Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

ЗВІТ

про виконання лабораторної роботи № 2

з дисципліни «Інтелектуальний аналіз даних»

Виконала:
Студентка III курсу
Групи КА-76
Оркуша А. Д.

Перевірила:
Недашківська Н. І.

Київ – 2020

Постановка задачі

11. 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) `sklearn.datasets.load_digits`

    (б) `sklearn.datasets.make_moons`

1. Представити початкові дані графічно.
2. Розбити дані на навчальний і перевірочний набори.
3. Побудувати моделі класифікації або регресії згідно з варіантом.
4. Представити моделі графічно.
5. Виконати прогнози на основі моделей.
6. Для кожної з моделей оцінити, чи має місце перенавчання.
7. Розрахувати додаткові результати моделей, наприклад, апостеріорні імовірності, опорні вектори або інші (згідно з варіантом).
8. Для задач класифікації розрахувати критерії якості для кожної моделі:
    - матрицю неточностей (confusion matrix),
    - точність (precision),
    - повноту (recall),
    - міру F1 (F1 score),
    - побудувати криву точності-повноти (precision-recall (PR) curve), ROC-криву, показник AUC.
9. Виконати перехресну перевірку і решітчатий пошук для підбору гіперпараметрів моделей.
10. Зробити висновки про якість роботи моделей на досліджених даних. Для кожної навчальної вибірки вибрати найкращу модель.
11. Навчити моделі на підмножинах навчальних даних. Оцінити, наскільки розмір навчальної множини впливає на якість моделі.

Виконання

1. Представити початкові дані графічно

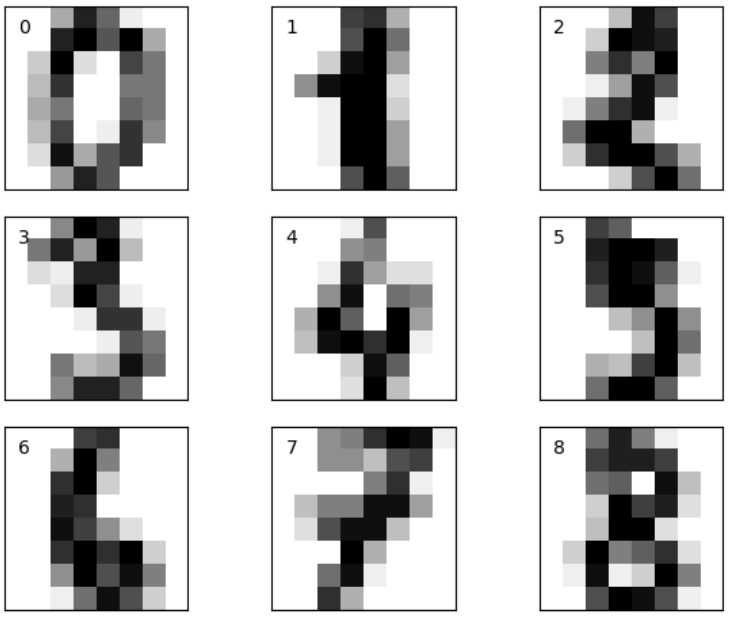
(a)load_digits

Цей датасет складається з набору n_samples зображень рукописних цифр (розміром 8x8), що представлені(закодовані) у вигляді цілочисельних векторів (1x64), та вектору розмірністю 1x n_samples - вектору класів до якого належить кожен з прикладів. Для графічного представлення покажемо декілька зображень цих цифр з підписом класу до якого вони належать у лівому верхньому куті.

Input:

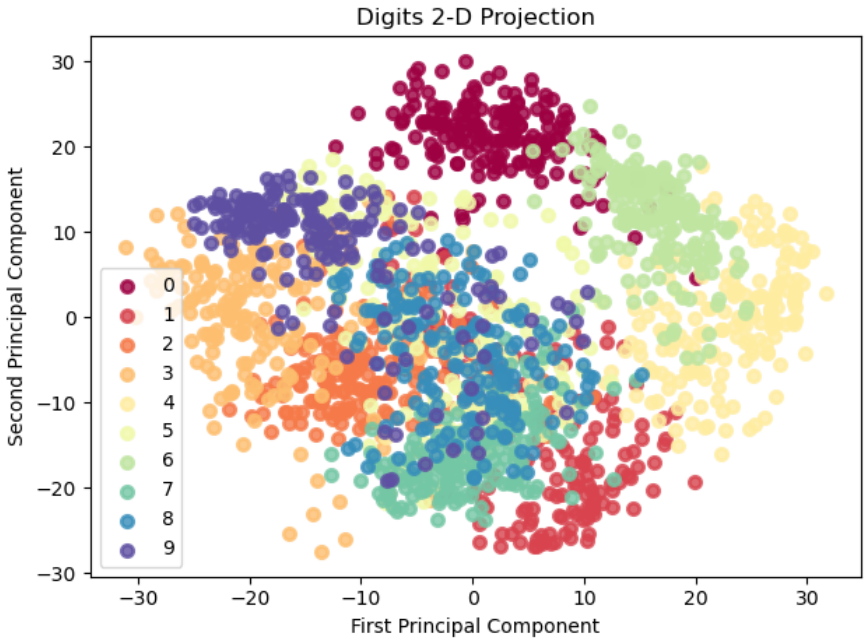
```
from sklearn import datasets, inspection, metrics
import matplotlib.pyplot as plt

digits = datasets.load_digits()
digits_X, digits_y = datasets.load_digits(return_X_y=True)

fig = plt.figure()
fig.subplots_adjust()

for i in range(9):
  ax = fig.add_subplot(3, 3, i + 1, xticks=[], yticks=[])
  ax.imshow(digits.images[i], cmap=plt.cm.gray_r, interpolation='nearest')
  ax.text(0.1, 0.7, str(digits_y[i]))
plt.show()
```

