

ЛАБОРАТОРНА РОБОТА № 5

ДОСЛІДЖЕННЯ МЕТОДІВ АНСАМБЛЕВОГО НАВЧАННЯ

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

Завдання 1

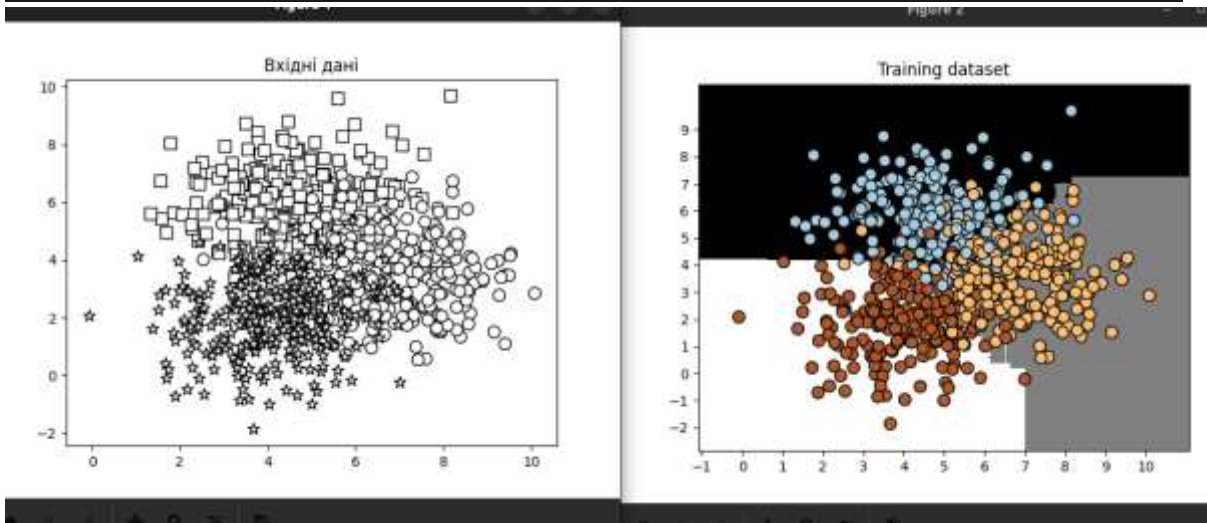
```
$ python3 random_forests.py --classifier-type rf
```

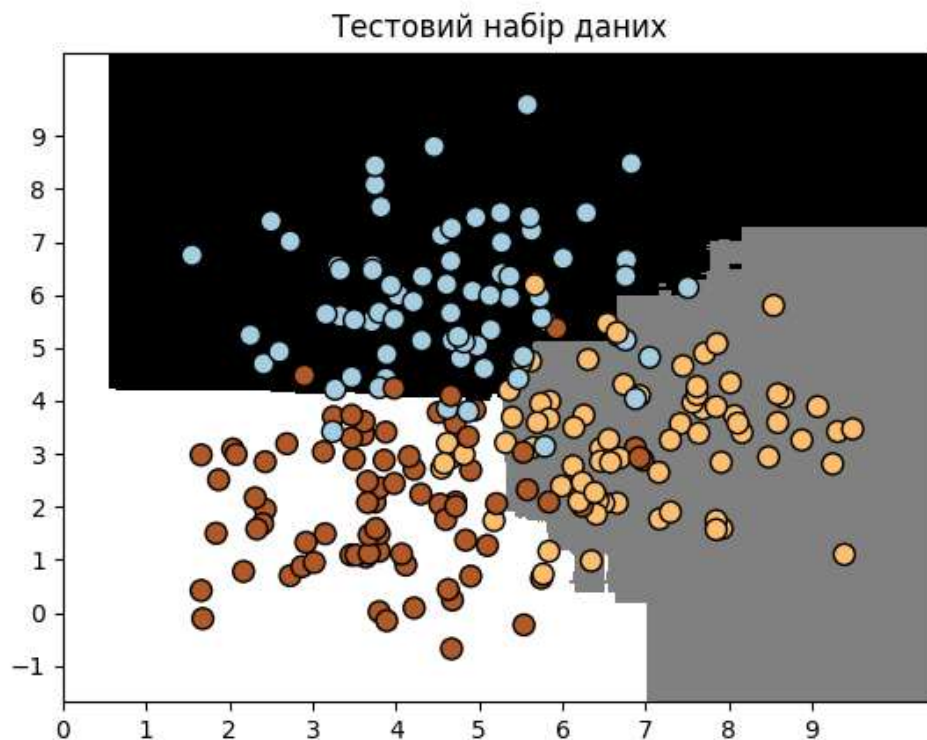
У процесі виконання цього коду отримайте **ряд зображень та занесіть їх у звіт**.

