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 № 3

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: Класифікація та регресія на основі багатошарового персептрона в Scikit-Learn Python

Мета: побудова та оцінювання якості моделей для класифікації засобами бібліотеки Scikit-Learn Python (MLPClassifier), та використання багатошарового персептрона.

Завдання:

Початкові дані:

(а) `from sklearn.datasets.samples_generator import make_blobs X, y_true = make_blobs (n_samples =400, centers =4, cluster _std =0.60 , random_state=0)`

```
rng = np.random.RandomState(13)

X_stretched = np.dot(X, rng.randn (2 , 2))
```

(б)

```
np.random.seed(1)
feature1 = np.random.randint(0, 10, size=300)
feature2 = np.random.randint(0, 10, size=300)
X2 = np.column_stack((feature1, feature2))
Y2 = np.random.randint(0, 2, size=300)
```

Початкові дані (два набори) і Хід виконання роботи такі ж як для Практикуму №2.
Використовувати sklearn.neural_network.MLPClassifier, якщо в роботі №2 була класифікація.

Починати з одношарової моделі нейронної мережі і визначити чи достатньо буде одношарової моделі для опису даних. Реалізуйте динамічне додавання нейронів до скритого шару. Перевірте скільки нейронів в одношаровій моделі буде достатньо для задовільного розв'язання задачі.

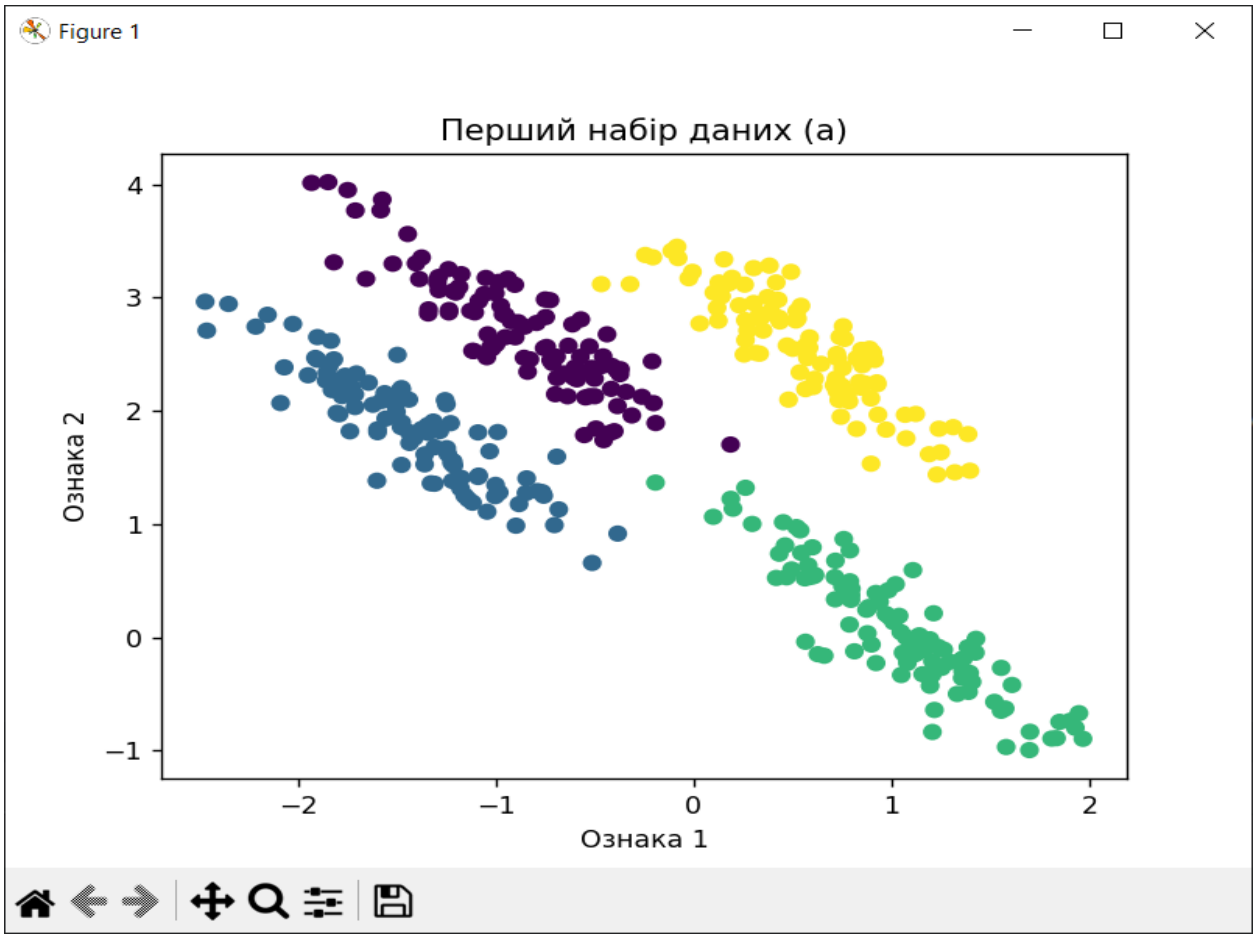
Хід роботи:

Згенеруємо наш набір даних (а) та підключимо необхідні інструменти з бібліотеки Scikit-Learn Python:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_blobs
from sklearn.naive_bayes import GaussianNB, MultinomialNB
from sklearn.metrics import confusion_matrix, precision_score,
```

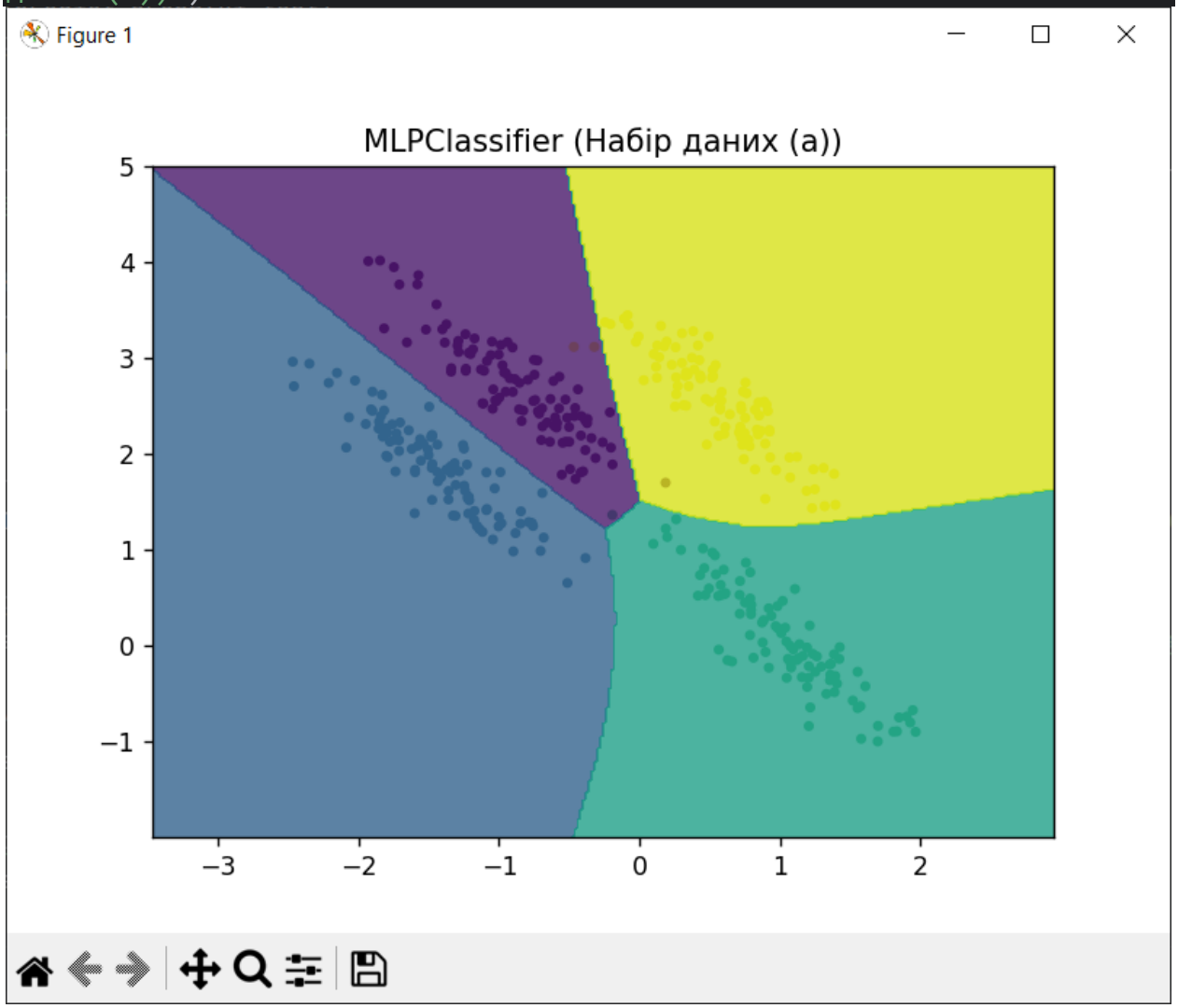
```
recall_score, f1_score, roc_curve, auc, precision_recall_curve,
accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.neural_network import MLPClassifier

