НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря Сікорського»
«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

ЗВІТ

про виконання комп'ютерного практикуму № 5

з дисципліни «Інтелектуальний аналіз даних»

Виконав:

Студент III курсу

Групи КА-13

Приймак Є.О.

Варіант № 25

Перевірила:

Недашківська Н. І.

Київ – 2023 рік

Тема: Побудова та оцінювання ансамблів моделей класифікації та регресії з використанням бібліотеки Scikit-Learn Python

Завдання (25):

BaggingClassifier. Розглянути різні значення параметрів learning_rate та algorithm.

Початкові дані:

(a) sklearn.datasets.samples_generator.make_circles

X, y = make_circles (noise =0.2, factor=0.5, random_state=1)

Виконання:

1. Початкові 2D-дані представити графічно.

2. Розбити дані на навчальний, перевірочний та тестовий набори. Перевірочний набір використати для налаштування гіперпараметрів. Тестовий набір використати для остаточної оцінки якості моделей.

3. Побудувати ансамблі моделей, використовуючи наступні методи (згідно з варіантом):

• **BaggingClassifier**. Розглянути різні значення max_samples, bootstrap, n_estimators.

В ансамблях **Bagging**:

• В якості base_estimator/estimators використати одну/ кілька моделей із параметрами по умовчанню: дерев рішень, логістичної регресії, svm тощо.

• Побудувати графіки залежності значень показника якості ансамбля та індивідуальної моделі від n_estimators на одній координатній вісі. Такий графік для індивідуальної моделі, очевидно, буде горизонтальною прямою. В задачах класифікації в якості показника якості можна обрати accuracy_score, f1_score або zero_one_loss.

• Оцінити якість ансамблю на основі прикладів oob (для ансамблів на основі беггінгу).

4. В задачах класифікації навести приклад границі рішень decision boundaries на основі окремої моделі та ансамблю.

5. Розрахувати значення зміщення та дисперсії для окремої моделі та ансамблю.

6. Що можна сказати про час навчання ансамблю порівняно з окремими моделями, які утворюють ці ансамблі?

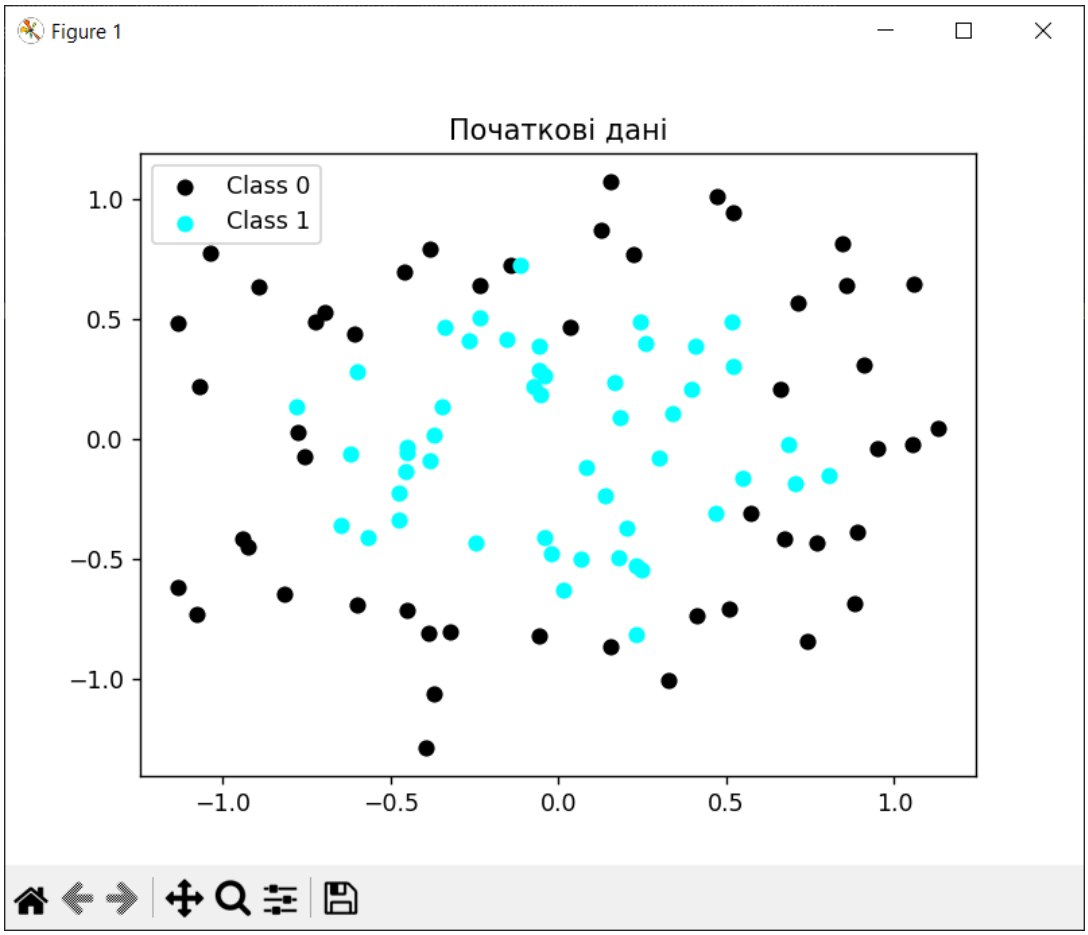
7. Зробити висновки. Чи краще на заданих даних виконується ансамбль порівняно з індивідуальними моделями?