Output:



Окрім цього корисно візуалізувати належність усіх прикладів конкретним класам. Оскільки дані багатовимірні, використаємо метод головних компонент(PCA) для зменшення вимірності.

Input:
```
from sklearn.decomposition import PCA
import scikitplot as skplt

pca = PCA(n_components=2)
pca.fit(digits_X)
skplt.decomposition.plot_pca_2d_projection(pca, digits_X, digits_y, title='Digits 2-D Projection')
```
Output:



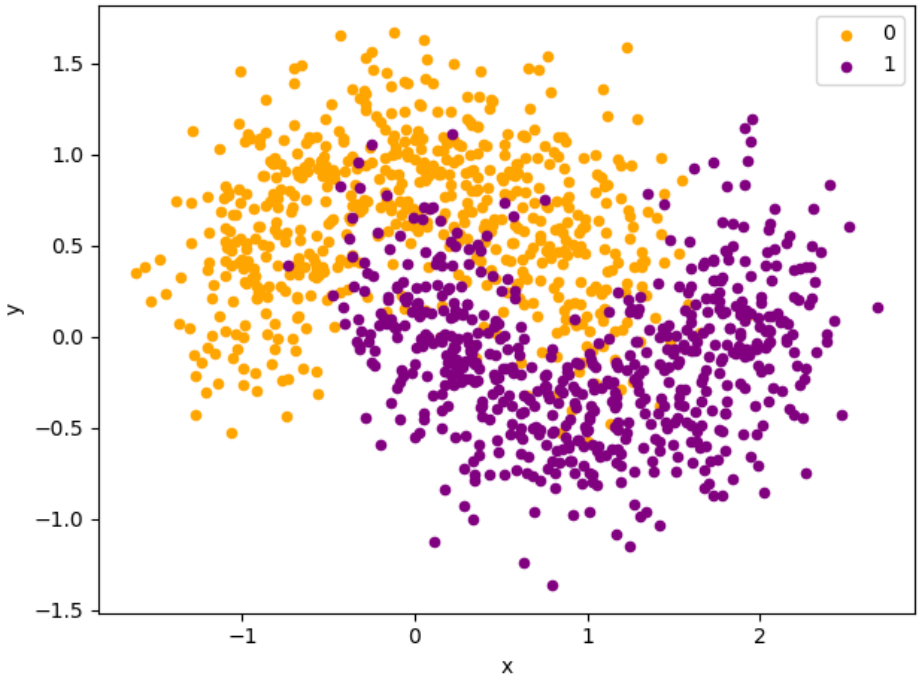
(b)make_moons

Цей датасет складається з набору n_samples двовимірних точок, що утворюють два півкола котрі напів перетинаються, та вектору з 0 та 1 що описує якому півколу(класу) належить точка. Параметр noise описує ступінь розкиданості прикладів в датасеті відносно ліній півкіл.

Input:
```
from pandas import DataFrame
moons_X, moons_y = datasets.make_moons(n_samples=1200, noise=0.3)

df = DataFrame(dict(x=moons_X[:, 0], y=moons_X[:, 1], label=moons_y))
colors = {0: 'orange', 1: 'purple'}
fig, ax = plt.subplots()
grouped = df.groupby('label')
for key, group in grouped:
```

```
    group.plot(ax=ax, kind='scatter', x='x', y='y', label=key, color=colors[key])
plt.show()
```

Output:



2. Розбити дані на навчальний і перевірочний набори

(a)load_digits

Input:

```
from sklearn.model_selection import train_test_split, GridSearchCV

digits_X_train, digits_X_test, digits_y_train, digits_y_test = train_test_split(
digits_X, digits_y, test_size=0.3, shuffle=False)
print(f"********Load digits**********\nTraining data shape:\n {digits_X_train.shape},
{digits_y_train.shape}\n"
                                f"Test data shape: \n {digits_X_test.shape}, {digits_y_test.shape}")
```

Output:

```
********Load digits**********
Training data shape:
 (1257, 64), (1257,)
Test data shape:
 (540, 64), (540,)
```

