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

ЗВІТ

про виконання практикуму № 2

з дисципліни «Інформаційний 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 якості моделей класифікації та регресії засобами бібліотеки Scikit-Learn Python

Мета: побудова та оцінювання якості моделей:

- дерев рішень,
- опорних векторів,
- логістичної регресії,
- наївної баєсівської моделі

для класифікації та регресії засобами бібліотеки Scikit-Learn Python.

Завдання:

25. Побудувати моделі наївної байесівської класифікації за припущень:

- Дані в кожному класі мають нормальний розподіл без коваріації між вимірами; використати клас sklearn.naive_bayes.GaussianNB.
- Дані в кожному класі мають поліноміальний розподіл; використати клас sklearn.naive_bayes.MultinomialNB.
- Для кожної моделі розрахувати апостеріорні імовірності для тестового прикладу, використовуючи метод predict_proba.

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)
```
import numpy as np

np.random.seed(0)

X = np.random.randn(300 , 2)

Y = np.logical_xor(X[:,0] > 0 , X[:,1] > 0)
```

Хід роботи:

1. Представити початкові дані графічно.

2. Розбити дані на навчальний та валідаційний набори.

3. Побудувати на навчальному наборі даних моделі класифікації задані згідно з варіантом.

4. Представити моделі графічно (наприклад вивести частину дерева рішень).

5. Виконати прогнози на основі побудованих моделей.

6. Для кожної з моделей оцінити, чи має місце перенавчання.

7. Розрахувати додаткові результати моделей, наприклад, апостеріорні імовірності або інші (згідно з варіантом).

8. В задачах класифікації побудувати границі рішень графічно для кожної з моделей.

9. В задачах класифікації розрахувати для кожної моделі значення наступних критеріїв якості, окремо на навчальній та валідаційній множинах:

- матрицю неточностей (confusion matrix),
- точність (precision),
- повноту (recall),
- міру F1 (F1 score),
- побудувати криву точності-повноти (precision-recall (PR) curve), ROC-криву, показник AUC.

10. Спробувати виконати решітчастий пошук (grid search) для підбору гіперпараметрів моделей.

11. Зробити висновки про якість роботи моделей на досліджених даних. На основі критеріїв якості спробувати обрати найкращу модель.

12. Навчити моделі на підмножинах навчальних даних. Оцінити, наскільки розмір навчальної множини впливає на якість моделі.

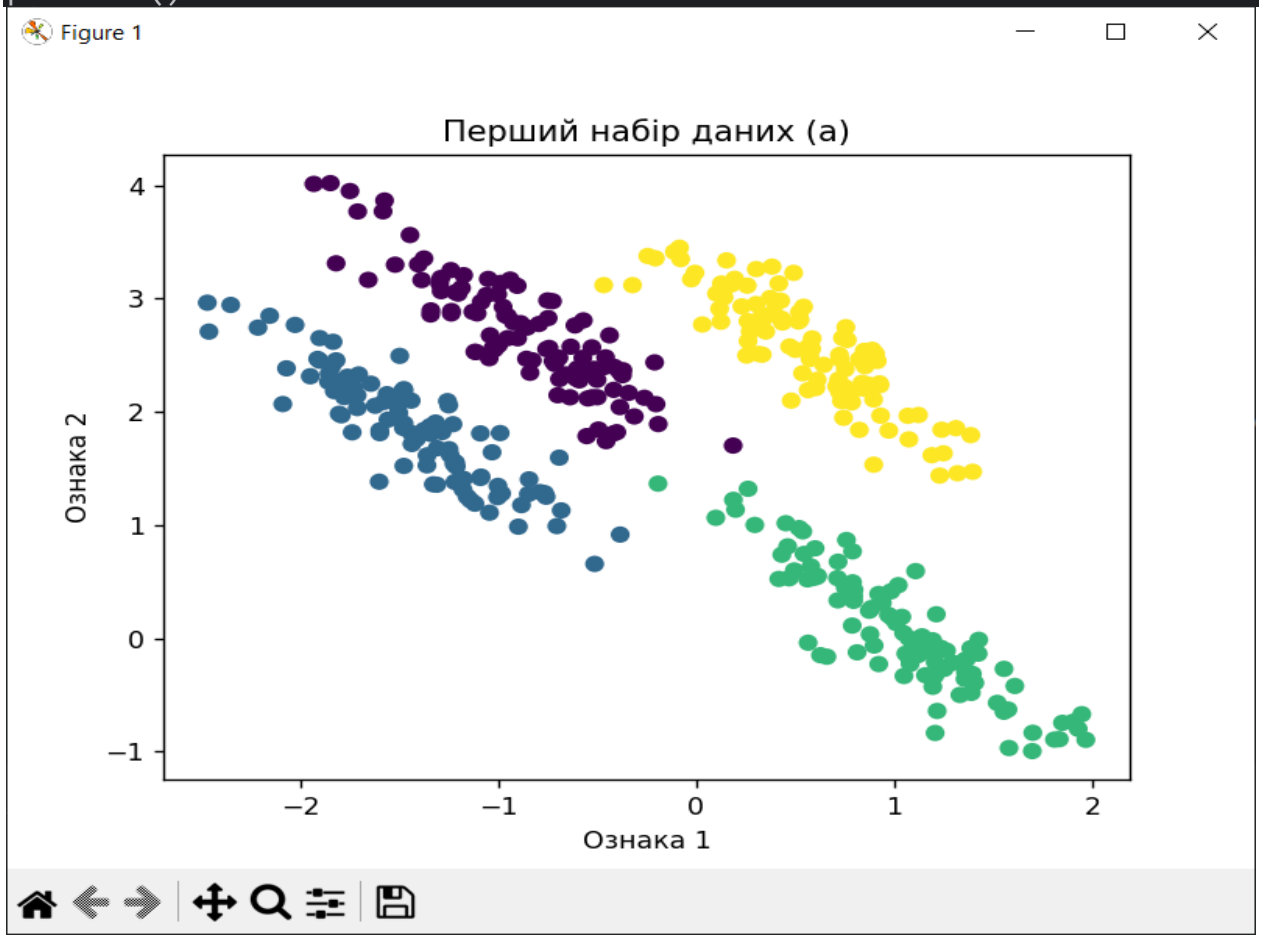
13. Кожний варіант містить два набори даних. Дослідити обидва набори за наведеними вище етапами. Можна обрати власний набір даних (повідомивши попередньо про це викладача), наприклад, з цікавої вам практичної задачі. Для кожного набору спробувати підібрати найкращу модель.

