**ЛАБОРАТОРНА РОБОТА №2**

**ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ**

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

**Хід роботи:**

**GitHub:** https://github.com/ipz201svo/AI

**Завдання 1.1:** Випишіть у звіт всі 14 ознак з набору даних – їх назви та що вони позначають та вид (числові чи категоріальні).

- age – вік (числова)

- workclass – робочий клас (категоріальна)

- fnlwgt – вага вибірки (числова)

- education – освіта (категоріальна)

- education-enum – рівень освіти (категоріальна)

- material-status – сімейний стан (категоріальна)

- occupation – сфера зайнятості (категоріальна)

- relationship – відносини (категоріальна)

- race – раса (категоріальна)

- sex – стать (категоріальна)

- capital-gain – отриманий капітал (числова)

- capital-loss – втрачений капітал (числова)

- hours-per-week – кількість робочих годин на тиждень (числова)

- native-country – країна-походження (категоріальна)

**Завдання 1.2:** Написання програми для роботи з цими даними на основі SVM. Обчисліть значення інших показників якості класифікації (акуратність, повнота, точність) та разом з F1 занесіть їх у звіт.

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

import warnings

warnings.filterwarnings("ignore")

# Вхідний файл, який містить дані

input\_file = "income\_data.txt"

# Читання даних

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)

# Створення SVМ-класифікатора

classifier = OneVsOneClassifier(LinearSVC(random\_state=0))

# Навчання класифікатора

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)

# Обчислення F-міри для SVМ-класифікатора

f1 = cross\_val\_score(classifier, X, y, scoring="f1\_weighted", cv=3)

print("F1 score: " + str(round(100 \* f1.mean(), 2)) + "%")

# Передбачення результату для тестової точки даних

input\_data = [

    "37",

    "Private",

    "215646",

    "HS-grad",

    "9",

    "Never-married",

    "Handlers-cleaners",

    "Not-in-family",

    "White",

    "Male",

    "0",

    "0",

    "40",

    "United-States",

]

# Кодування тестової точки даних

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:

        transformed\_value = label\_encoder[count].transform([input\_data[i]])[0]

        input\_data\_encoded[i] = int(transformed\_value)

        count = count + 1

input\_data\_encoded = np.array(input\_data\_encoded).reshape(1, -1)

# Використання класифікатора для кодованої точки даних

# та виведення результату

predicted\_class = classifier.predict(input\_data\_encoded)

print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])

num\_folds = 3

accuracy\_values = cross\_val\_score(classifier, X, y, scoring="accuracy", cv=num\_folds)

print("Accuracy: " + str(round(100 \* accuracy\_values.mean(), 2)) + "%")

recall\_values = cross\_val\_score(

    classifier, X, y, scoring="recall\_weighted", cv=num\_folds

)

print("Recall: " + str(round(100 \* recall\_values.mean(), 2)) + "%")

precision\_values = cross\_val\_score(

    classifier, X, y, scoring="precision\_weighted", cv=num\_folds

)

print("Precision: " + str(round(100 \* precision\_values.mean(), 2)) + "%")



Рис. 1.1 – Результат виконання програми

Зображення, що містить текст, Шрифт, знімок екрана, число

Автоматично згенерований опис

Рис. 1.2 – Результат виконання програми

Тестова точка належить до першого класу (class1) так як має позначку “<=50K”, тобто має дохід менше ніж $50 000.

**Завдання 2.1:** Використовуючи набір даних та код з попереднього завдання створіть та дослідіть нелінійні класифікатори SVM з поліноміальним ядром.

classifier = OneVsOneClassifier(SVC(kernel='poly', degree=8))

Зображення, що містить текст, Шрифт, білий, дизайн

Автоматично згенерований опис

Рис. 2.1 – Результат виконання програми

**Завдання 2.2:** Використовуючи набір даних та код з попереднього завдання створіть та дослідіть нелінійні класифікатори SVM з гаусовим ядром.

classifier = OneVsOneClassifier(SVC(kernel="rbf"))

Зображення, що містить текст, Шрифт, дизайн, типографія

Автоматично згенерований опис

Рис. 2.2 – Результат виконання програми

**Завдання 2.3:** Використовуючи набір даних та код з попереднього завдання створіть та дослідіть нелінійні класифікатори SVM з сигмоїдальним ядром.

classifier = OneVsOneClassifier(SVC(kernel='sigmoid'))

Зображення, що містить текст, Шрифт, білий, дизайн

Автоматично згенерований опис

Рис. 2.3 – Результат виконання програми

**Завдання 2.4:** Опишіть який з видів SVM найкраще виконує завдання класифікації за результатами тренування.

Порівнявши результати трьох ядер для SVM, можна сказати що найкраще справилося поліноміальне ядро, хоча час виконання роботи був значно більшим.