(b)make_moons

Input:

```
moons_X_train, moons_X_test, moons_y_train, moons_y_test = train_test_split(
                                moons_X, moons_y, test_size=0.3, shuffle=False)
print(f"********Make moons**********\nTraining data shape:\n {moons_X_train.shape}
{moons_y_train.shape}\n"
      f"Test data shape: \n {moons_X_test.shape} {moons_y_test.shape}")
```

Output:

```
********Make moons**********
```

```
Training data shape:
 (840, 2) (840,)
Test data shape:
 (360, 2) (360,)
```

3. Побудувати моделі класифікації або регресії згідно з варіантом

Припущення про поліноміальний розподіл не має сенсу для make_moons оскільки приклади цього набору даних - координати точок, що формують півкола що напів перетинаються - раціональні числа що можуть бути як від'ємними так і дробовими. MultinomialNB припускає що дані мають поліноміальний розподіл, який є узагальнення біноміального розподілу (цілі і невід'ємні значення). Отже для make_moons будемо розглядати лише припущення про нормальний розподіл і модель GaussianNB.

На наборі даних load_digits покажемо роботу обох моделей класифікації, оскільки обидві мають сенс. Кожен приклад у цьому наборі даних (що належить до одного з 10-ти класів) це зображення рукописної цифри розміру 8x8, кожен піксель якого закодований у вигляді беззнакового цілого числа(0-16). І хоча це і не є числом успіхів у незалежній схемі Бернулі у прямому сенсі, ми можемо розглядати ці числа як певну міру успішності(важливості) або частоти і застосувати припущення про поліноміальний розподіл. З припущенням про нормальний розподіл також не виникає проблем.

Input:
```
from sklearn.naive_bayes import GaussianNB, MultinomialNB
gaus_digits = GaussianNB()
gaus_digits.fit(digits_X_train, digits_y_train)

multinom_digits = MultinomialNB()
multinom_digits.fit(digits_X_train, digits_y_train)

gaus_moons = GaussianNB()
gaus_moons.fit(moons_X_train, moons_y_train)
```

4. Представити моделі графічно

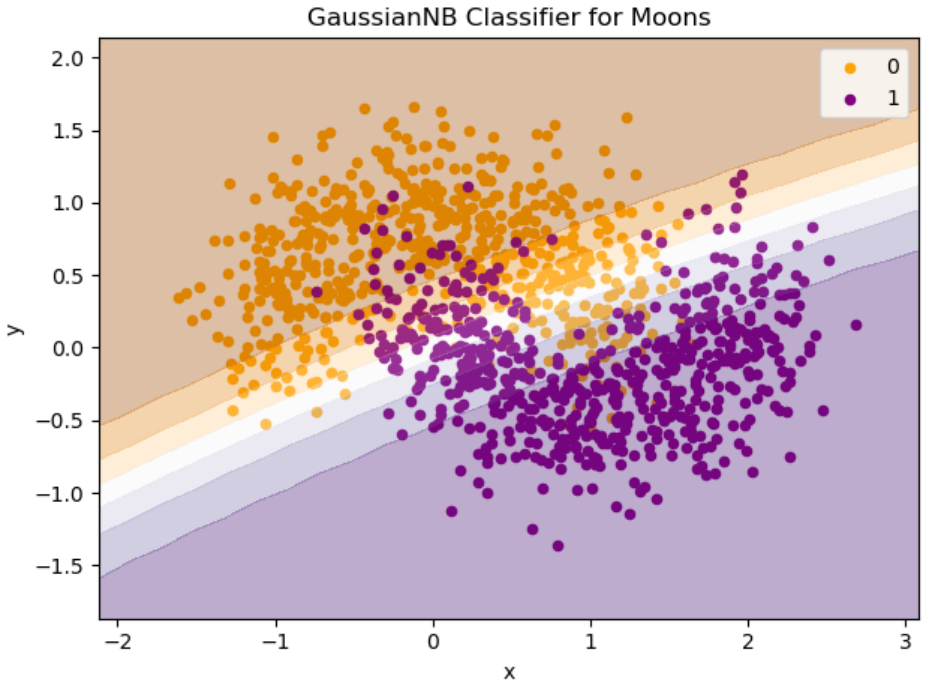
Make_moons

Для графічної візуалізації GaussianNB на make_moons використаємо апостеріорні ймовірності

Input:

```
 import numpy as np

x_min, x_max = moons_X[:, 0].min() - 0.5, moons_X[:, 0].max() + 0.5
y_min, y_max = moons_X[:, 1].min() - 0.5, moons_X[:, 1].max() + 0.5
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.2),
                     np.arange(y_min, y_max, 0.2))

df = DataFrame(dict(x=moons_X[:, 0], y=moons_X[:, 1], label=moons_y))
fig, ax = plt.subplots()
grouped = df.groupby('label')
for key, group in grouped:
  group.plot(ax=ax, kind='scatter', x='x', y='y', label=key, color=colors[key])
  Z = gaus_moons.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]
  Z = Z.reshape(xx.shape)
  ax.contourf(xx, yy, Z, alpha=0.2, cmap=plt.cm.PuOr)
  ax.set_title("GaussianNB Classifier for Moons")
plt.show()
```

Output:



Load_digits

Для цього датасету представляти моделі класифікації графічно не будемо, оскільки дані багатовимірні.