# Перший набір даних (a)
X1, y1 = make_blobs(n_samples=400, centers=4, cluster_std=0.60,
random_state=0)
rng = np.random.RandomState(13)
X1_stretched = np.dot(X1, rng.randn(2, 2))


# Перший набір даних (a): Представлення початкових даних графічно
plt.scatter(X1_stretched[:, 0], X1_stretched[:, 1], c=y1,
cmap='viridis')
plt.xlabel("Ознака 1")
plt.ylabel("Ознака 2")
plt.title("Перший набір даних (a)")
plt.show()
```

Набір (a):

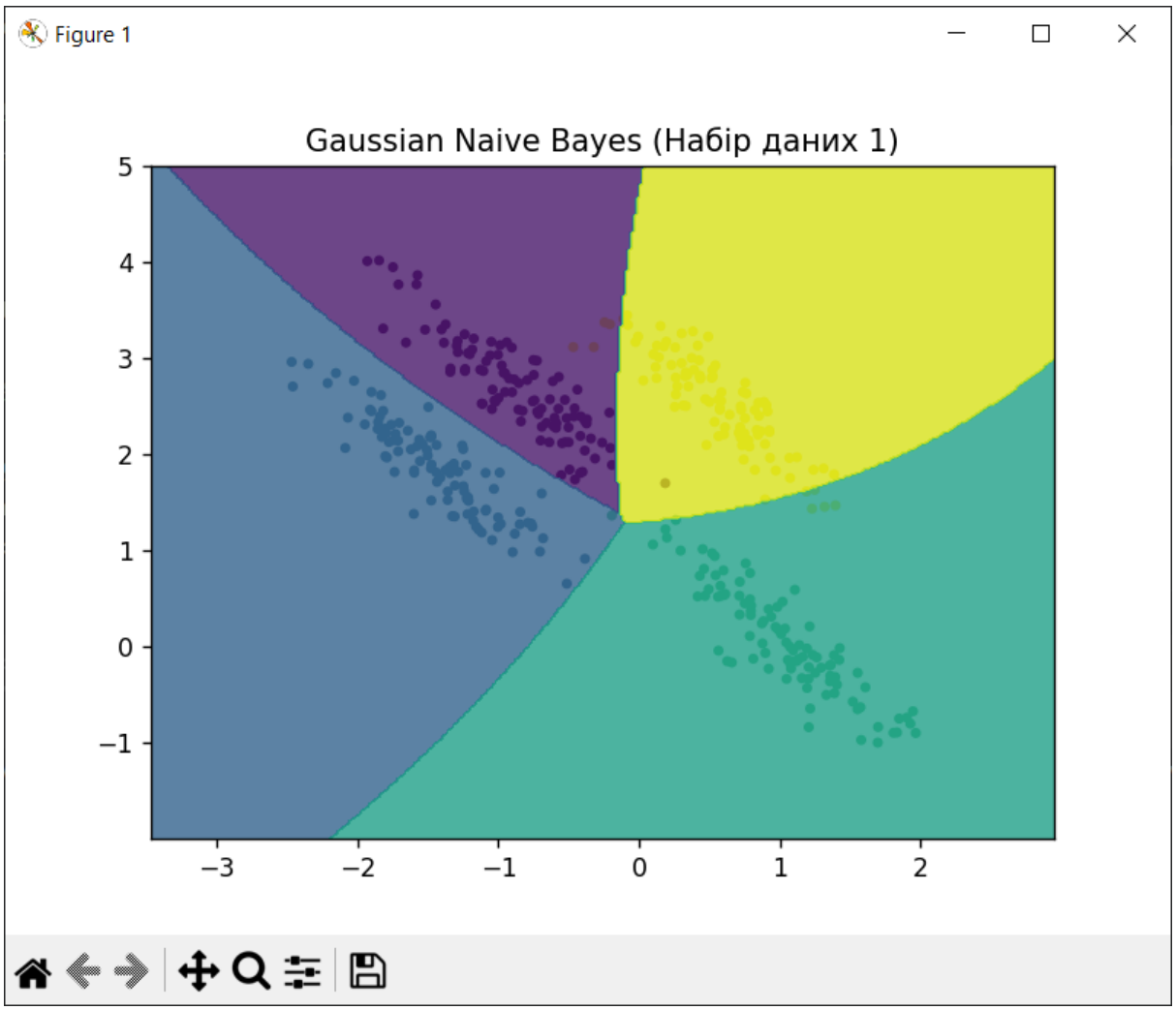


Поділимо згенерований набір (a) на навчальний та валідаційний, створимо та застосуємо модель і візуалізуємо границі рішень:

```python
# Поділ набору (a) на навчальний та валідаційний
X1_train, X1_test, y1_train, y1_test = train_test_split(X1_stretched,
y1, test_size=0.2, random_state=42)

Model1=MLPClassifier(random_state=1)
Model1.fit(X1_train, y1_train)

# Візуалізація границь рішень для MLPClassifier (перший набір даних)
plot_decision_boundary(Model1, X1_stretched, y1, "MLPClassifier (Набір
даних (a))")
```



Згадаємо, що в практикумі 2 методом GaussianNB були отримані трохи інші границі рішень цього ж набору:



Вже тут помітні переваги має MLPClassifier, хоча обидва досить добре виконують поставлену задачу класифікації.

Зробимо прогноз набору (а) та обчислимо апостеріорну ймовірність:

```
# Прогноз
y1_pred_M1 = Model1.predict(X1_test)
print("\nПрогноз навчальний набору(а):", y1_pred)


