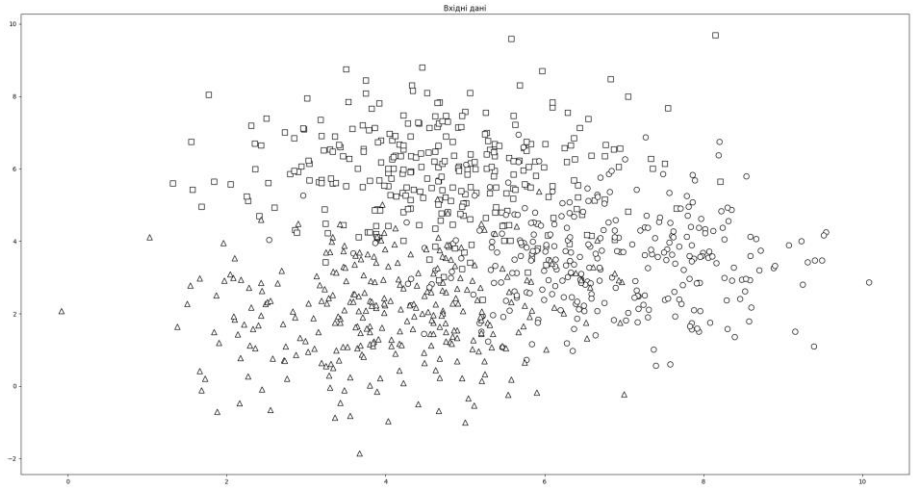
ЛАБОРАТОРНА РОБОТА № 5 ДОСЛІДЖЕННЯ МЕТОДІВ АНСАМБЛЕВОГО НАВЧАННЯ

Гітхаб: https://github.com/PanchukPetro/SShILabsPanchuk/tree/main/Lab5

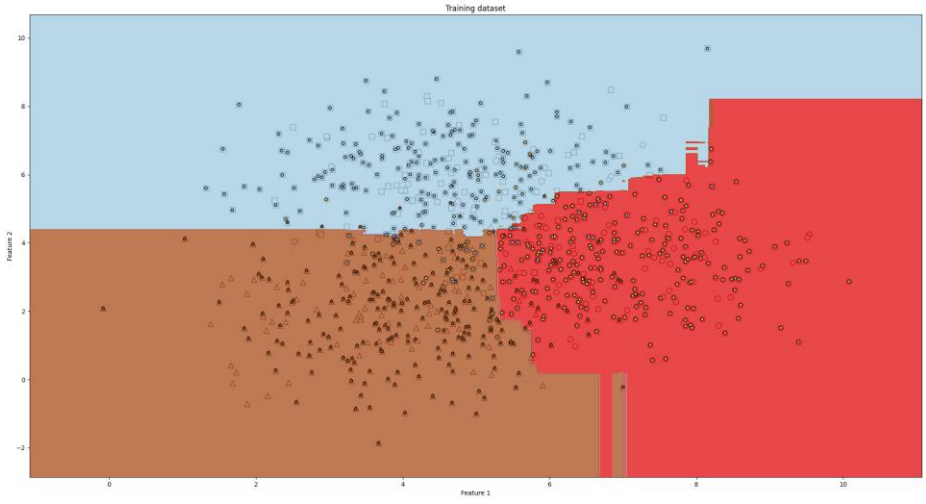
Мета роботи: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити методи ансамблів у машинному навчанні

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

Використовувати файл вхідних даних: data_random_forests.txt, побудувати класифікатори на основі випадкових та гранично випадкових лісів.