Графік вхідних даних. На графіку квадрати, кола та трикутники представляють три класи. Оцініть візуально, що класи значною мірою перекриваються, проте на цьому етапі це нормально. **Графік занесіть у звіт.**

Зображення на якому відображені границі класифікатора. **Графік занесіть у звіт.**

Classifier performance on training dataset					Classifier performance on test dataset				
	precision	recall	f1-score	support		precision	recall	f1-score	support
Class-0	0.90	0.88	0.89	227	Class-0	0.89	0.89	0.89	73
Class-1	0.84	0.87	0.86	226	Class-1	0.85	0.85	0.85	74
Class-2	0.88	0.87	0.88	222	Class-2	0.86	0.86	0.86	78
accuracy			0.87	675	accuracy			0.87	225
macro avg	0.87	0.87	0.87	675	macro avg	0.87	0.87	0.87	225
weighted avg	0.87	0.87	0.87	675	weighted avg	0.87	0.87	0.87	225





```
$ python3 random_forests.py --classifier-type erf
```

Отримайте зображення то порівняйте його з попереднім. **Графік занесіть у звіт.** Зверніть увагу, що в останньому випадку були отримані більш лагідні піки. Це обумовлено тим, що в процесі навчання гранично випадкові ліси мають більше можливостей для вибору оптимальних дерев рішень, тому, як правило, вони забезпечують отримання кращих границь.

Classifier performance on training dataset					Classifier performance on test dataset				
	precision	recall	f1-score	support		precision	recall	f1-score	support
Class-0	0.88	0.84	0.86	227	Class-0	0.92	0.84	0.88	73
Class-1	0.82	0.83	0.83	226	Class-1	0.83	0.85	0.84	74
Class-2	0.84	0.88	0.86	222	Class-2	0.84	0.90	0.87	78
accuracy			0.85	675	accuracy			0.86	225
macro avg	0.85	0.85	0.85	675	macro avg	0.87	0.86	0.86	225
weighted avg	0.85	0.85	0.85	675	weighted avg	0.86	0.86	0.86	225

Training dataset

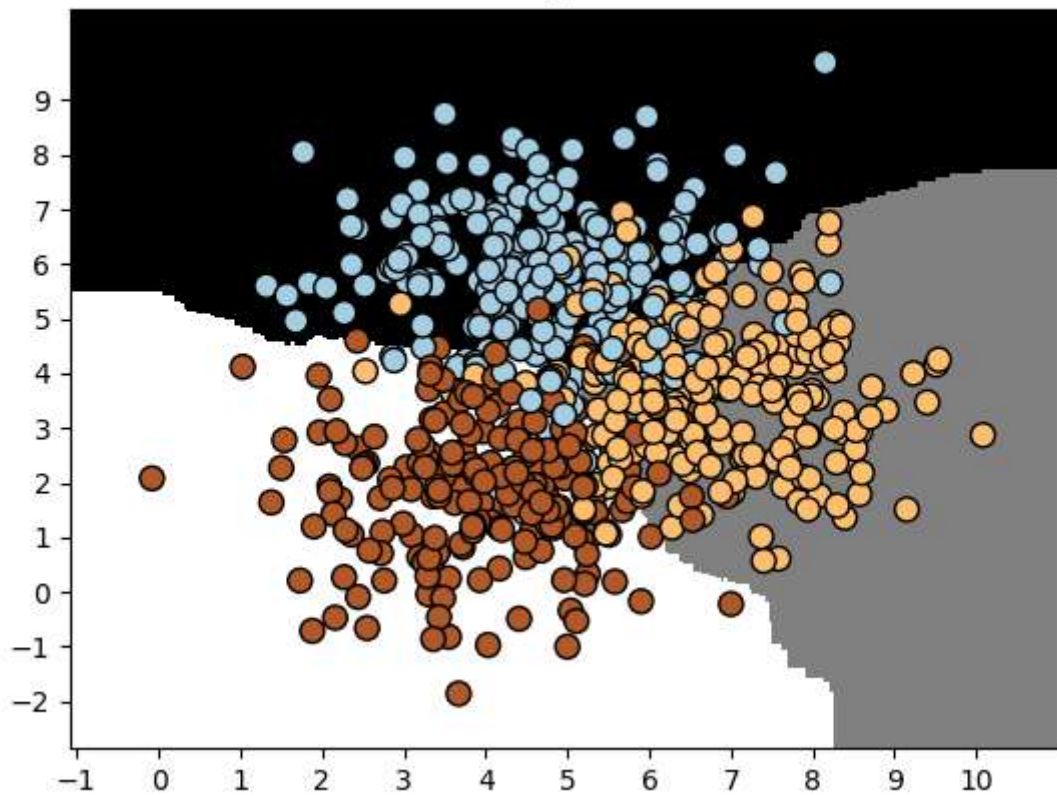
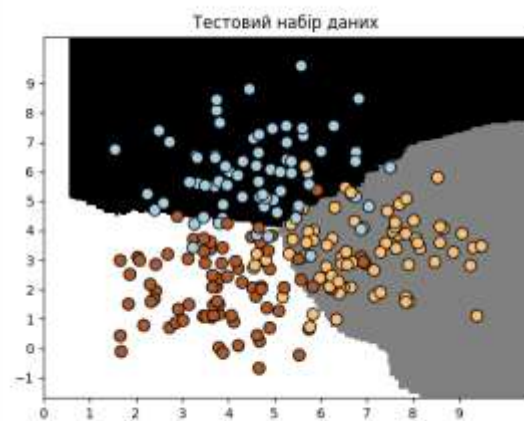
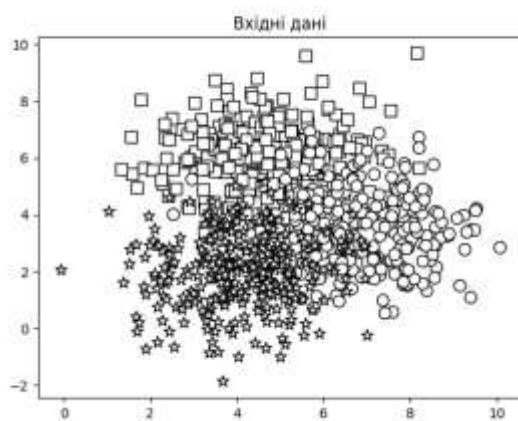


