ЛАБОРАТОРНА РОБОТА № 5
ДОСЛІДЖЕННЯ МЕТОДІВ АНСАМБЛЕВОГО НАВЧАННЯ

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

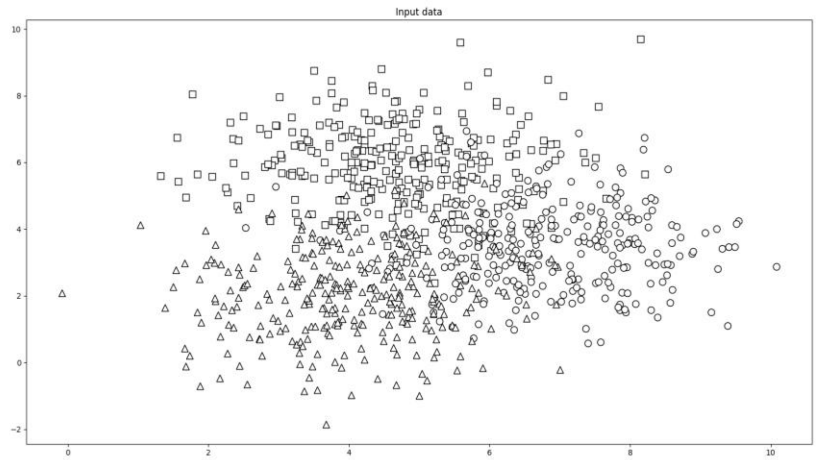
Хід роботи

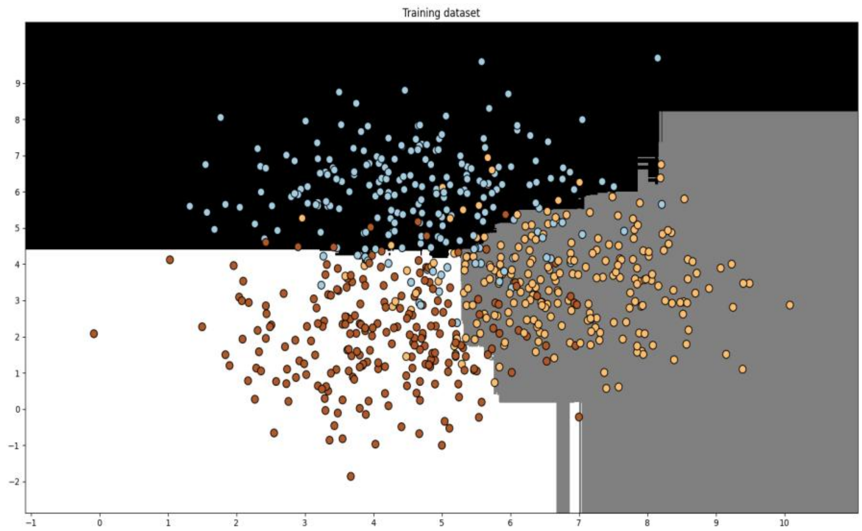
Завдання 2.1. Створення класифікаторів на основі випадкових та гранично випадкових лісів

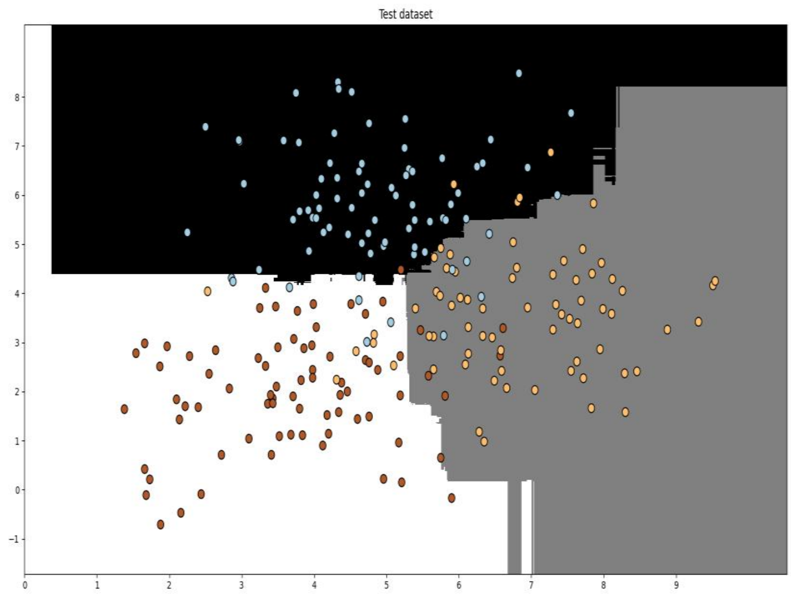
```
import argparse
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from sklearn.metrics import classification_report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


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('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()
```