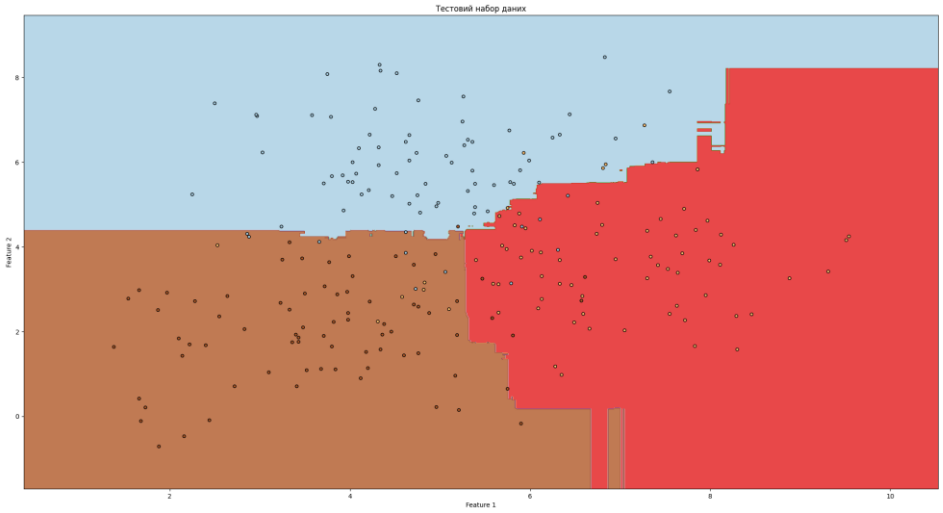


Графік вхідних даних

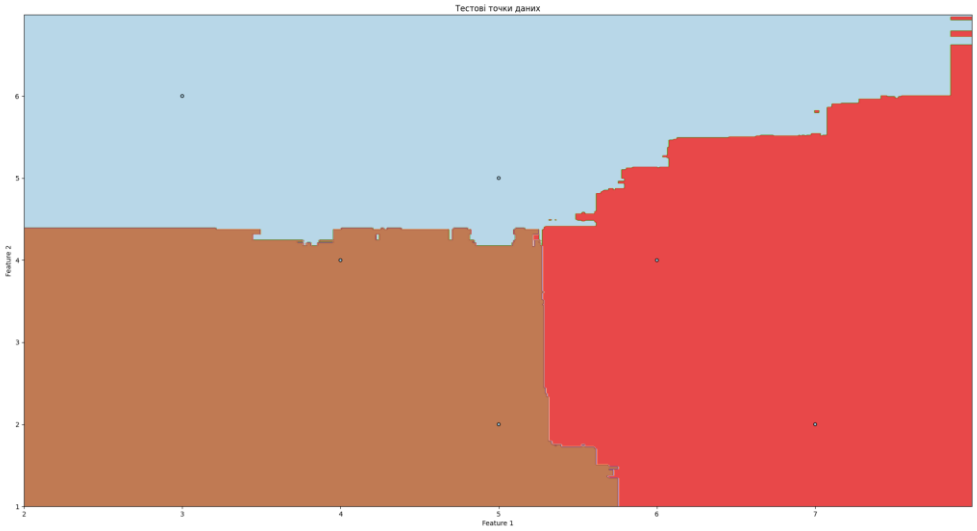


Тренувльний Графік rf

					ДУ «Житомирська політехніка».24.121.02.000 – ІПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Панчук П.С			СШІ Лабораторна робота №5	Літ.	Арк. 1 / Аркушів 14
Перевір.		Голенко М.Ю					
Керівник						ФІКТ Гр. ІПЗ-21-3	
Н. контр.							
Зав. каф.							