Figure 1

Figure 2



Результат виконання цього коду із прапором **rf** занесіть у звіт

У вікні терміналу з'явиться виведена інформація **Скріншот цієї інформації виріжте та занесіть у звіт.**

Для кожної точки даних обчислюється можливість її належності кожному з трьох класів. Ми вибираємо той клас, якому відповідає найвищий рівень довіри.

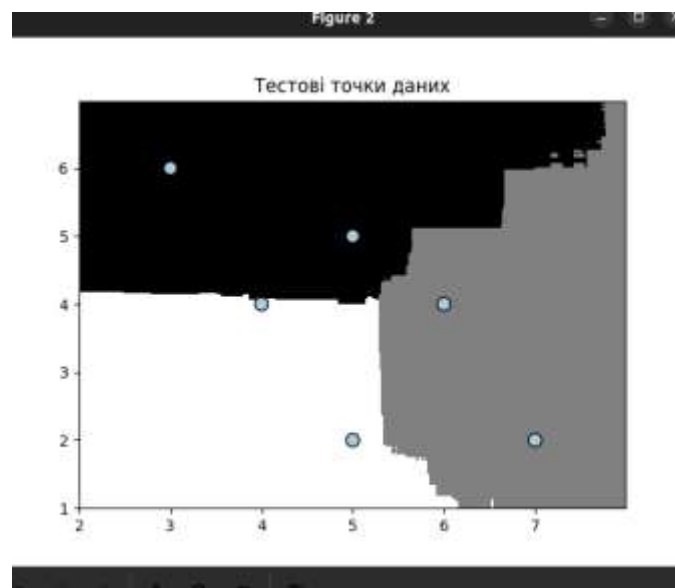
Результат виконання коду із прапором **erf** занесіть у звіт.

У вікні терміналу з'явиться виведена інформація **Скріншот цієї інформації виріжте та занесіть у звіт.**

Збережіть код робочої програми під назвою LR_5_task_1.py

Код програми, графік функції та результати оцінки якості занесіть у звіт.

