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.

```python
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 SVM-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 SVM-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)) + "%")
```

```
F1 score: 56.15%
```

Рис. 1.1 – Результат виконання програми

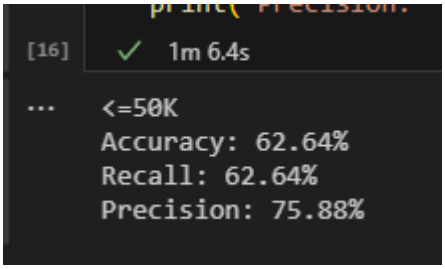

Рис. 1.2 – Результат виконання програми

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))
```

```
<=50K
Accuracy: 62.64%
Recall: 62.64%
Precision: 75.88%
```

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"))
```

```
F1 score: 56.15%
<=50K
Accuracy: 62.64%
Recall: 62.64%
Precision: 75.88%
```

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'))
```

```
F1 score: 56.15%
<=50K
Accuracy: 62.64%
Recall: 62.64%
Precision: 75.88%
```

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"]))
```

```
Ключі iris_dataset:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, pre
...
Назви відповідей: ['setosa' 'versicolor' 'virginica']
Назва ознак:
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Тип масиву data: <class 'numpy.ndarray'>
Форма масиву data: (150, 4)
Ознаки перших п'яти елементів:
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
Тип масиву target: <class 'numpy.ndarray'>
Відповіді:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

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()
```