Хід роботи:

1.Початкові 2D-дані представимо графічно:

```
import matplotlib.pyplot as plt
import numpy as np
import time
from sklearn.datasets import make_circles
from sklearn.model_selection import train_test_split
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import GridSearchCV

# Початкові дані
X, y = make_circles(noise=0.2, factor=0.5, random_state=1)

# 1. Початкові 2D-дані представити графічно.
plt.title('Початкові дані')
plt.scatter(X[y == 0][:, 0], X[y == 0][:, 1], c='black', label='Class
0')
plt.scatter(X[y == 1][:, 0], X[y == 1][:, 1], c='aqua', label='Class
1')
plt.legend()
plt.show()
```

Отримуємо:



2. Розіб’ємо дані на навчальний, перевірочний та тестовий набори:

```
# 2. Розбиття даних на навчальний, перевірочний та тестовий набори.
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train,
test_size=0.25, random_state=42)
```

Підбираємо гіперпараметри на перевірочному наборі:

```
# Пошук оптимальних гіперпараметрів базової моделі на перевірочному
наборі
param_grid = {'max_depth': [3, 5, 7], 'min_samples_split': [2, 5, 10]}
base_model = DecisionTreeClassifier()

grid_search = GridSearchCV(base_model, param_grid, cv=3)
grid_search.fit(X_train, y_train)

# Виведення найкращих гіперпараметрів
print("Best parameters found on validation set:")
print(grid_search.best_params_)
```

Маємо:

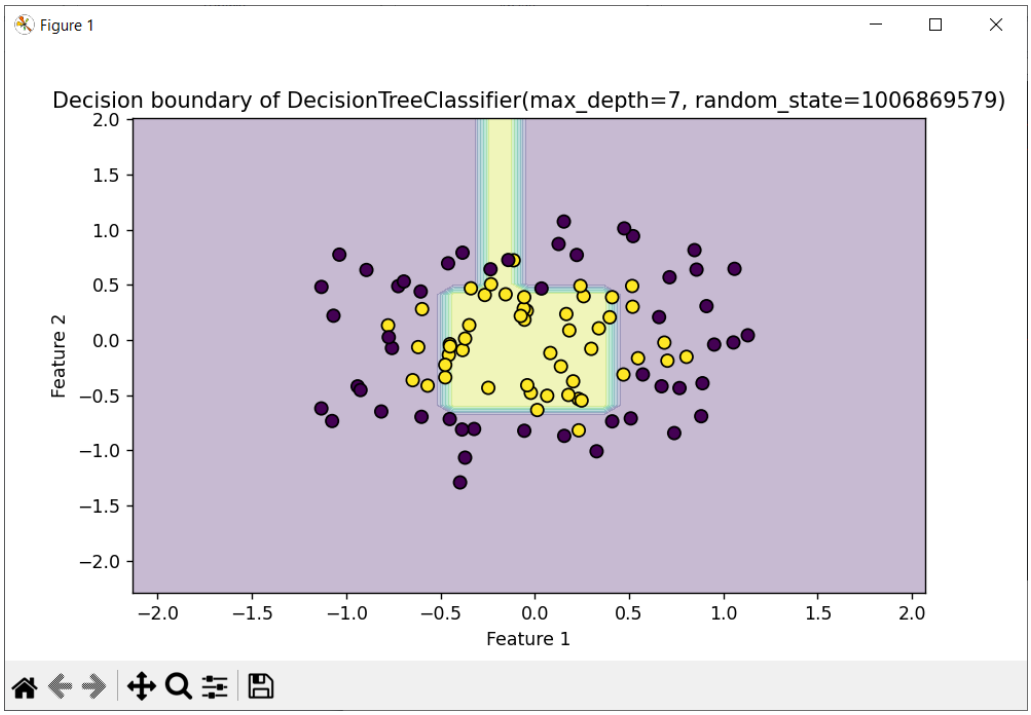
```
"E:\PyCharm\PyCharm projects\venv\Scripts\python.exe" "E:\PyCharm\PyCharm projects\IAD_Lab5.py"
Best parameters found on validation set:
{'max_depth': 7, 'min_samples_split': 2}
```