- Запуск для rf

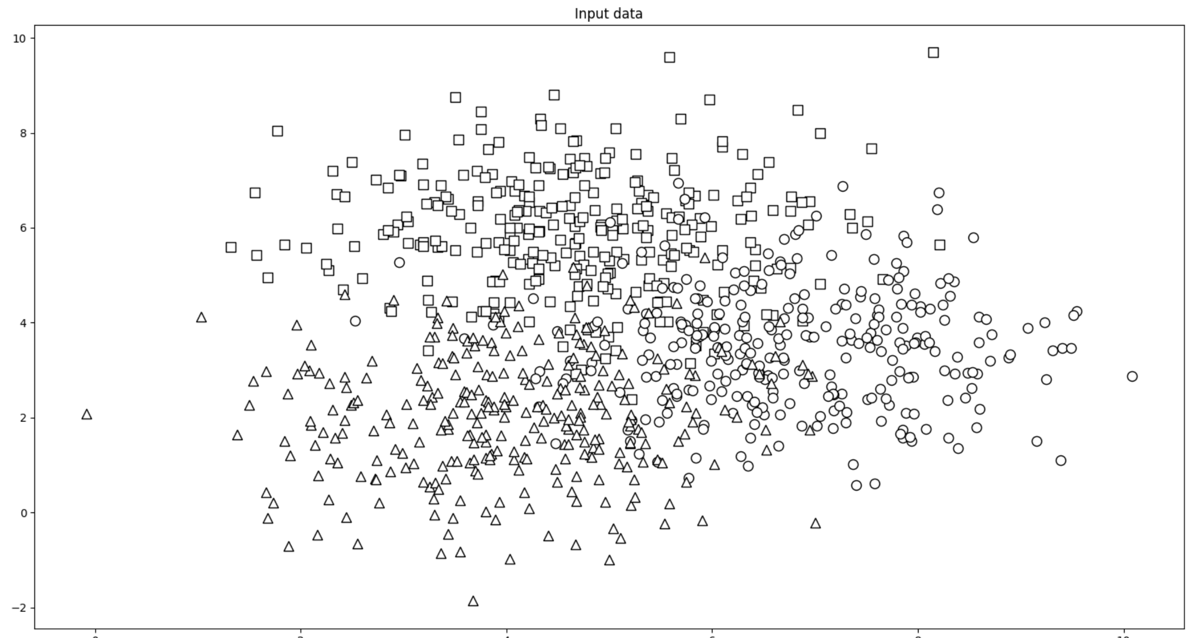


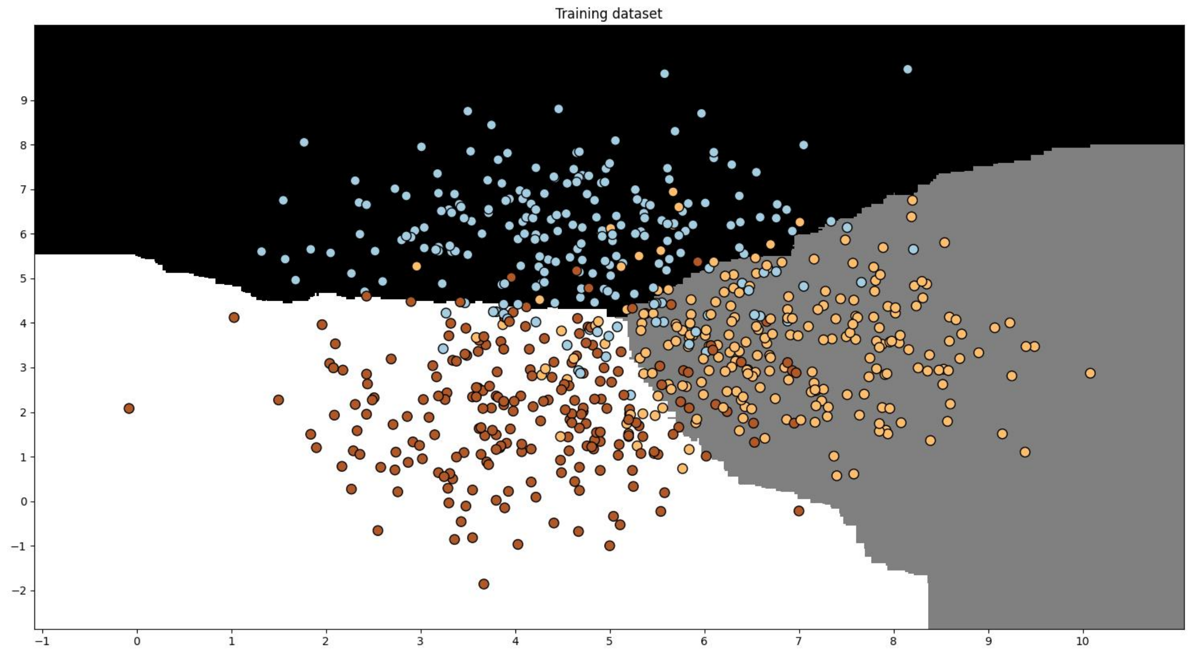


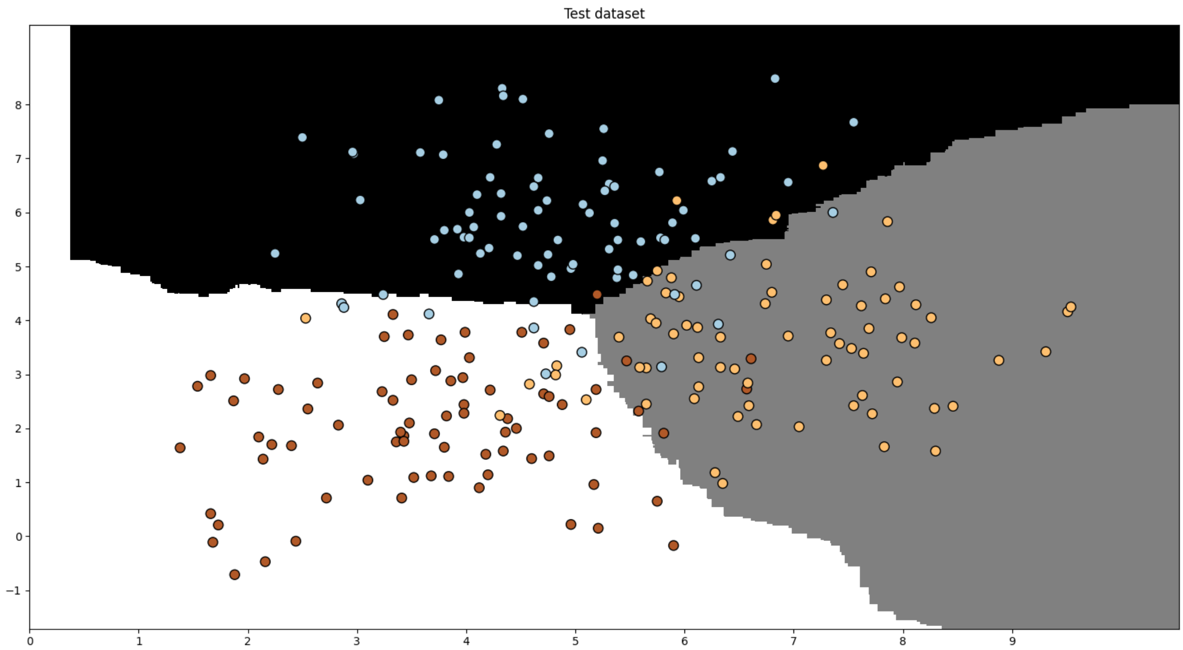


```
########################################

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

