ЛАБОРАТОРНА РОБОТА № 2
ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ
Варіант 13
Хід роботи:

Завдання 1: Випишіть всі 14 ознак з набору даних:

age	вік	числові
workclass	робочий клас (на кого працює, чи працював)	категоріальні
fnlwgt	вага – позначає довіру до опитуваного	числові
education	вид освіти	категоріальні
education-num	кількість освіт	числові
marital-status	одруженість (розлучений, чи ніколи не був)	категоріальні
occupation	ким працює	категоріальні
relationship	стосунки (чоловік, дружина, батько)	категоріальні
race	раса	категоріальні
sex	стать (чоловік, жінка)	категоріальні (бінарні)
capital-gain	скільки заробив, продаючи щось	числові
capital-loss	скільки втратив, продаючи щось (впливає також на податки)	числові
hours-per-week	скільки годин на тиждень працює	числові
native-country	місце народження	категоріальні

Код програми:

```
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
from sklearn.metrics import confusion_matrix as get_confusion_matrix
import warnings

warnings.filterwarnings("ignore")


def has_missing_values(row, missing_value_symbol, missing_value_columns):
    for index in missing_value_columns:
        if row[index] == missing_value_symbol:
            return True
    return False
```

```
def read_file_to_array(
    filename, max_datapoints, missing_value_symbol="?", missing_value_columns=(1, 6,
13)
):
    data = []

    data1_points = 0
    data2_points = 0

    with open(filename, "r") as file:
        for line in file.readlines():
            if data1_points >= max_datapoints and data2_points >= max_datapoints:
                break

            row = line[:-1].split(", ")

            if has_missing_values(
                row,
                missing_value_symbol=missing_value_symbol,
                missing_value_columns=missing_value_columns,
            ):
                continue

            is_data1_point = row[-1] == ">50K" and data1_points < max_datapoints
            is_data2_point = row[-1] == "<=50K" and data2_points < max_datapoints
            if is_data1_point:
                data1_points += 1
                data.append(row)
            elif is_data2_point:
                data2_points += 1
                data.append(row)

    data = np.array(data)
    return data


def encode_labels(
    array, encoders=None, categorical_columns=(1, 3, 5, 6, 7, 8, 9, 13, 14)
):
    encoders = {}
    encoded_array = np.empty(array.shape)
    for index in categorical_columns:
        if index not in encoders.keys():
            encoder_instance = preprocessing.LabelEncoder()
            encoders[index] = encoder_instance
        encoded_array[:, index] = encoders[index].fit_transform(array[:, index])

    return encoded_array.astype(int), encoders
```

```python
def split_array(array):
    x = array[:, :-1]
    y = array[:, -1]
    return x, y


def train_svm(x, y):
    classifier = OneVsOneClassifier(LinearSVC(random_state=0))
    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)
    return classifier, x_test, y_test


def evaluate_model(classifier, x, y, num_folds=3):
    accuracy = cross_val_score(
        classifier, x, y, cv=num_folds, scoring="accuracy"
    ).mean()
    precision = cross_val_score(
        classifier, x, y, cv=num_folds, scoring="precision_macro"
    ).mean()

    recall = cross_val_score(
        classifier, x, y, cv=num_folds, scoring="recall_macro"
    ).mean()
    f1 = cross_val_score(classifier, x, y, cv=num_folds, scoring="f1_macro").mean()

    print(f"Accuracy = {accuracy:.2f}")
    print(f"Precision = {precision:.2f}")
    print(f"Recall = {recall:.2f}")
    print(f"F1-score = {f1:.2f}")

    y_prediction = classifier.predict(x)
    confusion_matrix = get_confusion_matrix(y, y_prediction)
    print("Confusion Matrix:\n", confusion_matrix)


if __name__ == "__main__":
    data = read_file_to_array(
        "Лабораторна робота 2/income_data.txt", max_datapoints=25_000
    )
    data, encoders = encode_labels(data)
    x, y = split_array(data)
    svm, x_test, y_test = train_svm(x, y)

    input_array = [
        "37",
        "Private",
        "215646",
        "HS-grad",
```

```
        "9",
        "Never-married",
        "Handlers-cleaners",
        "Not-in-family",
        "White",
        "Male",
        "0",
        "0",
        "40",
        "United-States",
    ]
    print(f"Input: {input_array}")
    test_array = np.array([input_array])

    test_array, _ = encode_labels(
        test_array, encoders=encoders, categorical_columns=(1, 3, 5, 6, 7, 8, 9, 13)
    )

    prediction = np.array([svm.predict(test_array)])
    prediction = encoders[14].inverse_transform(prediction)[0]

    print(f"Prediction: {prediction}\n")

    evaluate_model(svm, x_test, y_test)
```