3+4+5+6. Побудуємо ансамбль BaggingClassifier, візуалізаємо границі рішень для нього, окремої моделі, та OOB-ансамбля, який йому відповідає:

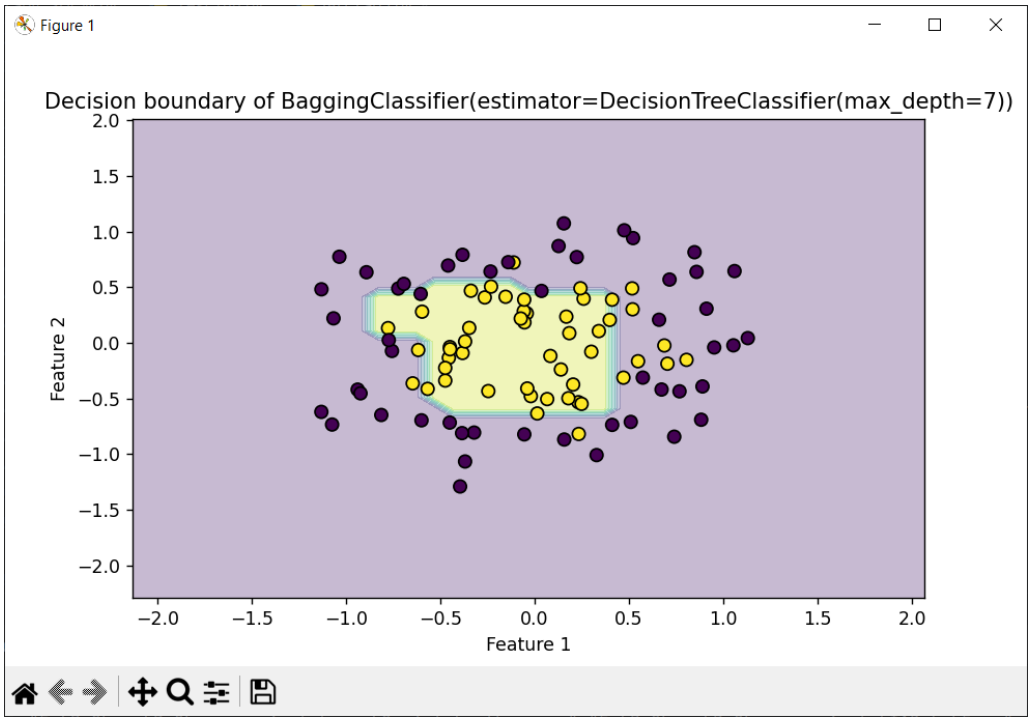
```
# 3-6. Побудова ансамблів BaggingClassifier та оцінка якості.

# Використання найкращих гіперпараметрів у моделі для ансамблю
best_params = grid_search.best_params_
best_base_model = DecisionTreeClassifier(**best_params)

# Побудова ансамблю
bagging_clf = BaggingClassifier(estimator=best_base_model,
n_estimators=10)
bagging_clf.fit(X_train, y_train)

# Побудова ансамблю з оцінкою якості на основі OOB
bagging_clf_extra = BaggingClassifier(estimator=best_base_model,
n_estimators=10, oob_score=True)
bagging_clf_extra.fit(X_train, y_train)


# Візуалізація класифікації дефолтної моделі та ансамблю для
початкових даних
def plot_decision_boundary(model, X, y):
    # Створення області
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                         np.arange(y_min, y_max, 0.1))

    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.contourf(xx, yy, Z, alpha=0.3)
    plt.scatter(X[:, 0], X[:, 1], c=y, s=50, edgecolor='k')
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.title(f'Decision boundary of {model}')
    plt.show()

base_model = bagging_clf.estimators_[0]
plot_decision_boundary(base_model, X, y)
plot_decision_boundary(bagging_clf, X, y)
plot_decision_boundary(bagging_clf_extra, X, y)
```