**Завдання 3.1:** Завантаження та вивчення даних:

from sklearn.datasets import load\_iris

iris\_dataset = load\_iris()

print("Ключі iris\_dataset: \n{}".format(iris\_dataset.keys()))

print(iris\_dataset["DESCR"][:193] + "\n...")

print("Назви відповідей: {}".format(iris\_dataset["target\_names"]))

print("Назва ознак: \n{}".format(iris\_dataset["feature\_names"]))

print("Тип масиву data: {}".format(type(iris\_dataset["data"])))

print("Форма масиву data: {}".format(iris\_dataset["data"].shape))

print("Ознаки перших п'яти елементів:\n{}".format(iris\_dataset["data"][:5]))

print("Тип масиву target: {}".format(type(iris\_dataset["target"])))

print("Відповіді:\n{}".format(iris\_dataset["target"]))

Зображення, що містить текст, знімок екрана

Автоматично згенерований опис

Рис. 3.1 – Результат виконання програми

**Завдання 3.2:** Візуалізація даних.

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)

print(dataset.shape)

print(dataset.head(20))

print(dataset.describe())

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()

Зображення, що містить знімок екрана, текст, схема

Автоматично згенерований опис

Рис. 3.2 – Результат виконання програми

Зображення, що містить знімок екрана, схема, текст, квадрат

Автоматично згенерований опис

Рис. 3.3 – Результат виконання програми

Зображення, що містить знімок екрана

Автоматично згенерований опис

Рис. 3.4 – Результат виконання програми

**Завдання 3.3:** Створення навчального та тестового наборів.

# Розділення датасету на навчальну та контрольну вибірки

array = dataset.values

# Вибір перших 4-х стовпців

X = array[:, 0:4]

# Вибір 5-го стовпця

y = array[:, 4]

# Разделение X и y на обучающую и контрольную выборки

X\_train, X\_validation, Y\_train, Y\_validation = train\_test\_split(

    X, y, test\_size=0.20, random\_state=1

)

**Завдання 3.4:** Класифікація (побудова моделі).

# Завантажуємо алгоритми моделі

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")))

# оцінюємо модель на кожній ітерації

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()))

pyplot.boxplot(results, labels=names)

pyplot.title('Algorithm Comparison')

pyplot.show()

**Зображення, що містить текст, Шрифт, знімок екрана, типографія

Автоматично згенерований опис**

Рис. 3.4 – Результат виконання програми

**Зображення, що містить знімок екрана, текст, схема, дизайн

Автоматично згенерований опис**

Рис. 3.5 – Результат виконання програми

**Завдання 3.5:** Оптимізація параметрів моделі.

Поки що не потрібно турбуватися про інші параметри моделей, частково ми доторкнулися до цього кроку у попередньому завданні.

**Завдання 3.6:** Отримання прогнозу (передбачення на тренувальному наборі).

# Створюємо прогноз на контрольній вибірці

model = SVC(gamma='auto')

model.fit(X\_train, Y\_train)

predictions = model.predict(X\_validation)

**Завдання 3.7:** Отримання прогнозу (передбачення на тренувальному наборі).

# Оцінюємо прогноз

print(accuracy\_score(Y\_validation, predictions))

print(confusion\_matrix(Y\_validation, predictions))

print(classification\_report(Y\_validation, predictions))

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Рис. 3.6 – Результат виконання програми

**Завдання 3.8:** Отримання прогнозу (застосування моделі для передбачення).

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 массива X\_new: {}".format(X\_new.shape))

prediction = knn.predict(X\_new)

print("Прогноз: {}".format(prediction))

print("Оцінка тестового набору: {:.2f}".format(knn.score(X\_validation, Y\_validation)))

Зображення, що містить текст, Шрифт, білий, типографія

Автоматично згенерований опис

Рис. 3.7 – Результат виконання програми

Тобто квітка належить до класу «Iris-setosa». Крім того, нам вдалося досягти якості класифікації у 96.67%.

