
feature\_importances = 100.0 \* (feature\_importances / max(feature\_importances))  
index\_sorted = np.flipud(np.argsort(feature\_importances))  
pos = np.arange(index\_sorted.shape[0]) + 0.5  
plt.figure()  
plt.bar(pos, feature\_importances[index\_sorted], align="center")  
plt.xticks(pos, feature\_names[index\_sorted])  
plt.ylabel("Relative Importance")  
plt.title("Adaboost")  
plt.show()

A graph of blue bars

Description automatically generated

Рис. 21. Результат виконання програми

A black background with white text

Description automatically generated

Рис. 22. Результат виконання програми

Ознаки MedInc, Longitude і Latitude мають вирішальне значення, тоді як HouseAge і AveBedrms є менш важливими.

**Завдання 5:**

Лістинг коду:

import numpy as np  
from sklearn.metrics import mean\_absolute\_error  
from sklearn import preprocessing  
from sklearn.ensemble import ExtraTreesRegressor  
from sklearn.model\_selection import train\_test\_split  
  
input\_file = "traffic\_data.txt"  
data = []  
with open(input\_file, "r") as f:  
 for line in f.readlines():  
 items = line[:-1].split(",")  
 data.append(items)  
  
data = np.array(data)  
label\_encoder = []  
X\_encoded = np.empty(data.shape)  
for i, item in enumerate(data[0]):  
 if item.isdigit():  
 X\_encoded[:, i] = data[:, i]  
 else:  
 label\_encoder.append(preprocessing.LabelEncoder())  
 X\_encoded[:, i] = label\_encoder[-1].fit\_transform(data[:, i])  
X = X\_encoded[:, :-1].astype(int)  
y = X\_encoded[:, -1].astype(int)  
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}  
regressor = ExtraTreesRegressor(\*\*params)  
regressor.fit(X\_train, y\_train)  
y\_pred = regressor.predict(X\_test)  
print("Mean absolute error:", round(mean\_absolute\_error(y\_test, y\_pred), 2))  
test\_datapoint = ["Saturday", "10:20", "Atlanta", "no"]  
test\_datapoint\_encoded = [-1] \* len(test\_datapoint)  
count = 0  
for i, item in enumerate(test\_datapoint):  
 if item.isdigit():  
 test\_datapoint\_encoded[i] = int(test\_datapoint[i])  
 else:  
 test\_datapoint\_encoded[i] = int(  
 label\_encoder[count].transform([test\_datapoint[i]])[0]  
 )  
 count += 1  
test\_datapoint\_encoded = np.array(test\_datapoint\_encoded)  
print("Predicted traffic:", int(regressor.predict([test\_datapoint\_encoded])[0]))

A black background with white text

Description automatically generated

Рис. 23. Результат виконання програми

Посилання на репозиторій на GitHub: <https://github.com/vladyslavgeyna/artificial-intelligence-systems/tree/main/lab5>.

***Висновки:*** в ході виконання лабораторної роботи ми,використовуючи спеціалізовані бібліотеки та мову програмування Python дослідили методи ансамблів у машинному навчанні.