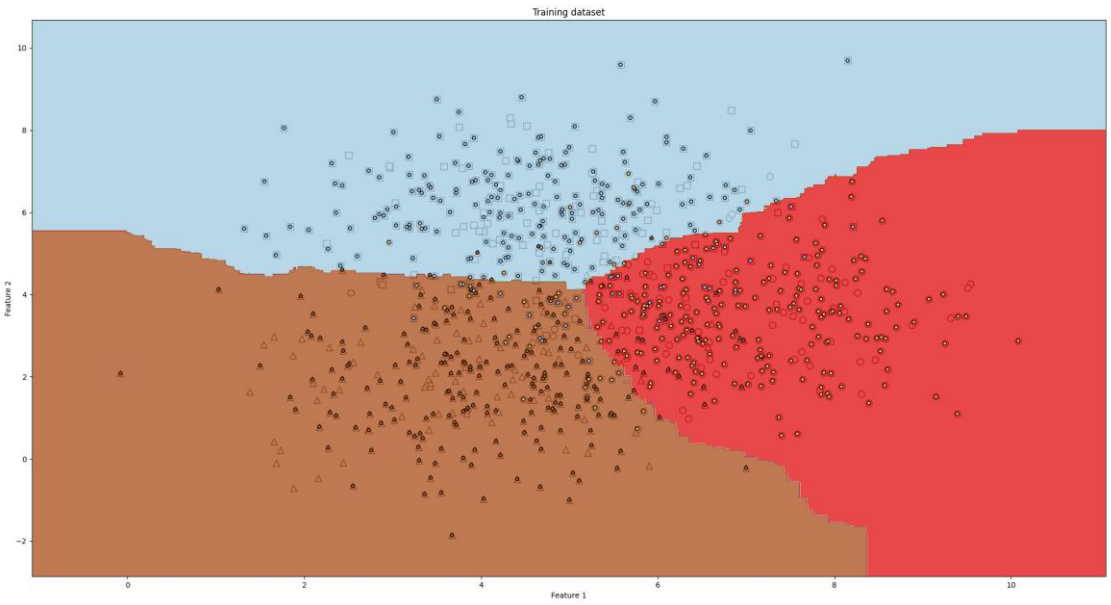
Тестовий Граіфк erf



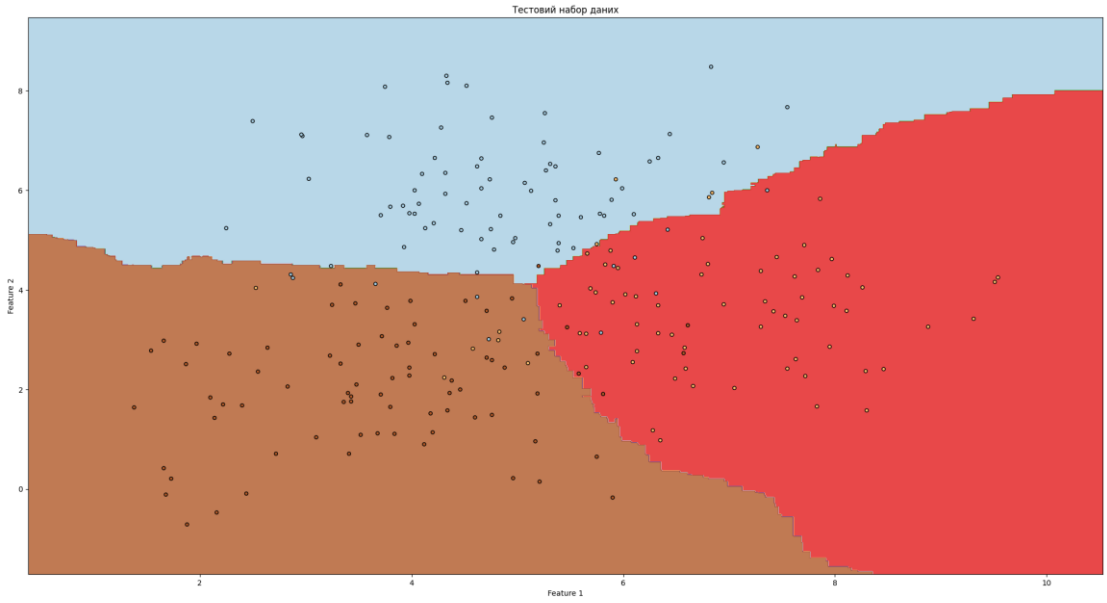
Тестові точки даних rf

```
PS E:\Politech\2024\AI Systems\Lab5> python LR_5_Task_1.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]

 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
```

