**ЛАБОРАТОРНА РОБОТА № 2**

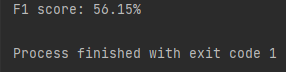
**ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ**

**Мета роботи:** використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити різні методи класифікації даних та навчитися їх порівнювати.

**Хід роботи**

**Завдання 1. Класифікація за допомогою машин опорних векторів (SVM)**

import numpy as np  
from sklearn import preprocessing  
from sklearn.svm import LinearSVC  
from sklearn.multiclass import OneVsOneClassifier  
from sklearn.model\_selection import train\_test\_split, cross\_val\_score  
  
input\_file = 'income\_data.txt'  
  
X = []  
y = []  
count\_class1 = 0  
count\_class2 = 0  
max\_datapoints = 25000  
  
with open(input\_file, 'r') as f:  
 for line in f.readlines():  
 if count\_class1 >= max\_datapoints and count\_class2 >= max\_datapoints:  
 break  
  
 if '?' in line:  
 continue  
  
 data = line[:-1].split(', ')  
  
 if data[-1] == '<=50K' and count\_class1 < max\_datapoints:  
 X.append(data)  
 count\_class1 += 1  
  
 if data[-1] == '>50K' and count\_class2 < max\_datapoints:  
 X.append(data)  
 count\_class2 += 1  
  
X = np.array(X)  
  
# Перетворення рядкових даних на числові  
label\_encoder = []  
X\_encoded = np.empty(X.shape)  
for i, item in enumerate(X[0]):  
 if item.isdigit():  
 X\_encoded[:, i] = X[:, i]  
 else:  
 label\_encoder.append(preprocessing.LabelEncoder())  
 X\_encoded[:, i] = label\_encoder[-1].fit\_transform(X[:, i])  
  
X = X\_encoded[:, :-1].astype(int)  
y = X\_encoded[:, -1].astype(int)  
  
classifier = OneVsOneClassifier(LinearSVC(random\_state=0))  
  
classifier.fit(X, y)  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
classifier = OneVsOneClassifier(LinearSVC(random\_state=0))  
classifier.fit(X\_train, y\_train)  
y\_test\_pred = classifier.predict(X\_test)  
  
f1 = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)  
print("F1 score: " + str(round(100 \* f1.mean(), 2)) + "%")  
  
input\_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White',  
 'Male', '0', '0', '40', 'United-States']  
  
input\_data\_encoded = [-1] \* len(input\_data)  
count = 0  
for i, item in enumerate(input\_data):  
 if item.isdigit():  
 input\_data\_encoded[i] = int(input\_data[i])  
 else:  
 input\_data\_encoded[i] = int(label\_encoder[count].transform(input\_data[i]))  
 count += 1  
  
input\_data\_encoded = np.array(input\_data\_encoded)  
  
predicted\_class = classifier.predict(input\_data\_encoded)  
print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])



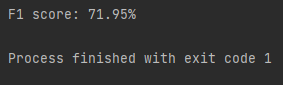
**Завдання 2. Порівняння якості класифікаторів SVM з нелінійними ядрами**

Поліноміальне ядро

import numpy as np  
from sklearn import preprocessing  
from sklearn.svm import SVC  
from sklearn.multiclass import OneVsOneClassifier  
from sklearn.model\_selection import train\_test\_split, cross\_val\_score  
  
input\_file = 'income\_data.txt'  
  
X = []  
y = []  
count\_class1 = 0  
count\_class2 = 0  
max\_datapoints = 10000  
  
with open(input\_file, 'r') as f:  
 for line in f.readlines():  
 if count\_class1 >= max\_datapoints and count\_class2 >= max\_datapoints:  
 break  
  
 if '?' in line:  
 continue  
  
 data = line[:-1].split(', ')  
  
 if data[-1] == '<=50K' and count\_class1 < max\_datapoints:  
 X.append(data)  
 count\_class1 += 1  
  
 if data[-1] == '>50K' and count\_class2 < max\_datapoints:  
 X.append(data)  
 count\_class2 += 1  
  
