НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Лабораторна робота №2 з дисципліни

«Програмування інтелектуальних інформаційних систем»

Прийняв

аспірант кафедри ІПІ

Баришич Л.М.

Виконав

студент групи ІП-12

Казаков В.О.

“19” вересня 2023 р.

Київ – 2023

Імпорт необхідних модулів:

import pandas as pd

import seaborn as sns

import scipy as sp

import warnings

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans, Birch, AgglomerativeClustering, DBSCAN, AffinityPropagation

from sklearn import cluster, datasets, mixture

from sklearn.naive\_bayes import GaussianNB

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.metrics import adjusted\_rand\_score, classification\_report, confusion\_matrix, normalized\_mutual\_info\_score, \

silhouette\_score, ConfusionMatrixDisplay, confusion\_matrix

from sklearn.model\_selection import train\_test\_split

Функції:

def check\_null\_hypothesis(dataset):

alpha = 0.05

for column in dataset.columns:

s, p = sp.stats.normaltest(dataset[column])

if p > alpha:

print('The null hypothesis CANNOT be rejected for ' + column)

else:

print('The null hypothesis CAN be rejected for ' + column)

def confusion\_matrix\_plot(y\_true, y\_pred):

cm = confusion\_matrix(y\_true, y\_pred)

disp = ConfusionMatrixDisplay(confusion\_matrix=cm)

disp.plot(cmap='Greens')

plt.title('Confusion Matrix')

plt.show()

def calculate\_metrics(y\_true, y\_pred):

ari\_score = adjusted\_rand\_score(y\_true, y\_pred)

nmi\_score = normalized\_mutual\_info\_score(y\_true, y\_pred)

silhouette\_avg = silhouette\_score(y\_true.reshape(-1, 1), y\_pred)

print(f'Adjusted Rand Index score (ARI): {ari\_score:.3f}')

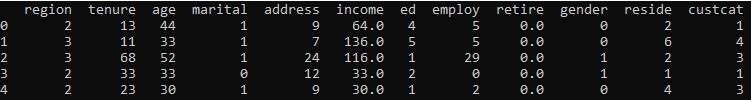
print(f'Normalized Mutual Information score (NMI): {nmi\_score:.3f}')

print(f'Silhouette Coefficient: {silhouette\_avg:.3f}')

Перший датасет:

first\_dataset = pd.read\_csv('teleCust1000t.csv')

print(first\_dataset.head())

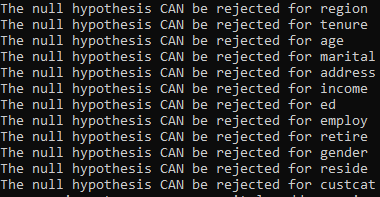


Датасети для третього та четвертого завдань:

X1, Y1 = datasets.make\_moons(n\_samples=2000, noise=.09, random\_state=42)

Перевірка нульової гіпотези для першого датасету:

check\_null\_hypothesis(first\_dataset)



Розділення першого датасету на тренувальну та тестову вибірки:

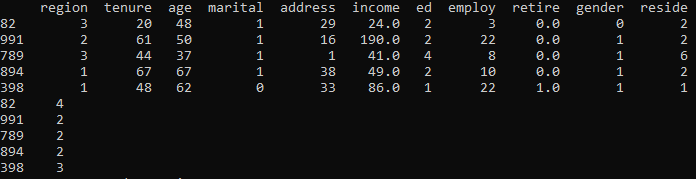
X = first\_dataset.drop(['custcat'], axis=1)

y = first\_dataset['custcat']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42)

print(x\_train.head())

print(y\_train.head())



Bayesian Classification:

gaussian\_nb = GaussianNB()

gaussian\_nb.fit(x\_train, y\_train)

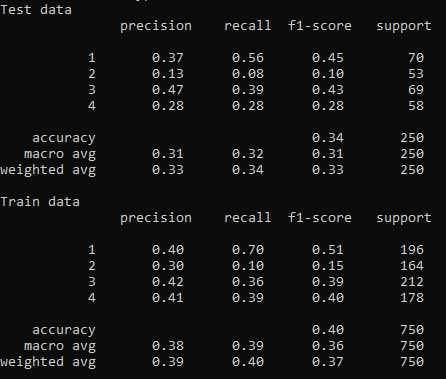
predictions\_test = gaussian\_nb.predict(x\_test)