Зауваження: було змінено заданий набір даних (б) для MultinomialNB на більш зручний поліноміальний набір з індикатором ознак для наочності демонстрації методу.

Хід роботи:

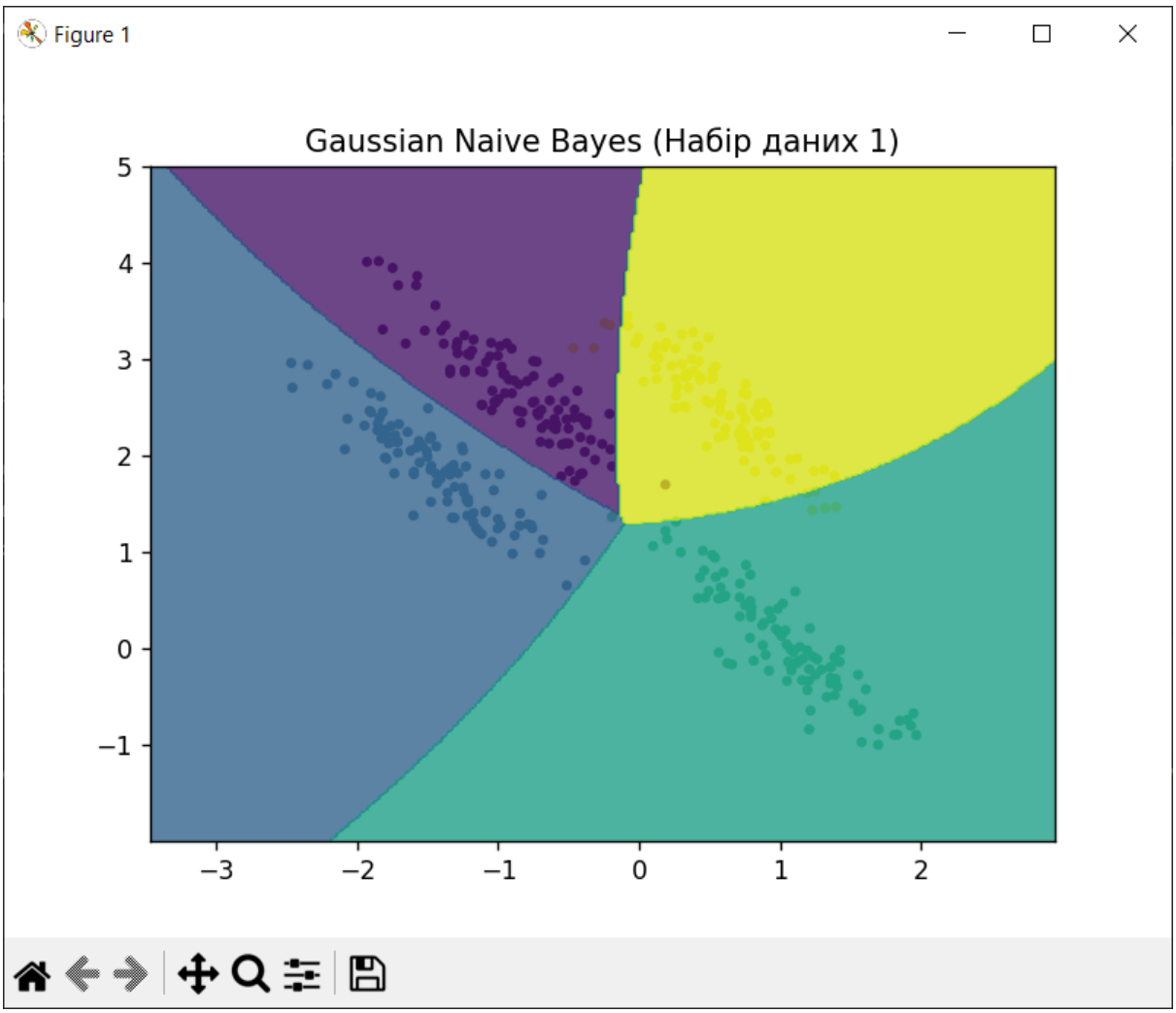
1.Згенеруємо та опрацюємо набір (а):

```
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
from sklearn.model_selection import GridSearchCV

# Перший набір даних (а)
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 (а): 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 (а)")
plt.show()
```



2. Розіб’ємо дані на навчальні та валідаційні набори, побудуємо моделі класифікації та представимо їх графічно:

```
# Поділ набору даних 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)

# Gaussian Naive Bayes для першого набору даних (a)
gnb1 = GaussianNB()
gnb1.fit(X1_train, y1_train)

# Функція для візуалізації границь рішень
def plot_decision_boundary(model, X, y, title):
    plt.figure()
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis', marker='.')
    h = .02
    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, h), np.arange(y_min,
y_max, h))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap='viridis', alpha=0.8)
    plt.title(title)
    plt.show()

# Візуалізація границь рішень для Gaussian Naive Bayes (перший набір
даних)
plot_decision_boundary(gnb1, X1_stretched, y1, "Gaussian Naive Bayes
(Набір даних 1)")
```



3. Виконаємо прогноз:

```
# Прогноз
y1_pred = gnb1.predict(X1_test)
```

```
Прогноз 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]
```

4. Оцінимо, чи має місце перенавчання:

```
# Оцінка перенавчання
def check_overfitting(model, X_train, y_train, X_test, y_test):
    train_acc = model.score(X_train, y_train)
    test_acc = model.score(X_test, y_test)
    return train_acc, test_acc

train_acc1, test_acc1 = check_overfitting(gnb1, X1_train, y1_train,
X1_test, y1_test)

print("Перенавчання для Gaussian Naive Bayes (Набір даних 1):")
print(f"Точність на навчальних даних: {train_acc1:.2f}")
print(f"Точність на валідаційних даних: {test_acc1:.2f}")
```

```
Перенавчання для Gaussian Naive Bayes (Набір даних 1):
Точність на навчальних даних: 0.98
Точність на валідаційних даних: 0.95
```