RF:



```
Confidence measure:
Datapoint: [5 5]
Predicted class: Class-0

Datapoint: [3 6]
Predicted class: Class-0

Datapoint: [6 4]
Predicted class: Class-1

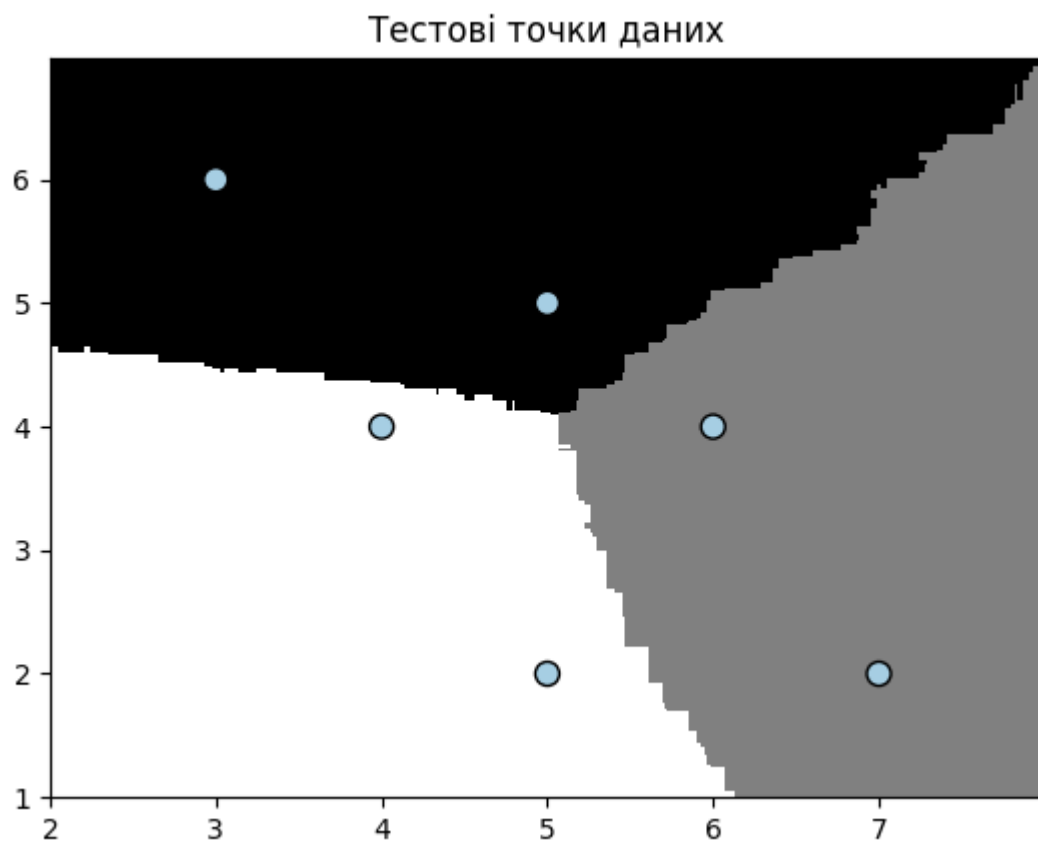
Datapoint: [7 2]
Predicted class: Class-1

Datapoint: [4 4]
Predicted class: Class-2

Datapoint: [5 2]
Predicted class: Class-2
root@dmytro-X550CL:/home/alhim/Завантаження/ШІ/lab5#
```

ERF:

```
Confidence measure:  
  
Datapoint: [5 5]  
Predicted class: Class-0  
  
Datapoint: [3 6]  
Predicted class: Class-0  
  
Datapoint: [6 4]  
Predicted class: Class-1  
  
Datapoint: [7 2]  
Predicted class: Class-1  
  
Datapoint: [4 4]  
Predicted class: Class-2  
  
Datapoint: [5 2]  
Predicted class: Class-2
```



ЛІСТИНГ:

```
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 # Updated import
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 classifier to use; can be either 'rf'
or 'erf'")
    return parser

if __name__ == '__main__':
    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')

    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('Вхідні дані')

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=2)

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, 'Тестовий набір
даних')

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]