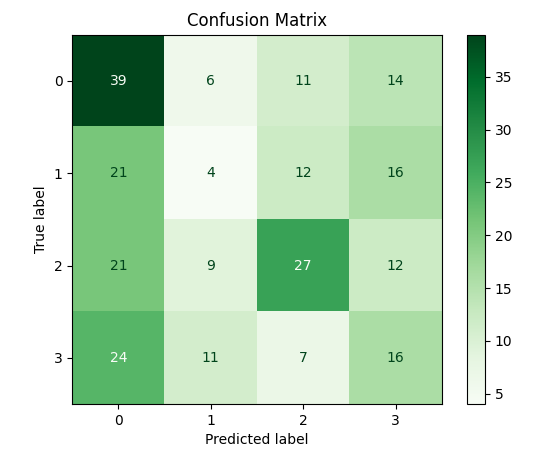
predictions\_train = gaussian\_nb.predict(x\_train)

print("Test data\n", classification\_report(y\_test, predictions\_test))

print("Train data\n", classification\_report(y\_train, predictions\_train))

confusion\_matrix\_plot(y\_test, predictions\_test)





Support Vector Machine:

svm\_model = SVC()

svm\_model.fit(x\_train, y\_train)

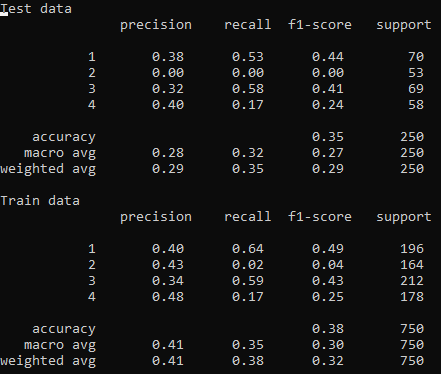
predictions\_test = svm\_model.predict(x\_test)

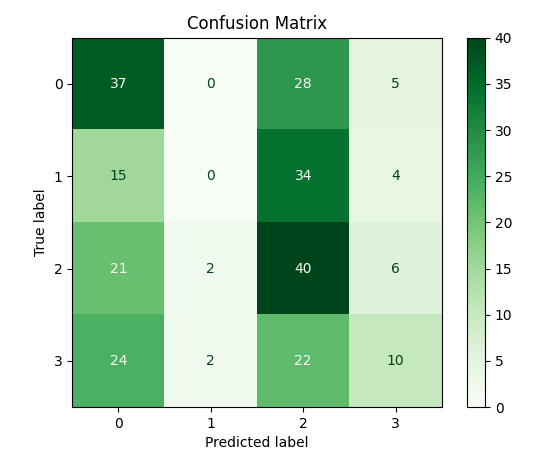
predictions\_train = svm\_model.predict(x\_train)

print("Test data\n", classification\_report(y\_test, predictions\_test))

print("Train data\n", classification\_report(y\_train, predictions\_train))

confusion\_matrix\_plot(y\_test, predictions\_test)





Minkowski:

knn = KNeighborsClassifier(6, metric='minkowski')

knn.fit(X=x\_train, y=y\_train)

predictions\_train = knn.predict(x\_train)

predictions\_test = knn.predict(x\_test)

print("Test data\n", classification\_report(y\_test, predictions\_test))

print("Train data\n", classification\_report(y\_train, predictions\_train))

confusion\_matrix\_plot(y\_test, predictions\_test)

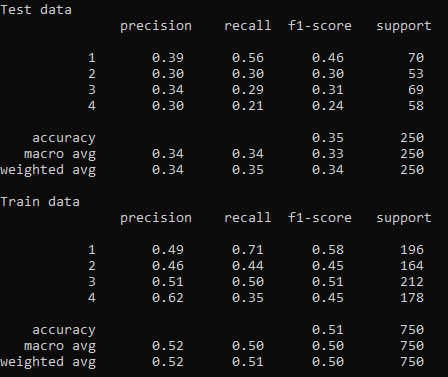
plt.scatter(x\_test['tenure'], x\_test['income'], marker='.', c=predictions\_test)