5.Обчислимо додаткові результати для моделі (апостеріорні ймовірності):

```
# Розрахунок додаткових результатів
probs1 = gnb1.predict_proba(X1_test)
```

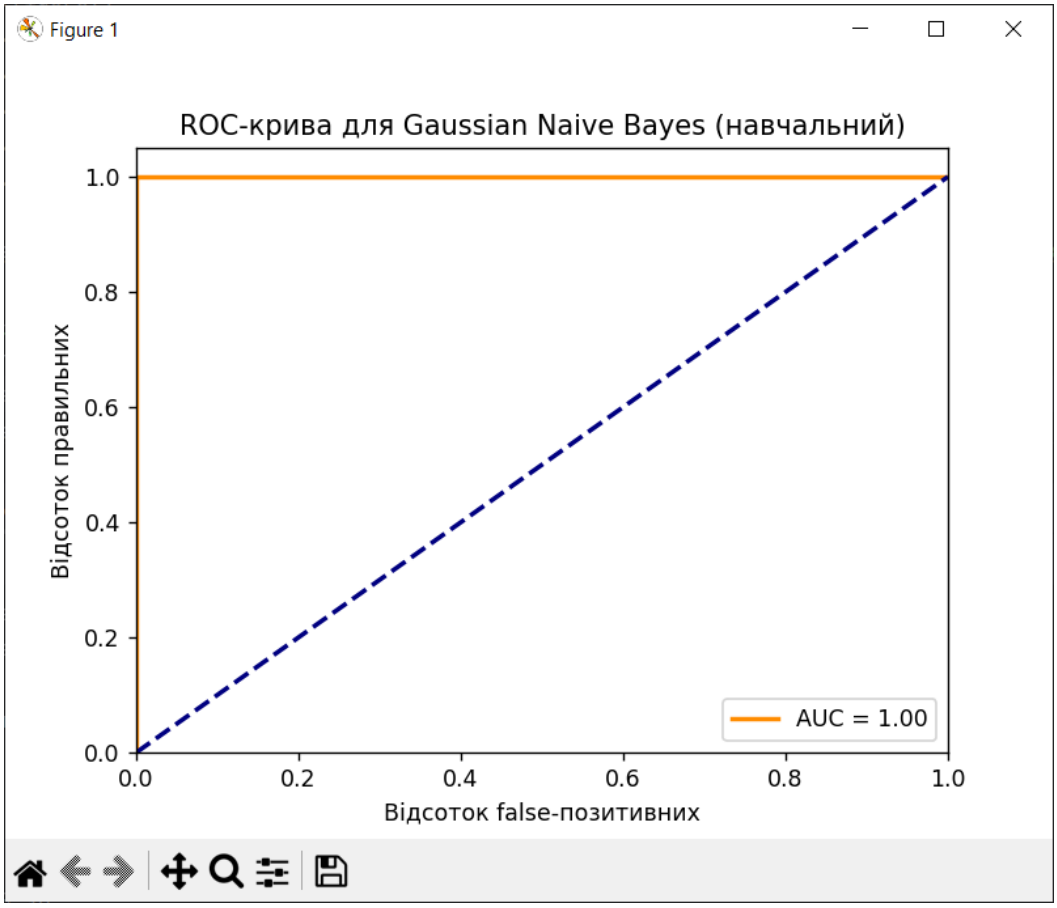
6. Розрахуємо значення критеріїв якості моделі, побудуємо PR, ROC-криві, знайдемо показник AUC:

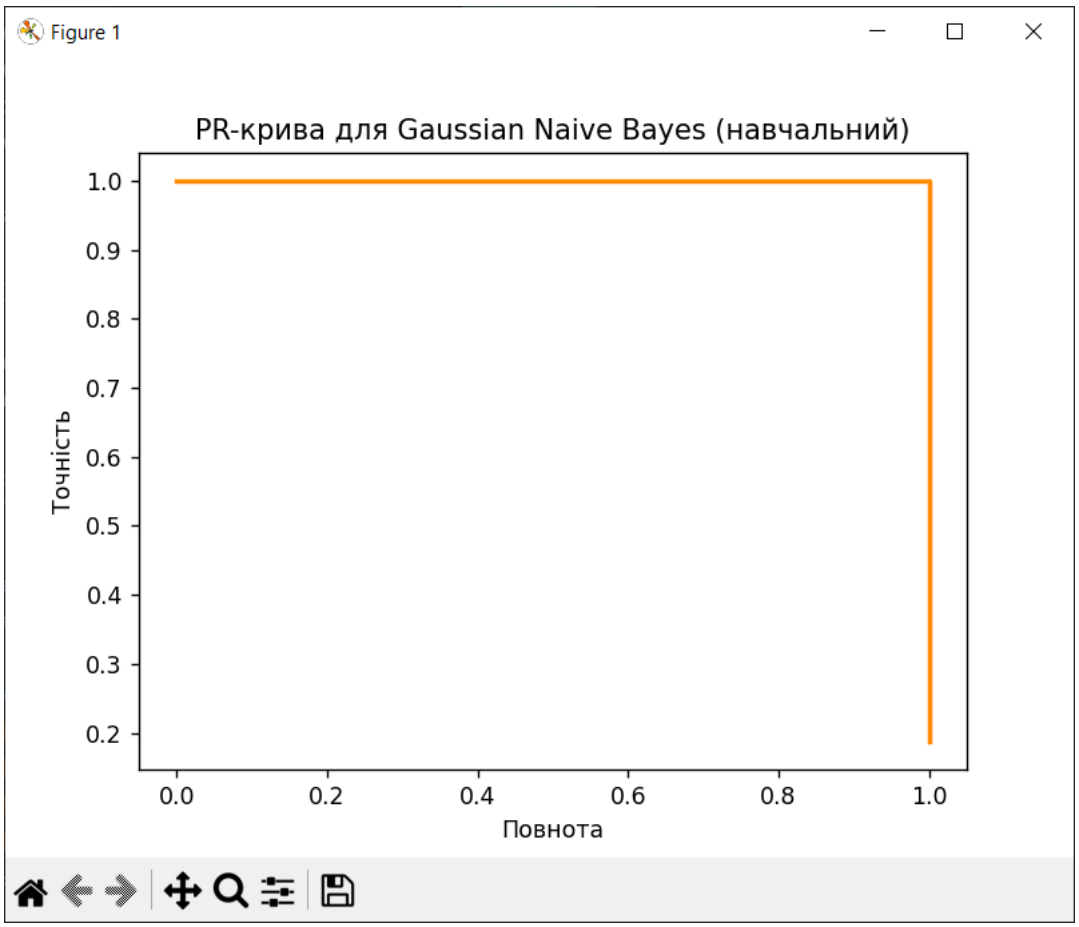
```
# Розрахунок критеріїв якості
def calculate_metrics(y_true, y_pred, probs):
    cm = confusion_matrix(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='weighted')
    recall = recall_score(y_true, y_pred, average='weighted')
    f1 = f1_score(y_true, y_pred, average='weighted')
    fpr, tpr, _ = roc_curve(y_true, probs[:, 1],pos_label=1)
    auc_score = auc(fpr, tpr)
    precision_recall = precision_recall_curve(y_true, probs[:, 1],
pos_label=1)
    return cm, precision, recall, f1, fpr, tpr, auc_score,
precision_recall

cm1, precision1, recall1, f1_1, fpr1, tpr1, auc1, pr1 =
calculate_metrics(y1_test, y1_pred, probs1)
print("\nМатриця неточностей (Confusion Matrix) для Gaussian Naive
Bayes (навчальний набір):")
print(cm1)
print(f"Точність: {precision1:.2f}")
print(f"Повнота: {recall1:.2f}")
print(f"F1-міра: {f1_1:.2f}")
print(f"AUC: {auc1:.2f}")

# Візуалізація ROC-кривих
def plot_roc_curve(fpr, tpr, auc, title):
    plt.figure()
    plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'AUC =
{auc:.2f}')
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('Відсоток false-позитивних')
    plt.ylabel('Відсоток правильних')
    plt.title(title)
    plt.legend(loc="lower right")
    plt.show()
```

```
plot_roc_curve(fpr1, tpr1, auc1, "ROC-крива для Gaussian Naive Bayes
(навчальний)")

# Візуалізація PR-кривих
def plot_precision_recall_curve(precision, recall, title):
    plt.figure()
    plt.plot(recall, precision, color='darkorange', lw=2)
    plt.xlabel('Повнота')
    plt.ylabel('Точність')
    plt.title(title)
    plt.show()

plot_precision_recall_curve(pr1[0], pr1[1], "PR-крива для Gaussian
Naive Bayes (навчальний)")
```