Отримуємо:

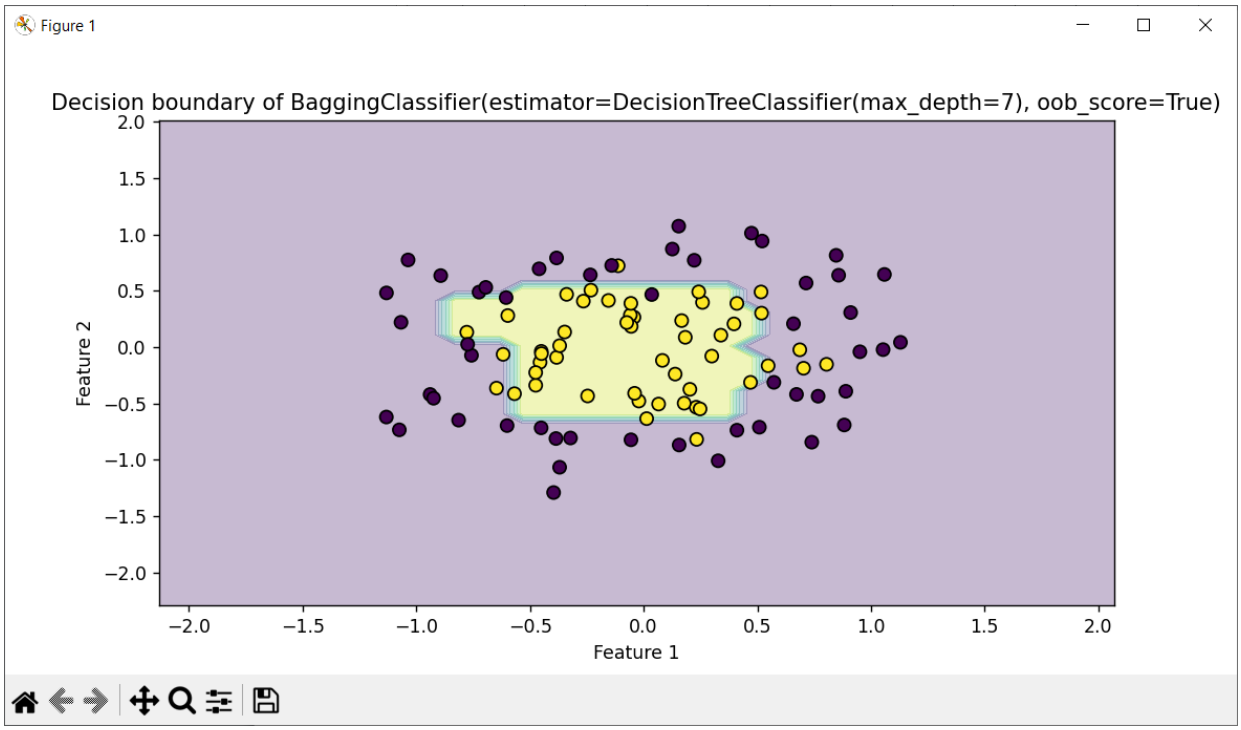
(окрема модель з ансамблю):



(дефолтний ансамбль):



(OOB ансамбль):



Тепер оцінимо якість окремої моделі та ансамблів.

