ЛАБОРАТОРНА РОБОТА № 5

ДОСЛІДЖЕННЯ МЕТОДІВ АНСАМБЛЕВОГО НАВЧАННЯ

Лукашевич Влада ІПЗ-21-1

https://github.com/vladalukashevych/artificial-intelligence-systems

Завдання №1. Створення класифікаторів на основі випадкових та гранично випадкових лісів.

```
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

# Argument parser
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

if __name__=='__main__':
    # Parse the input arguments
    args = build_arg_parser().parse_args()
    classifier_type = args.classifier_type

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

    # Separate input data into three classes based on labels
    class_0 = np.array(X[y==0])
    class_1 = np.array(X[y==1])
    class_2 = np.array(X[y==2])

    # Visualize input data
    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')

    # Split data into training and testing datasets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=5)

    # Ensemble Learning classifier
    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')

    # Evaluate classifier performance
    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")

    # Compute confidence
    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 the datapoints
    visualize_classifier(classifier, test_datapoints, [0]*len(test_datapoints),
'Тестові точки даних')
    plt.show()
```

rf

```
(.venv) PS D:\ZTU\Нейронка\AI\lab5> python random_forests.py --classifier-type rf

########################################

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
     Class-2       0.86      0.87      0.86       224

    accuracy                           0.87       675
   macro avg       0.87      0.87      0.87       675
weighted avg       0.87      0.87      0.87       675

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

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

Classifier performance on test dataset

              precision    recall  f1-score   support

     Class-0       0.92      0.85      0.88        79
     Class-1       0.86      0.84      0.85        70
     Class-2       0.84      0.92      0.88        76

    accuracy                           0.87       225
   macro avg       0.87      0.87      0.87       225
weighted avg       0.87      0.87      0.87       225

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