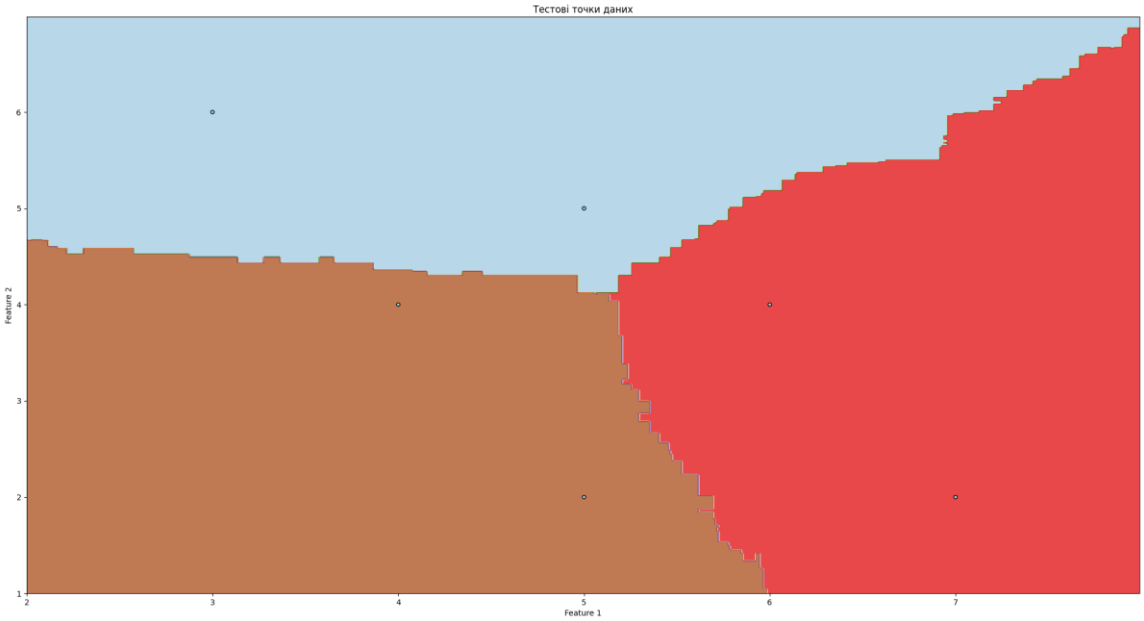
Консольний аутпут rf



Тестовий Графік rf



Тренувальний графік erf



Тестові точки даних erf

```
PS E:\Politech\2024\AI Systems\Lab5> python LR_5_Task_1.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
```

Консольний аутпут erf

Програмний код:

```
import argparse
import numpy as np
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from sklearn.metrics import classification_report
from visualize classifier 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'])
    return parser

if __name__ == '__main__':
    # Вилучення вхідних аргументів
    args = build arg parser().parse args()
    classifier type = args.classifier type

    # У файлі data_random_forests.txt кожен рядок містить значення розділені комою.
    # Перші два значення відповідають вхідним даним, останнє — цільовій мітці.
    # У цьому наборі даних містяться три різні класи. Завантажимо дані із цього файлу.

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

    # Розіб'ємо вхідні дані на три класи.
```