Отримуємо:

```
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
```





7. Спробуємо виконати решітчастий пошук, та дослідимо модель на покращених валідаційних параметрах

```
# Решітчастий пошук для Gaussian Naive Bayes
param_grid = {'var_smoothing': np.logspace(0, -9, num=100)}
grid_search = GridSearchCV(GaussianNB(), param_grid, cv=5)
grid_search.fit(X1_train, y1_train)
best_params = grid_search.best_params_
best_gnb = grid_search.best_estimator_

print("Найкращі гіперпараметри для Gaussian Naive Bayes (Набір даних
1):", best_params)

# Навчання моделі з найкращими параметрами
best_gnb.fit(X1_train, y1_train)

# Оцінка моделі з найкращими параметрами
y1_pred_best = best_gnb.predict(X1_test)
print("\nПрогноз валідаційний набору(а):", y1_pred_best)
probs1_best = best_gnb.predict_proba(X1_test)

cm1_best, precision1_best, recall1_best, f1_1_best, fpr1_best,
tpr1_best, auc1_best, pr1_best = calculate_metrics(
    y1_test, y1_pred_best, probs1_best)
```

```python
print("Матриця неточностей (Confusion Matrix) для Gaussian Naive Bayes (з найкращими параметрами):")
print(cm1_best)
print(f"Точність: {precision1_best:.2f}")
print(f"Повнота: {recall1_best:.2f}")
print(f"F1-міра: {f1_1_best:.2f}")
print(f"AUC: {auc1_best:.2f}")

plot_roc_curve(fpr1_best, tpr1_best, auc1_best, "ROC-крива для Gaussian Naive Bayes (валідаційний)")
plot_precision_recall_curve(pr1_best[0], pr1_best[1], "PR-крива для Gaussian Naive Bayes (валідаційний)")
```

```
Найкращі гіперпараметри для Gaussian Naive Bayes (Набір даних a): {'var_smoothing': 0.8111308307896871}

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 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
```

Бачимо, що зі згладженим гіперпараметром точність результатів втрачається, хоча все ще є порівняно високою