X = np.array(X)  
  
# Перетворення рядкових даних на числові  
label\_encoder = []  
X\_encoded = np.empty(X.shape)  
for i, item in enumerate(X[0]):  
 if item.isdigit():  
 X\_encoded[:, i] = X[:, i]  
 else:  
 label\_encoder.append(preprocessing.LabelEncoder())  
 X\_encoded[:, i] = label\_encoder[-1].fit\_transform(X[:, i])  
  
X = X\_encoded[:, :-1].astype(int)  
y = X\_encoded[:, -1].astype(int)  
  
classifier = OneVsOneClassifier(SVC(random\_state=0, kernel='poly', degree=8))  
  
classifier.fit(X, y)  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
classifier = OneVsOneClassifier(SVC(random\_state=0, kernel='poly', degree=8))  
classifier.fit(X\_train, y\_train)  
y\_test\_pred = classifier.predict(X\_test)  
  
f1 = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)  
print("F1 score: " + str(round(100 \* f1.mean(), 2)) + "%")  
  
input\_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White',  
 'Male', '0', '0', '40', 'United-States']  
  
input\_data\_encoded = [-1] \* len(input\_data)  
count = 0  
for i, item in enumerate(input\_data):  
 if item.isdigit():  
 input\_data\_encoded[i] = int(input\_data[i])  
 else:  
 input\_data\_encoded[i] = int(label\_encoder[count].transform(input\_data[i]))  
 count += 1  
  
input\_data\_encoded = np.array(input\_data\_encoded)  
  
predicted\_class = classifier.predict(input\_data\_encoded)  
print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])

Гаусове ядро

import numpy as np  
from sklearn import preprocessing  
from sklearn.svm import SVC  
from sklearn.multiclass import OneVsOneClassifier  
from sklearn.model\_selection import train\_test\_split, cross\_val\_score  
  
input\_file = 'income\_data.txt'  
  
X = []  
y = []  
count\_class1 = 0  
count\_class2 = 0  
max\_datapoints = 25000  
  
with open(input\_file, 'r') as f:  
 for line in f.readlines():  
 if count\_class1 >= max\_datapoints and count\_class2 >= max\_datapoints:  
 break  
  
 if '?' in line:  
 continue  
  
 data = line[:-1].split(', ')  
  
 if data[-1] == '<=50K' and count\_class1 < max\_datapoints:  
 X.append(data)  
 count\_class1 += 1  
  
 if data[-1] == '>50K' and count\_class2 < max\_datapoints:  
 X.append(data)  
 count\_class2 += 1  
  
X = np.array(X)  
  
# Перетворення рядкових даних на числові  
label\_encoder = []  
X\_encoded = np.empty(X.shape)  
for i, item in enumerate(X[0]):  
 if item.isdigit():  
 X\_encoded[:, i] = X[:, i]  
 else:  
 label\_encoder.append(preprocessing.LabelEncoder())  
 X\_encoded[:, i] = label\_encoder[-1].fit\_transform(X[:, i])  
  
X = X\_encoded[:, :-1].astype(int)  
y = X\_encoded[:, -1].astype(int)  
  
classifier = OneVsOneClassifier(SVC(random\_state=0, kernel='rbf'))  
  
classifier.fit(X, y)  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
classifier = OneVsOneClassifier(SVC(random\_state=0, kernel='rbf'))  
classifier.fit(X\_train, y\_train)  
y\_test\_pred = classifier.predict(X\_test)  
  
f1 = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)  
print("F1 score: " + str(round(100 \* f1.mean(), 2)) + "%")  
  
input\_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White',  
 'Male', '0', '0', '40', 'United-States']  
  
input\_data\_encoded = [-1] \* len(input\_data)  
count = 0  
for i, item in enumerate(input\_data):  
 if item.isdigit():  
 input\_data\_encoded[i] = int(input\_data[i])  
 else:  
 input\_data\_encoded[i] = int(label\_encoder[i].transform(input\_data[i]))  
 count += 1  
  
input\_data\_encoded = np.array(input\_data\_encoded)  
  
predicted\_class = classifier.predict(input\_data\_encoded)  
print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])