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

- Запуск для ERF


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

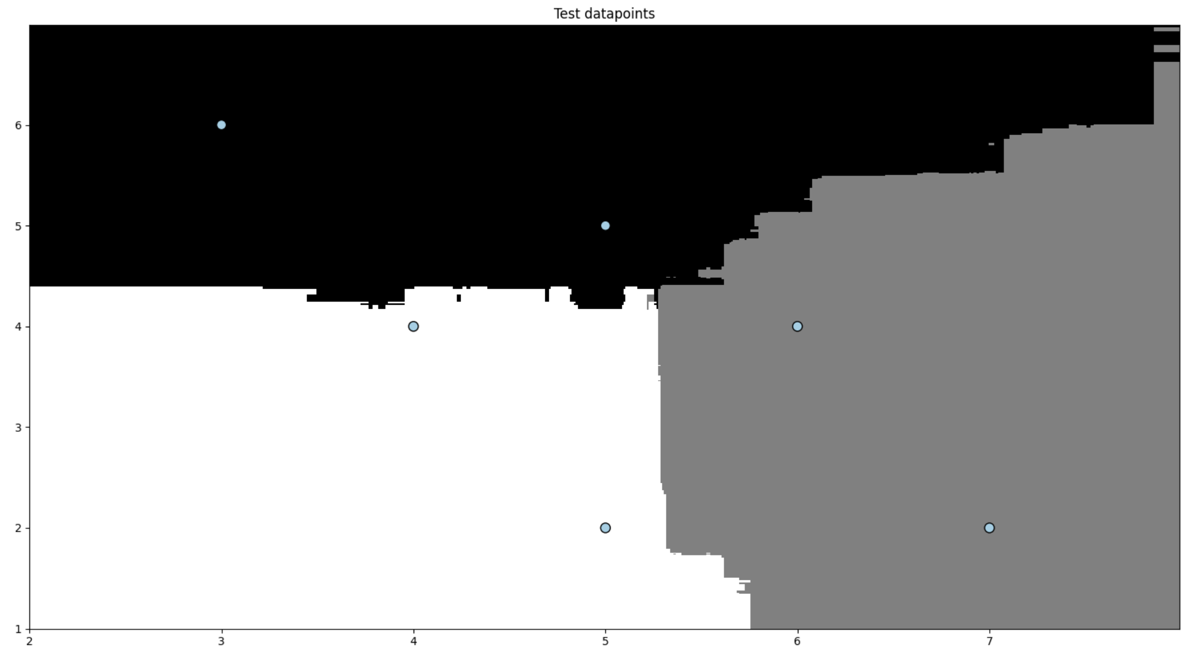

Training dataset
9
8
7
6
5
4
3
2
1
0
−1
−2
−1
0
1
2
3
4
5
6
7
8
9
10




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