5. Виконати прогнози на основі моделей

Input:

```
gaus_digits_predicted = gaus_digits.predict(digits_X_test)
print("********Load digits**********\nGaussianNB \n test score: ",
    gaus_digits.score(digits_X_test, digits_y_test), "\n",
    f" train score: {gaus_digits.score(digits_X_train, digits_y_train)}")

multinom_digits_predicted = multinom_digits.predict(digits_X_test)
print("MultinomialNB \n test score: ", multinom_digits.score(digits_X_test, digits_y_test), "\n",
    f" train score: {multinom_digits.score(digits_X_train, digits_y_train)}")

gaus_moons_predicted = gaus_moons.predict(moons_X_test)
print("********Make moons**********\nGaussianNB\n test score: ",
gaus_moons.score(moons_X_test, moons_y_test),
    f"\n train score: {gaus_moons.score(moons_X_train, moons_y_train)}")
```

Output:

```
********Load digits**********
GaussianNB
 test score: 0.825925925925926
 train score: 0.8774860779634049
MultinomialNB
 test score: 0.8611111111111112
 train score: 0.9132856006364359
********Make moons**********
GaussianNB
 test score: 0.8666666666666667
 train score: 0.8742857142857143
```

6. Для кожної з моделей оцінити чи має місце перенавчання

Перенавчена модель має високу дисперсію. Для моделей з високою дисперсією ефективність моделі на перевірочному наборі даних є істотно гіршою за її ефективність на навчальній множині. Як бачимо з результатів виконання попереднього пункту перенавченими можуть бути обидві моделі на датасеті load_digits, поліноміальний наївний байєс на make-moons скоріше за все не є перенавченим. Детальніший аналіз проведемо побудувавши навчальні криві у пункті 10.

7. Розрахувати апостеріорні ймовірності

Input:

```
gaus_digits_predicted_probas = gaus_digits.predict_proba(digits_X_test)
gaus_moons_predicted_probas = gaus_moons.predict_proba(moons_X_test)
multinom_digits_predicted_probas = multinom_digits.predict_proba(digits_X_test)
```

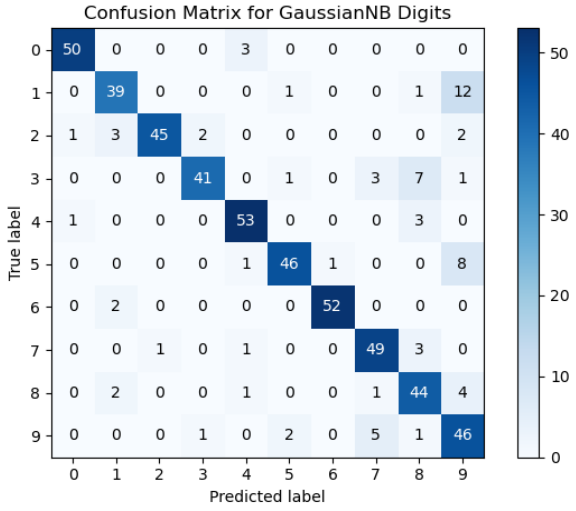
8. Розрахувати критерії якості для кожної з моделей :
   - Матрицю неточностей(confusion matrix)
   - Точність(precision)
   - Повноту (recall)
   - Міру F1(F1 score)
   - Побудувати криву точності-повноти(precision-recall(PR) curve), ROC-криву, показник AUC

Input:

```
def model_quality(clf_name, data_name, y_test, predicted, predicted_probas):
    skplt.metrics.plot_confusion_matrix(digits_y_test, multinom_digits_predicted,
                                        title=f'Confusion Matrix for {clf_name} {data_name}')
    plt.show()
    print(f"classiffication report for {clf_name} {data_name}\n------------------------------\n"
          f"{metrics.classification_report(y_test, predicted)}")
    skplt.metrics.plot_precision_recall(y_test, predicted_probas,
                                        title=f'Precision-Recall curves for {clf_name} {data_name}')
    plt.show()
    skplt.metrics.plot_roc(y_test, predicted_probas,
                           title=f'ROC curves for {clf_name} {data_name}')
```