**Завдання 4:** Отримання прогнозу (застосування моделі для передбачення).

import numpy as np

from sklearn import preprocessing

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.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

import warnings

warnings.filterwarnings("ignore")

# Вхідний файл, який містить дані

input\_file = "income\_data.txt"

# Читання даних

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)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)

# Завантажуємо алгоритми моделі

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')))

# оцінюємо модель на кожній ітерації

results = []

names = []

for name, model in models:

    kfold = StratifiedKFold(n\_splits=10, random\_state=1, shuffle=True)

    print('%s: ' % name)

    f1 = cross\_val\_score(model, X, y, scoring='f1\_weighted', cv=3)

    print("F1 score: " + str(round(100 \* f1.mean(), 2)) + "%")

    accuracy\_values = cross\_val\_score(model, X, y, scoring='accuracy', cv=3)

    print("Accuracy: " + str(round(100 \* accuracy\_values.mean(), 2)) + "%")

    recall\_values = cross\_val\_score(model, X, y, scoring='recall\_weighted', cv=3)

    print("Recall: " + str(round(100 \* recall\_values.mean(), 2)) + "%")

    precision\_values = cross\_val\_score(model, X, y, scoring='precision\_weighted',

    cv=3)

    print("Precision: " + str(round(100 \* precision\_values.mean(), 2)) + "%")

    results.append(f1)

    names.append(name)

    print()

# Порівняння алгоритмів

pyplot.boxplot(results, labels=names)

pyplot.title('Algorithm Comparison')

pyplot.show()

LR:

F1 score: 75.66%

Accuracy: 78.85%

Recall: 78.85%

Precision: 77.03%

LDA:

F1 score: 79.35%

Accuracy: 81.14%

Recall: 81.14%

Precision: 79.86%

KNN:

F1 score: 74.16%

Accuracy: 76.67%

Recall: 76.67%

Precision: 73.99%

CART:

F1 score: 80.69%

Accuracy: 80.62%

Recall: 80.71%

Precision: 80.91%

NB:

F1 score: 75.89%

Accuracy: 78.87%

Recall: 78.87%

Precision: 76.97%

SVM:

F1 score: 64.52%

Accuracy: 75.1%

Recall: 75.1%

Precision: 68.18%

Можна зробити висновок що класифікатор CART справився найкраще а даному випадку. Хоча і програє в параметрах accuracy та recall класифікатору LDA.

**Завдання 5:** 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, cross\_val\_score

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)

from sklearn import metrics

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))

from sklearn.metrics import confusion\_matrix

from io import BytesIO  # neded for plot

import seaborn as sns

sns.set()

import matplotlib.pyplot as plt

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")

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Рис. 5.1 – Результат виконання програми

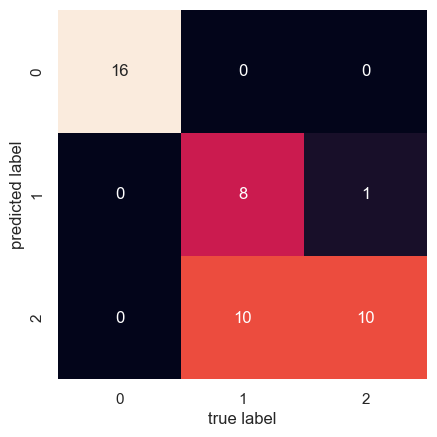


Рис. 5.2 – Результат виконання програми

При створенні класифікатора Ridge використовуємо параметри tol (точність рішення) та solver (розв’язувач для використання в обчислювальних процедурах).

При обрахунках якості класифікатора використовували:

- accuracy (акуратність) ≈ 0.76

- precision (точність) ≈ 0.83

- recall (повнота) ≈ 0.76

- f1 score ≈ 0.75

- Cohen Kappa Score (Коефіцієнт каппа Коена) ≈ 0.64

- Matthews Corrcoef (Коефіцієнт кореляції Метьюза) ≈ 0.68

Коефіцієнт Каппа Коена (K) – це статистичний показник, який використовують для вимірювання надійності між оцінювачами для якісних елементів.

Коефіцієнт кореляції Метьюза - у статистиці коефіцієнт фі є мірою асоціації для двох двійкових змінних. У машинному навчанні він відомий як коефіцієнт кореляції Метьюза та використовується як міра якості бінарних класифікацій.

Коефіцієнт Коена Каппа і коефіцієнт кореляції Метьюза застосовуються для визначення ступеня відповідності між передбачуваними та спостережуваними класами, і вони ураховують можливість випадкової згоди.

Значення цих коефіцієнтів може коливатися від -1 до 1. Вище значення вказує на більшу ступінь узгодженості. Значення нуль свідчить про те, що узгодженість випадкова, тоді як від'ємне значення вказує на гіршу узгодженість, ніж при випадковому виборі

**Висновок:** в ході виконання лабораторної роботи використовуючи спеціалізовані бібліотеки та мову програмування Python досліджено різні методи класифікації даних та навчився їх порівнювати