Сигмоїдальне ядро

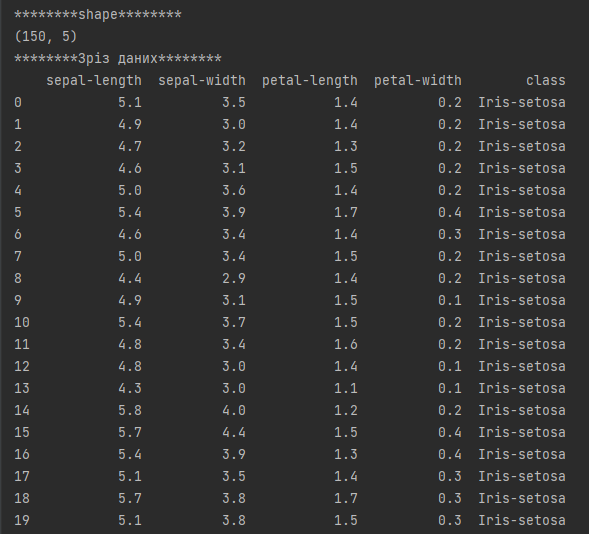
import numpy as np  
from sklearn import preprocessing  
from sklearn.svm import SVC  
from sklearn.multiclass import OneVsOneClassifier  
from sklearn.model\_selection import train\_test\_split, cross\_val\_score  
  
input\_file = 'income\_data.txt'  
  
X = []  
y = []  
count\_class1 = 0  
count\_class2 = 0  
max\_datapoints = 25000  
  
with open(input\_file, 'r') as f:  
 for line in f.readlines():  
 if count\_class1 >= max\_datapoints and count\_class2 >= max\_datapoints:  
 break  
  
 if '?' in line:  
 continue  
  
 data = line[:-1].split(', ')  
  
 if data[-1] == '<=50K' and count\_class1 < max\_datapoints:  
 X.append(data)  
 count\_class1 += 1  
  
 if data[-1] == '>50K' and count\_class2 < max\_datapoints:  
 X.append(data)  
 count\_class2 += 1  
  
X = np.array(X)  
  
# Перетворення рядкових даних на числові  
label\_encoder = []  
X\_encoded = np.empty(X.shape)  
for i, item in enumerate(X[0]):  
 if item.isdigit():  
 X\_encoded[:, i] = X[:, i]  
 else:  
 label\_encoder.append(preprocessing.LabelEncoder())  
 X\_encoded[:, i] = label\_encoder[-1].fit\_transform(X[:, i])  
  
X = X\_encoded[:, :-1].astype(int)  
y = X\_encoded[:, -1].astype(int)  
  
classifier = OneVsOneClassifier(SVC(random\_state=0, kernel='sigmoid'))  
  
classifier.fit(X, y)  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
classifier.fit(X\_train, y\_train)  
y\_test\_pred = classifier.predict(X\_test)  
  
f1 = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)  
print("F1 score: " + str(round(100 \* f1.mean(), 2)) + "%")  
  
input\_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White',  
 'Male', '0', '0', '40', 'United-States']  
  
input\_data\_encoded = [-1] \* len(input\_data)  
count = 0  
for i, item in enumerate(input\_data):  
 if item.isdigit():  
 input\_data\_encoded[i] = int(input\_data[i])  
 else:  
 input\_data\_encoded[i] = int(label\_encoder[i].transform(input\_data[i]))  
 count += 1  
  
input\_data\_encoded = np.array(input\_data\_encoded)  
  
predicted\_class = classifier.predict(input\_data\_encoded)  
print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])