```
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
```

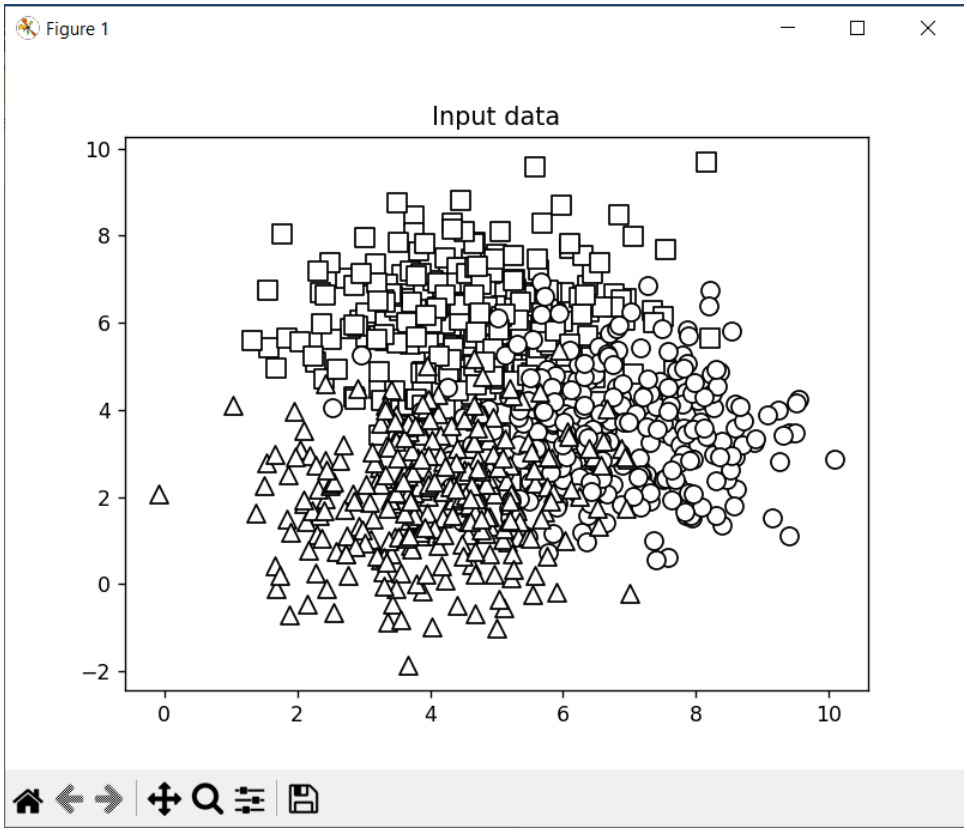


Figure 1
Input data
10
8
6
4
2
0
−2
0
2
4
6
8
10

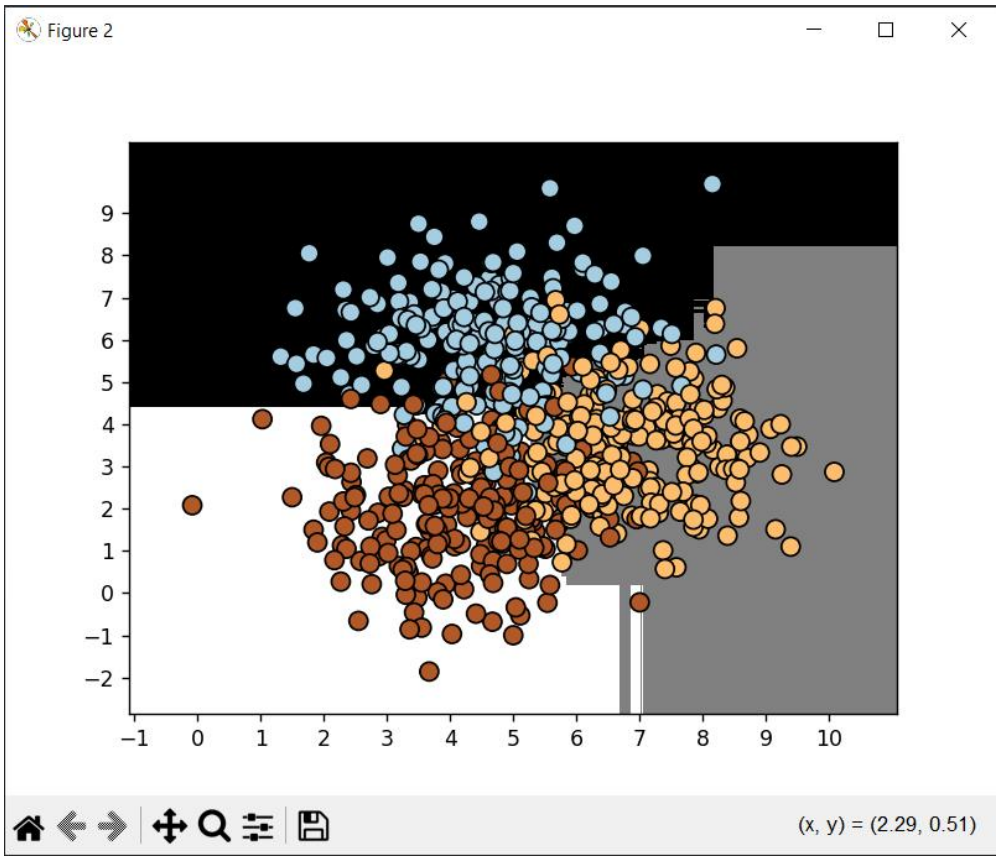
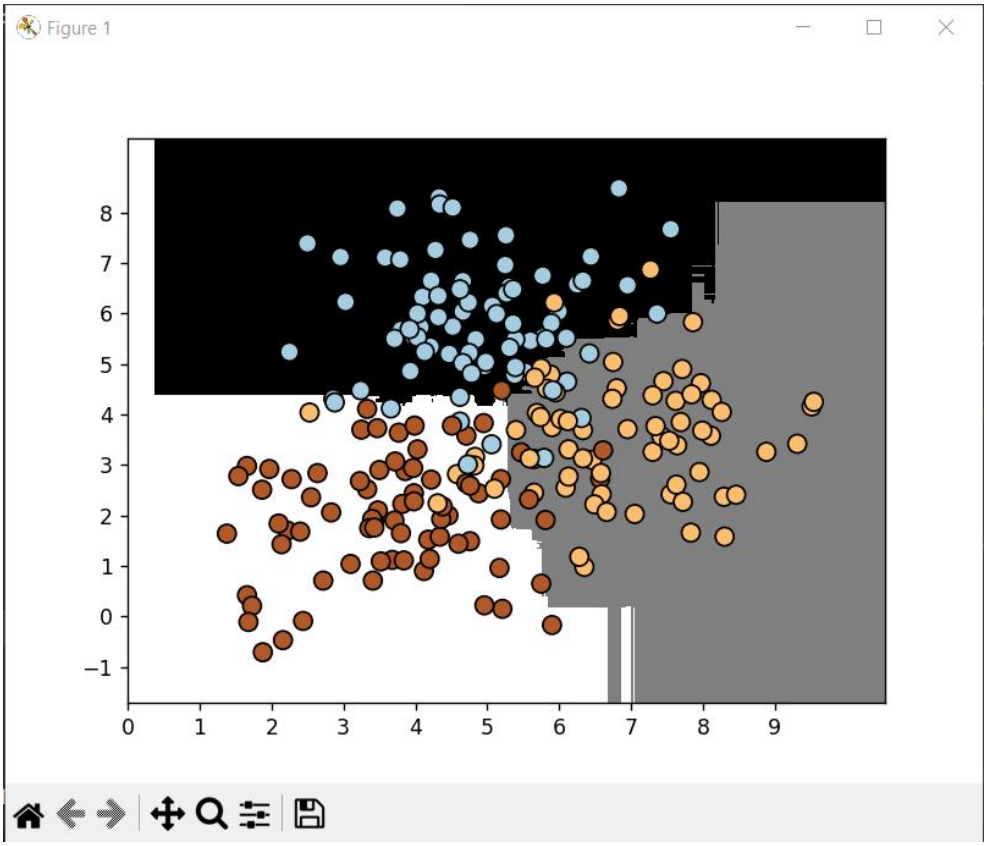
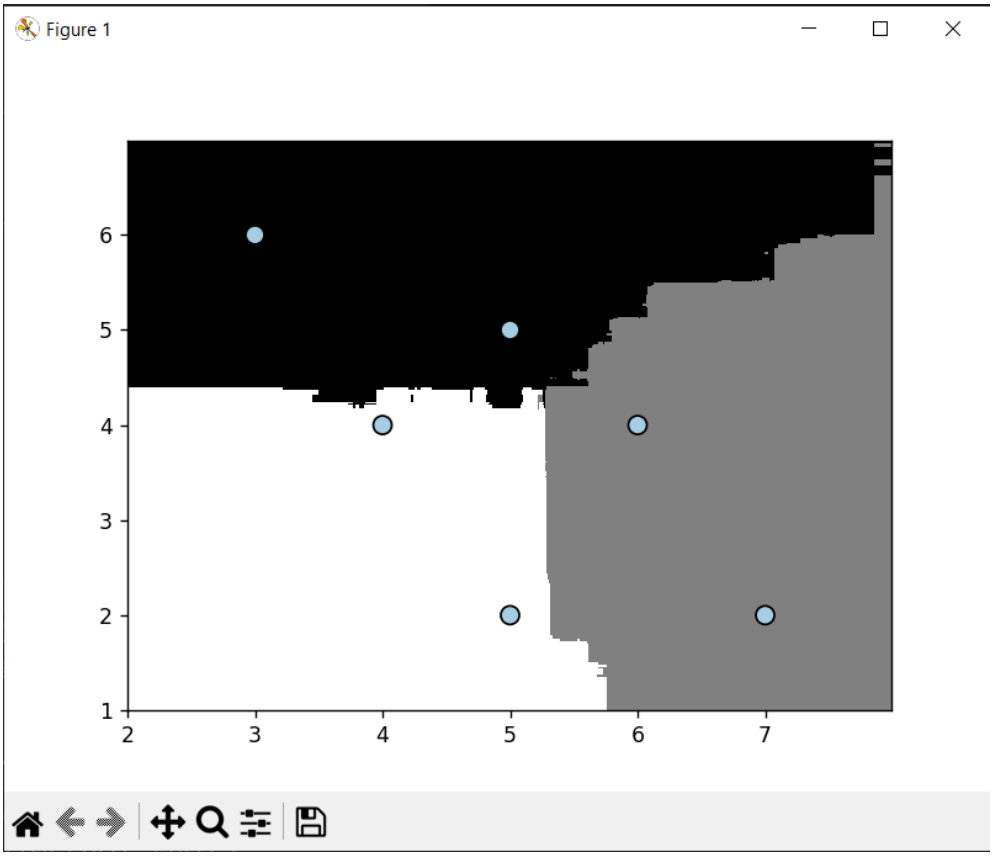


Figure 2
9
8
7
6
5
4
3
2
1
0
−1
−2
−1
0
1
2
3
4
5
6
7
8
9
10
(x, y) = (2.29, 0.51)





Модель Random Forest демонструє стабільну продуктивність як на навчальній, так і на тестовій вибірках. На навчальному наборі середні значення точності, відгуку та F1-міри складають 0.87 для всіх класів. Найвища точність