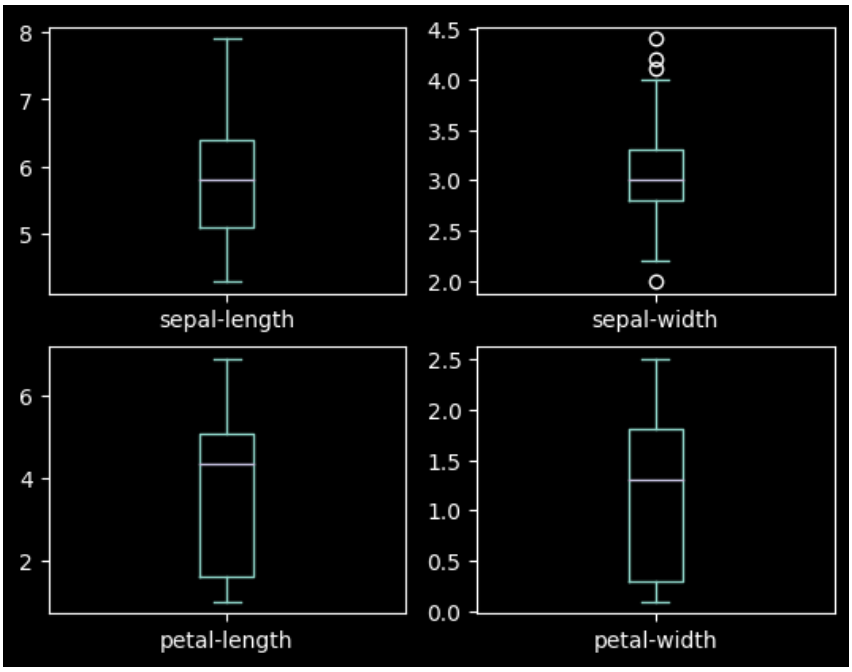

Рис. 3.2 – Результат виконання програми

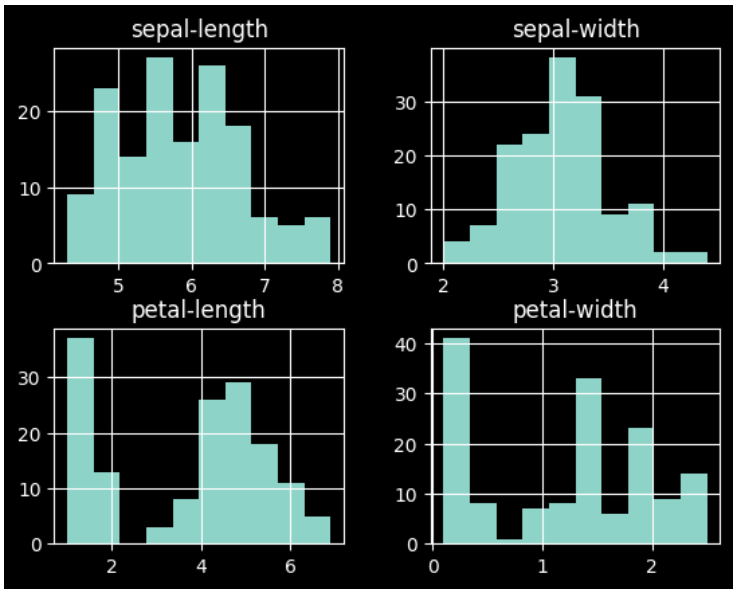

Рис. 3.3 – Результат виконання програми

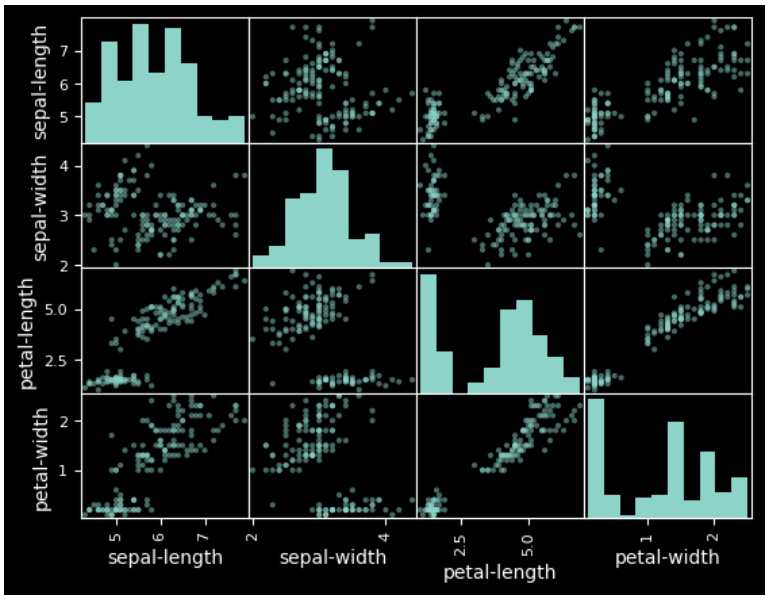

Рис. 3.4 – Результат виконання програми

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()
```

```
LR: 0.941667 (0.065085)
LDA: 0.975000 (0.038188)
KNN: 0.958333 (0.041667)
CART: 0.933333 (0.050000)
NB: 0.950000 (0.055277)
SVM: 0.983333 (0.033333)
```

Рис. 3.4 – Результат виконання програми

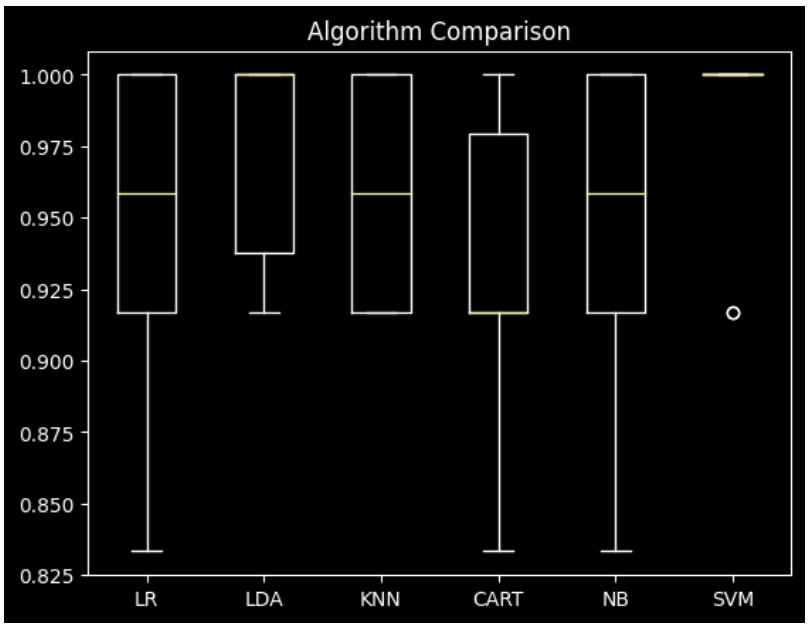

Рис. 3.5 – Результат виконання програми

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))
```

```
0.9666666666666667
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.92      0.96        13
 Iris-virginica       0.86      1.00      0.92         6

       accuracy                           0.97        30
      macro avg       0.95      0.97      0.96        30
   weighted avg       0.97      0.97      0.97        30
```

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)))
```

```
форма массива X_new: (1, 4)
Прогноз: ['Iris-setosa']
Оцінка тестового набору: 1.00
```

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")
```

```
Accuracy: 0.7556
Precision: 0.8333
Recall: 0.7556
F1 Score: 0.7503
Cohen Kappa Score: 0.6431
Matthews Corrcoef: 0.6831
              Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.44      0.89      0.59         9
           2       0.91      0.50      0.65        20

    accuracy                           0.76        45
   macro avg       0.78      0.80      0.75        45
weighted avg       0.85      0.76      0.76        45
```

Рис. 5.1 – Результат виконання програми

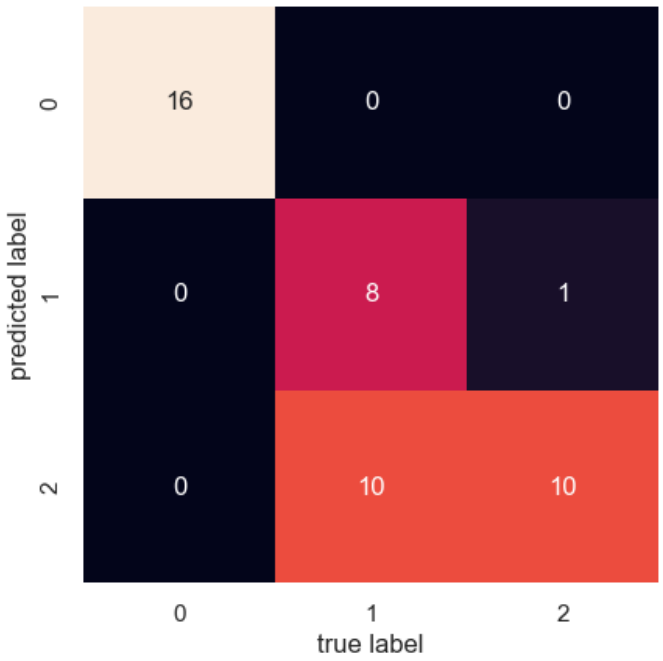

Рис. 5.2 – Результат виконання програми

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