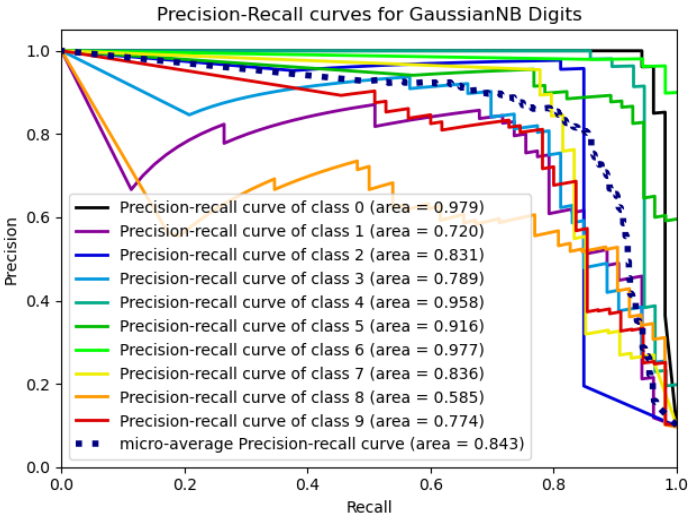
```
for clf_name, data_name, clf, X, y, \
    test, predicted, probas in [['GaussianNB', 'Digits', gaus_digits, digits_X, digits_y, digits_y_test,
                                 gaus_digits_predicted, gaus_digits_predicted_probas],
                                ['MultinomialNB', 'Digits', multinom_digits, digits_X, digits_y, digits_y_test,
                                 multinom_digits_predicted, multinom_digits_predicted_probas],
                                ['GaussianNB', 'Moons', gaus_moons, moons_X, moons_y, moons_y_test,
                                 gaus_moons_predicted, gaus_moons_predicted_probas]]:
    model_quality(clf_name, data_name, test, predicted, probas)
    print("*****************************************\n"
          " Tuning  Test Size for ", clf_name, "for", data_name, "\n")

    skplt.estimators.plot_learning_curve(clf, X, y, title=f'Learning Curve for {clf_name} {data_name}')
    plt.show()
    split_evaluating(clf, X, y, [0.2, 0.3, 0.4, 0.5])
```

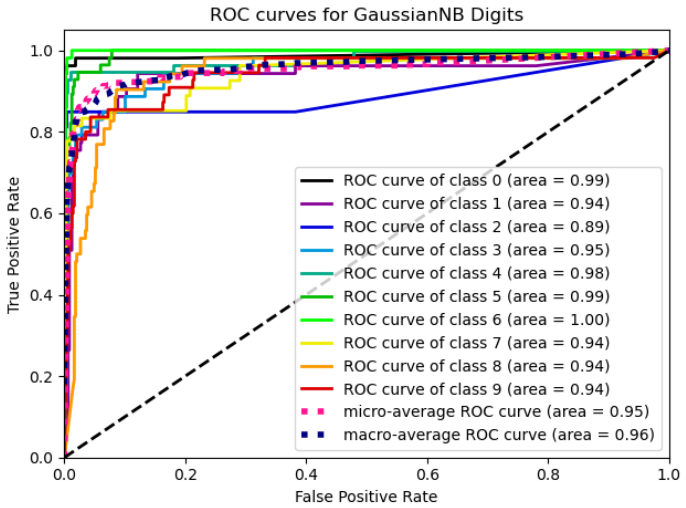
plt.show()

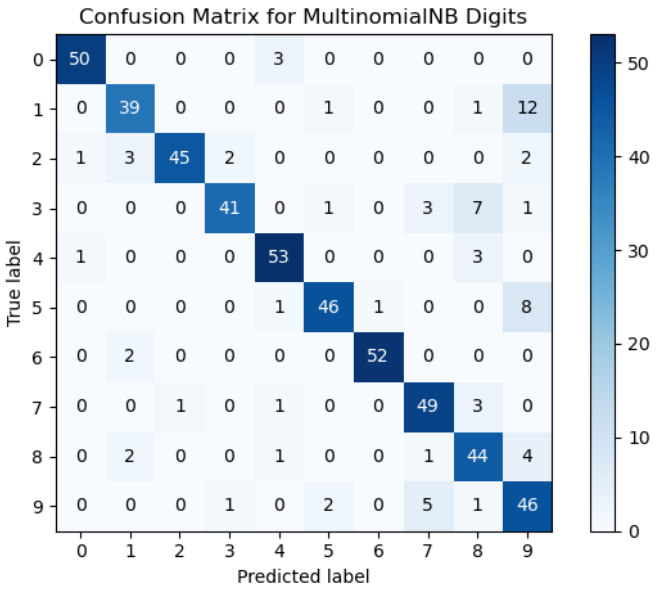



```
classiffication report for GaussianNB Digits
------------------------------
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        53
           1       0.65      0.79      0.71        53
           2       0.98      0.81      0.89        53
           3       0.92      0.64      0.76        53
           4       1.00      0.86      0.92        57
           5       0.83      0.93      0.87        56
           6       0.96      0.98      0.97        54
           7       0.73      0.83      0.78        54
           8       0.59      0.71      0.64        52
           9       0.82      0.73      0.77        55

    accuracy                           0.83       540
   macro avg       0.84      0.82      0.83       540
weighted avg       0.84      0.83      0.83       540
```

Output:



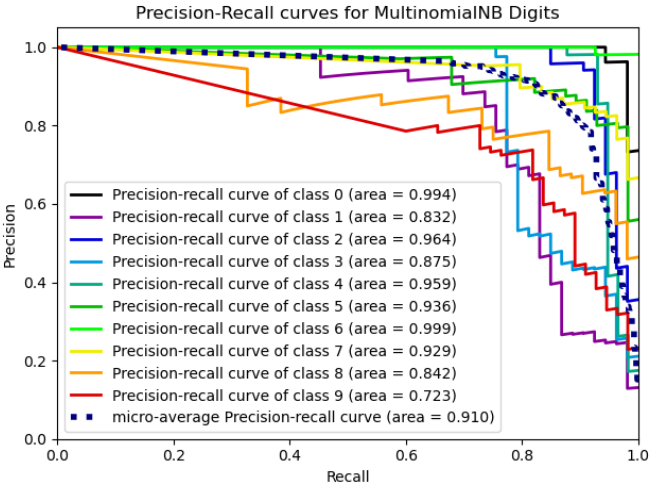


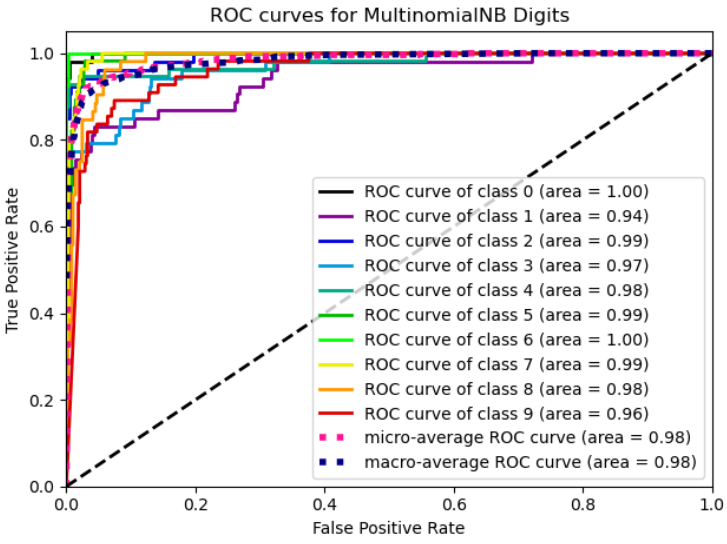


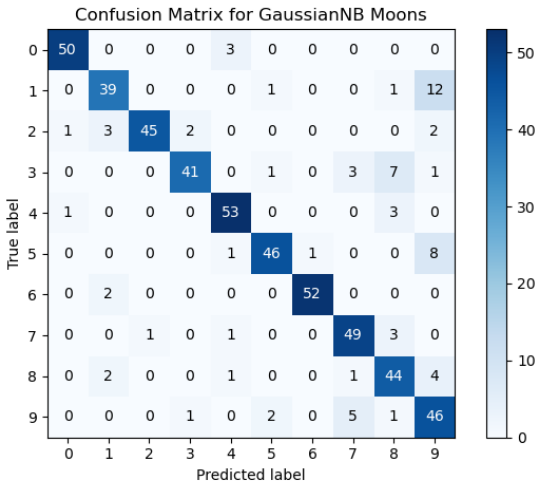
classiffication report for MultinomialNB Digits

---

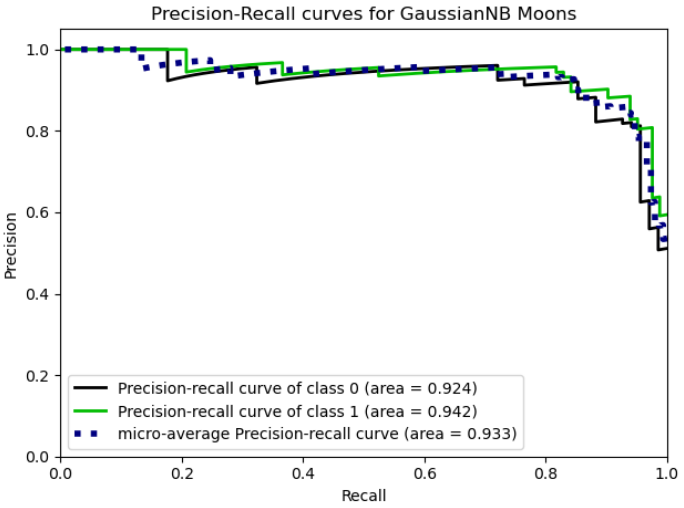
	precision	recall	f1-score	support
0	0.96	0.94	0.95	53
1	0.85	0.74	0.79	53
2	0.98	0.85	0.91	53
3	0.93	0.77	0.85	53
4	0.90	0.93	0.91	57
5	0.92	0.82	0.87	56
6	0.98	0.96	0.97	54
7	0.84	0.91	0.88	54
8	0.75	0.85	0.79	52
9	0.63	0.84	0.72	55
accuracy			0.86	540
macro avg	0.87	0.86	0.86	540
weighted avg	0.87	0.86	0.86	540