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

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

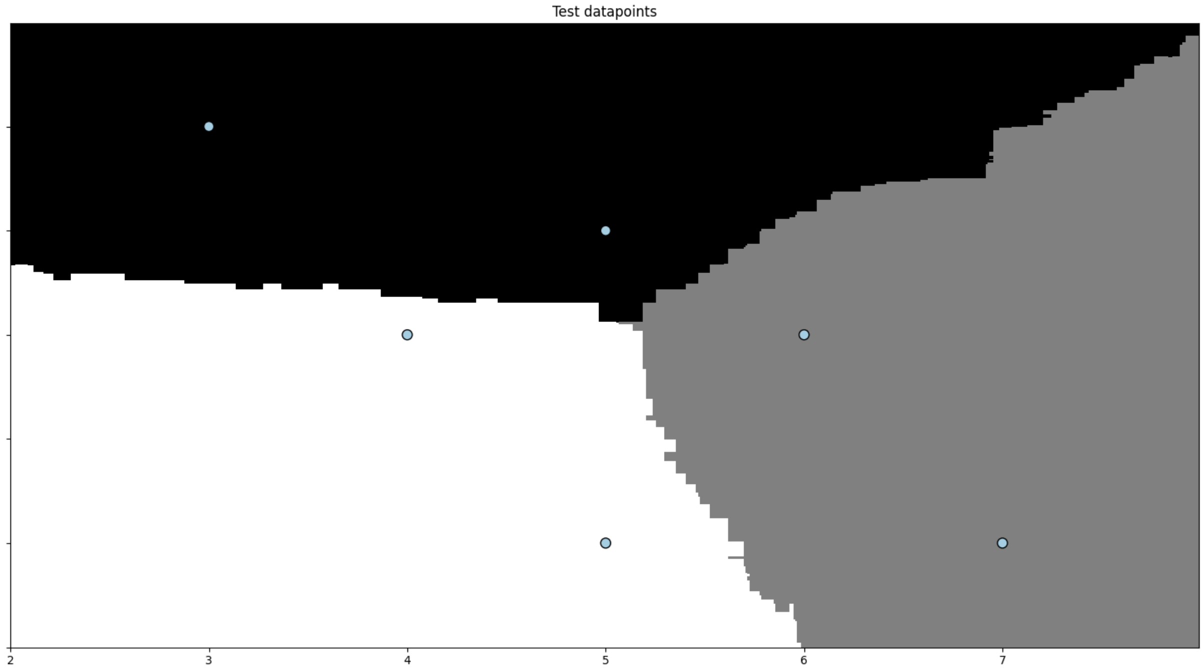
Оцінка мір достовірності прогнозу

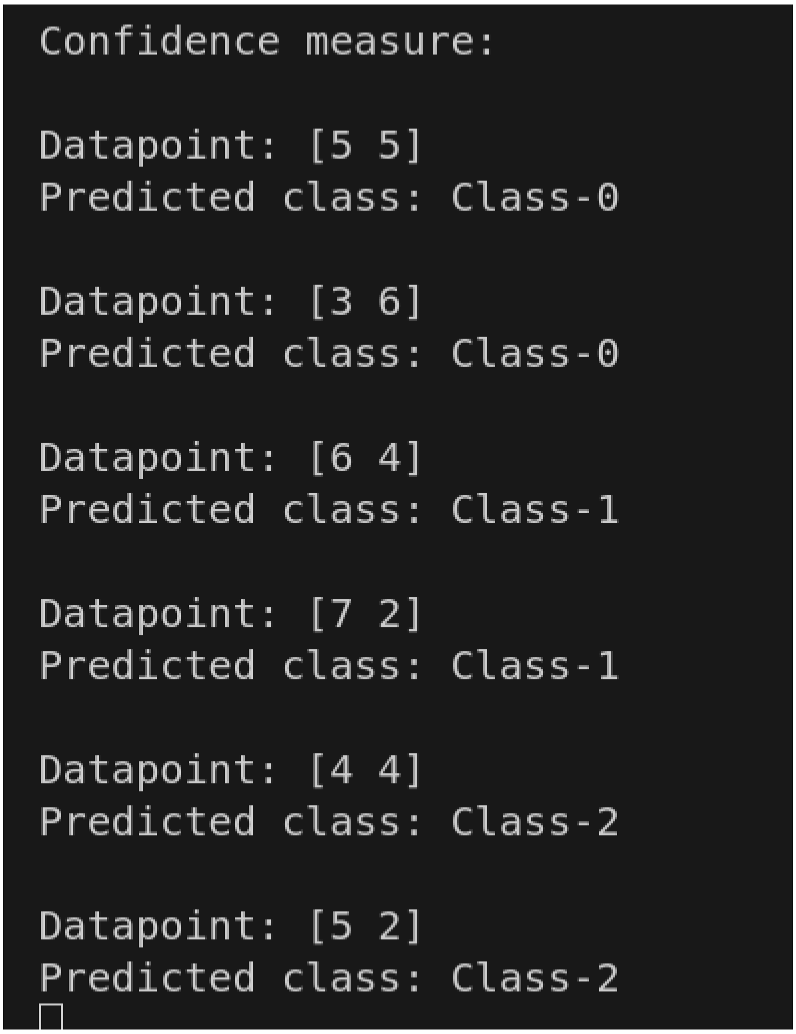
- для rf