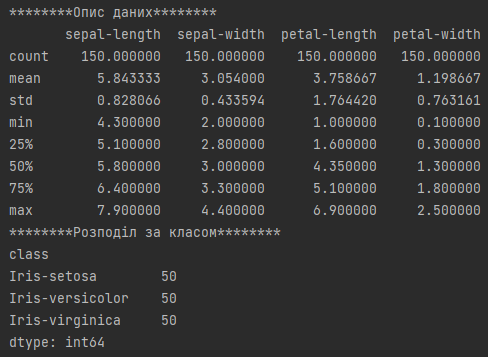
**Завдання 3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів**

from pandas import read\_csv  
from pandas.plotting import scatter\_matrix  
from matplotlib import pyplot  
from sklearn.model\_selection import train\_test\_split  
from sklearn.model\_selection import cross\_val\_score  
from sklearn.model\_selection import StratifiedKFold  
from sklearn.metrics import classification\_report  
from sklearn.metrics import confusion\_matrix  
from sklearn.metrics import accuracy\_score  
from sklearn.linear\_model import LogisticRegression  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.svm import SVC  
  
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"  
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width',  
 'class']  
dataset = read\_csv('url', names=names)  
# shape  
print("\*\*\*\*\*\*\*\*shape\*\*\*\*\*\*\*\*")  
print(dataset.shape)  
  
print("\*\*\*\*\*\*\*\*Зріз даних\*\*\*\*\*\*\*\*")  
print(dataset.head(20))  
  
# description  
print("\*\*\*\*\*\*\*\*Опис даних\*\*\*\*\*\*\*\*")  
print(dataset.describe())  
  
# classdistribution  
print("\*\*\*\*\*\*\*\*Розподіл за класом\*\*\*\*\*\*\*\*")  
print(dataset.groupby('class').size())  
  
# Діаграма розмаху  
dataset.plot(kind='box', subplots=True, layout=(2, 2), sharex=False, sharey=False)  
pyplot.show()  
  
# Гістограма розподілу атрибутів датасета  
dataset.hist()  
pyplot.show()  
  
# Матриця діаграм розсіювання  
scatter\_matrix(dataset)  
pyplot.show()  
  
# Розділення датасету  
array = dataset.values  
X = array[:, 0:4]  
y = array[:, 4]  
X\_train, X\_validation, Y\_train, Y\_validation = train\_test\_split(X, y, test\_size=0.20, random\_state=1)  
  
print("\*\*\*\*\*\*\*\*Оцінка моделей\*\*\*\*\*\*\*\*")  
models = []  
models.append(('LR', LogisticRegression(solver='liblinear', multi\_class='ovr')))  
models.append(('LDA', LinearDiscriminantAnalysis()))  
models.append(('KNN', KNeighborsClassifier()))  
models.append(('CART', DecisionTreeClassifier()))  
models.append(('NB', GaussianNB()))  
models.append(('SVM', SVC(gamma='auto')))  
# evaluate each model in turn  
results = []  
names = []  
for name, model in models:  
 kfold = StratifiedKFold(n\_splits=10, random\_state=1, shuffle=True)  
 cv\_results = cross\_val\_score(model, X\_train, Y\_train, cv=kfold, scoring='accuracy')  
 results.append(cv\_results)  
 names.append(name)  
 print('%s: %f (%f)' % (name, cv\_results.mean(), cv\_results.std()))  
  