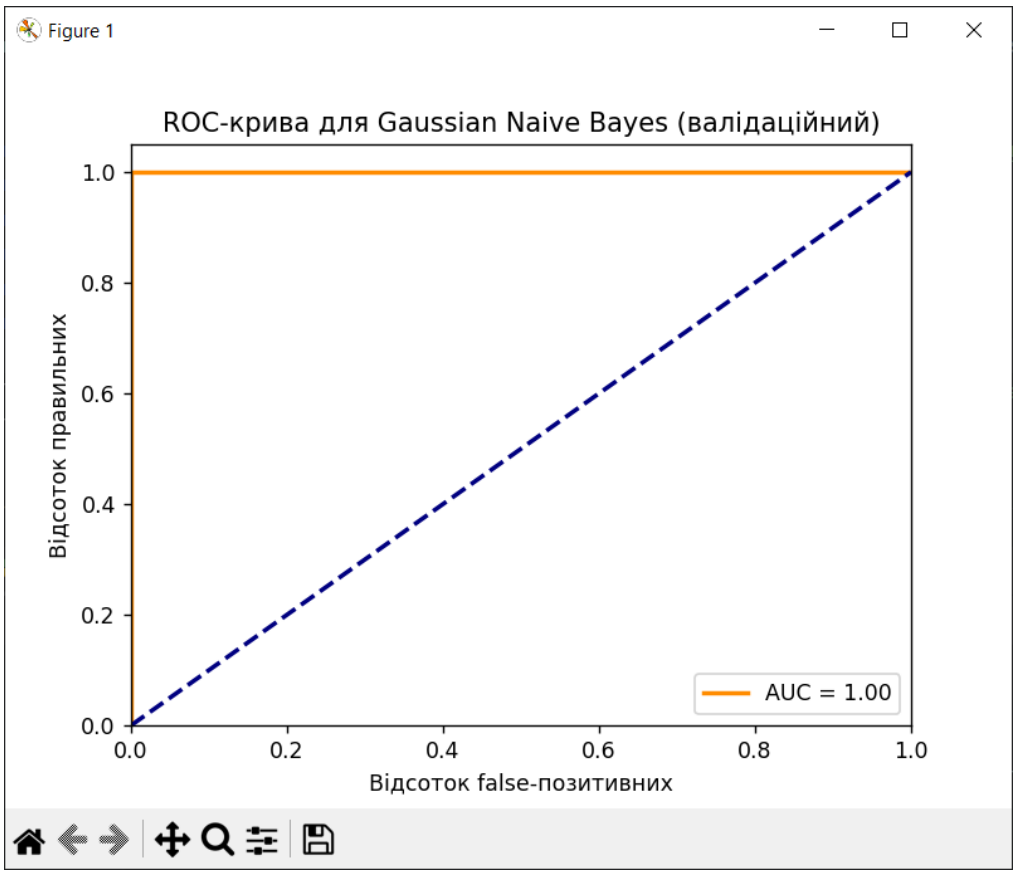


Figure 1
ROC-крива для Gaussian Naive Bayes (валідаційний)
Відсоток правильних
Відсоток false-позитивних
AUC = 1.00
0.0
0.2
0.4
0.6
0.8
1.0

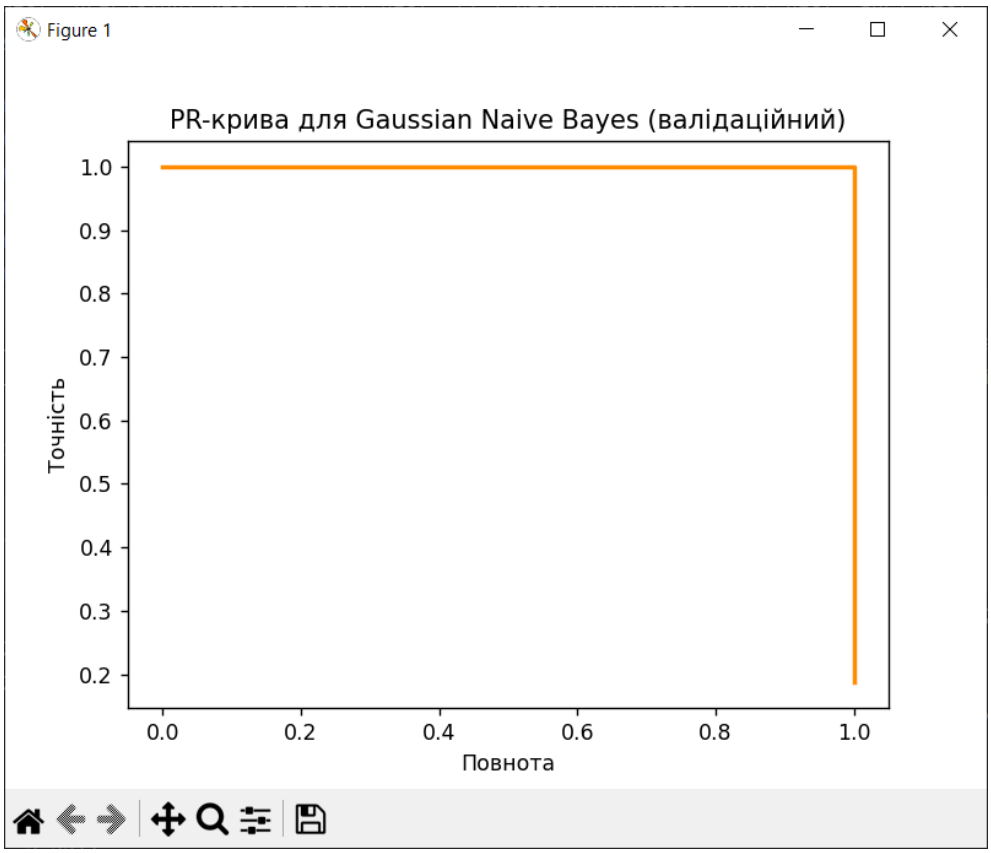
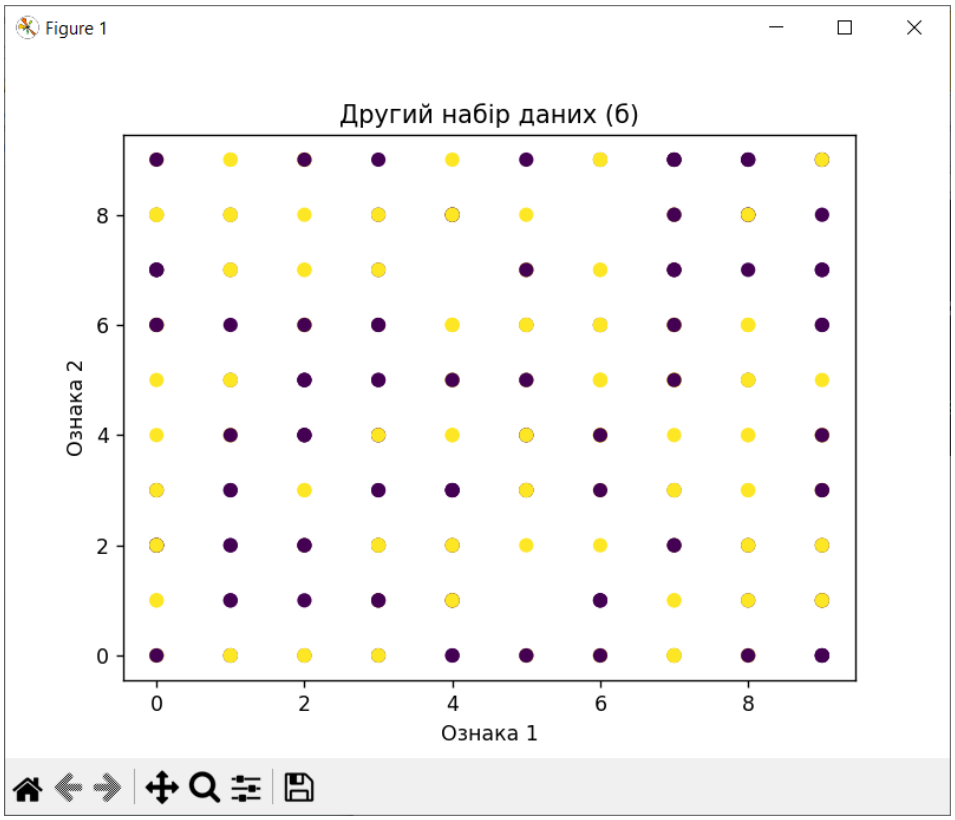