```
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
```

- Для 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
```

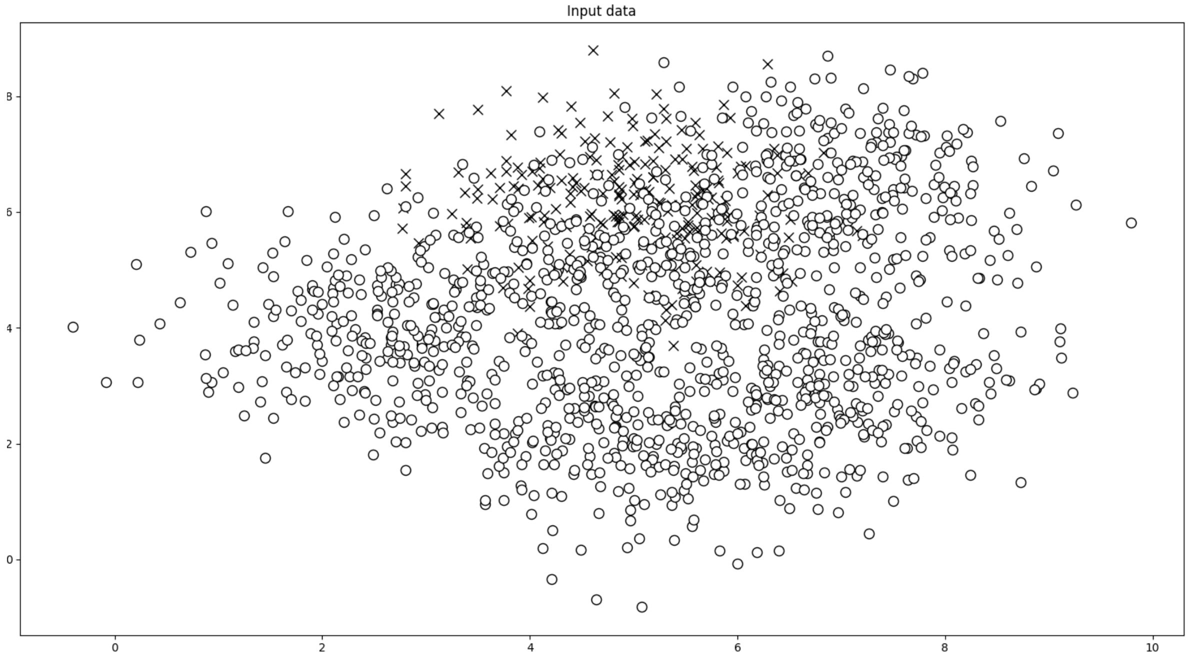
Висновок: У цьому коді розглядається класифікація даних за допомогою двох різних моделей ансамблевого навчання: Random Forest (RF) та Extra Trees Classifier (ERF). Використано набір даних з трьома класами, які представлені двома вимірюваннями для кожного з об'єктів. Моделі тренуються на навчальних даних і перевіряються на тестових даних. Для кожної моделі також виводиться класифікаційний звіт (classification report), який включає точність, відчутливість і специфічність для кожного класу.