```

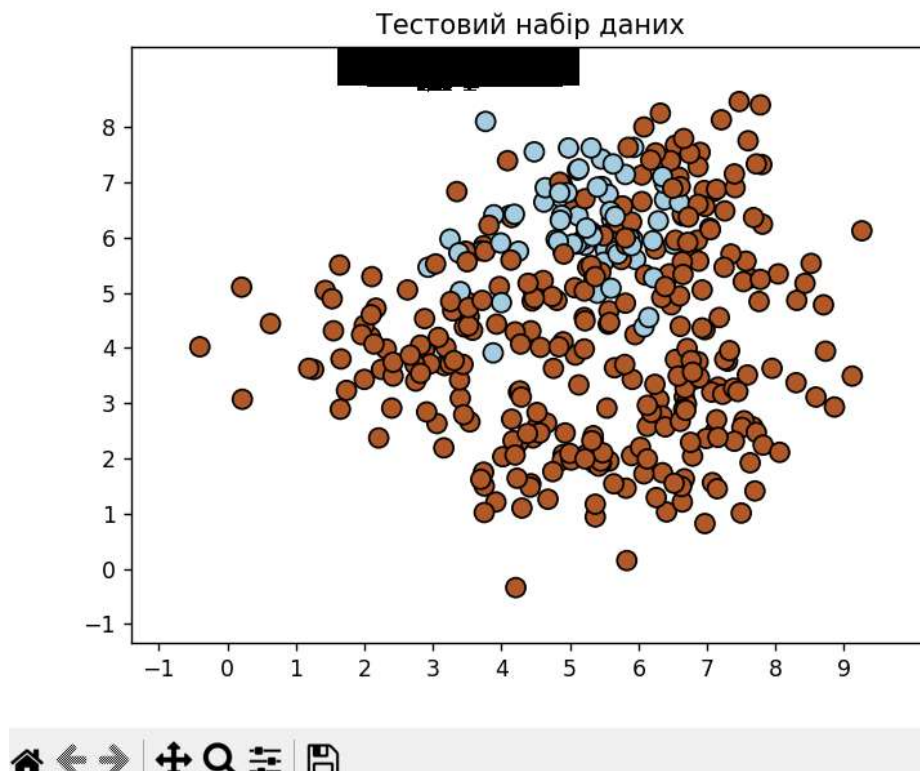
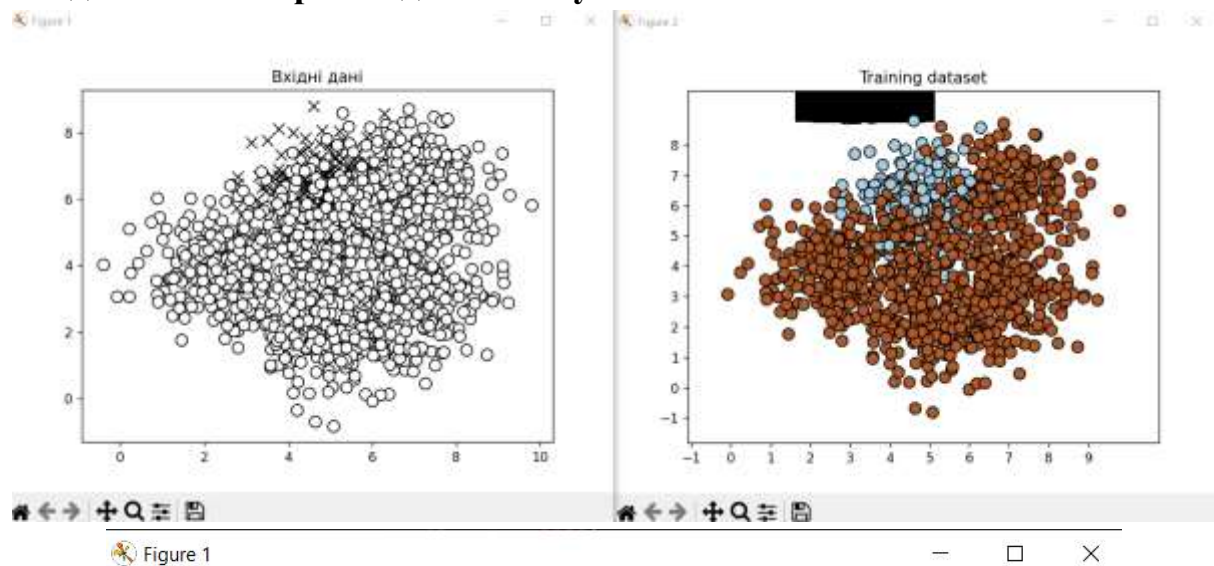
```

predicted_class = 'Class-' + str (np.argmax (probabilities) )
print ('\nDatapoint:', datapoint)
print ('Predicted class:', predicted_class)

visualize_classifier(classifier, test_datapoints,
[0]*len(test_datapoints), 'Тестові точки даних')
plt.show()

```

Завдання 2.2. Обробка дисбалансу класів




```
#####
```

Classifier performance on training dataset

	precision	recall	f1-score	support
Class-0	1.00	0.01	0.01	181
Class-1	0.84	1.00	0.91	944
accuracy			0.84	1125
macro avg	0.92	0.50	0.46	1125
weighted avg	0.87	0.84	0.77	1125

```
#####
```

	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

```
#####
```

Тепер запусимо програму з команди:

```
python3 --W ignore class_imbalance.py
```