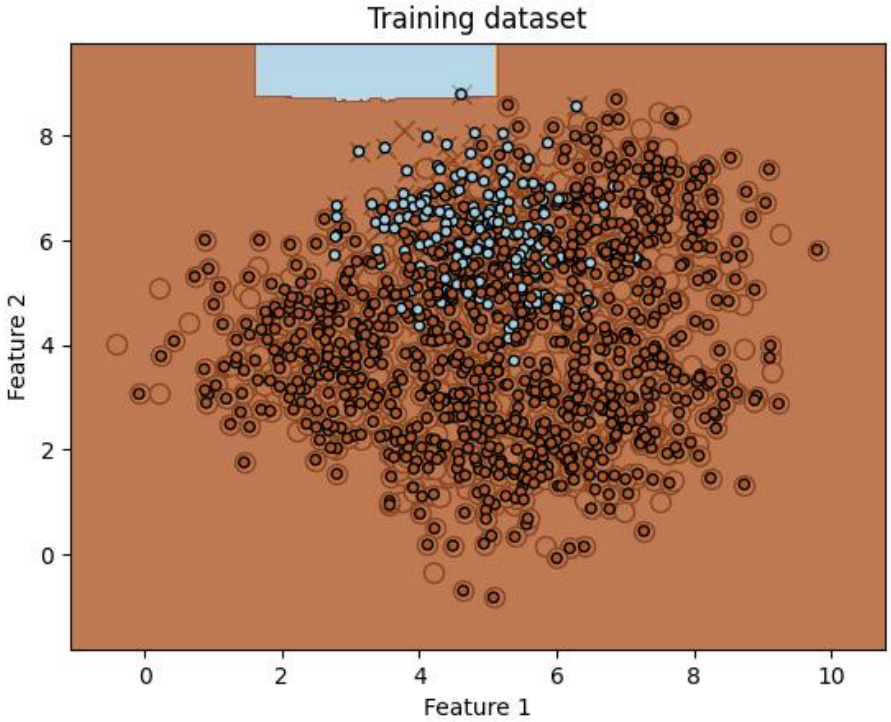
```
    # Розбиття вхідних даних на три класи
    class_0 = np.array(X[y == 0])
    class_1 = np.array(X[y == 1])
    class_2 = np.array(X[y == 2])

    # Візуалізуємо вхідні дані.

    # Візуалізація вхідних даних
    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")
    #plt.show()

    # Розіб'ємо дані на навчальний і тестовий набори.
    # Розбиття даних на навчальний та тестовий набори
    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 classifier_type == 'rf':
        classifier = RandomForestClassifier(**params)
    else:
        classifier = ExtraTreesClassifier(**params)

     #Навчимо та візуалізуємо класифікатор.
    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 набор даних')

    # Перевірка роботи класифікатора
    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)

    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("\n Confidence measure")
    for datapoint in test_datapoints:
        probabilities = classifier.predict_proba([datapoint])[0]
        predicted_class = 'Class - ' + str(np.argmax(probabilities))
        print('\n Datapoint: ', 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

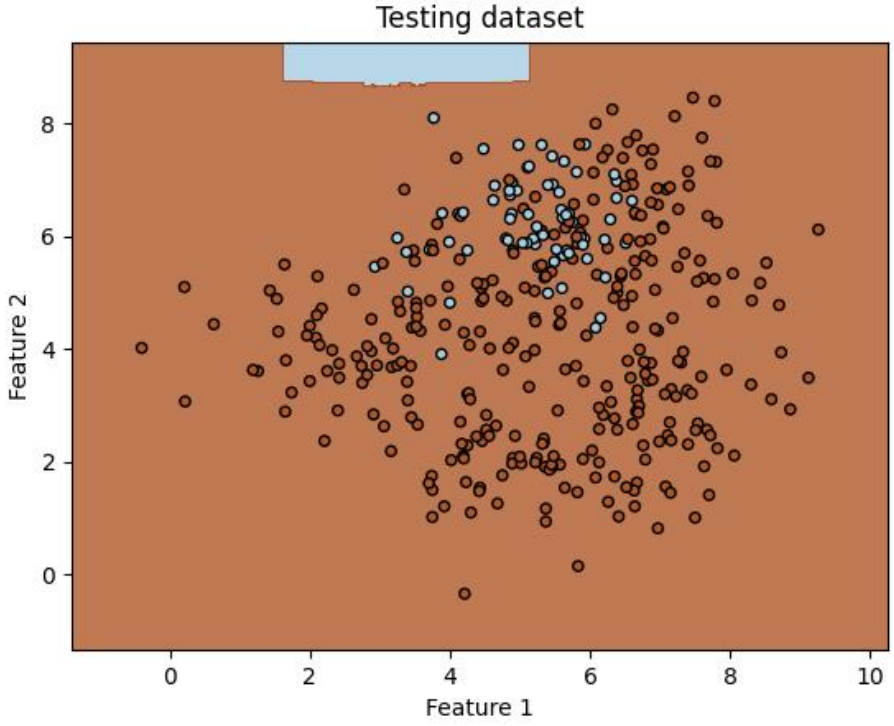
Використовуючи для аналізу дані, які містяться у файлі data_imbalance.txt проведіть обробку з урахуванням дисбалансу класів.

Програмний код:

```
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 visualize_classifier 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',
linewidths=1, marker='x')
plt.scatter(class_1[:, 0], class_1[:, 1], s=75, facecolors='white', edgecolors='black',
linewidths=1, marker='o')
plt.title('Вхідні дані')

X train, X test, y train, y test = model selection.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')

y test pred = classifier.predict(X test)
visualize classifier(classifier, X test, y test, 'Testing dataset')