# Розрахунок додаткових результатів
probs1_M1 = Model1.predict_proba(X1_test)
#print("\nАпостеріорна ймовірність:", probs1_M1)
```

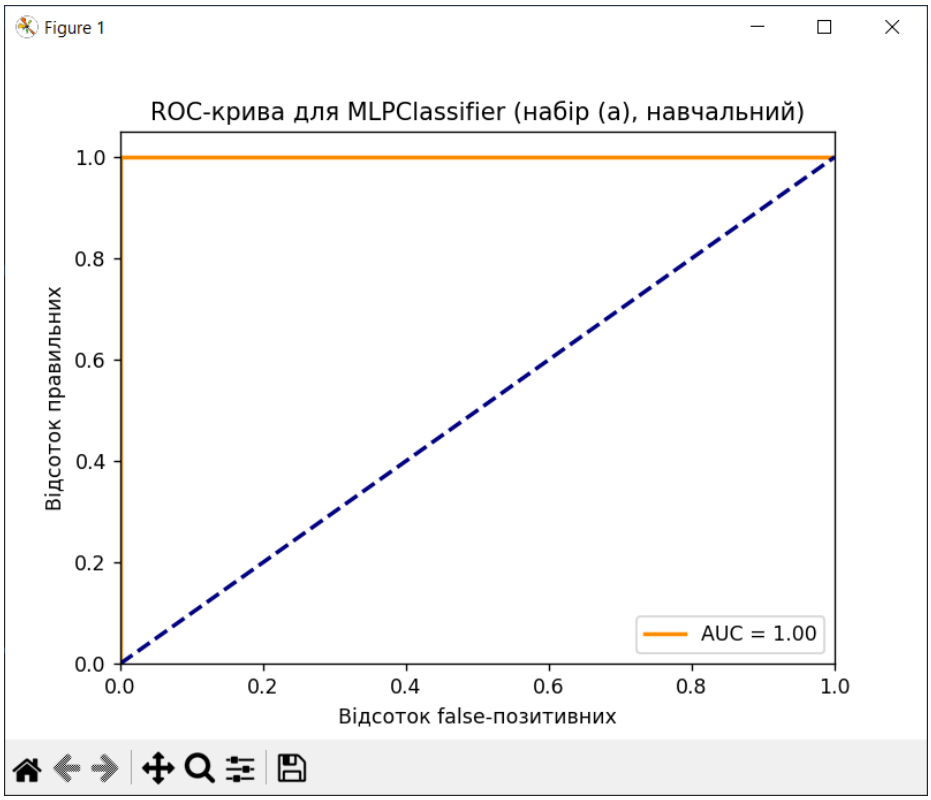
```
Прогноз навчальний набору(а): [3 3 0 0 2 0 2 2 0 1 1 3 0 2 3 1 1 3 2 2 2 1 3 0 0 0 3 2 2 2 0 2 2 1 0 2 2
 1 2 2 3 2 0 2 1 0 1 3 3 1 1 2 0 2 2 0 2 3 0 1 0 2 3 3 1 0 3 0 1 3 1 0 3 3
 3 3 3 3 0 0]
```

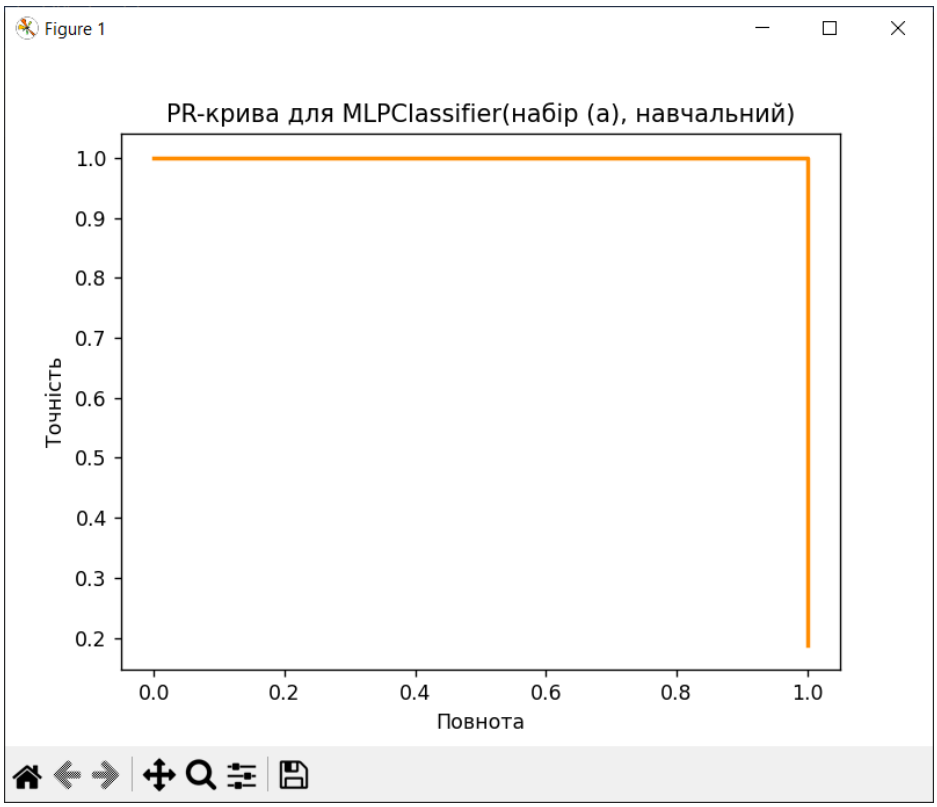
Розрахуємо критерії якості нашої моделі та візуалізуємо ROC і PR криві для тренувального набору (а):

```
# Розрахунок критеріїв якості
cm1_M1, precision1_M1, recall1_M1, f1_M1, fpr1_M1, tpr1_M1, auc1_M1,
pr1_M1 = calculate_metrics(y1_test, y1_pred_M1, probs1_M1)
print("\nМатриця неточностей (Confusion Matrix) для MLPClassifier
(навчальний набір):")
print(cm1_M1)
print(f"Точність: {precision1_M1:.2f}")
print(f"Повнота: {recall1_M1:.2f}")
print(f"F1-міра: {f1_M1:.2f}")
print(f"AUC: {auc1_M1:.2f}")

# Візуалізація ROC-кривих
plot_roc_curve(fpr1_M1, tpr1_M1, auc1_M1, "ROC-крива для MLPClassifier
(набір (а), навчальний)")

#Візуалізація PR-кривих
plot_precision_recall_curve(pr1_M1[0], pr1_M1[1], "PR-крива для
MLPClassifier(набір (а), навчальний)")
```

Отримуємо:

```
Матриця неточностей (Confusion Matrix) для MLPClassifier (навчальний набір):
[[20  0  0  0]
 [ 0 15  0  0]
 [ 0  0 20  0]
 [ 1  0  0 24]]
Точність: 0.99
Повнота: 0.99
F1-міра: 0.99
AUC: 1.00
```



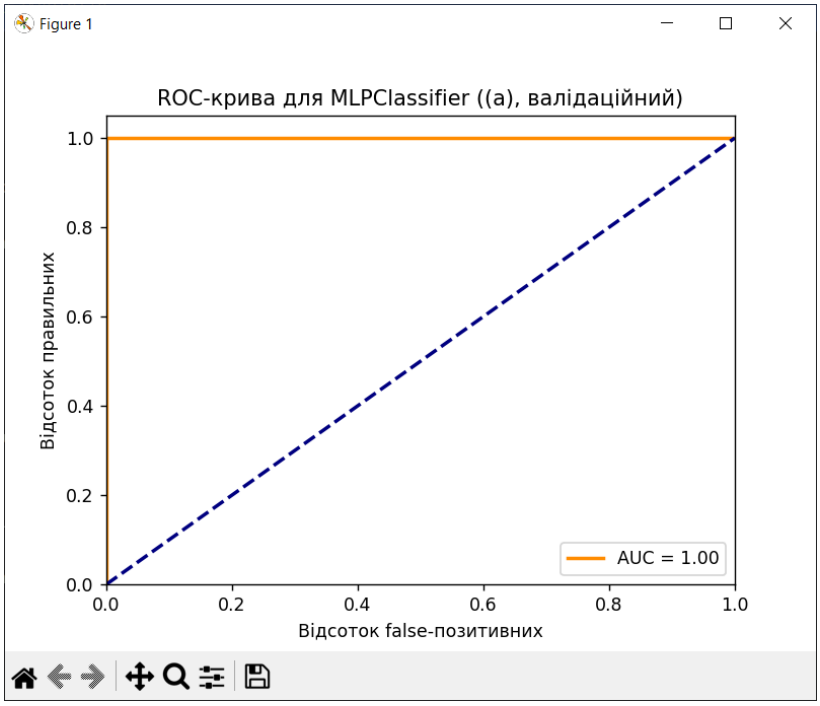


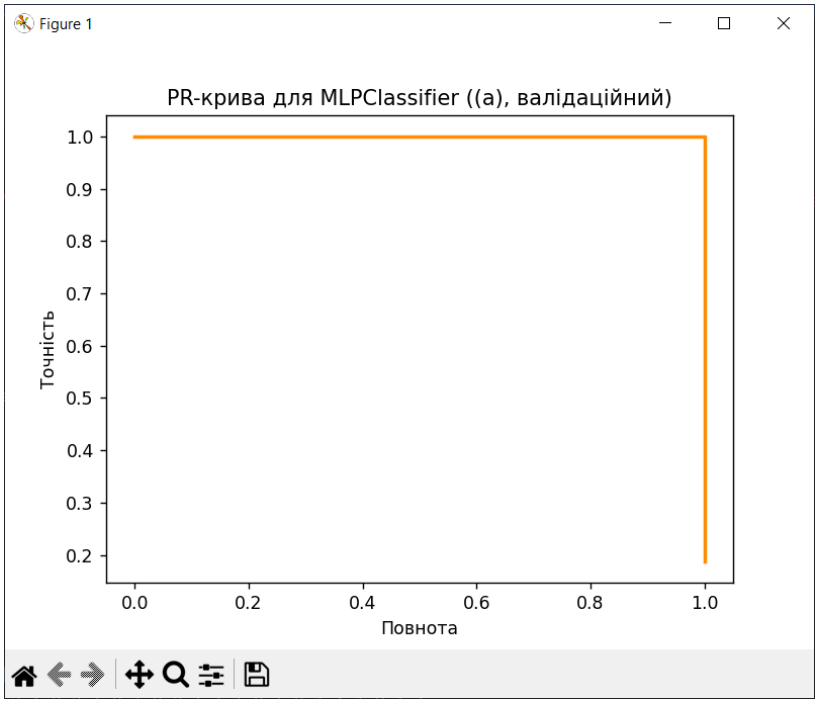
Аналогічні розрахунки робимо для застосованого на моделі валідаційного набору:

```
mlp_M1_v = MLPClassifier()
# Навчання отриманої моделі з валідаційними параметрами
mlp_M1_v.fit(X1_train, y1_train)
# Оцінка моделі з валідаційними параметрами
y1_pred_M1 = mlp_M1_v.predict(X1_test)

print("\nПрогноз валідаційний набору(а):", y1_pred_M1)
probs1_M1_v = mlp_M1_v.predict_proba(X1_test)

cm1_M1_v, precision1_M1_v, recall1_M1_v, f1_M1_v, fpr1_M1_v,
tpr1_M1_v, auc1_M1_v, pr1_M1_v = calculate_metrics(
    y1_test, y1_pred_M1, probs1_M1_v)

print("Матриця неточностей (Confusion Matrix) для MLPClassifier
((а),валідаційна):")
print(cm1_M1_v)
print(f"Точність: {precision1_M1_v:.2f}")
print(f"Повнота: {recall1_M1_v:.2f}")
print(f"F1-міра: {f1_M1_v:.2f}")
print(f"AUC: {auc1_M1_v:.2f}")

plot_roc_curve(fpr1_M1_v, tpr1_M1_v, auc1_M1_v, "ROC-крива для
MLPClassifier ((а), валідаційний)")
plot_precision_recall_curve(pr1_M1_v[0], pr1_M1_v[1], "PR-крива для
MLPClassifier ((а), валідаційний)")
```

Отримуємо:

```
Прогноз валідаційний набору(a): [3 3 0 0 2 0 2 2 0 1 1 3 0 2 3 1 1 3 2 3 2 1 3 0 0 0 3 2 2 2 0 2 2 1 0 3 2
 1 2 2 3 2 0 3 1 0 1 3 3 1 1 2 0 2 3 0 2 3 0 1 0 2 3 3 1 0 3 0 1 3 1 0 3 3
 3 3 3 3 0 0]
Матриця неточностей (Confusion Matrix) для MLPClassifier ((a),валідаційна):
[[20  0  0  0]
 [ 0 15  0  0]
 [ 0  0 19  1]
 [ 1  0  0 24]]
Точність: 0.98
Повнота: 0.97
F1-міра: 0.97
AUC: 1.00
```





Бачимо, що на валідаційному наборі точність моделі трохи падає, що очікувано, але загалом результати класифікації дуже точні, про що свідчать ROC і PR- криві.

Можна порівняти результати з методом GaussianNB з минулої роботи:

(навчальний набір (а))

```
Матриця неточностей (Confusion Matrix) для Gaussian Naive Bayes (навчальний набір):
[[20  0  0  0]
 [ 0 15  0  0]
 [ 0  0 20  0]
 [ 1  0  3 21]]
Точність: 0.96
Повнота: 0.95
F1-міра: 0.95
AUC: 1.00
```

(валідаційний набір (а))

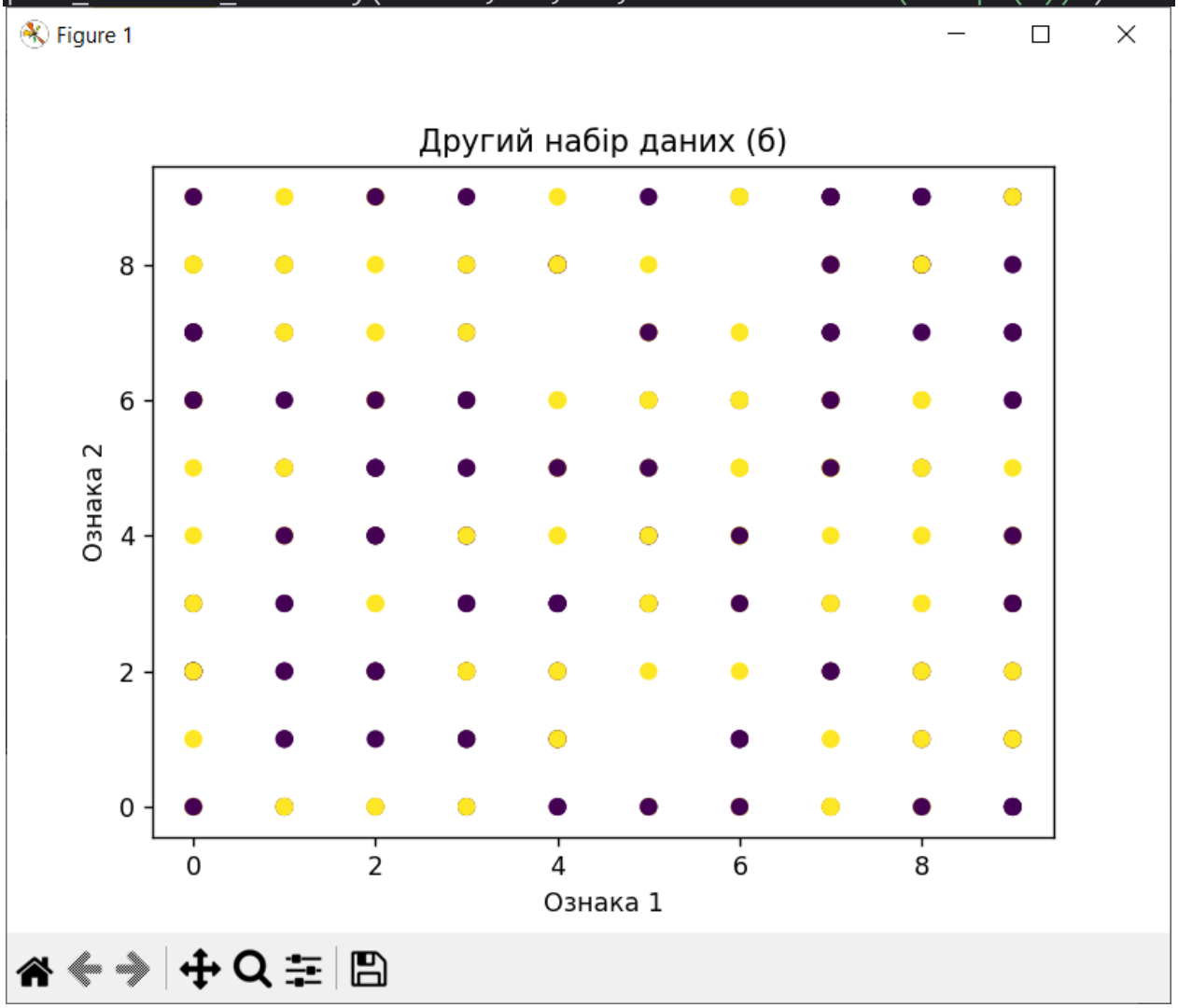
```
Прогноз валідаційний набору(а): [3 3 0 0 2 0 2 2 0 1 1 3 0 2 3 1 1 3 2 3 2 1 3 0 0 0 3 2 2 2 0 2 2 1 0 2 2
 1 2 2 3 2 1 3 1 0 1 3 3 1 1 2 0 2 2 0 2 3 0 1 0 2 3 3 1 0 3 0 1 3 1 0 3 3
 3 3 3 3 0 0]
Матриця неточностей (Confusion Matrix) для Gaussian Naive Bayes (з найкращими параметрами):
[[19  1  0  0]
 [ 0 15  0  0]
 [ 0  0 19  1]
 [ 1  0  2 22]]
Точність: 0.94
Повнота: 0.94
F1-міра: 0.94
AUC: 1.00
```

Отже MLPClassifier отримує більшу точність як на тренувальному, так і тестовому наборах.

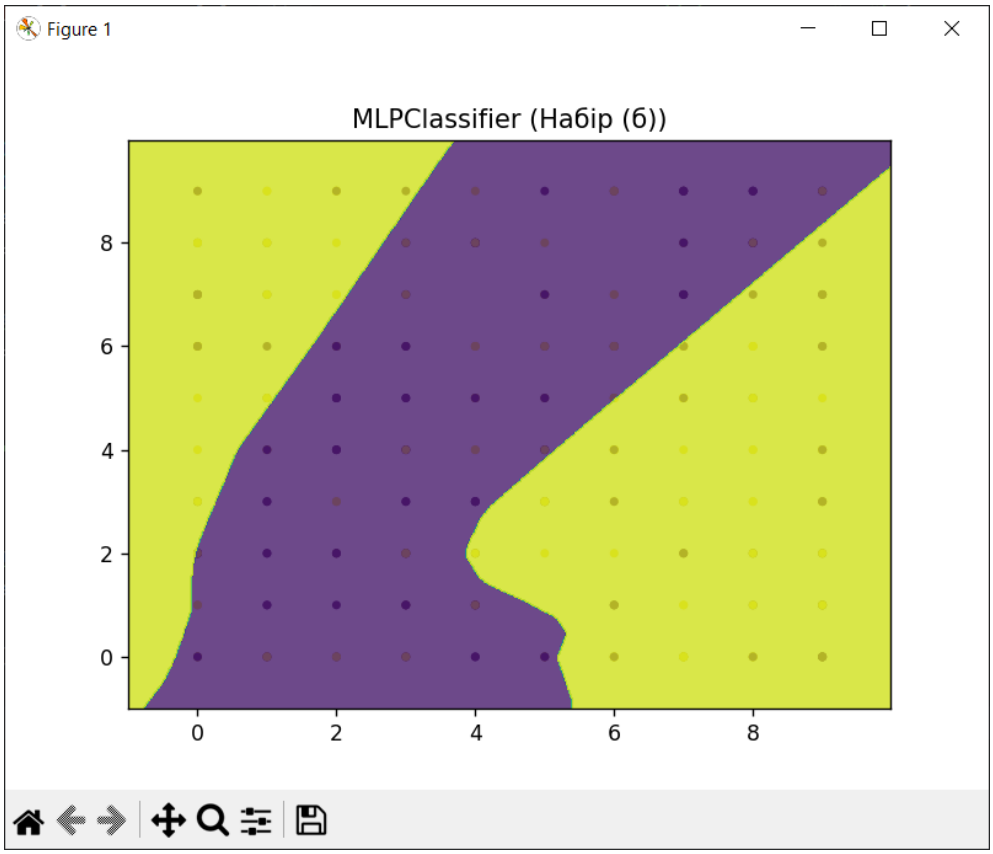
Аналогічно створимо нову MLP модель та дамо їй опрацювати згенерований набір (б), який також розіб’ємо на тренувальний та тестовий:

```python
# Другий набір даних (б)
np.random.seed(1)
feature1 = np.random.randint(0, 10, size=300)
feature2 = np.random.randint(0, 10, size=300)
X2 = np.column_stack((feature1, feature2))
Y2 = np.random.randint(0, 2, size=300)

# Другий набір даних (б): Представлення початкових даних графічно
plt.scatter(X2[:, 0], X2[:, 1], c=Y2, cmap='viridis')
plt.xlabel("Ознака 1")
plt.ylabel("Ознака 2")
plt.title("Другий набір даних (б)")
plt.show()
```

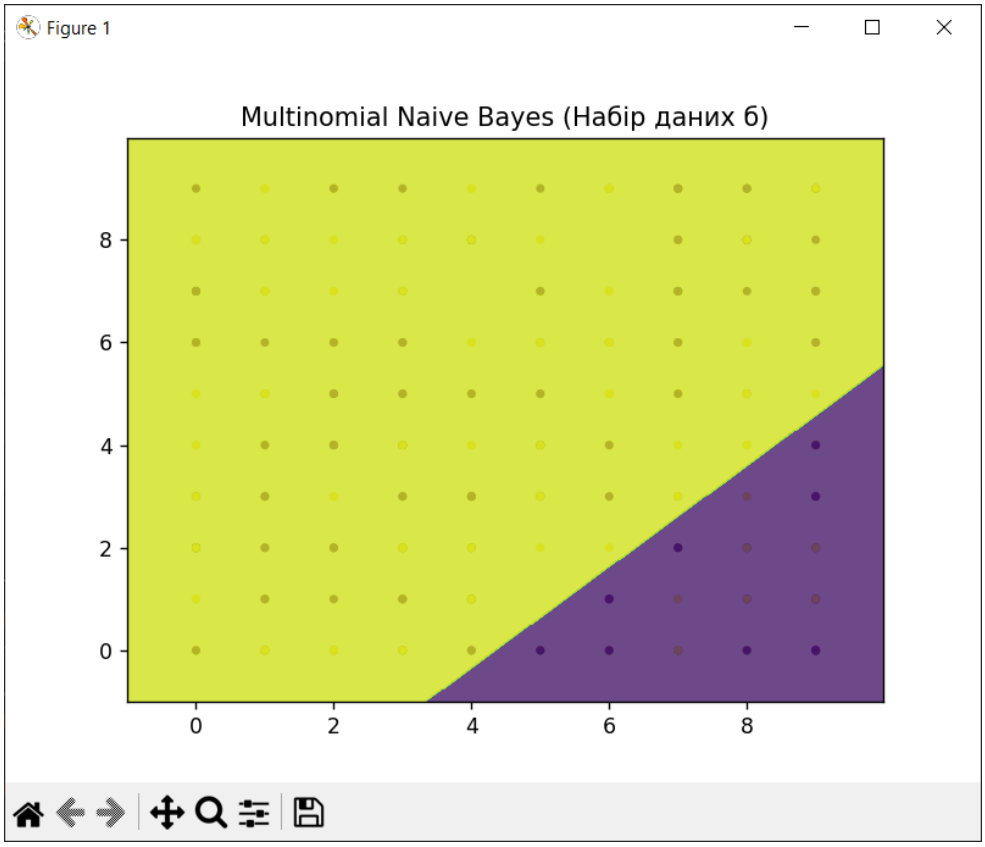
```python
# Поділ набору даних (б) на навчальний та валідаційний
X2_train, X2_test, y2_train, y2_test = train_test_split(X2, Y2,
test_size=0.2, random_state=42)