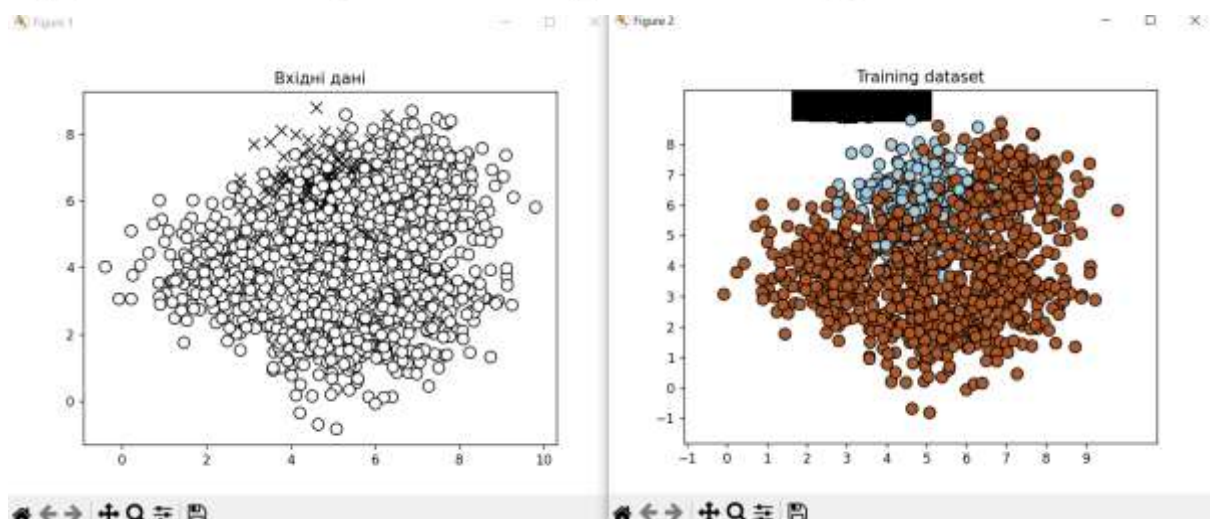
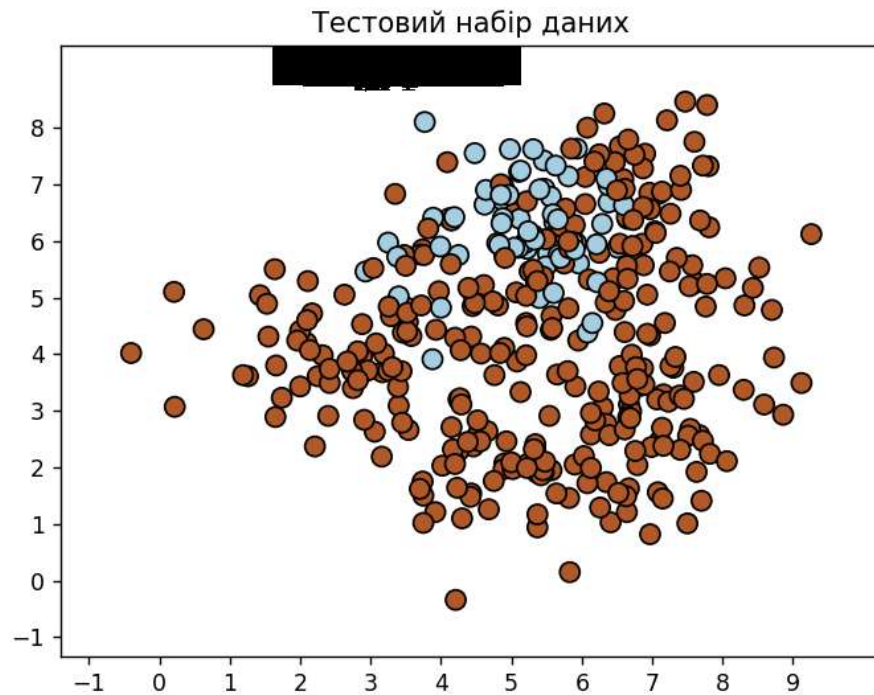