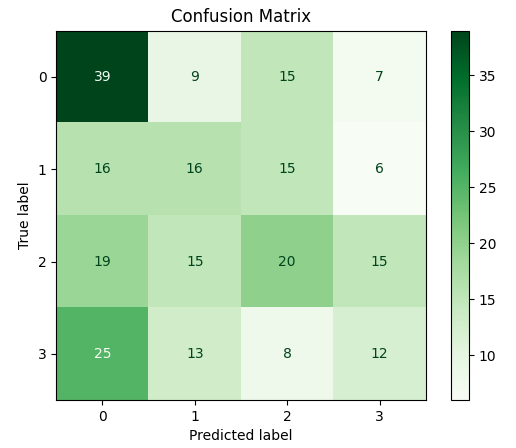
plt.ylabel('Income')

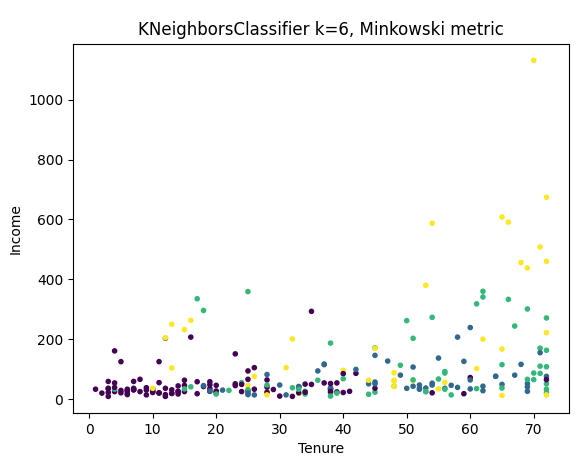
plt.xlabel('Tenure')

plt.title('KNeighborsClassifier k=6, Minkowski metric')

plt.show()







Euclidean:  
knn = KNeighborsClassifier(6, metric='euclidean')

knn.fit(X=x\_train, y=y\_train)

predictions\_train = knn.predict(x\_train)

predictions\_test = knn.predict(x\_test)

print("Test data\n", classification\_report(y\_test, predictions\_test))

print("Train data\n", classification\_report(y\_train, predictions\_train))

confusion\_matrix\_plot(y\_test, predictions\_test)

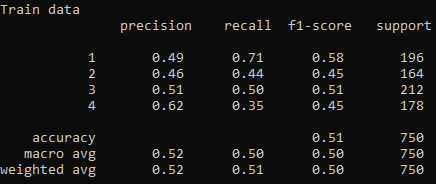
plt.scatter(x\_test['tenure'], x\_test['income'], marker='.', c=predictions\_test)

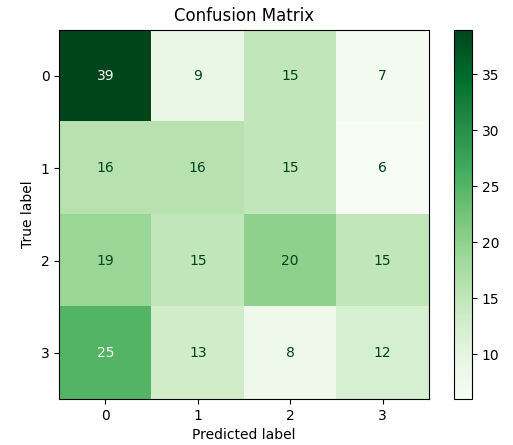
plt.ylabel('Income')