Для отримання часу та дисперсії нам знадобляться функції:

```
# Розрахунок зміщення та дисперсії для окремої моделі та ансамблю
def calculate_bias_variance(model, X, y):
    y_pred = model.predict(X)
    # Знаходимо середнє значення прогнозу
    avg_y_pred = np.mean(y_pred, axis=0)
    # Знаходимо зміщення та дисперсію
    bias = np.mean((y - avg_y_pred) ** 2)
    variance = np.mean(np.var(y_pred, axis=0))
    return bias, variance


# Розрахунок часу навчання для окремої моделі та ансамблю
def calculate_training_time(model, X, y):
    start_time = time.time()
    model.fit(X, y)
    end_time = time.time()
    return end_time - start_time
```

Обчислення метрик якості:

```
# Оцінка якості окремої модельки
y_pred_val1 = (bagging_clf.estimators_[0]).predict(X_val)
accuracy_1 = accuracy_score(y_val, y_pred_val1)
f1_1 = f1_score(y_val, y_pred_val1)
bias1, variance1 = calculate_bias_variance(base_model, X_val, y_val)
training_time1 = calculate_training_time(base_model, X_train, y_train)
print(f'\nModel accuracy on validation set: {accuracy_1}')
```

```
print(f'Model F1 score on validation set: {f1_1}')
print(f'Bias for base model: {bias1:.4f}, Variance for base model:
{variance1:.4f}')
print(f'Training time for base model: {training_time1:.4f} seconds')

# Оцінка якості дефолтного ансамблю
y_pred_val2 = bagging_clf.predict(X_val)
accuracy_2 = accuracy_score(y_val, y_pred_val2)
f1_2 = f1_score(y_val, y_pred_val2)
bias2, variance2 = calculate_bias_variance(bagging_clf, X_val, y_val)
training_time2 = calculate_training_time(bagging_clf, X_train,
y_train)
print(f'\nDefault ensemble Accuracy on validation set
(n_estimators=10): {accuracy_2}')
print(f'Default ensemble F1 score on validation set (n_estimators=10):
{f1_2}')
print(f'Bias for ensemble: {bias2:.4f}, Variance for ensemble:
{variance2:.4f}')
print(f'Training time for ensemble: {training_time2:.4f} seconds')

# Оцінка якості ансамблю на основі OOB
y_pred_val3 = bagging_clf_extra.predict(X_val)
accuracy_3 = accuracy_score(y_val, y_pred_val3)
f1_3 = f1_score(y_val, y_pred_val3)
oob_score = bagging_clf_extra.oob_score_
print(f'\nOOB ensemble Accuracy on validation set (n_estimators=10):
{accuracy_3}')
print(f'OOB ensemble F1 score on validation set (n_estimators=10):
{f1_3}')
print(f'Out-of-Bag Score: {oob_score:.4f}')
```

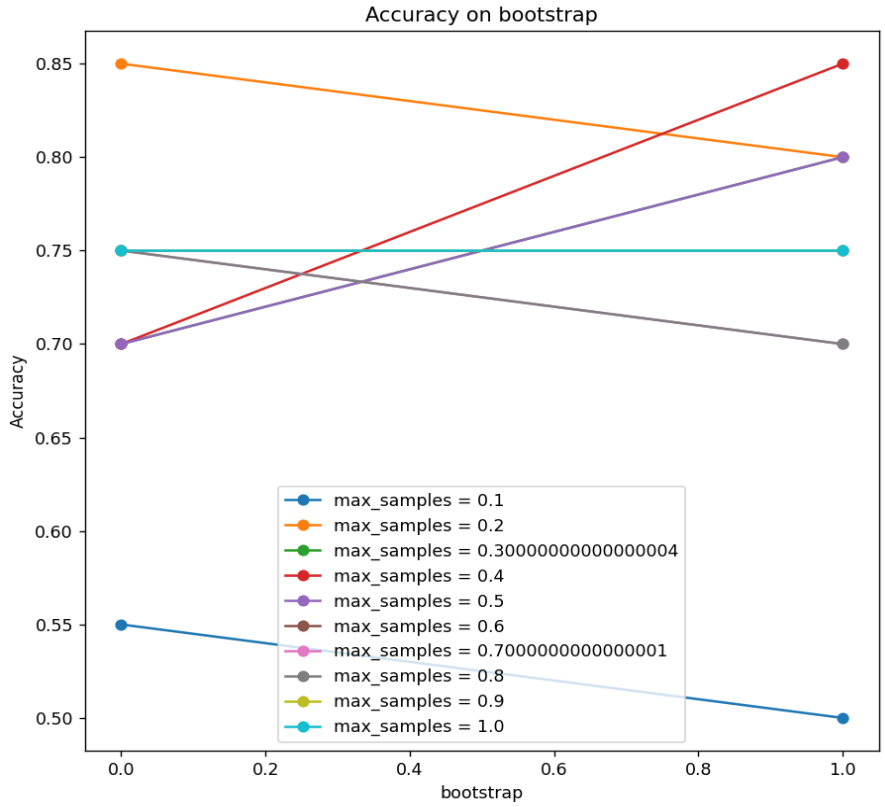
Отримуємо:

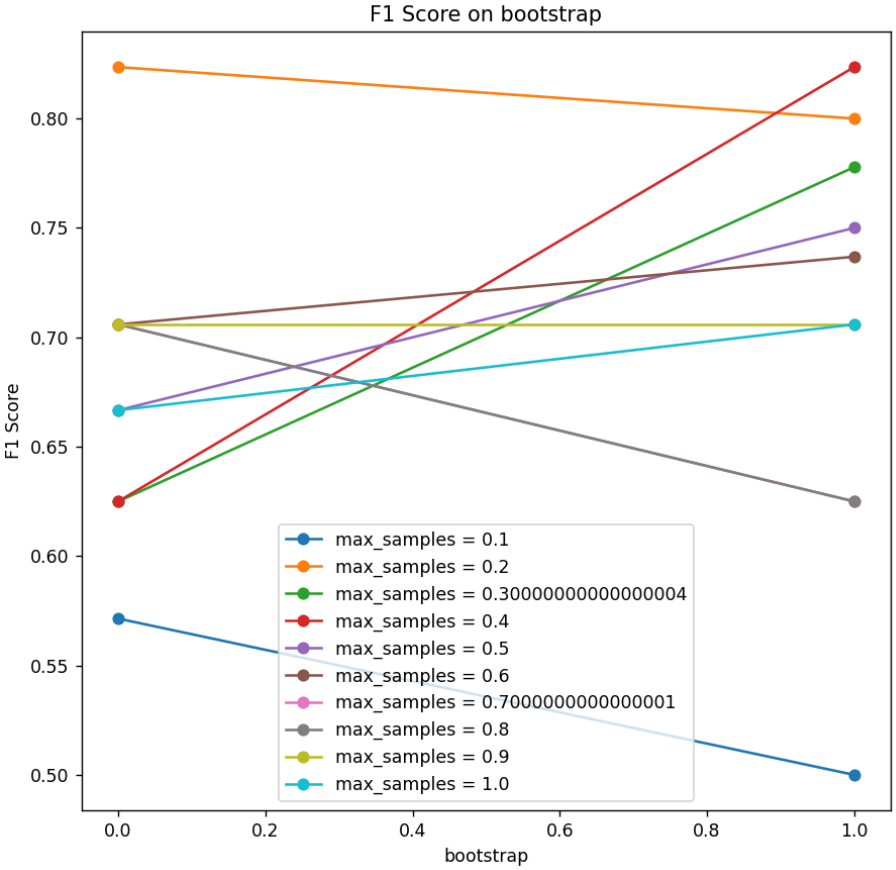
```
Model accuracy on validation set: 0.8
Model F1 score on validation set: 0.7777777777777777
Bias for base model: 0.3400, Variance for base model: 0.1600
Training time for base model: 0.0030 seconds

Default ensemble Accuracy on validation set (n_estimators=10): 0.7
Default ensemble F1 score on validation set (n_estimators=10): 0.625
Bias for ensemble: 0.2900, Variance for ensemble: 0.2100
Training time for ensemble: 0.0210 seconds

OOB ensemble Accuracy on validation set (n_estimators=10): 0.7
OOB ensemble F1 score on validation set (n_estimators=10): 0.625
Out-of-Bag Score: 0.8000
```