спостерігається для класу Class-0 (0.91), а відгук і F1-міра залишаються збалансованими для всіх класів. На тестовому наборі загальна точність моделі становить 0.87. F1-міра для класів знаходиться в діапазоні від 0.85 до 0.88, що свідчить про хорошу узгодженість моделі. Передбачені класи для вибраних тестових точок узгоджуються з навченою моделлю, що підтверджує її стабільність. Графіки показують розподіл вхідних даних і рішення моделі, демонструючи межі між класами та ілюструючи передбачення та впевненість моделі. Загальний висновок: модель має високу точність і добре справляється із завданням класифікації, демонструючи збалансовані показники точності, відгуку та F1-міри.

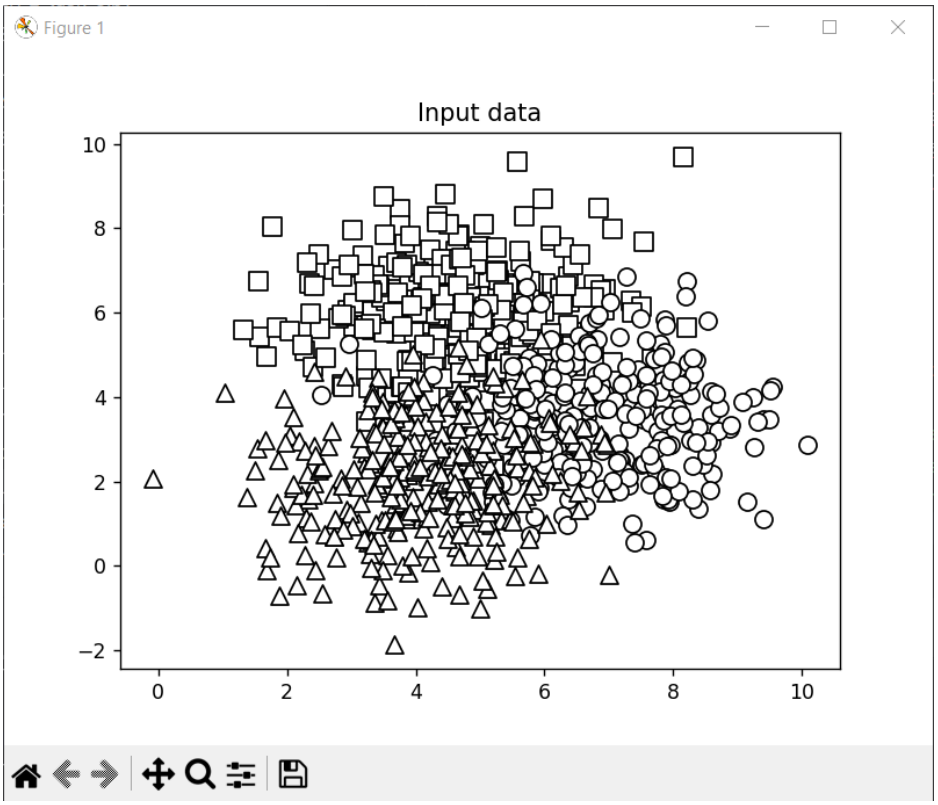
erf