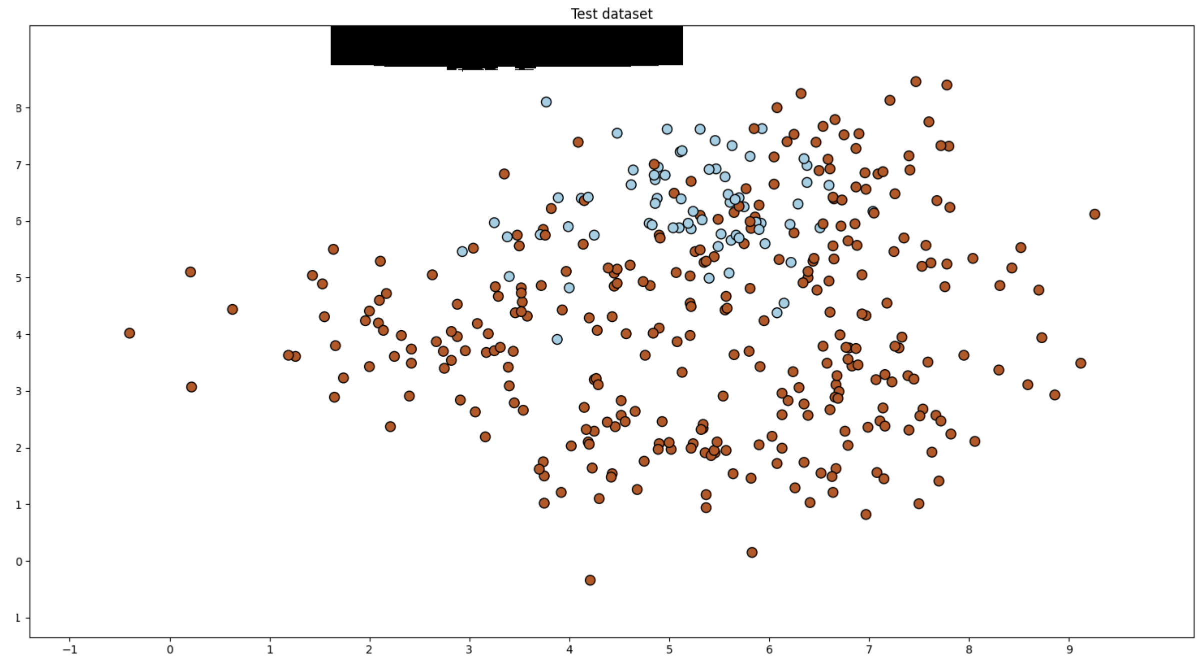
Додатково, перевіряється здатність класифікаторів до прогнозування класу для кількох нових тестових точок і демонструється візуалізація результатів. Це дає змогу оцінити, як кожен з класифікаторів працює не лише на тренувальних і тестових даних, але й на нових, раніше не бачених даних.

Завдання 2.2. Обробка дисбалансу класів

```
import sys
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.model_selection import train_test_split, cross_val_score
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 = 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:
```

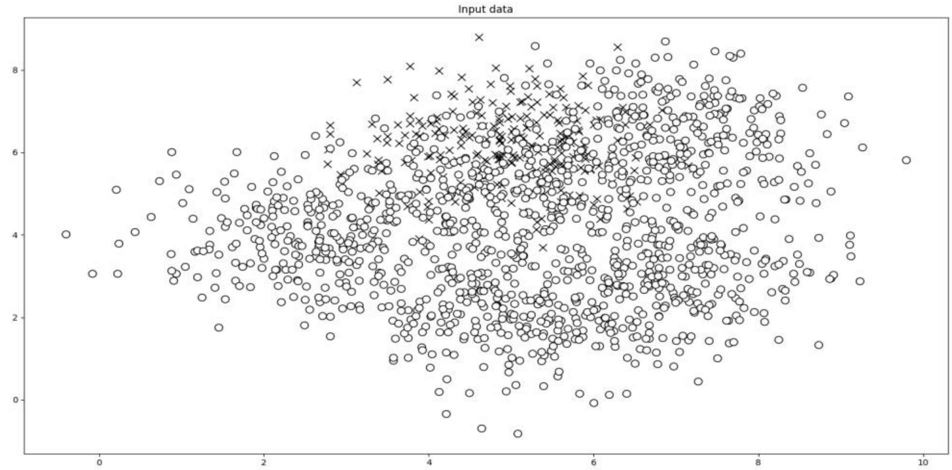
```
    if sys.argv[1] == 'balance':
        params = {'n_estimators': 100, 'max_depth': 4, 'random_state': 0, 
'class_weight': 'balanced'}
    else:
        raise TypeError("Invalid input argument; should be 'balance'")

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()
```