#Обчислення показників ефективності класифікатора
class_names = ['Class-0','Class-1']
print("\n" + "#"*40)
print("\n Classifier performance on training dataset \n")
print(classification report(y train,classifier.predict(X train), target names = class names))
print("\n" + "#"*40)
print("\n Classifier performance on test dataset \n")
print(classification_report(y_test, y_test_pred, target_names = class_names))
print("\n" + "#"*40)

plt.show()
```



Графік вхідних даних

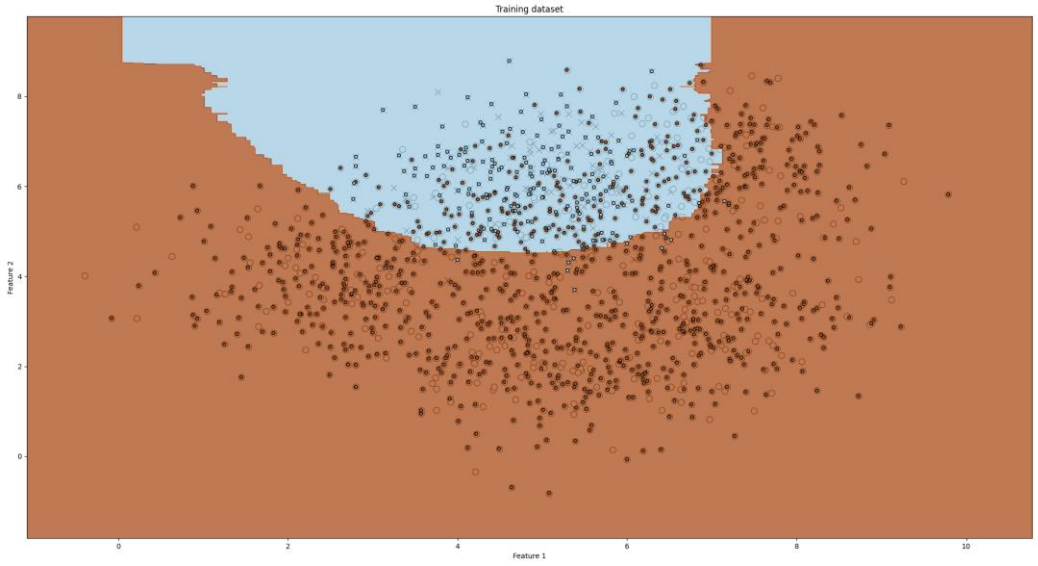


Графік для тестового набору

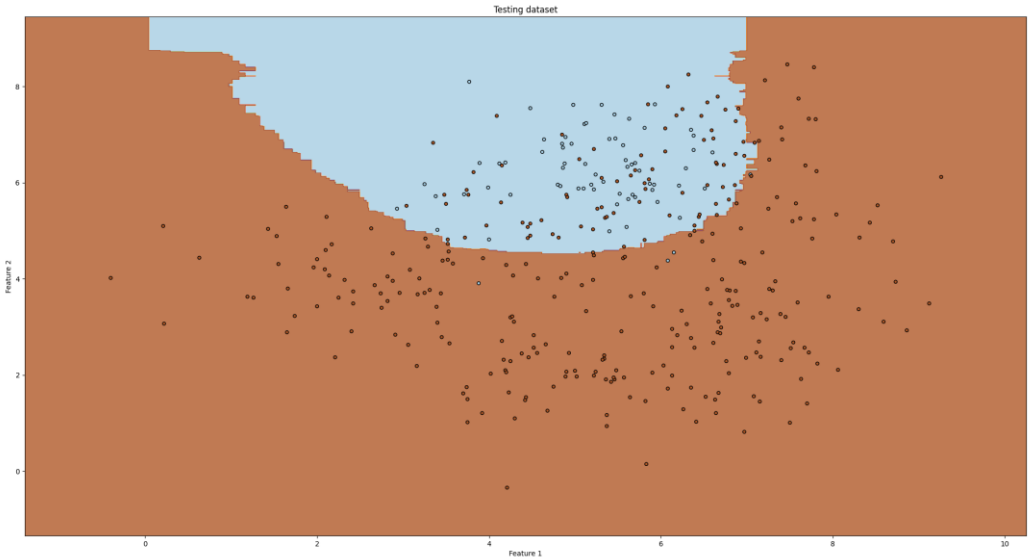
```
########################################

 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


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

Оцінка перформанса класифікатора

Виконуємо python LR_5_Task_2.py balance



Графік тренувальних даних, збалансований



Графік тестових даних, збалансований

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

 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


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

Оцінка перформанса збалансованого класифікатора

Завдання 2.3. Знаходження оптимальних навчальних параметрів за допомогою сіткового пошуку

Використовуючи дані, що містяться у файлі data_random_forests.txt. знайти оптимальних навчальних параметрів за допомогою сіткового пошуку.