# MLPClassifier для набору даних (б)
Model2 = MLPClassifier()
Model2.fit(X2_train, y2_train)

# Візуалізація границь рішень для MLPClassifier (б)
plot_decision_boundary(Model2, X2, Y2, "MLPClassifier (Набір (б))")
```



Візуалізуємо границі рішень MLP класифікатора:



Згадаємо, що попередній роботі з MultinomialNB для цього ж набору було отримано менш точніші рішення класифікації:

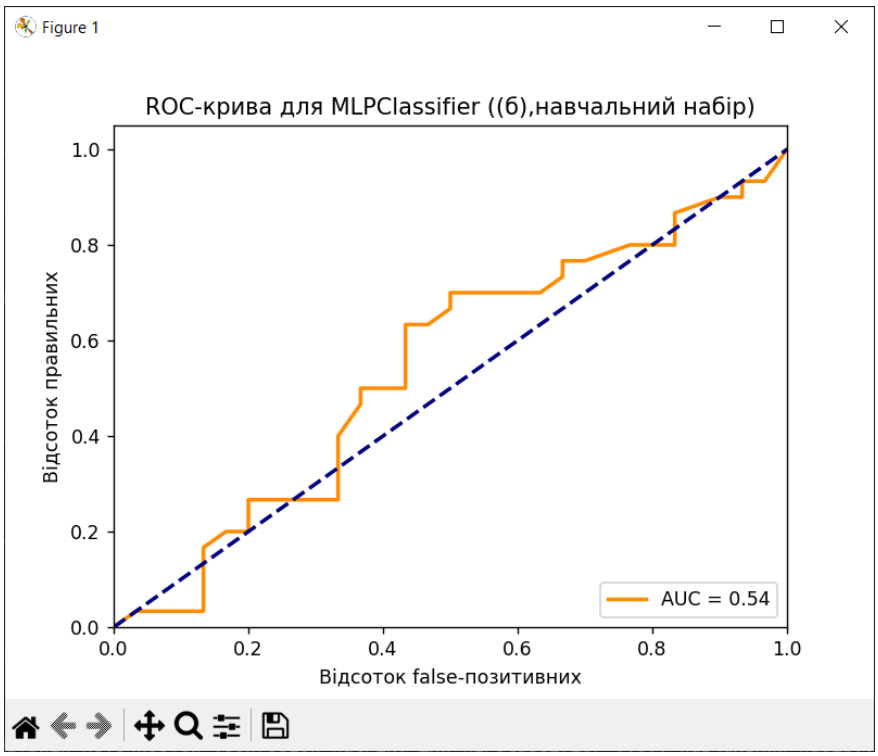


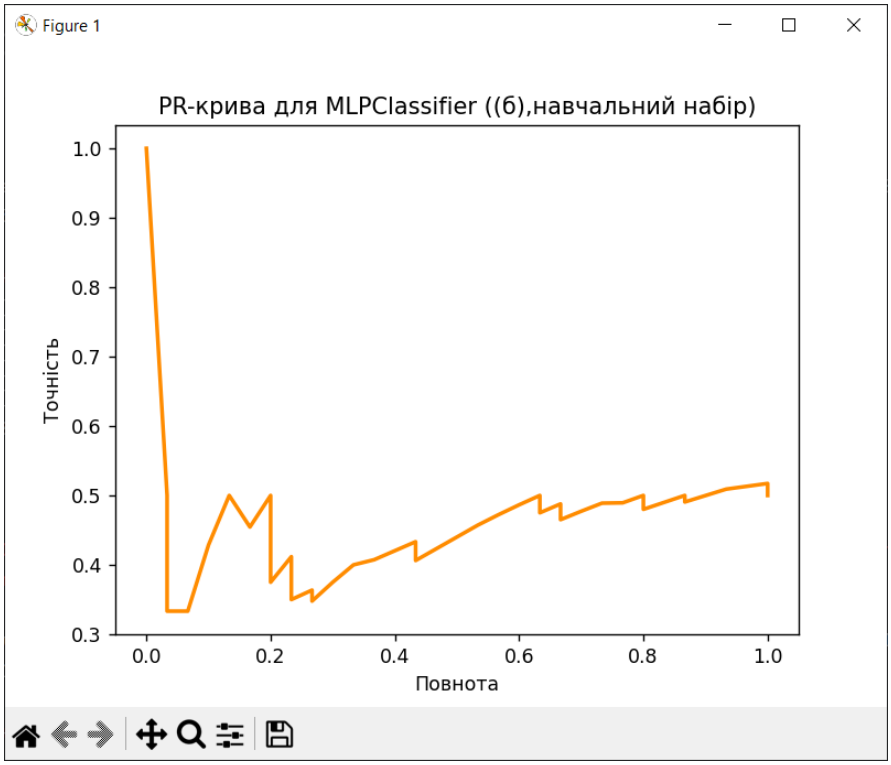
Розрахуємо прогноз навчального набору, апостеріорні ймовірності, критерії якості нашої моделі, ROC та PR- криві:

```
#Прогноз
y2_pred_M2 = Model2.predict(X2_test)
print("\nПрогноз навчальний набору(6):", y2_pred_M2)

# Розрахунок додаткових результатів
probs2_M2 = Model2.predict_proba(X2_test)
#print("\nАпостеріорна ймовірність:", probs2_M2)

# Розрахунок критеріїв якості
cm2_M2, precision2_M2, recall2_M2, f2_M2, fpr2_M2, tpr2_M2, auc2_M2,
pr2_M2 = calculate_metrics(y2_test, y2_pred_M2, probs2_M2)

print("\nМатриця неточностей (Confusion Matrix) для MLPClassifier
((6), навчальний набір):")
print(cm2_M2)
print(f"Точність: {precision2_M2:.2f}")
print(f"Повнота: {recall2_M2:.2f}")
print(f"F1-міра: {f2_M2:.2f}")
print(f"AUC: {auc2_M2:.2f}")

# Візуалізація ROC-кривих
plot_roc_curve(fpr2_M2, tpr2_M2, auc2_M2, "ROC-крива для MLPClassifier
((6),навчальний набір)")

# Візуалізація PR-кривих
plot_precision_recall_curve(pr2[0], pr2[1], "PR-крива для
MLPClassifier ((6),навчальний набір)")
```

Отримуємо:

```
Прогноз навчальний набору(6): [0 0 1 0 1 0 1 0 0 0 0 1 1 1 0 1 0 1 0 0 1 1 0 1 1 0 1 0 1 1 1 1 0 1 0 0 1
 0 0 0 0 1 0 1 0 1 1 0 0 1 0 1 1 1 1 1 0 0 1 1]

Матриця неточностей (Confusion Matrix) для MLPClassifier ((6), навчальний набір):
[[17 13]
 [12 18]]
Точність: 0.58
Повнота: 0.58
F1-міра: 0.58
AUC: 0.54
```





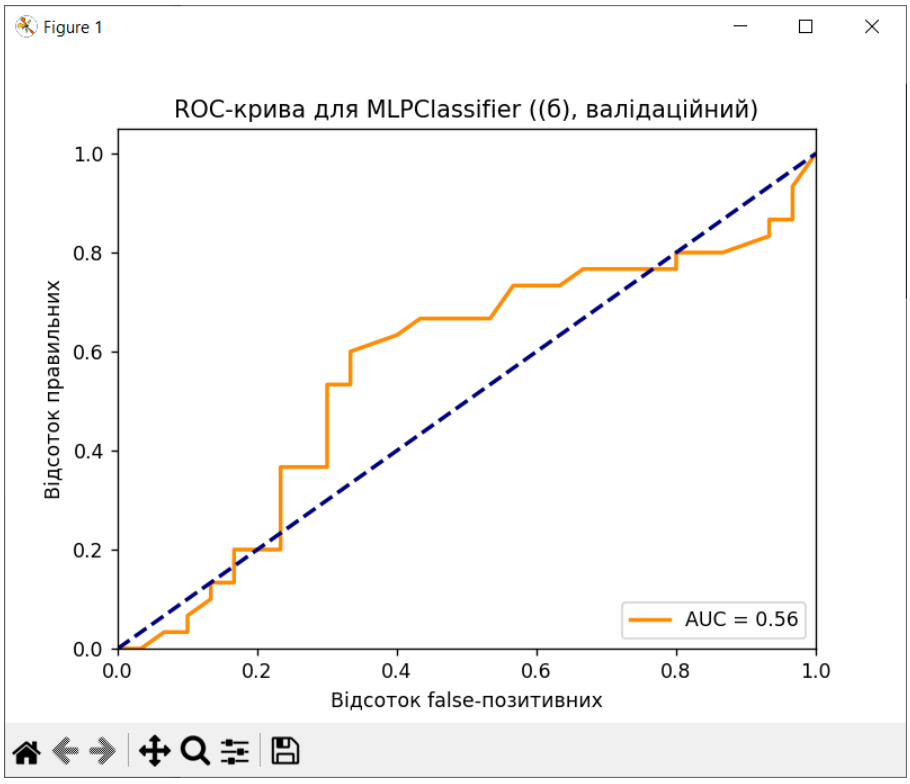
Аналогічні розрахунки для валідаційного набору (б):

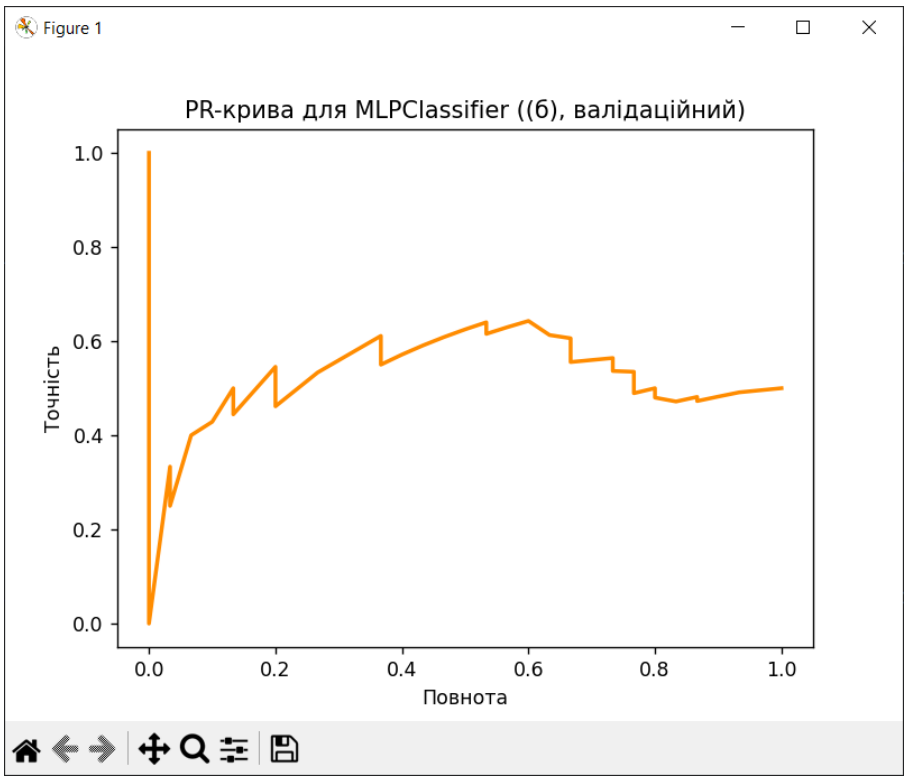
```
mlp_M2_v = MLPClassifier()
# Навчання отриманої моделі з валідаційними параметрами
mlp_M2_v.fit(X2_train, y2_train)
# Оцінка моделі з валідаційними параметрами
y2_pred_M2 = mlp_M2_v.predict(X2_test)