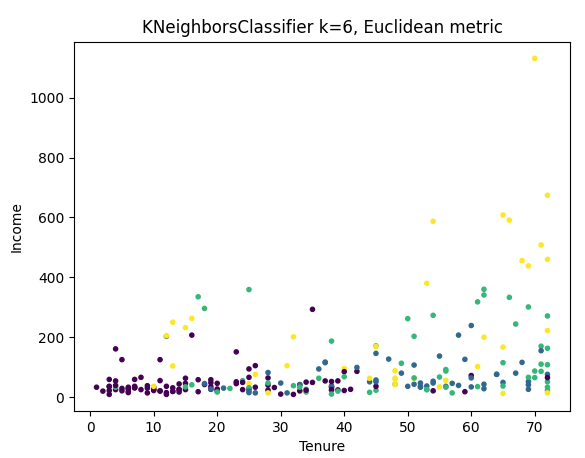
plt.xlabel('Tenure')

plt.title('KNeighborsClassifier k=6, Euclidean metric')

plt.show()







Manhattan:

knn = KNeighborsClassifier(6, metric='manhattan')

knn.fit(X=x\_train, y=y\_train)

predictions\_train = knn.predict(x\_train)

predictions\_test = knn.predict(x\_test)

print("Test data\n", classification\_report(y\_test, predictions\_test))

print("Train data\n", classification\_report(y\_train, predictions\_train))

confusion\_matrix\_plot(y\_test, predictions\_test)

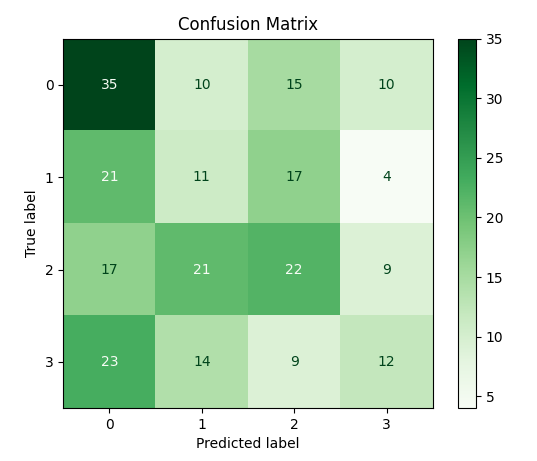
plt.scatter(x\_test['tenure'], x\_test['income'], marker='.', c=predictions\_test)

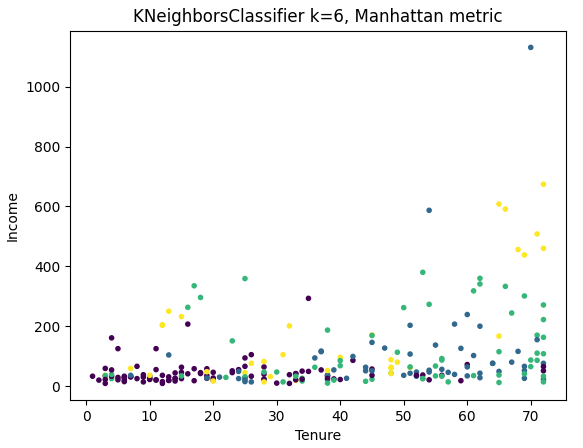
plt.ylabel('Income')

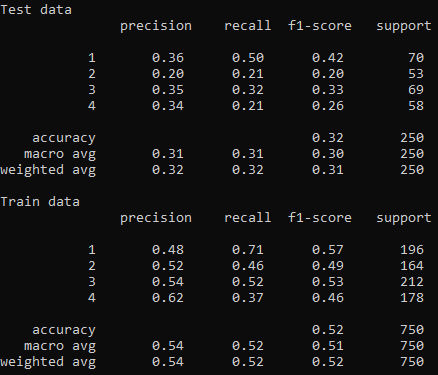
plt.xlabel('Tenure')

plt.title('KNeighborsClassifier k=6, Manhattan metric')

plt.show()







Agnes:

agnes = AgglomerativeClustering(n\_clusters=2)