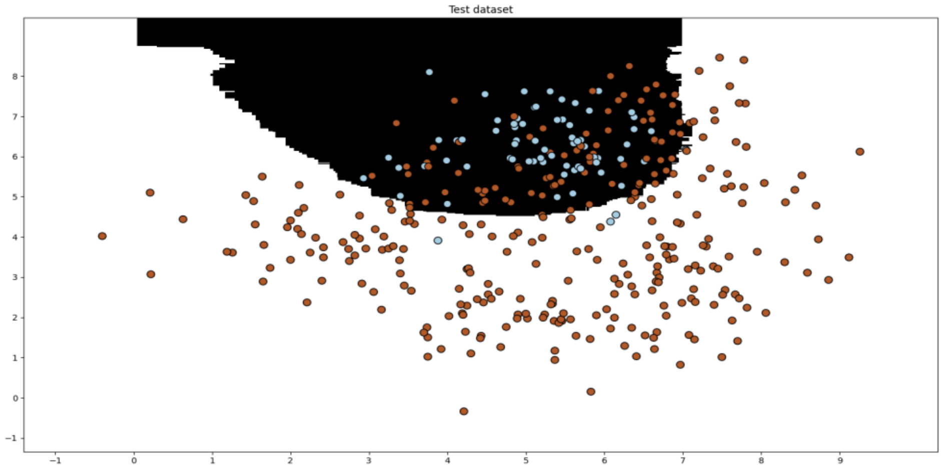


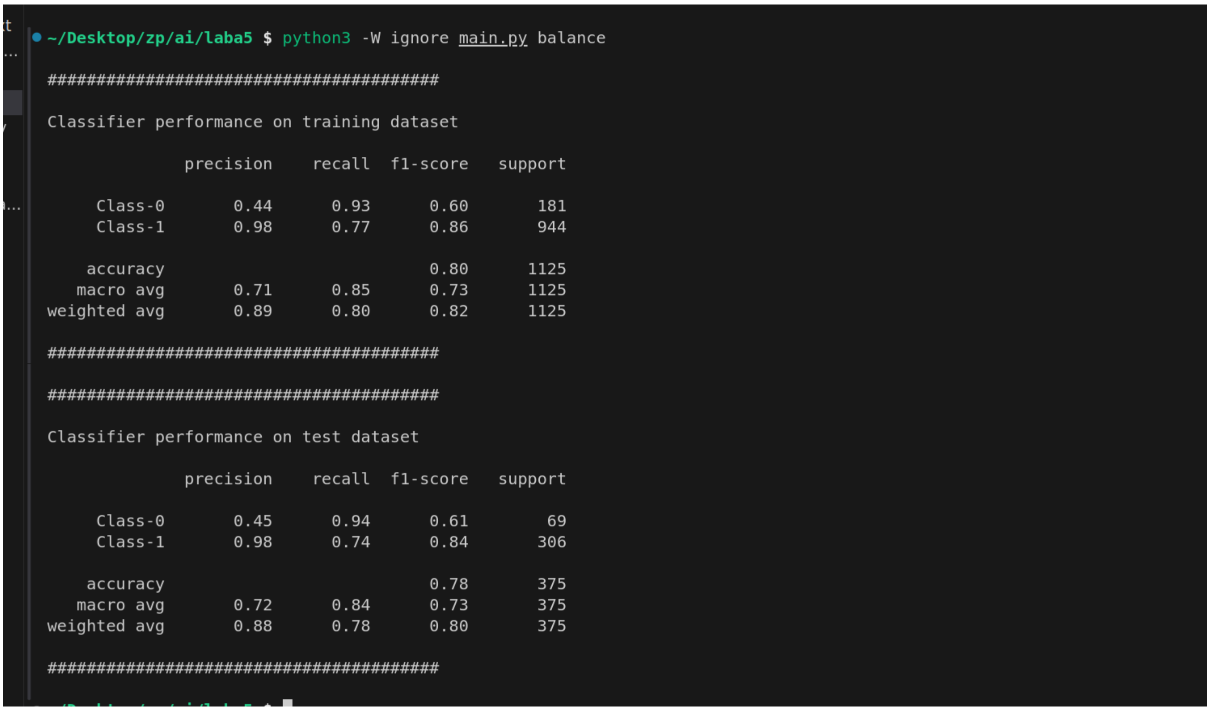



```
Try `python -h' for more information.
~/Desktop/zp/ai/laba5 $ python3 -W ignore main.py

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

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

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

3 balance







```
~/Desktop/zp/ai/laba5 $ python3 -W ignore main.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

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

Знаходження оптимальних навчальних параметрів за допомогою сіткового пошуку

Завдання 2.3. Знаходження оптимальних навчальних параметрів за допомогою сіткового пошуку