```
(.venv) PS D:\ZTU\Нейронка\AI\lab5> python random_forests.py --classifier-type erf

########################################

Classifier performance on training dataset

              precision    recall  f1-score   support

     Class-0       0.89      0.83      0.86       221
     Class-1       0.82      0.84      0.83       230
     Class-2       0.83      0.86      0.85       224

    accuracy                           0.85       675
   macro avg       0.85      0.85      0.85       675
weighted avg       0.85      0.85      0.85       675

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

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

Classifier performance on test dataset

              precision    recall  f1-score   support

     Class-0       0.92      0.85      0.88        79
     Class-1       0.84      0.84      0.84        70
     Class-2       0.85      0.92      0.89        76

    accuracy                           0.87       225
   macro avg       0.87      0.87      0.87       225
weighted avg       0.87      0.87      0.87       225

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

```
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
```



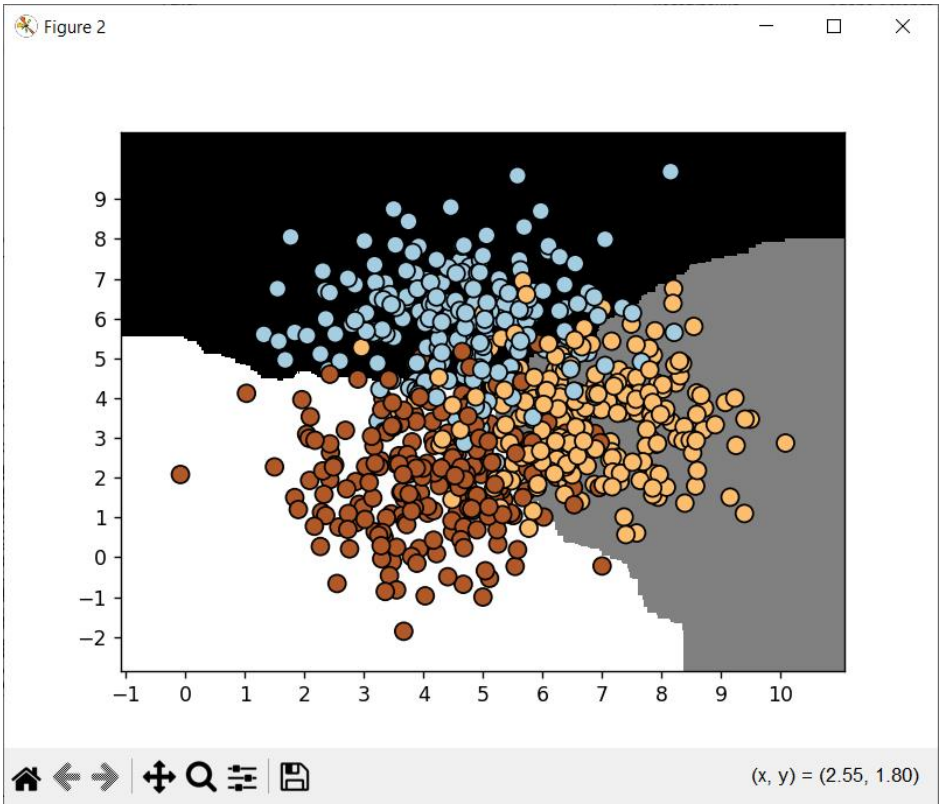


Figure 2
9
8
7
6
5
4
3
2
1
0
−1
−2
−1 0 1 2 3 4 5 6 7 8 9 10
(x, y) = (2.55, 1.80)