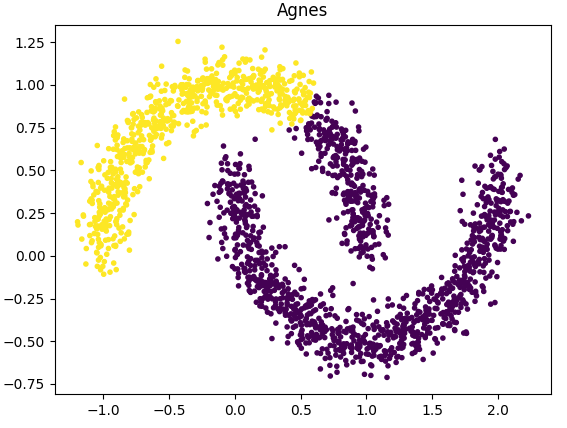
y\_agnes = agnes.fit\_predict(X1)

plt.scatter(X1[:, 0], X1[:, 1], marker='.', c=y\_agnes)

plt.title('Agnes')

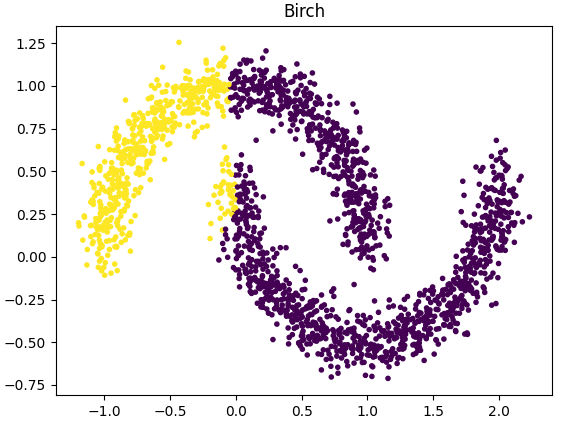
plt.show()

calculate\_metrics(y\_agnes, Y1)



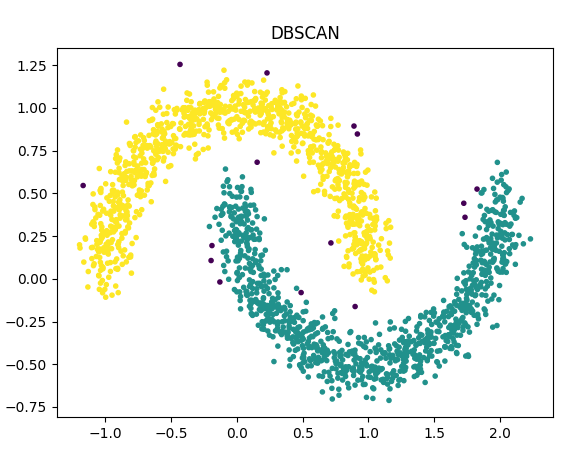


Birch:





DBSCAN:





Affinity Propagation:

affinitypropagation = AffinityPropagation(damping=0.95)

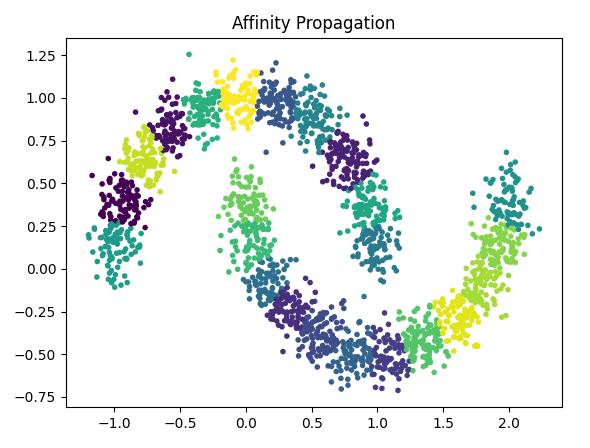
y\_affinitypropagation = affinitypropagation.fit\_predict(X1)

plt.scatter(X1[:, 0], X1[:, 1], marker='.', c=y\_affinitypropagation)

plt.title('Affinity Propagation')

plt.show()

calculate\_metrics(y\_affinitypropagation, Y1)





K-Means:

kmeans = KMeans(n\_clusters=2, n\_init=10)

y\_kmeans = kmeans.fit\_predict(X1)

plt.scatter(X1[:, 0], X1[:, 1], marker='.', c=y\_kmeans)

plt.title('K-means')

plt.show()

calculate\_metrics(y\_kmeans, Y1)