Figure 1
PR-крива для Gaussian Naive Bayes (валідаційний)
Точність
Повнота
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
0.0
0.2
0.4
0.6
0.8
1.0

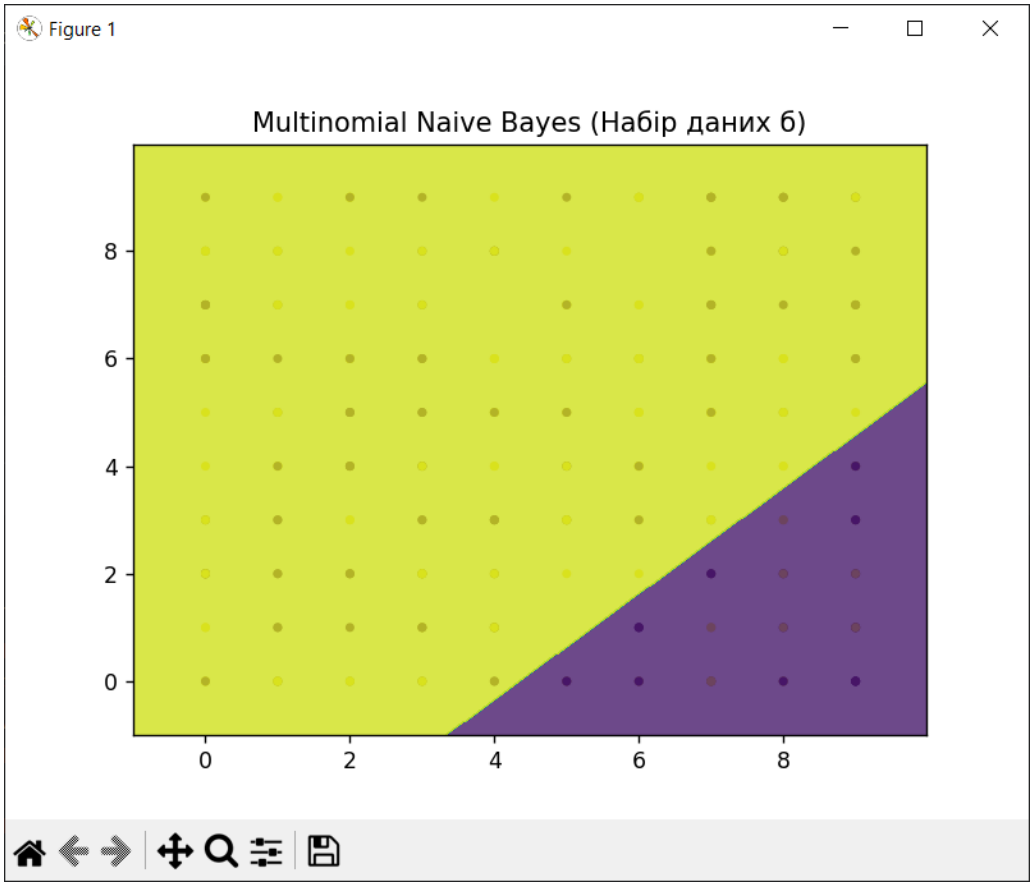
8. За аналогічним алгоритмом згенеруємо набір (б) та проведемо експерименти:

```
#--------------------------2й Датасет-----------------------------------
----#
np.random.seed(1)
# Генеруємо випадковий count для ознаки 1
feature1 = np.random.randint(0, 10, size=300)

# Генеруємо випадковий count для ознаки 2
feature2 = np.random.randint(0, 10, size=300)

# Побудова вибірки, де кожний рядок - це вектор ознак
X2 = np.column_stack((feature1, feature2))

# Генеруємо вектор класів (0 або 1)
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()
```



Розділимо дані на навчальний та валідаційний набори, створимо модель та візуалізуємо:

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

# Multinomial Naive Bayes для другого набору даних (б)
mnb2 = MultinomialNB()
mnb2.fit(X2_train, y2_train)

# Візуалізація границь рішень для Multinomial Naive Bayes (другий
набір даних)
plot_decision_boundary(mnb2, X2, Y2, "Multinomial Naive Bayes (Набір
даних б)")
```



Прогноз, оцінка перенавчання та апостеріорна ймовірність:

```
#Прогноз
y2_pred = mnb2.predict(X2_test)
print("\nПрогноз навчальний набору(б):", y2_pred)

#Оцінка перенавчання
train_acc2, test_acc2 = check_overfitting(mnb2, X2_train, y2_train,
X2_test, y2_test)

print("\nПеренавчання для Multinomial Naive Bayes (Набір даних 2):")
print(f"Точність на навчальних даних: {train_acc2:.2f}")
print(f"Точність на валідаційних даних: {test_acc2:.2f}")
```

```
# Розрахунок додаткових результатів
probs2 = mnb2.predict_proba(X2_test)
```

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

Перенавчання для Multinomial Naive Bayes (Набір даних 2):
Точність на навчальних даних: 0.51
Точність на валідаційних даних: 0.48
```

Критерії якості:

```
# Розрахунок критеріїв якості
cm2, precision2, recall2, f2, fpr2, tpr2, auc2, pr2 =
calculate_metrics(y2_test, y2_pred, probs2)

print("\nМатриця неточностей (Confusion Matrix) для MultinomialNB
(навчальний набір):")
print(cm2)
print(f"Точність: {precision2:.2f}")
print(f"Повнота: {recall2:.2f}")
print(f"F1-міра: {f2:.2f}")
print(f"AUC: {auc2:.2f}")

# Візуалізація ROC-кривих
plot_roc_curve(fpr2, tpr2, auc2, "ROC-крива для Multinomial Naive
Bayes (навчальний набір)")

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