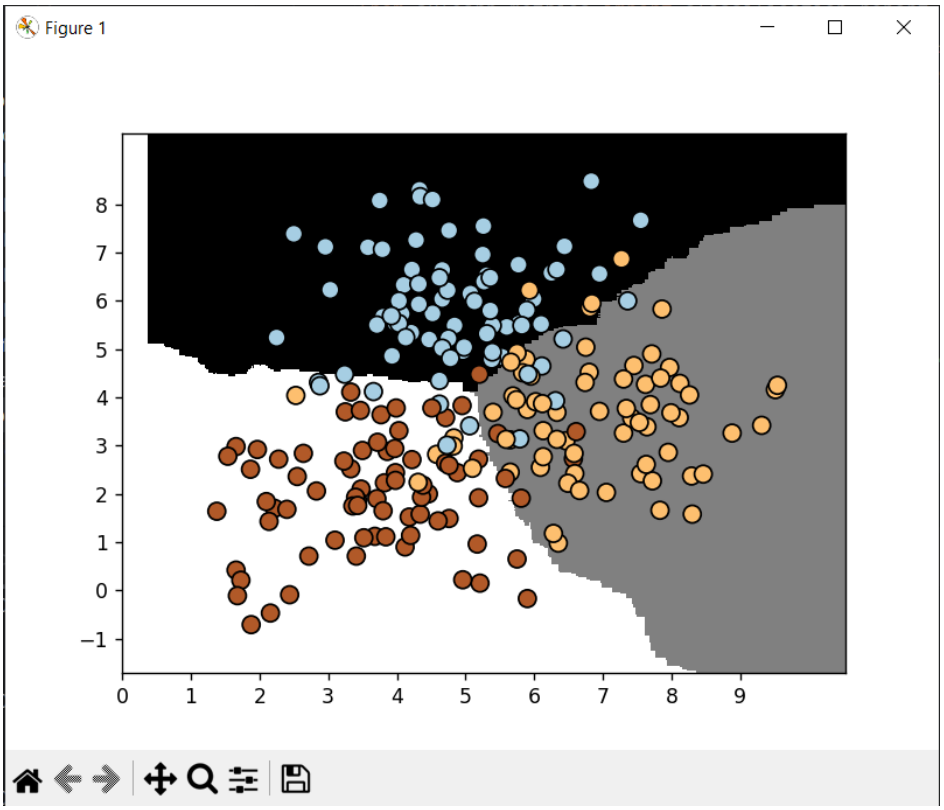
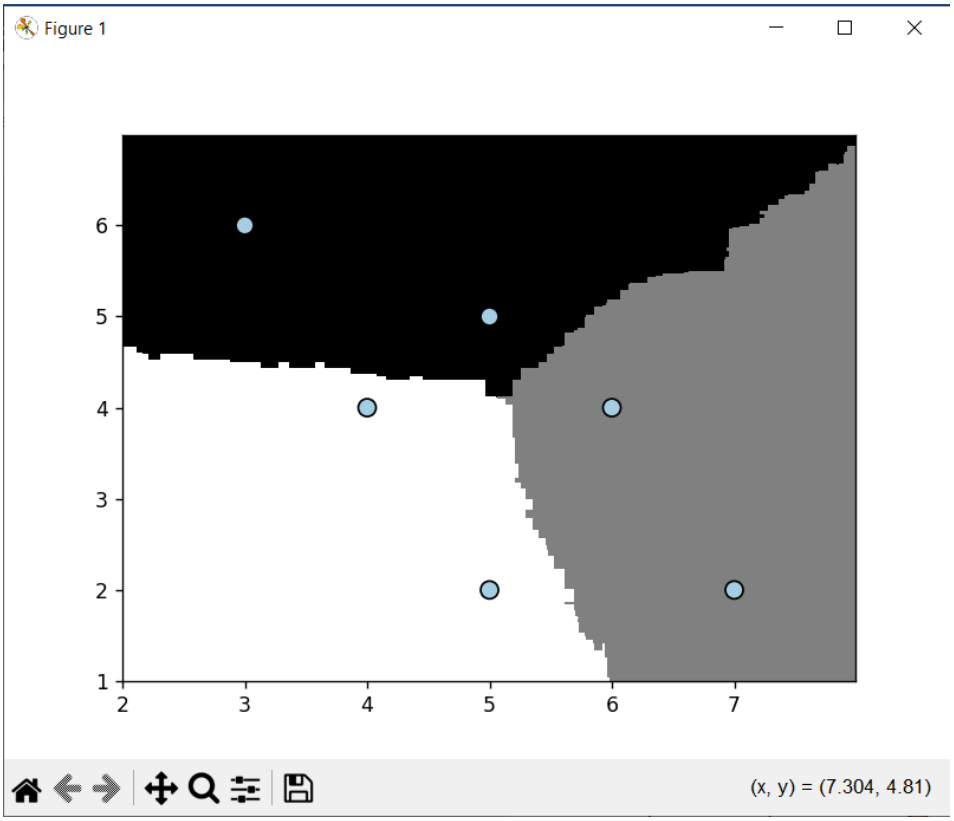