# Порівняння алгоритмів  
pyplot.boxplot(results, labels=names)  
pyplot.title('Algorithm Comparison')  
pyplot.show()  
  
model = SVC(gamma='auto')  
model.fit(X\_train, Y\_train)  
predictions = model.predict(X\_validation)  
  
print("\*\*\*\*\*\*\*\*Оцінюємо прогноз\*\*\*\*\*\*\*\*")  
print(accuracy\_score(Y\_validation, predictions))  
print(confusion\_matrix(Y\_validation, predictions))  
print(classification\_report(Y\_validation, predictions))  
  
from sklearn.neighbors import KNeighborsClassifier  
import numpy as np  
  
knn = KNeighborsClassifier(n\_neighbors=1)  
  
knn.fit(X\_train, Y\_train)  
X\_new = np.array([[5, 2.9, 1, 0.2]])  
print("Форма масиву X\_new: {}".format(X\_new.shape))  
prediction = knn.predict(X\_new)  
print("Прогноз: {}".format(prediction))  
print("Спрогнозированная метка: {}".format(dataset['class']))  
  
y\_pred = knn.predict(X\_validation)  
print("Прогнози для тестового набору:\n {}".format(y\_pred))  
  
print("Правильність тестового набору: {:.2f}".format(np.mean(y\_pred == Y\_validation)))  
  
print("Правильність тестового набору: {:.2f}".format(knn.score(X\_validation, Y\_validation)))

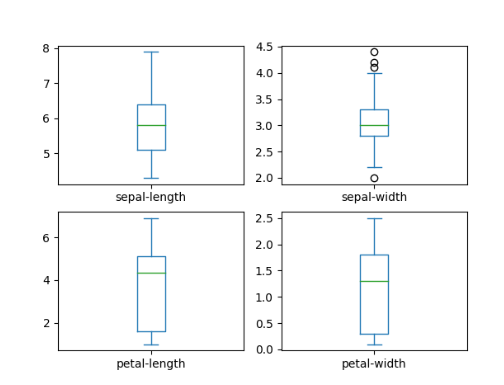
Форма масиву та зріз даних



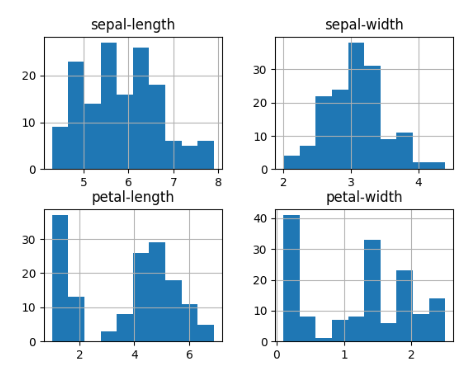
Статистичне зведення та розподіл за класом.



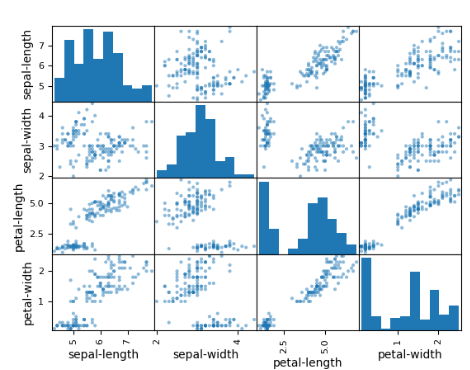
Діаграма розмаху



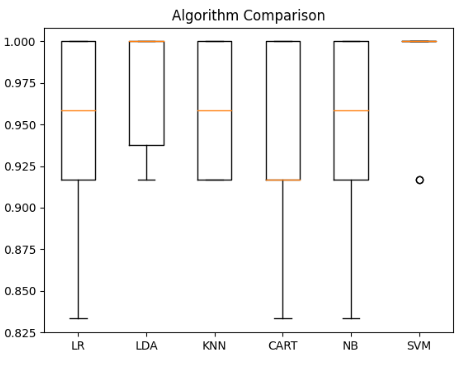
Гістограма розподілу атрибутів датасета

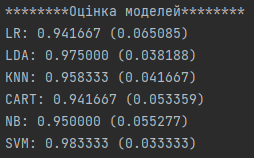


Матриця діаграм розсіювання



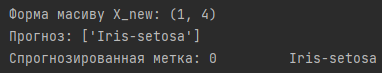
Порівняння алгоритмів





Найкращий результат має алгоритм SVM.