```
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
```

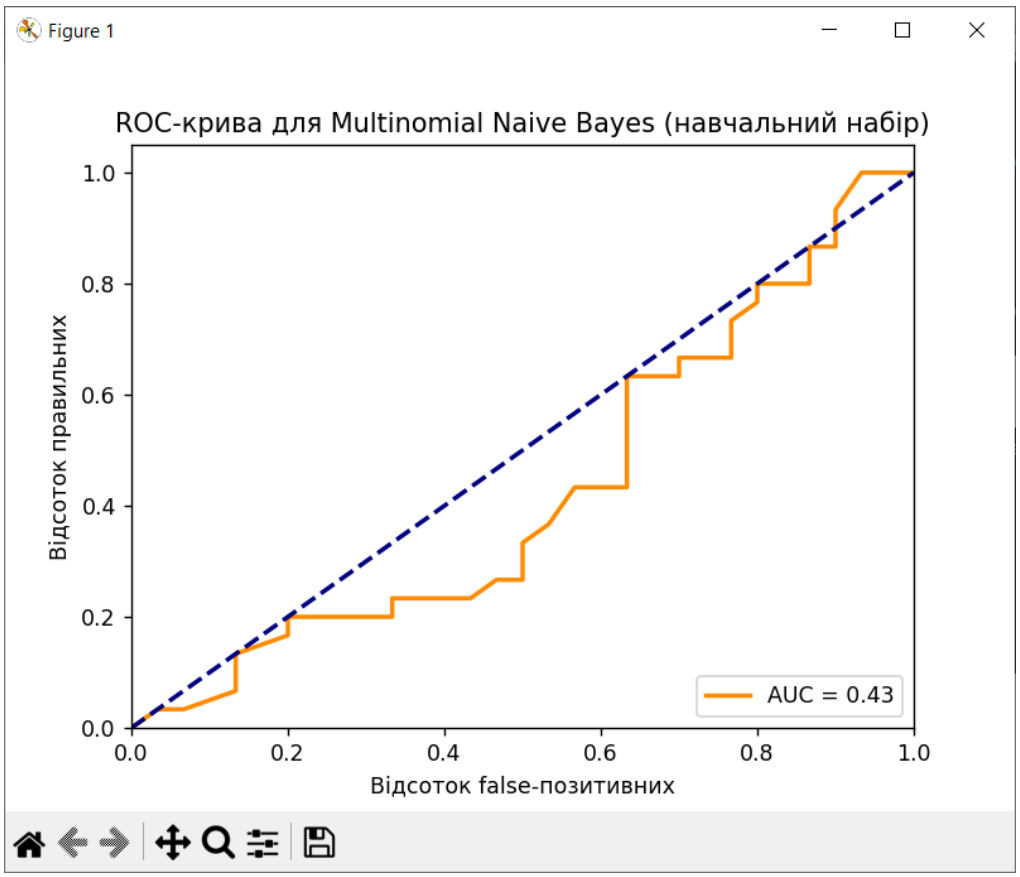


Figure 1
ROC-крива для Multinomial Naive Bayes (навчальний набір)
Відсоток правильних
Відсоток false-позитивних
AUC = 0.43
0.0
0.2
0.4
0.6
0.8
1.0

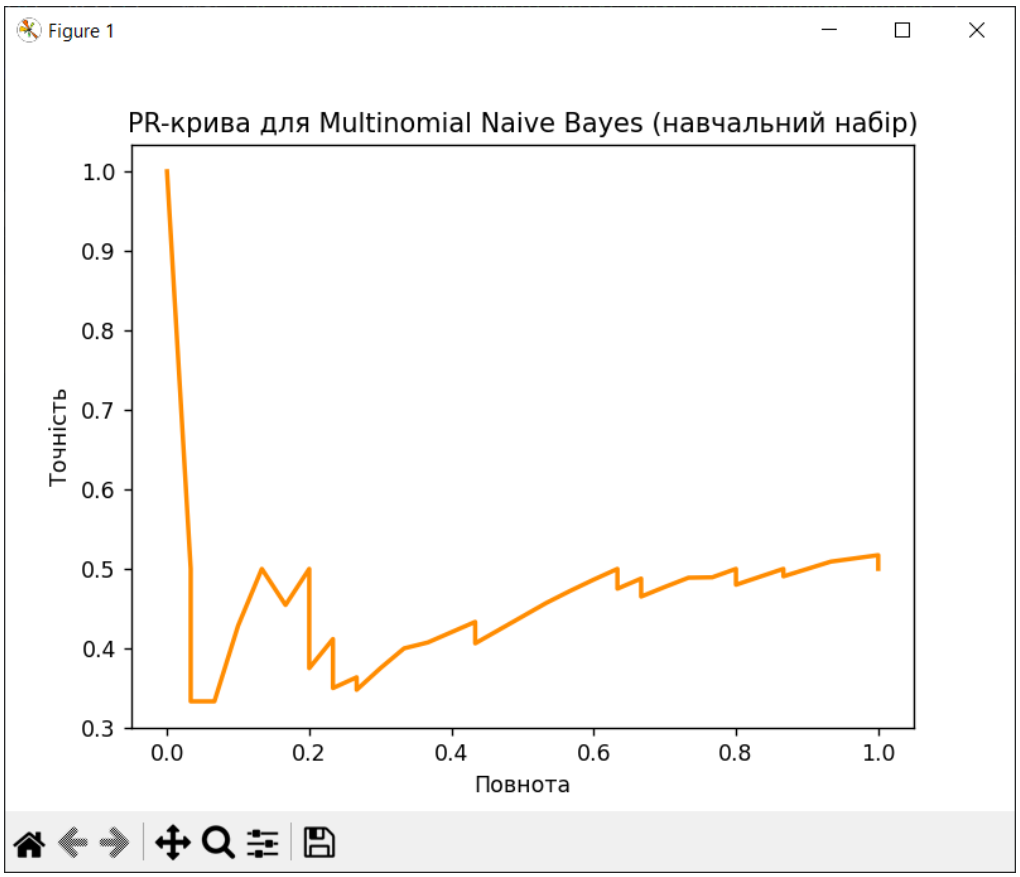


Figure 1
PR-крива для Multinomial Naive Bayes (навчальний набір)
Точність
Повнота
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
0.0
0.2
0.4
0.6
0.8
1.0

Решітчастий пошук для гіперпараметрів:

```
# Решітчастий пошук для Multinomial Naive Bayes
param_grid = {'alpha': np.logspace(0, 1, num=10)}
grid_search = GridSearchCV(MultinomialNB(), param_grid, cv=5)
grid_search.fit(X2_train, y2_train)
best_params = grid_search.best_params_
best_mnb2 = grid_search.best_estimator_

print("Найкращі гіперпараметри для Multinomial Naive Bayes (Набір
даних (6)):", best_params)
```

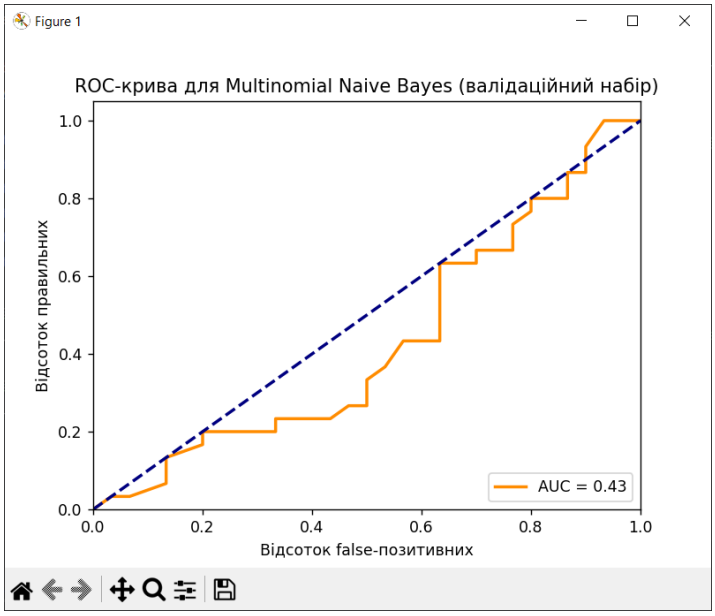
```
Найкращі гіперпараметри для Multinomial Naive Bayes (Набір даних (6)): {'alpha': 1.0}
```

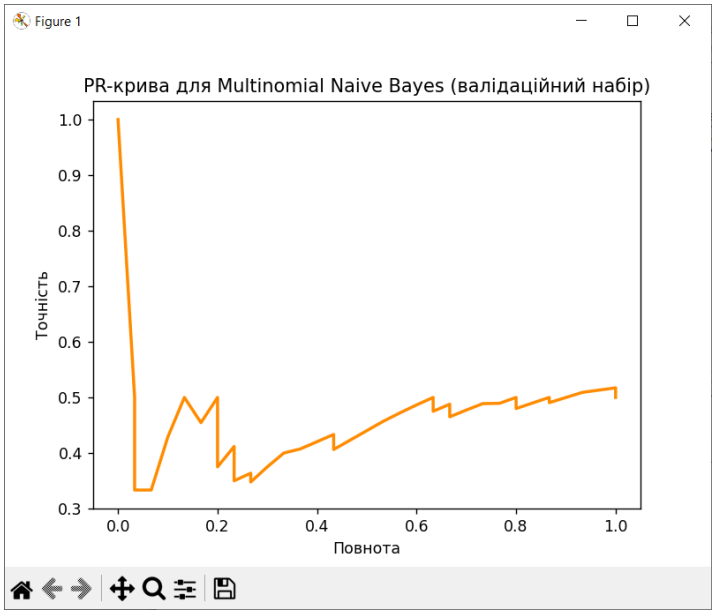
Оскільки це значення, яке використовується за замовчуванням в MultinomialNB, для знайденого гіперпараметру моделі точність та результати співпадатимуть з навчальними:

```
# Навчання моделі з найкращими параметрами
best_mnb2.fit(X2_train, y2_train)

# Оцінка моделі з найкращими параметрами
y2_pred_best = best_mnb2.predict(X2_test)
print("\nПрогноз валідаційний набору(6):", y2_pred_best)

probs2_best = best_mnb2.predict_proba(X2_test)

cm2_best, precision2_best, recall2_best, f2_best, fpr2_best,
tpr2_best, auc2_best, pr2_best = calculate_metrics(
    y2_test, y2_pred_best, probs2_best)

print("\nМатриця неточностей (Confusion Matrix) для Multinomial Naive
Bayes (з найкращими параметрами):")
print(cm2_best)
print(f"Точність: {precision2_best:.2f}")
print(f"Повнота: {recall2_best:.2f}")
print(f"F1-міра: {f2_best:.2f}")
print(f"AUC: {auc2_best:.2f}")

# Візуалізація ROC-кривих
plot_roc_curve(fpr2_best, tpr2_best, auc2_best, "ROC-крива для
Multinomial Naive Bayes (навчальний набір)")

# Візуалізація PR-кривих
plot_precision_recall_curve(pr2_best[0], pr2_best[1], "PR-крива для
Multinomial Naive Bayes (навчальний набір)")
```

Отримуємо:

```
Прогноз валідаційний набору(б): [1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 1 1 1 0 1 0 0 0 1 1 0 1 1 1
 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 0 0]

Матриця неточностей (Confusion Matrix) для Multinomial Naive Bayes (з найкращими параметрами):
[[ 6 24]
 [ 7 23]]
Точність: 0.48
Повнота: 0.48
F1-міра: 0.44
AUC: 0.43
```





Висновок: У ході тестування та навчання моделей наївної баєсівської класифікації засобами бібліотеки Scikit-Learn Python було порівняно точність та якість методів GaussianNB та MultinomialNB, для відповідних видів задач класифікації і розподілів. З отриманих результатів можна зробити висновок, що модель, яка навчалася з GaussianNB показала високу точність для свого

завдання (майже всі елементи розподілено правильно), в той час як модель тренована з MultinomialNB правильно класифікувала приблизно половину індикаторів, що є непоганим результатом для класифікатора поліноміально розподілених даних.