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.model_selection import train_test_split, cross_val_score,
GridSearchCV
from sklearn.metrics import classification_report
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:")
    results = classifier.cv_results_
    for mean_score, params in zip(results['mean_test_score'],
results['params']):
        print(params, '-->', round(mean_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))
```

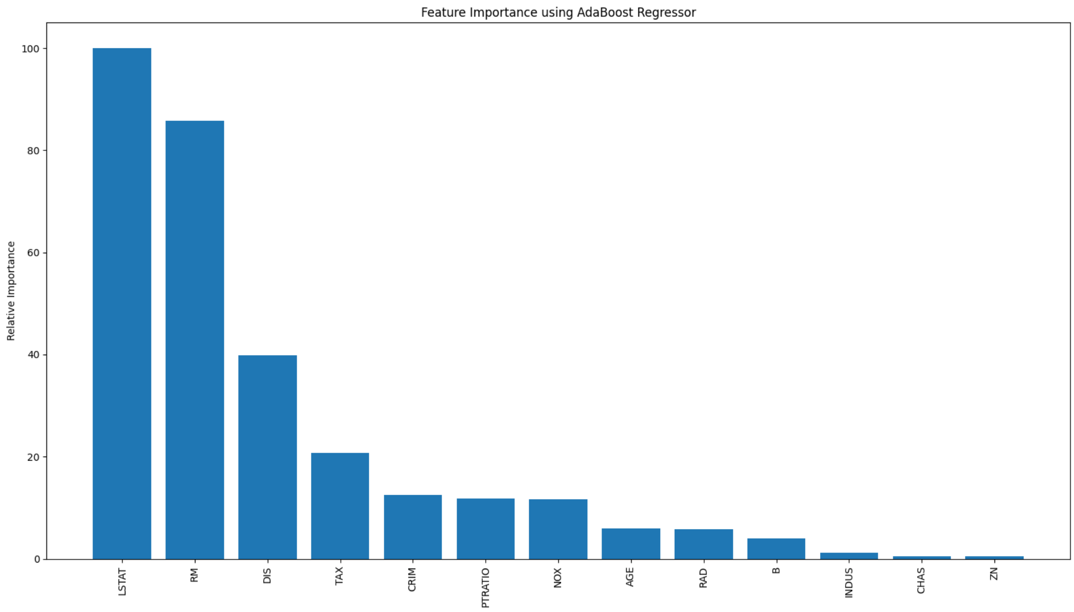
```
● ~/Desktop/zp/ai/laba5 $ python3 main.py

 #### Searching optimal parameters for precision_weighted
 Params: {'max_depth': 2, 'n_estimators': 100} --> Mean score: 0.85
 Params: {'max_depth': 4, 'n_estimators': 100} --> Mean score: 0.841
 Params: {'max_depth': 7, 'n_estimators': 100} --> Mean score: 0.844
 Params: {'max_depth': 12, 'n_estimators': 100} --> Mean score: 0.832
 Params: {'max_depth': 16, 'n_estimators': 100} --> Mean score: 0.816
 Params: {'max_depth': 4, 'n_estimators': 25} --> Mean score: 0.846
 Params: {'max_depth': 4, 'n_estimators': 50} --> Mean score: 0.84
 Params: {'max_depth': 4, 'n_estimators': 100} --> Mean score: 0.841
 Params: {'max_depth': 4, 'n_estimators': 250} --> Mean score: 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
 Params: {'max_depth': 2, 'n_estimators': 100} --> Mean score: 0.843
 Params: {'max_depth': 4, 'n_estimators': 100} --> Mean score: 0.837
 Params: {'max_depth': 7, 'n_estimators': 100} --> Mean score: 0.841
 Params: {'max_depth': 12, 'n_estimators': 100} --> Mean score: 0.83
```

```
          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
 Params: {'max_depth': 2, 'n_estimators': 100} --> Mean score: 0.843
 Params: {'max_depth': 4, 'n_estimators': 100} --> Mean score: 0.837
 Params: {'max_depth': 7, 'n_estimators': 100} --> Mean score: 0.841
 Params: {'max_depth': 12, 'n_estimators': 100} --> Mean score: 0.83
 Params: {'max_depth': 16, 'n_estimators': 100} --> Mean score: 0.815
 Params: {'max_depth': 4, 'n_estimators': 25} --> Mean score: 0.843
 Params: {'max_depth': 4, 'n_estimators': 50} --> Mean score: 0.836
 Params: {'max_depth': 4, 'n_estimators': 100} --> Mean score: 0.837
 Params: {'max_depth': 4, 'n_estimators': 250} --> Mean score: 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

○ ~/Desktop/zp/ai/laba5 $
```

Завдання 2.4. Обчислення відносної важливості ознак

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

```python
# Load dataset
data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
target = raw_df.values[1::2, 2]

# Shuffle and split dataset
X, y = shuffle(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)

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

# Predictions and evaluation
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 importance visualization
feature_importances = regressor.feature_importances_
feature_names = [
    "CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM",
    "AGE", "DIS", "RAD", "TAX", "PTRATIO", "B", "LSTAT"
]

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('Feature Importance using AdaBoost Regressor')
plt.show()
```




```
ADABOOST REGRESSOR
Mean squared error = 22.7
Explained variance score = 0.79
```

Можна знехтувати ZN, CHAS, INDUS

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

```
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, cross_val_score
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.metrics import classification_report
from sklearn import preprocessing

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 = []
```

```python
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_test_pred = regressor.predict(X_test)
print("Mean absolute error:", round(mean_absolute_error(y_test, y_test_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]]))
        count += 1

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

```
~/Desktop/zp/ai/laba5 $ python3 -W ignore main.py
Mean absolute error: 7.42
Predicted traffic: 26
```

Посилання на GitHub: https://github.com/missShevel/SHI_Shevel_Olha_IPZ-21-1/tree/master/Lab5