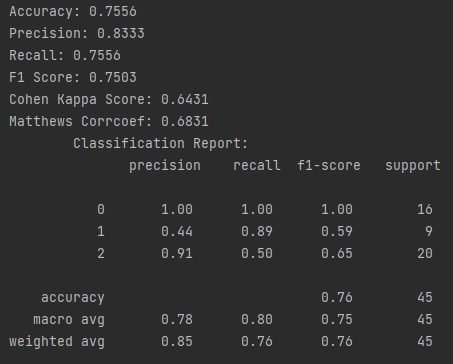
Мій прогноз

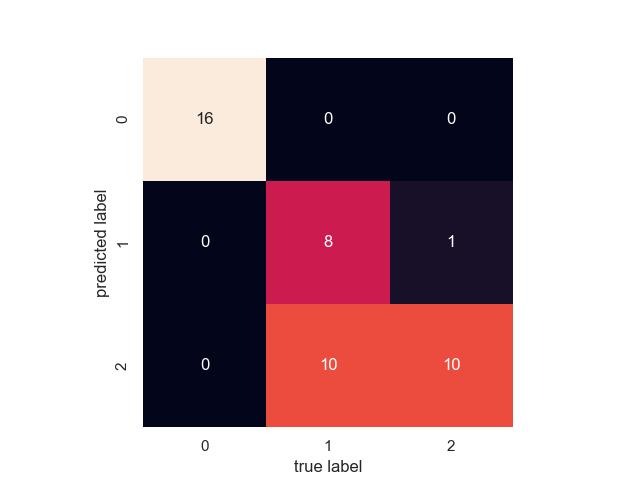


Квітка належить до класу Iris-setosa

**Завдання 4. Класифікація даних лінійним класифікатором Ridge**

import numpy as np  
from sklearn.datasets import load\_iris  
from sklearn.linear\_model import RidgeClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn import metrics  
from sklearn.metrics import confusion\_matrix  
from io import BytesIO  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
iris = load\_iris()  
X, y = iris.data, iris.target  
Xtrain, Xtest, ytrain, ytest = train\_test\_split(X, y, test\_size=0.3,  
 random\_state=0)  
clf = RidgeClassifier(tol=1e-2, solver="sag")  
clf.fit(Xtrain, ytrain)  
ypred = clf.predict(Xtest)  
  
print('Accuracy:', np.round(metrics.accuracy\_score(ytest, ypred), 4))  
print('Precision:', np.round(metrics.precision\_score(ytest, ypred, average='weighted'), 4))  
print('Recall:', np.round(metrics.recall\_score(ytest, ypred, average='weighted'), 4))  
print('F1 Score:', np.round(metrics.f1\_score(ytest, ypred, average='weighted'), 4))  
print('Cohen Kappa Score:',  
 np.round(metrics.cohen\_kappa\_score(ytest, ypred), 4))  
print('Matthews Corrcoef:',  
 np.round(metrics.matthews\_corrcoef(ytest, ypred), 4))  
print('\t\tClassification Report:\n',  
 metrics.classification\_report(ypred, ytest))  
  
sns.set()  
  
mat = confusion\_matrix(ytest, ypred)  
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)  
plt.xlabel('true label')  
plt.ylabel('predicted label')  
plt.savefig("Confusion.jpg")  
# Save SVG in a fake file object.  
f = BytesIO()  
plt.savefig(f, format="svg")





Ця картинка показує, що програма прогнозує 16 із 16 класів 0, 8 з 18 класів 1 та 10 з 11 класів 2.

Посилання на Git: <https://github.com/Grum74/AI>

**Висновок**

Я, використовуючи спеціалізовані бібліотеки та мову програмування Python дослідив різні методи класифікації даних та навчився їх порівнювати.