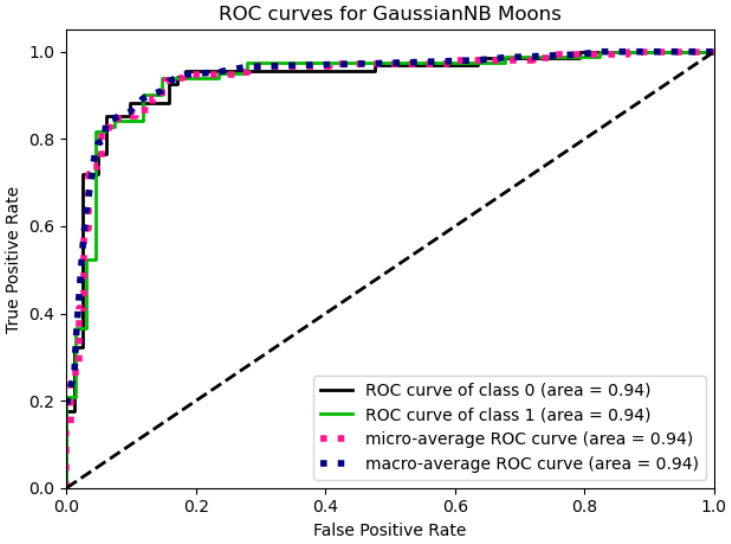







```
classiffication report for GaussianNB Moons
--------------------------------
              precision    recall  f1-score   support

           0       0.82      0.85      0.83       182
           1       0.84      0.81      0.82       178

    accuracy                           0.83       360
   macro avg       0.83      0.83      0.83       360
weighted avg       0.83      0.83      0.83       360
```





9. Виконати перехресну перевірку і решітчастий пошук для підбору гіперпараметрів моделей

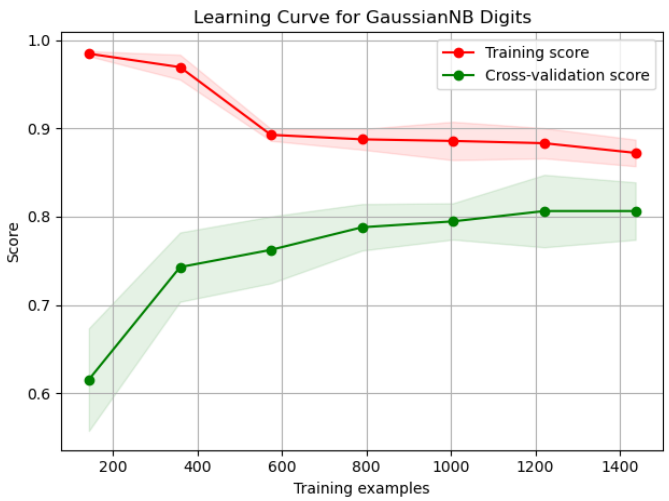
Для мого варіанту не має сенсу, оскільки у моделей GaussianNB та MultinomialNB відсутні гіперпараметри

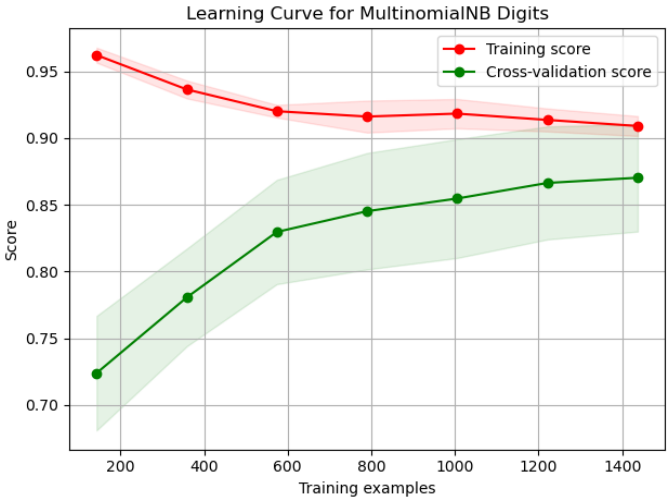
10. Зробити висновки про якість роботи моделей на досліджених даних.Для кожної навчальної вибірки вибрати найкращу модель