print("\nПрогноз валідаційний набору(б):", y2_pred_M2)
```

```
probs2_M2_v = mlp_M2_v.predict_proba(X2_test)

cm2_M2_v, precision2_M2_v, recall2_M2_v, f2_M2_v, fpr2_M2_v,
tpr2_M2_v, auc2_M2_v, pr2_M2_v = calculate_metrics(
    y2_test, y2_pred_M2, probs2_M2_v)

print("Матриця неточностей (Confusion Matrix) для MLPClassifier
((6),валідаційна):")
print(cm2_M2_v)
print(f"Точність: {precision2_M2_v:.2f}")
print(f"Повнота: {recall2_M2_v:.2f}")
print(f"F1-міра: {f2_M2_v:.2f}")
print(f"AUC: {auc2_M2_v:.2f}")

plot_roc_curve(fpr2_M2_v, tpr2_M2_v, auc2_M2_v, "ROC-крива для
MLPClassifier ((6), валідаційний)")
plot_precision_recall_curve(pr2_M2_v[0], pr2_M2_v[1], "PR-крива для
MLPClassifier ((6), валідаційний)")
```

Отримуємо:

```
Прогноз валідаційний набору(6): [1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1
 1 0 0 0 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1]
Матриця неточностей (Confusion Matrix) для MLPClassifier ((6),валідаційна):
[[ 4 26]
 [ 6 24]]
Точність: 0.44
Повнота: 0.47
F1-міра: 0.40
AUC: 0.56
```

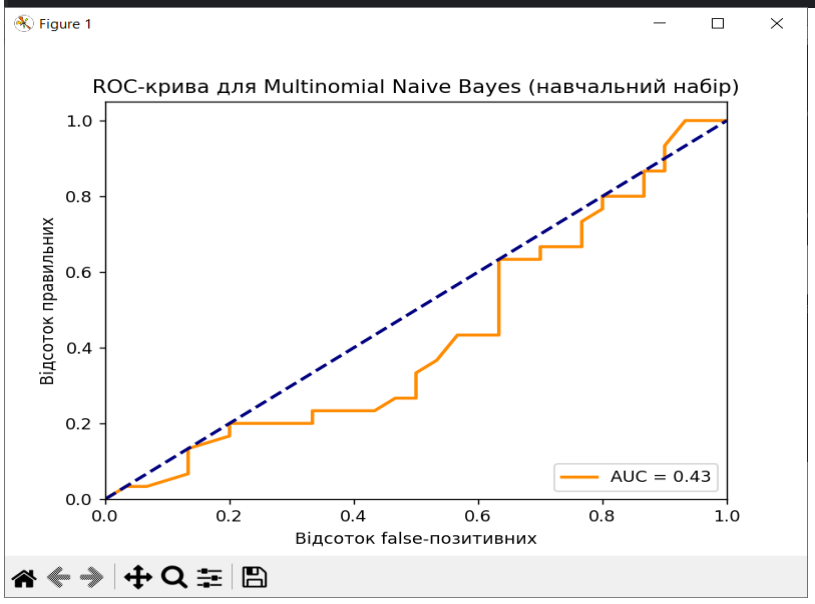


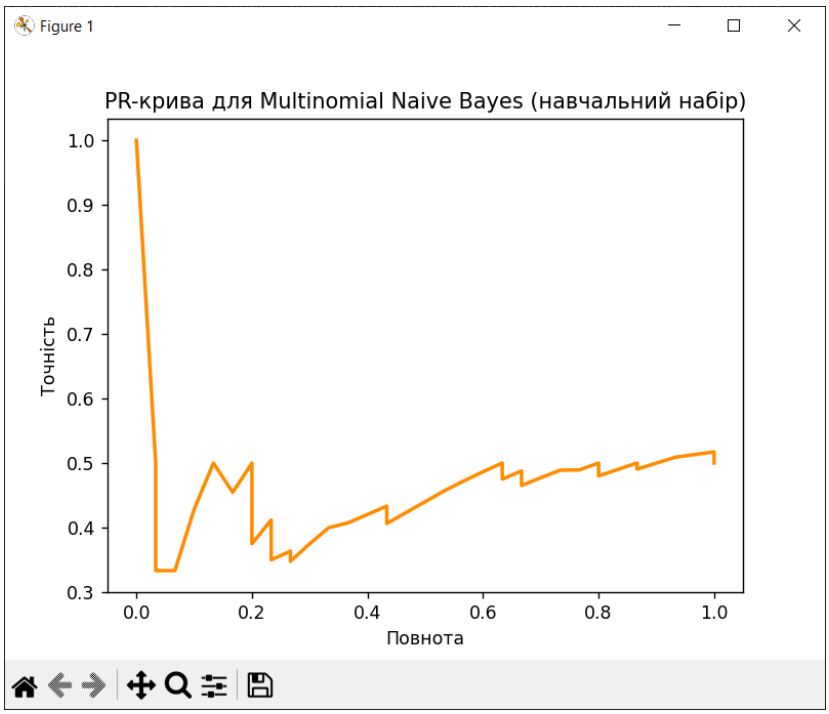


Загалом, модель демонструє набагато кращі навички, ніж її попередник, тренований на цьому ж складному наборі з MultinomialNB в попередній роботі:

(навчальний набір (б))

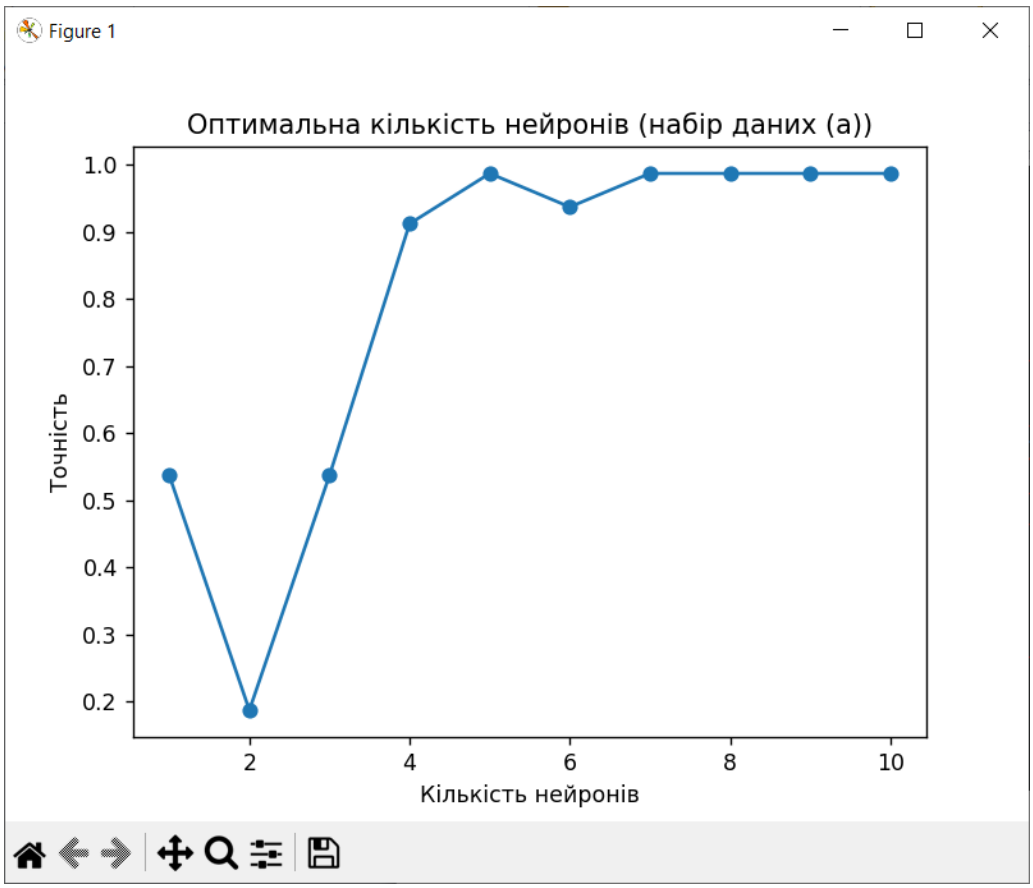
```
Матриця неточностей (Confusion Matrix) для MultinomialNB (навчальний набір):
[[ 6 24]
 [ 7 23]]
Точність: 0.48
Повнота: 0.48
F1-міра: 0.44
AUC: 0.43
```





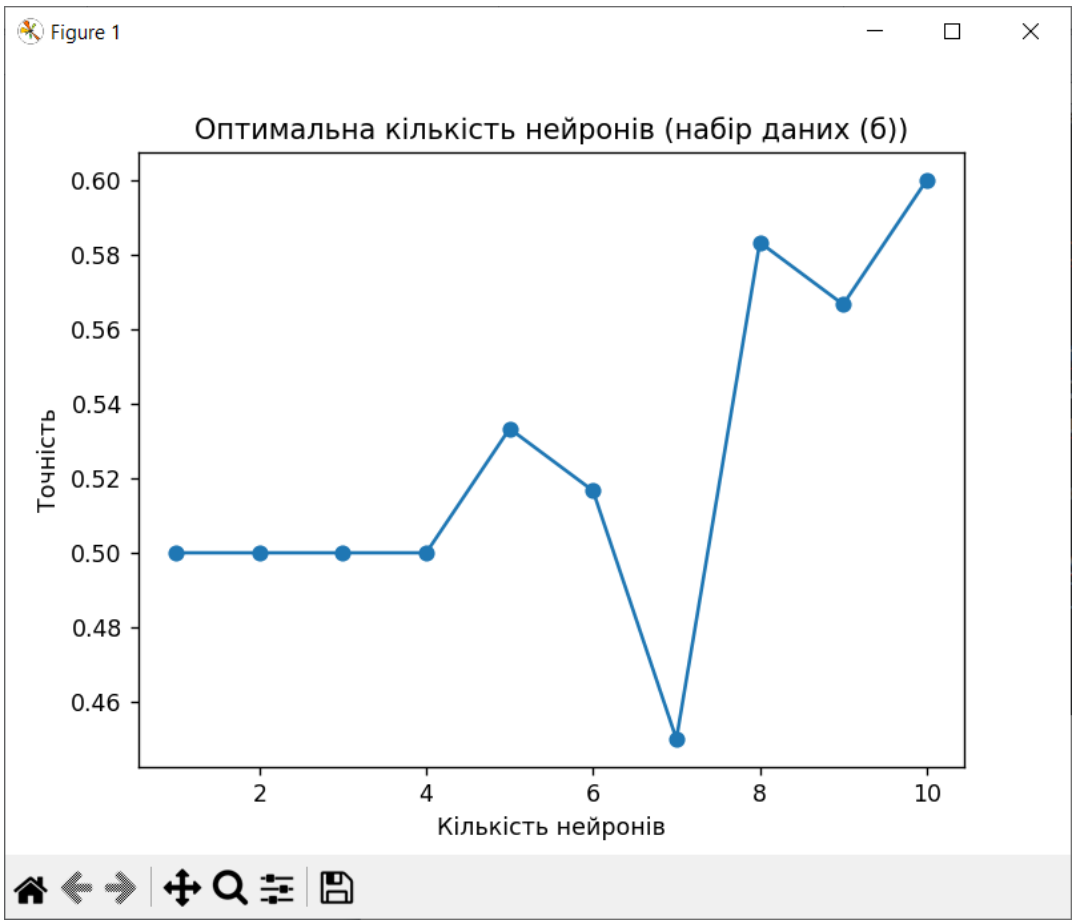
Знайдемо оптимальну кількість нейронів для найточнішої класифікації моделлю набору (a), та візуалізуємо результати:

```
# Пошук оптимальної кількості нейронів для першого набору даних
neuron_counts = range(1, 11)
accuracy_scores1 = []

for n in neuron_counts:
    mlp = MLPClassifier(hidden_layer_sizes=(n,), max_iter=1000,
random_state=1)
    mlp.fit(X1_train, y1_train)
    y_pred = mlp.predict(X1_test)
    accuracy = accuracy_score(y1_test, y_pred)
    accuracy_scores1.append(accuracy)

# Графік для оптимальної кількості нейронів (перший набір даних)
plt.figure()
plt.plot(neuron_counts, accuracy_scores1, marker='o')
plt.xlabel("Кількість нейронів")
plt.ylabel("Точність")
plt.title("Оптимальна кількість нейронів (набір даних (a))")
plt.show()
```



Аналогічно для набору (б):

```
# Пошук оптимальної кількості нейронів для другого набору даних
accuracy_scores2 = []

for n in neuron_counts:
    mlp = MLPClassifier(hidden_layer_sizes=(n,), max_iter=1000,
random_state=1)
    mlp.fit(X2_train, y2_train)
    y_pred = mlp.predict(X2_test)
    accuracy = accuracy_score(y2_test, y_pred)
    accuracy_scores2.append(accuracy)

# Графік для оптимальної кількості нейронів (другий набір даних)
plt.figure()
plt.plot(neuron_counts, accuracy_scores2, marker='o')
plt.xlabel("Кількість нейронів")
plt.ylabel("Точність")
plt.title("Оптимальна кількість нейронів (набір даних (б))")
plt.show()
```



Задіємо динамічне додавання нейронів в циклі до скритого шару, та визначимо, скільки треба додати для задовільного розв'язку з заданою точністю моделей:

```
# Динамічне додавання нейронів до скритого шару (перший набір даних)
desired_accuracy1 = 0.95  # Задовільна точність для першого набору
даних
hidden_neurons1 = 0

while True:
    hidden_neurons1 += 1
    mlp = MLPClassifier(hidden_layer_sizes=(hidden_neurons1,),
max_iter=1000, random_state=1)
    mlp.fit(X1_train, y1_train)
    y_pred = mlp.predict(X1_test)
    accuracy1 = accuracy_score(y1_test, y_pred)

    if accuracy1 >= desired_accuracy1:
        break

print(f"Для досягнення точності {desired_accuracy1:.2f} (набір даних
(а)) було додано {hidden_neurons1} нейронів в скритий шар.")

# Динамічне додавання нейронів до скритого шару (другий набір даних)
desired_accuracy2 = 0.6  # Задовільна точність для другого набору
```

```
даних
hidden_neurons2 = 0

while True:
    hidden_neurons2 += 1
    mlp = MLPClassifier(hidden_layer_sizes=(hidden_neurons2,),
max_iter=1000, random_state=1)
    mlp.fit(X2_train, y2_train)
    y_pred = mlp.predict(X2_test)
    accuracy2 = accuracy_score(y2_test, y_pred)

    if accuracy2 >= desired_accuracy2:
        break

print(f"Для досягнення точності {desired_accuracy2:.2f} (набір даних
(б)) було додано {hidden_neurons2} нейронів в скритий шар.")
```

Остаточно, маємо:

```
Для досягнення точності 0.95 (набір даних (а)) було додано 5 нейронів в скритий шар.
Для досягнення точності 0.60 (набір даних (б)) було додано 10 нейронів в скритий шар.
```

Бачимо, що результат відповідає зображеним вище графікам.

Висновки: У ході виконання комп'ютерного практикуму було проведено навчання та оцінювання моделей класифікації засобами бібліотеки Scikit-Learn Python, зокрема досліджено моделі, натреновану класифікатором MLPClassifier на різних видах наборів даних, також задіяно механізми багатошарового персептрону для збільшення класифікаційних можливостей створених моделей та покращення їх точності шляхом вибору оптимальної кількості нейронів.