Figure 1



(x, y) = (3.90, 6.34)

```
#####
Classifier performance on training dataset

              precision    recall  f1-score   support

   Class-0       1.00      0.01      0.01       181
   Class-1       0.84      1.00      0.91       944

 accuracy              0.84       1125
 macro avg           0.92      0.50      0.46       1125
 weighted avg        0.87      0.84      0.77       1125

#####

#####

Classifier performance on test dataset

              precision    recall  f1-score   support

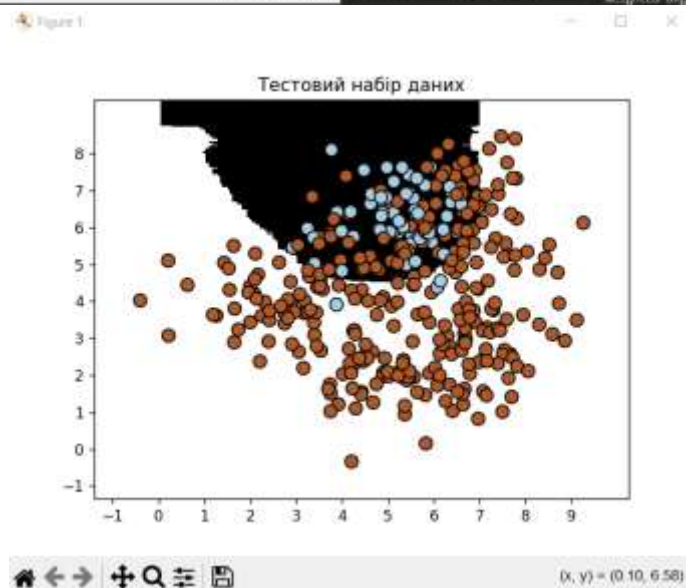
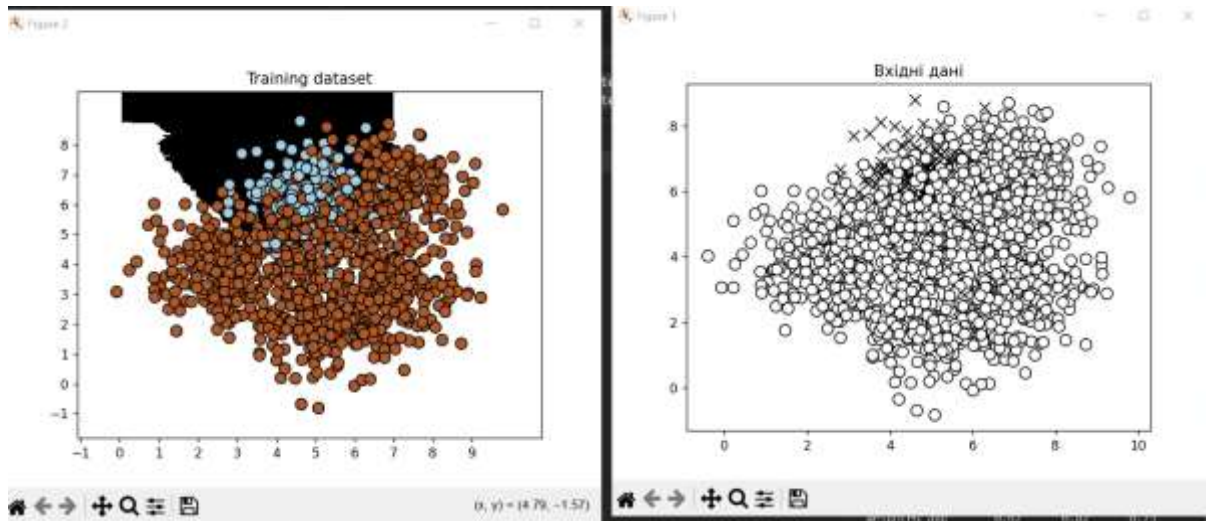
   Class-0       0.00      0.00      0.00        69
   Class-1       0.82      1.00      0.90       306

 accuracy              0.82       375
 macro avg           0.41      0.50      0.45       375
 weighted avg        0.67      0.82      0.73       375

#####
```

Тепер з використанням

```
python3 class_imbalance.py balance
```



```
#####
Classifier performance on training dataset
      precision    recall  f1-score   support

 Class-0       0.44      0.93      0.60       181
 Class-1       0.98      0.77      0.86       944

 accuracy              0.80       1125
 macro avg           0.71      0.85      0.73       1125
 weighted avg        0.89      0.80      0.82       1125

#####
#####
Classifier performance on test dataset
      precision    recall  f1-score   support

 Class-0       0.45      0.94      0.61        69
 Class-1       0.98      0.74      0.84       306

 accuracy              0.78       375
 macro avg           0.72      0.84      0.73       375
 weighted avg        0.88      0.78      0.80       375

#####
```

ЛІСТИНГ:

```
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_1[:, 0], class_1[:, 1], s=75,
            facecolors='white', edgecolors='black',
            linewidth=1, marker='o')

plt.title("Вхідні дані")

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['class_weight'] = 'balanced'
    else:
        raise ValueError("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, 'Тестовий набір даних')

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()

```

Завдання 2.3. Знаходження оптимальних навчальних параметрів за допомогою сіткового пошуку

```

##### Searching optimal parameters for precision_weighted
C:\Users\alhim\AppData\Local\Programs\Python\Python119\lib\site-packages\mapy\io\core.py:288: RuntimeWarning: invalid value encountered in cast
  _data = np.array(data, dtype=dtype, copy=copy,

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} --> 0.812
{'max_depth': 16, 'n_estimators': 100} --> 0.816
{'max_depth': 4, 'n_estimators': 25} --> 0.846
{'max_depth': 4, 'n_estimators': 50} --> 0.84
{'max_depth': 4, 'n_estimators': 100} --> 0.841
{'max_depth': 4, 'n_estimators': 250} --> 0.845

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

Performance report:

```

	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
{'max_depth': 4, 'n_estimators': 25} --> 0.843
{'max_depth': 4, 'n_estimators': 50} --> 0.836
{'max_depth': 4, 'n_estimators': 100} --> 0.837
{'max_depth': 4, 'n_estimators': 250} --> 0.841

```

```

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

```

```

Performance report:

```

	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

Лістинг:

```
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]},
    {'max_depth': [4], 'n_estimators': [25, 50, 100, 250]}
]

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))