Figure 1
8
7
6
5
4
3
2
1
0
−1
0 1 2 3 4 5 6 7 8 9



Модель ExtraTreesClassifier (ERF) демонструє стабільну продуктивність на навчальному і тестовому наборах. На навчальній вибірці середні значення точності, відгуку та F1-міри становлять 0.85, що свідчить про узгодженість моделі. Найвища точність серед класів спостерігається для `Class-0` - 0.89. На тестовій вибірці загальна точність моделі складає 0.87. Значення F1-міри для всіх класів також близькі до 0.87, що підтверджує надійність моделі.

Передбачення для окремих точок підтверджують, що модель добре класифікує вибрані приклади, наприклад, точки `[5; 5]` та `[3; 6]` віднесено до `Class-0`, а точки `[6; 4]` та `[7; 2]` — до `Class-1`.

Графіки, що відображають вхідні дані і зони прийняття рішень моделі, підтверджують збалансовану продуктивність, ілюструючи межі між класами та області впевненості моделі у передбаченнях.

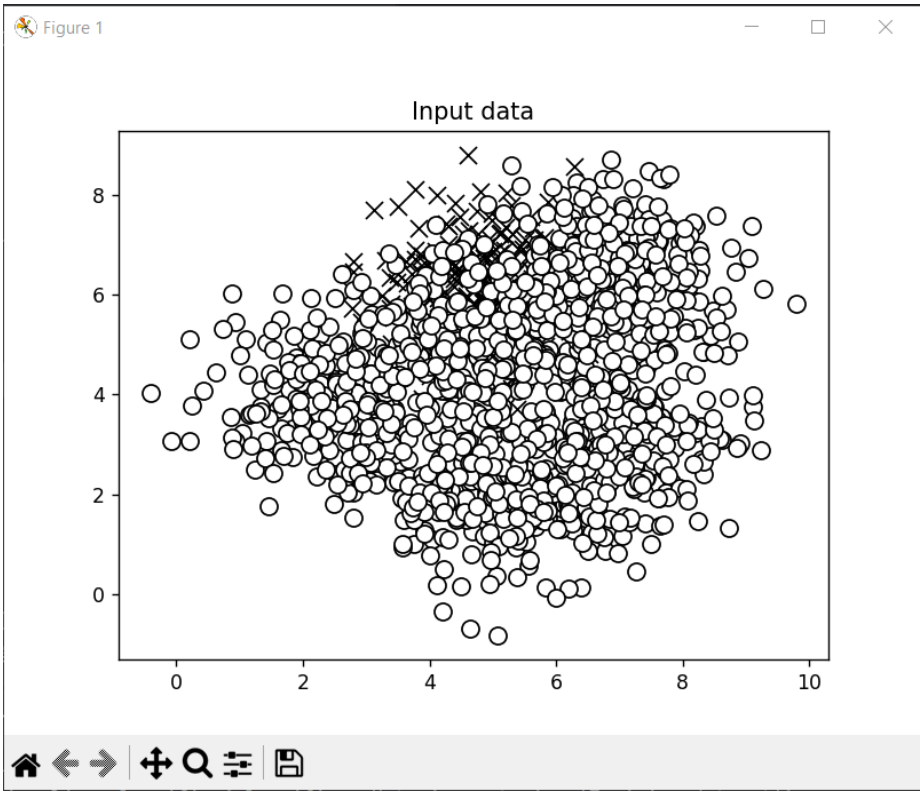
Завдання №2. Обробка дисбалансу класів.