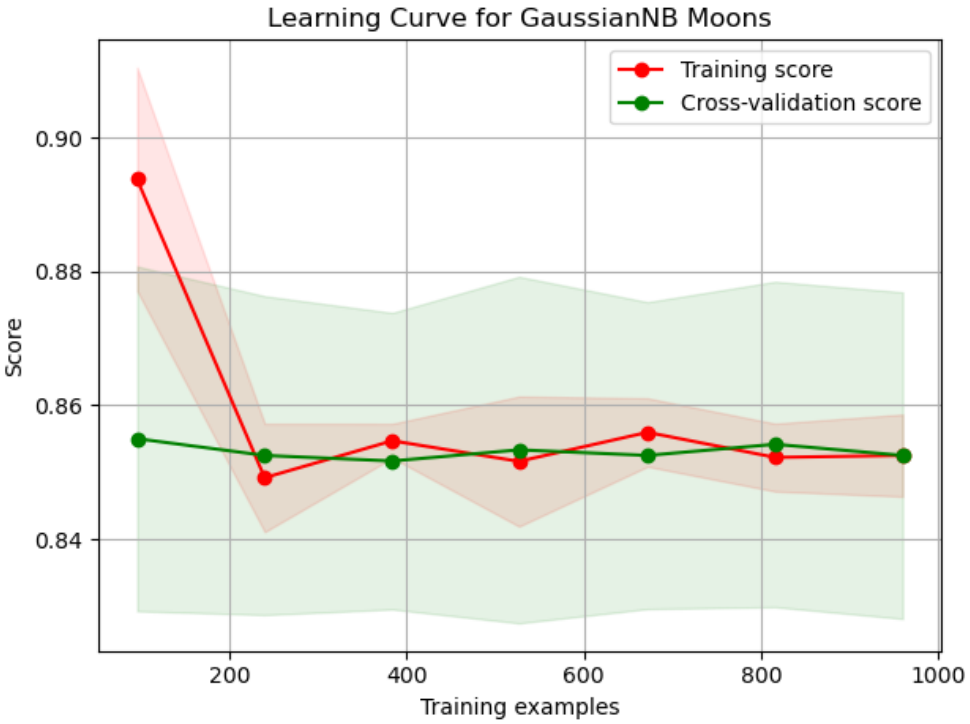
Основне підґрунтя для висновків представлене у пункті 8. Доповнимо оцінки якості моделей побудовою навчальних кривих. У циклі з п. 8 викличемо:

Input: skplt.estimators.plot_learning_curve(clf, X, y, title=**f'Learning Curve for** {clf_name} {data_name}**'**,
train_sizes=np.linspace(.1, 1, 7))

Output:







(a)load_digits

З середніх показників AUC ROC-кривої, PR-кривої та міри f1 можемо зробити висновок що модель MultinomialNB більше підходить для класифікації цього набору даних ніж GaussianNB. Проте у обох моделей непогані якісні показники.

Розглянемо навчальні криві. Дисперсія GaussianNB доволі суттєва і криві міри якості класифікації тренувального і перевіркового наборів так і не перетнулися навіть при використанні майже усього датасету для навчання, що свідчить про певну ступінь непідходящості даної моделі для

даного набору даних, має місце перенавчання моделі. І оскільки графіки йдуть майже паралельно під кінець, покращити гнучкість моделі збільшивши кількість навчальних прикладів не вдасться.

У MultinomialNB дисперсія помітно менше, загальна якість моделі краща, задовільна якість досягається вже при ~1100 навчальних прикладах, під кінець обидві криві перетинаються, тому вибираємо MultinomialNB для цього набору даних

(б)make_moons

Єдиний вибір класифікатора для цього класифікатора обґрунтовано у п.3. З середніх показників AUC ROC-кривої, PR-кривої та міри f1 можемо зробити висновок що модель GaussianlNB на задовільному рівні справляється з задачею класифікації на цьому наборі даних.

Розглянемо навчальні криві. Вже при ~220 навчальних прикладах досягається стабільна якість моделі і вона далі зі збільшенням кількості навчальних прикладів коливається в дуже незначних межах . Перенавчання відсутнє.

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

Використаємо навчальні криві з п.10 та функцію, яку викличемо в циклі з п. 8

Input:

```
def split_evaluating(clf, X, y, test_sizes):
    for size in test_sizes:
        X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size=size, shuffle=False)
        clf.fit(X_train, y_train)
        print(f"train score for test size = {size}\n{clf.score(X_train, y_train)}"
              f"\n---------------------------------------\n")
    return 0
```

Output:

```
*******************************************
 Tuning  Test Size for  GaussianNB for Digits

train score for test size = 0.2
0.8803061934585943
---------------------------------------

train score for test size = 0.3
```

```
0.8774860779634049
---------------------------------------

train score for test size = 0.4
0.8608534322820037
---------------------------------------

train score for test size = 0.5
0.8585746102449888
---------------------------------------

*******************************************
 Tuning  Test Size for  MultinomialNB for Digits

train score for test size = 0.2
0.9151009046624913
---------------------------------------

train score for test size = 0.3
0.9132856006364359
---------------------------------------

train score for test size = 0.4
0.9090909090909091
---------------------------------------

train score for test size = 0.5
0.9020044543429844
---------------------------------------

*******************************************
 Tuning  Test Size for  MultinomialNB for Digits

train score for test size = 0.2
0.9151009046624913
---------------------------------------

train score for test size = 0.3
0.9132856006364359
---------------------------------------

train score for test size = 0.4
0.9090909090909091
---------------------------------------

train score for test size = 0.5
0.9020044543429844
---------------------------------------
```

Для make_moons після ~220 розмір навчальної множини практично не впливає на якість моделі. До 220 модель зі збільшенням навчальної множини покращується

Для load_digits GaussianNB має місце перенавчання. Зі збільшенням розміру навчальної вибірки після 100 якість моделі на навчальних даних падає, а на тестових росте, проте розрив між ними занадто великий і вони так і не виходять на приблизно однаковий рівень, навіть при використанні для навчання майже усього датасету.

Для load_digits MultinomialNB ситуація схожа, проте початковий розрив менше і загальна якість моделі краща.