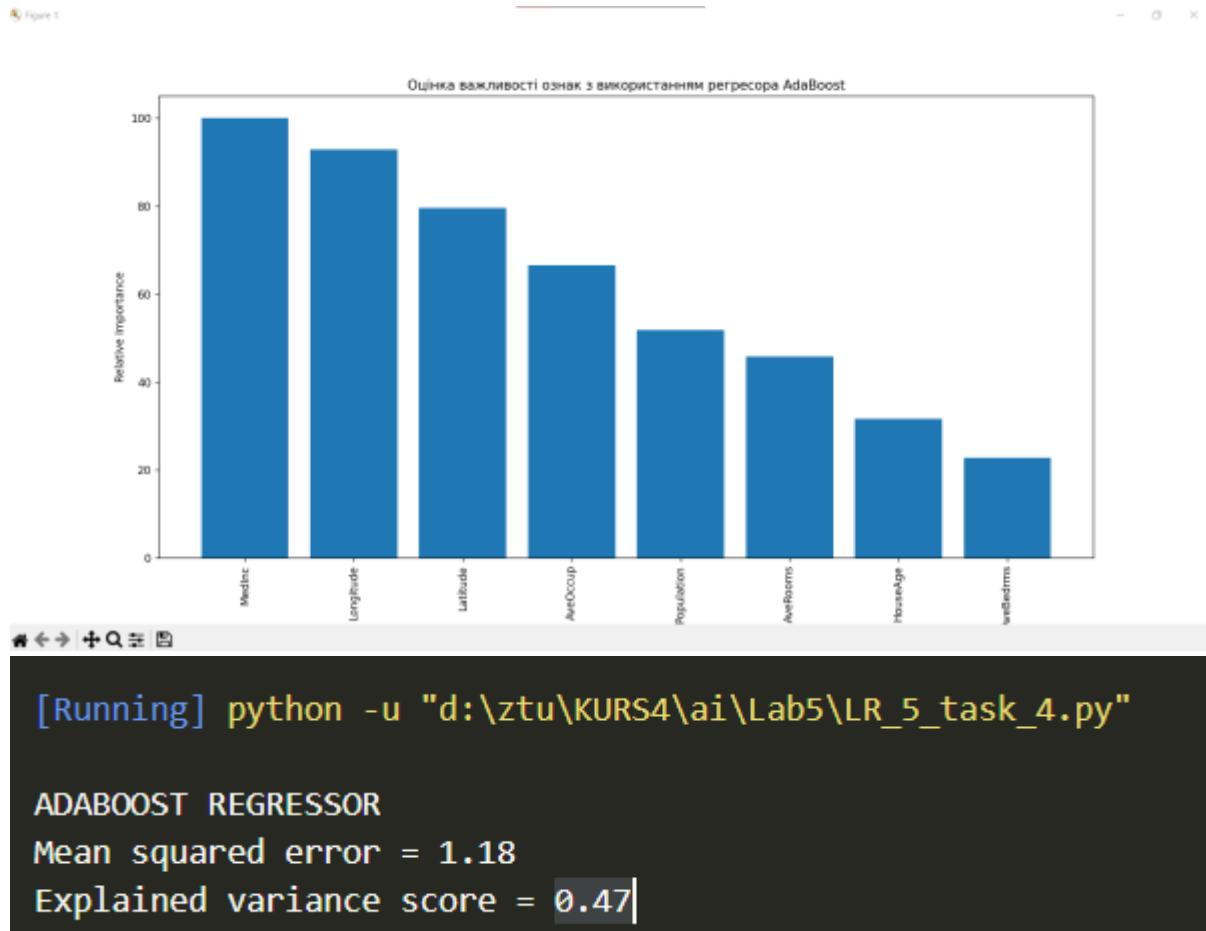
    print("\nBest parameters:", classifier.best_params_)

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

Завдання 2.4. Обчислення відносної важливості ознак

Код з завдання було змінено з даної причини:

```
ImportError:
`load_boston` has been removed from scikit-learn since version 1.2.
```



Лістинг:

```
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

# Load the California housing dataset
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(index_sorted.shape[0]) + 0.5

plt.figure()
plt.bar(pos, feature_importances[index_sorted], align='center')
plt.xticks(pos, np.array(feature_names)[index_sorted], rotation=90)
plt.ylabel("Relative Importance")
plt.title("Оцінка важливості ознак з використанням регресора AdaBoost")
plt.show()

```

Завдання 2.5. Прогнозування інтенсивності дорожнього руху за допомогою класифікатора на основі гранично випадкових лісів

```

Mean absolute error: 7.42
Predicted traffic: 26

```

Лістинг:

```

import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.ensemble import ExtraTreesRegressor

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([item])[0])
        count = count + 1

test_datapoint_encoded = np.array(test_datapoint_encoded)

print("Predicted traffic:",
int(regressor.predict([test_datapoint_encoded])[0]))

```

GIT: https://github.com/Alhim616/AI_Labs_Yanushevych