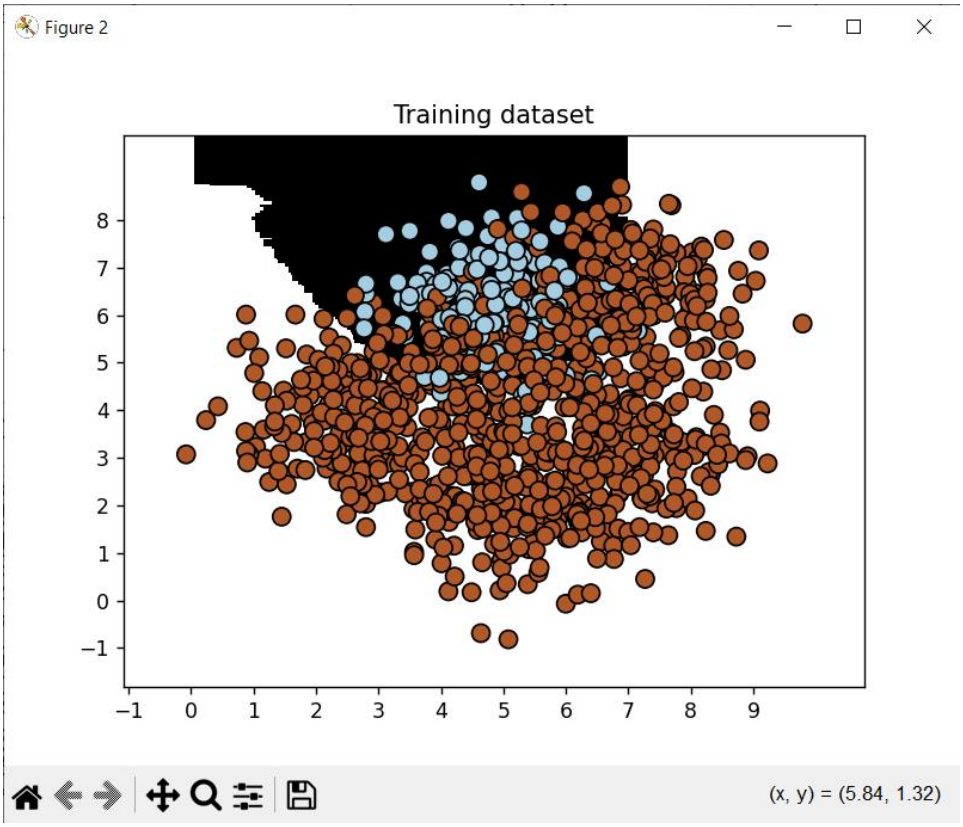
```
import sys
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split

from utilities import visualize_classifier

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

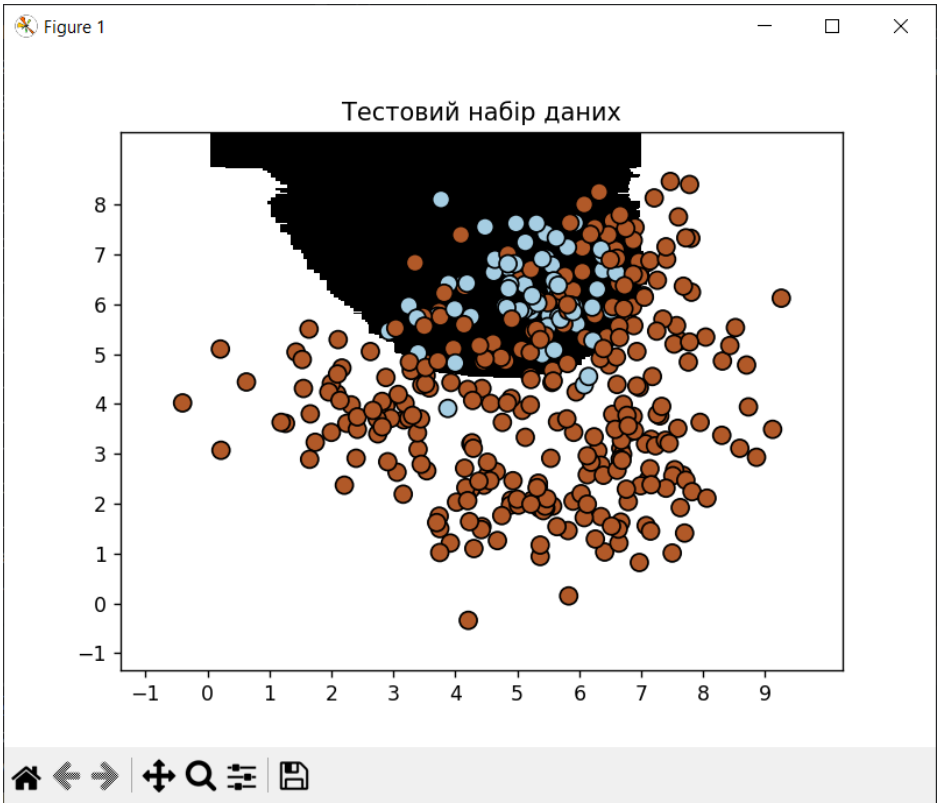
```
class_0 = np.array(X[y==0])
class_1 = np.array(X[y==1])

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 = train_test_split(X, y, test_size=0.25,
random_state=5)

# Класифікатор на основі гранично випадкових лісів
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')

# Передбачимо та візуалізуємо результат для тестового набору даних
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()
```

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

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

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







Результати скріншотів показують, що модель має проблеми з дисбалансом класів. На навчальній вибірці точність для `Class-0` низька (0.44) з високим відгуком (0.93), що свідчить про багато помилкових позитивних результатів. Для `Class-1` точність висока (0.98), але відгук менший (0.77), що вказує на

пропуски. Загальна точність становить 0.80, а макро F1-міра — 0.73, що вказує на значну різницю між класами.

На тестовій вибірці ситуація подібна: точність для `Class-0` — 0.45, відгук — 0.94, для `Class-1` точність 0.98, відгук 0.74. Загальна точність 0.78, макро F1-міра — 0.73, що підтверджує проблему з дисбалансом.

Завдання №3. Знаходження оптимальних навчальних параметрів за допомогою сіткового пошуку.

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

# Визначення сітки значень параметрів
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:")
    means = classifier.cv_results_['mean_test_score']
    params = classifier.cv_results_['params']
    for mean, param in zip(means, params):
        print(param, '-->', round(mean, 3))
    print("\nBest parameters:", classifier.best_params_)

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