Продемонструємо вплив параметрів **max_samples, bootstrap**, на метрики точності та F1 для ансамблю:

```
# Змінні для зберігання метрик
accuracy_scores = []
f1_scores = []
max_samples_range = np.arange(0.1, 1.1, 0.1)  # Значення для
max_samples
bootstrap_options = [True, False]  # Опції для bootstrap

# Цикл для max_samples
for max_samples in max_samples_range:
    accuracy_scores_for_max_samples = []
    f1_scores_for_max_samples = []

    # Цикл для bootstrap
    for bootstrap in bootstrap_options:
        bagging_clf = BaggingClassifier(estimator=best_base_model,
n_estimators=10, max_samples=max_samples,
                                        bootstrap=bootstrap)
        bagging_clf.fit(X_train, y_train)
        y_pred = bagging_clf.predict(X_val)

        accuracy = accuracy_score(y_val, y_pred)
        f1 = f1_score(y_val, y_pred)

        accuracy_scores_for_max_samples.append(accuracy)
        f1_scores_for_max_samples.append(f1)

    accuracy_scores.append(accuracy_scores_for_max_samples)
    f1_scores.append(f1_scores_for_max_samples)

# Побудова графіків
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
for i, accuracy_scores_for_max_samples in enumerate(accuracy_scores):
    plt.plot(bootstrap_options, accuracy_scores_for_max_samples,
marker='o',
             label=f'max_samples = {max_samples_range[i]}')
plt.xlabel('bootstrap')
plt.ylabel('Accuracy')
plt.title('Accuracy on bootstrap')
plt.legend()

plt.subplot(1, 2, 2)
for i, f1_scores_for_max_samples in enumerate(f1_scores):
    plt.plot(bootstrap_options, f1_scores_for_max_samples, marker='o',
label=f'max_samples = {max_samples_range[i]}')
plt.xlabel('bootstrap')
plt.ylabel('F1 Score')
plt.title('F1 Score on bootstrap')
plt.legend()
plt.tight_layout()
plt.show()
```

Отримуємо:

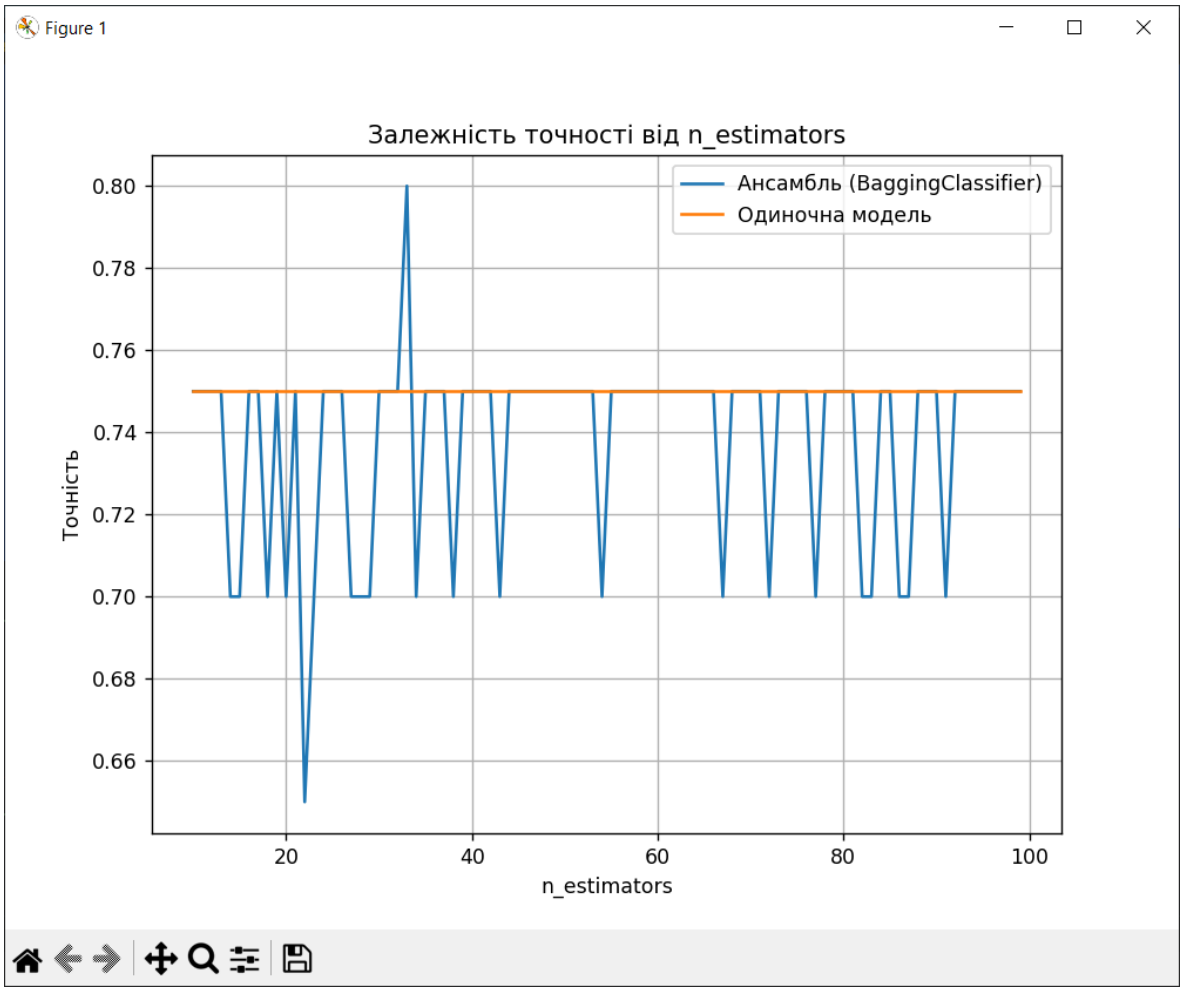

Accuracy on bootstrap
Accuracy
bootstrap
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.0
0.2
0.4
0.6
0.8
1.0
max_samples = 0.1
max_samples = 0.2
max_samples = 0.30000000000000004
max_samples = 0.4
max_samples = 0.5
max_samples = 0.6
max_samples = 0.7000000000000001
max_samples = 0.8
max_samples = 0.9
max_samples = 1.0


F1 Score on bootstrap
F1 Score
bootstrap
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.0
0.2
0.4
0.6
0.8
1.0
max_samples = 0.1
max_samples = 0.2
max_samples = 0.30000000000000004
max_samples = 0.4
max_samples = 0.5
max_samples = 0.6
max_samples = 0.7000000000000001
max_samples = 0.8
max_samples = 0.9
max_samples = 1.0

Оцінимо залежність точності ансамблю від параметру **n_estimators**:

```
# Діапазон n_estimators
n_estimators_range = range(10, 100)
scores_ensemble = []
scores_base_model = []

for n_estimators in n_estimators_range:
    # Створення ансамблю з BaggingClassifier
    bagging_clf_test = BaggingClassifier(estimator=best_base_model,
n_estimators=n_estimators)
    bagging_clf_test.fit(X_train, y_train)

    # Оцінка точності ансамблю
    y_val_pred = bagging_clf_test.predict(X_val)
    ensemble_accuracy = accuracy_score(y_val, y_val_pred)
    scores_ensemble.append(ensemble_accuracy)

    # Оцінка точності окремої моделі
    best_base_model.fit(X_train, y_train)
    y_val_pred_base = best_base_model.predict(X_val)
    base_accuracy = accuracy_score(y_val, y_val_pred_base)
    scores_base_model.append(base_accuracy)

# Візуалізація результатів
plt.figure(figsize=(8, 6))
plt.plot(n_estimators_range, scores_ensemble, label='Ансамбль
(BaggingClassifier)')
plt.plot(n_estimators_range, scores_base_model, label='Одиночна
модель')
plt.xlabel('n_estimators')
plt.ylabel('Точність')
plt.title('Залежність точності від n_estimators')
plt.legend()
plt.grid(True)
plt.show()
```

Отримуємо:



В даному випадку найвище значення точності досягається для n_estimators=33

7. З огляду отриманих результатів, можна зазначити, що окрема модель мала трохи більшу точність за ансамбль з 10 таких моделей на заданому наборі даних, хоча його застосування і дозволило згладити показник зміщення класифікації. Також слід відмітити, що час навчання ансамблю зайняв в 7 разів більше, ніж окремої його моделі

Висновки: У ході виконання комп’ютерного практикуму було отримано навички побудови та оцінювання ансамблів моделей класифікації та регресії з використанням бібліотеки Scikit-Learn Python, досліджено вплив різних параметрів (max_samples, bootstrap, n_estimators) на точність ансамбля для заданого набору даних, проведено порівняльний аналіз роботи ансамбля та його окремої моделі, а також перевірено альтернативний ансамбль OOB-класифікації.