Програмний код:

```
import numpy as np
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.metrics import classification report
from sklearn.model selection import GridSearchCV
from sklearn.model selection import train test split

# Використовуємо для нашого аналізу дані, що містяться у файлі data_random_forests.txt.
input file = 'data random forests.txt'
data = np.loadtxt(input file, delimiter=',')
X, y = data[:, :-1], data[:, -1]

# Розіб'ємо дані на три класи.
class_0 = np.array(X[y == 0])
class 1 = np.array(X[y == 1])
class 2 = np.array(X[y == 2])

# Розіб'ємо дані на навчальний та тестовий набори.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=5)

# Задамо сітку значень параметрів, де будемо тестувати класифікатор.
parameter grid = [
    {'n estimators': [100], 'max depth': [2, 4, 7, 12, 16]},
    {'max_depth': [4], 'n_estimators': [25, 50, 100, 250]},
]

# Визначимо метрики, які використовуватимемо для оцінки.
metrics = ['precision weighted', 'recall weighted']

# Виконуємо пошук оптимальних параметрів для кожної метрики.
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")
results = classifier.cv_results_
for mean_score, params in zip(results['mean_test_score'], results['params']):
    print(params, '-->', mean score)

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

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

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

Grid scores for the parameter grid
{'max_depth': 2, 'n_estimators': 100} --> 0.842962962962963
{'max_depth': 4, 'n_estimators': 100} --> 0.837037037037037
{'max_depth': 7, 'n_estimators': 100} --> 0.8414814814814815
{'max_depth': 12, 'n_estimators': 100} --> 0.8296296296296297
{'max_depth': 16, 'n_estimators': 100} --> 0.8148148148148149
{'max_depth': 4, 'n_estimators': 25} --> 0.842962962962963
{'max_depth': 4, 'n_estimators': 50} --> 0.8355555555555556
{'max_depth': 4, 'n_estimators': 100} --> 0.837037037037037
{'max_depth': 4, 'n_estimators': 250} --> 0.8414814814814815

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
```

Інформація після виконання коду

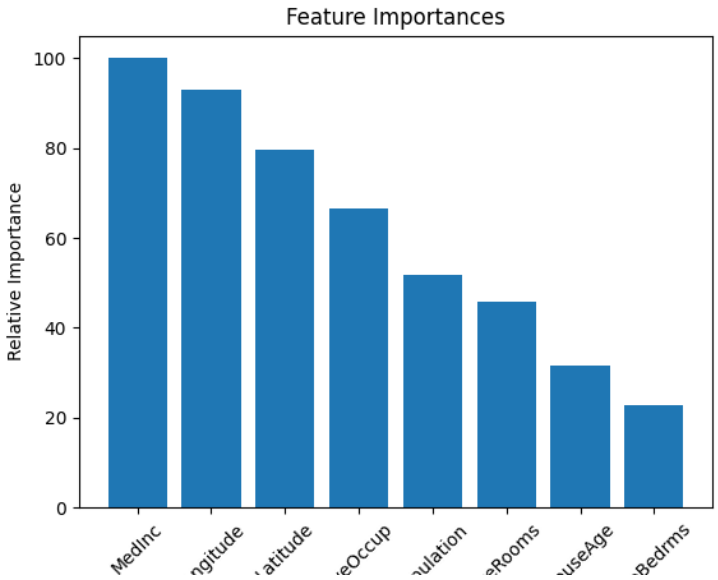
Завдання 2.4. Обчислення відносної важливості ознак

Програмний код:

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

housing_data = fetch_california_housing(as_frame=True)
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)

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

# Обчислення показників ефективності регресора AdaBoost
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

# Нормалізуємо значення відносної ваги ознак
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[i] for i in index_sorted], rotation=45)
plt.ylabel("Relative Importance")
plt.xlabel("Features")
plt.title("Feature Importances")
plt.show()
```

Я використав California_housing, бо boston_housing було видалено з sklearn



Графік важливостей ознак

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

Виведення в консоль

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

Програмний код:

```
import numpy as np
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)

# Перетворення рядкових даних на числові
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)

# Розіб'ємо дані на навчальний та тестовий набори.
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))

# Тестування кодування на одному прикладі
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]]))
        count += 1
test_datapoint_encoded = np.array(test_datapoint_encoded)

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

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

Виведення в консоль