```
##### Searching optimal parameters for precision_weighted
D:\ZTU\Нейронка\AI\.venv\Lib\site-packages\numpy\ma\core.py:2881: RuntimeWarning:
  _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} --> 0.832
{'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}
```

```
##### 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
```

Параметричний пошук із метриками precision_weighted та recall_weighted показав, що найкращі параметри для моделі — це max_depth: 2 і n_estimators:

100. Ці параметри забезпечили найвищі значення точності та відгуку серед перевірених комбінацій.

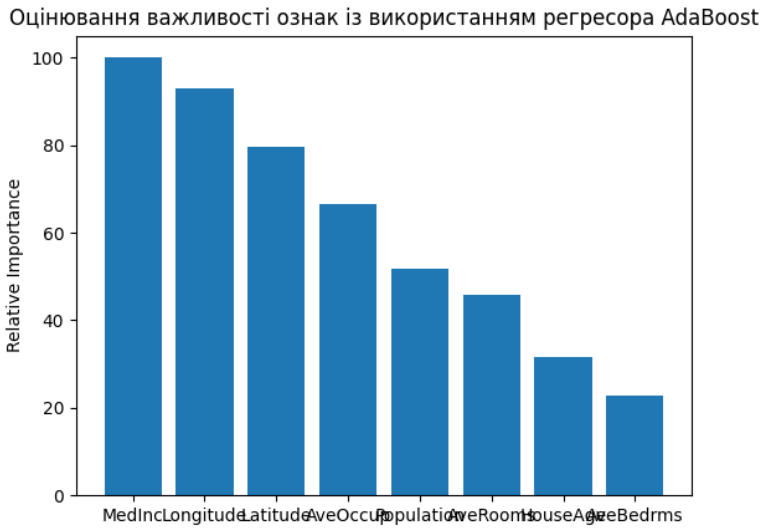
Загальна точність моделі становить 0.86, що є хорошим результатом. Макро- та зважена середні оцінки для точності, відгуку та F1-міри також дорівнюють 0.86, що вказує на збалансовану продуктивність моделі на різних класах.

Завдання №4. Обчислення відносної важливості ознак.

```
import numpy as np
import pandas as pd
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

# Завантаження даних із цінами на нерухомість
# housing_data = datasets.load_boston()
housing_data = datasets.fetch_california_housing()

# Перемішаємо дані, щоб підвищити об'єктивність нашого аналізу
X, y = shuffle(housing_data.data, housing_data.target, random_state=7)

# Роздіб'ємо дані на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=7)

# Модель на основі регресора AdaBoost
regressor = AdaBoostRegressor(
    DecisionTreeRegressor(max_depth=4),
    n_estimators=400, random_state=7)
regressor.fit(X_train, y_train)

# Оцінюємо ефективність регресора
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))

# Вилучення важливості ознак
feature_importances = regressor.feature_importances_
feature_names = housing_data.feature_names

# Нормалізація значень важливості ознак
feature_importances = 100.0 * (feature_importances / max(feature_importances))

# Сортування та перестановка значень
index_sorted = np.flipud(np.argsort(feature_importances))

# Розміщення міток уздовж осі X
pos = np.arange(index_sorted.shape[0]) + 0.5

feature_names = np.array(housing_data.feature_names)

# Побудова стовпчастої діаграми
plt.figure()
```

```
plt.bar(pos, feature_importances[index_sorted], align='center')
plt.xticks(pos, feature_names[index_sorted])
plt.ylabel('Relative Importance')
plt.title('Оцінювання важливості ознак із використанням регресора AdaBoost')
plt.show()
```

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



Відповідно до діаграми, можемо зробити висновки, що найважливішими ознаками є MedInc, Longitude, Latitude. Натомість AveBedrms, HouseAge, AveRooms, Population є ознаками, якими можна знехтувати.

Завдання №5. Прогнозування інтенсивності дорожнього руху за допомогою класифікатора на основі гранично випадкових лісів.

```
import numpy as np
from sklearn.metrics import classification_report, mean_absolute_error
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
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([test_datapoint[i]])[0])
        count += 1
test_datapoint_encoded = np.array(test_datapoint_encoded)
test_datapoint_encoded = test_datapoint_encoded.reshape(1, -1)

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

```
Mean absolute error:  7.42
Predicted traffic:  26
```

Висновок: виконуючи лабораторну роботу, я дослідила методи ансамблів у машинному навчанні, використовуючи спеціалізовані бібліотеки та мову програмування Python.