Виконання:

```
Input: ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']
Prediction: <=50K

Accuracy = 0.74
Precision = 0.41
Recall = 0.50
F1-score = 0.43
Confusion Matrix:
 [[4492    0]
 [1541    0]]
```

Рисунок 1.1 – Виконання програми

Висновок: Тестова точка входить до класу людей, що заробляють менше 50 тисяч.

Завдання 2: Використовуючи набір даних та код з попереднього завдання створіть та дослідіть нелінійні класифікатори SVM.

Виконання:

```
Input: ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']
Prediction: <=50K

Accuracy = 0.74
Precision = 0.37
Recall = 0.50
F1-score = 0.43
Confusion Matrix:
 [[4492    0]
 [1541    0]]
PS B:\University\Системи штучного інтелекту>
```

Рисунок 2.1 – Поліноміальне ядро

```
Input: ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']
Prediction: <=50K

Accuracy = 0.74
Precision = 0.37
Recall = 0.50
F1-score = 0.43
Confusion Matrix:
 [[4492    0]
 [1541    0]]
PS B:\University\Системи штучного інтелекту>
```

Рисунок 2.2 – Гаусове ядро

```
Input: ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']
Prediction: >50K

Accuracy = 0.74
Precision = 0.41
Recall = 0.50
F1-score = 0.43
Confusion Matrix:
 [[4018  474]
 [1303  238]]
PS B:\University\Системи штучного інтелекту>
```

Рисунок 2.3 – Сигмоїдальне ядро

Висновок: Найбільша точність та детальність у сигмоїдального ядра

Завдання 3.1: ознайомлення зі структурою даних.

Код програми:

```python
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("Значення ознак:\n{}".format(iris_dataset["data"][:5]))

print("Тип масиву target:{}".format(type(iris_dataset["target"])))
print("Відповіді:\n{}".format(iris_dataset["target"]))
```

Виконання:

```
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, predictive
...
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)
Значення ознак:
[[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 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]
PS B:\University\Системи штучного інтелекту>
```

Рисунок 3.1.1 – Виконання програми

Завдання 3.2: візуалізація.

Код програми:

```
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())

dataset.plot(kind="box", subplots=True, layout=(2, 2), sharex=False, sharey=False)
pyplot.show()

dataset.hist()
pyplot.show()

scatter_matrix(dataset)
pyplot.show()
```

Виконання:

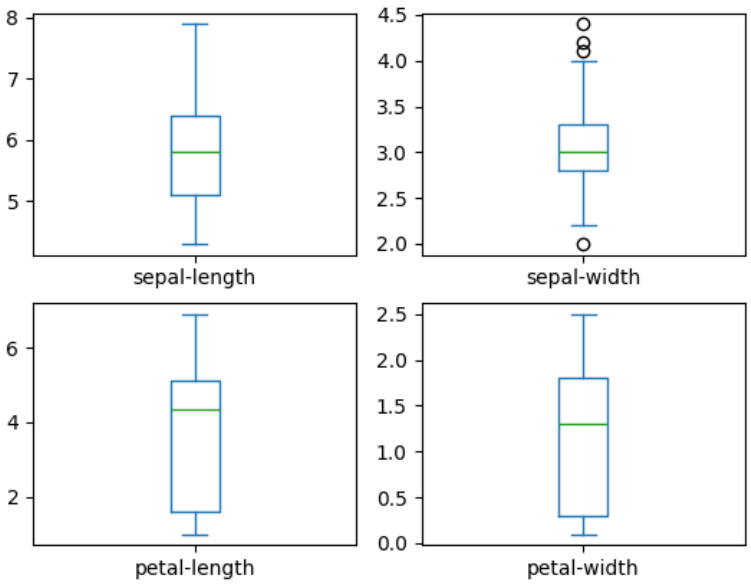

Рисунок 3.2.1 - Графік

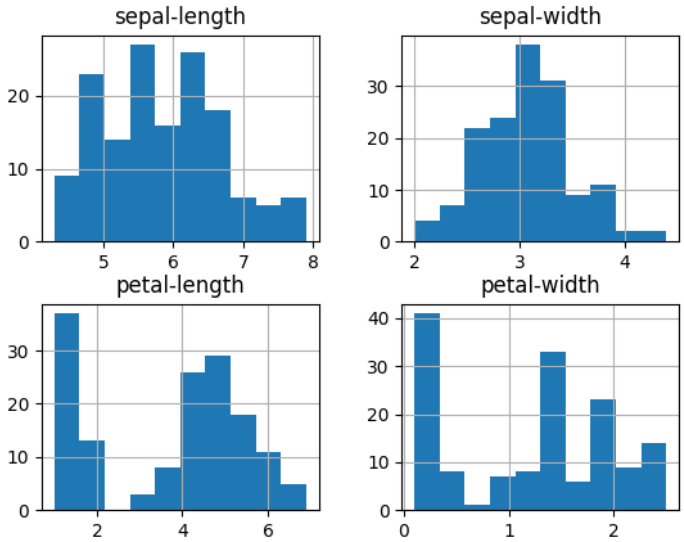

Рисунок 3.2.2 - Графік

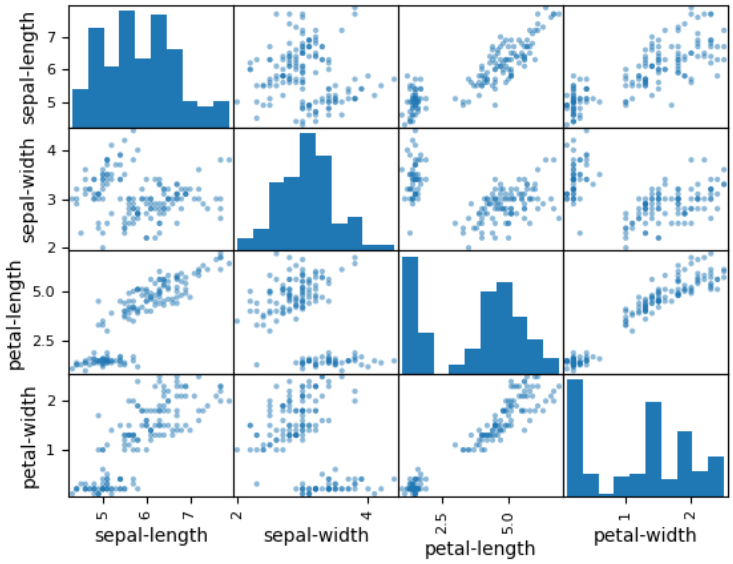

Рисунок 3.2.3 - Графік

Завдання 3.3: Порівняння алгоритмів.

Код програми:

```
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
)

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")))
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, scor-
ing="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()

model = SVC(gamma="auto")
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))

scatter_matrix(dataset)
pyplot.show()
```

Виконання:

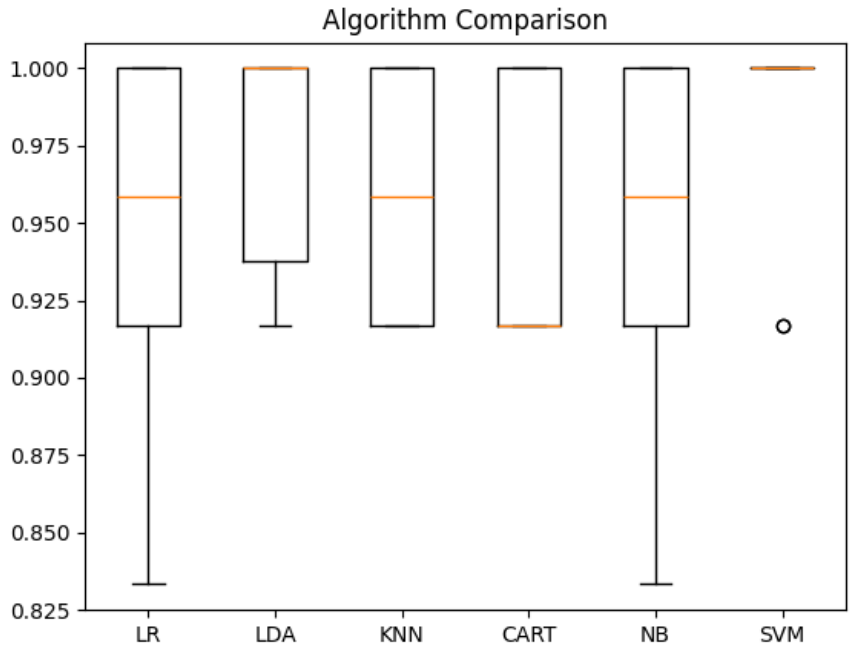

Рисунок 3.3.1 - Графік

```
LR: 0.941667 (0.065085)
LDA: 0.975000 (0.038188)
KNN: 0.958333 (0.041667)
CART: 0.950000 (0.040825)
NB: 0.950000 (0.055277)
SVM: 0.983333 (0.033333)
```

Рисунок 3.3.2 – Порівняння

Завдання 3.4: Передбачення.

Код програми:

```
X_new = np.array([[5, 2.9, 1, 0.2]])
print("Форма масива X_new: {}".format(X_new.shape))
knn = KNeighborsClassifier()
knn.fit(X_train, Y_train)
prediction = knn.predict(X_new)
print("Прогноз: {}".format(prediction))

dataset.hist()
pyplot.show()
```

Виконання:

```
Форма масива X_new: (1, 4)
Прогноз: ['Iris-setosa']
Backend tkagg is interactive backend. Turning
PS B:\University\Системи штучного інтелекту>
```

Рисунок 3.4.1 – Виконання програми

Висновок: Квітка належить до Iris-setosa. Найкращим методом є SVM у даному випадку – найвища точність.

Завдання 4: Порівняння алгоритмів на income_data.txt.

Код програми:

```
import pandas as pd
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.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
from LR_2_task_1 import read_file_to_array, encode_labels, split_array


data = read_file_to_array("Лабораторна робота 2/income_data.txt",
max_datapoints=25_000)

data, encoders = encode_labels(data)
X, y = split_array(data)
```

```
X_train, X_validation, Y_train, Y_validation = train_test_split(
    X, y, test_size=0.20, random_state=1
)

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")))
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, scor-
ing="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()
```

Виконання:

```
LR: 0.748560 (0.001652)
LDA: 0.751751 (0.000880)
KNN: 0.802395 (0.008081)
CART: 0.809524 (0.006284)
NB: 0.740686 (0.006838)
SVM: 0.805504 (0.006865)
PS B:\University\Системи штучного інтелекту>
```

Рисунок 4.1 – Виконання програми

Завдання 5: Класифікатор Ridge.

Код програми:

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import RidgeClassifier
from sklearn.model_selection import train_test_split

iris = load_iris()
X, y = iris.data, iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ran-
dom_state=0)
clf = RidgeClassifier(tol=1e-2, solver="sag")
clf.fit(X_train, y_train)
ypred = clf.predict(X_test)
from sklearn import metrics
```

```
print("Accuracy:", np.round(metrics.accuracy_score(y_test, ypred), 4))
print(
    "Precision:",
    np.round(metrics.precision_score(y_test, ypred, average="weighted"), 4),
)
print("Recall:", np.round(metrics.recall_score(y_test, ypred, average="weighted"),
4))
print("F1 Score:", np.round(metrics.f1_score(y_test, ypred, average="weighted"), 4))
print("Cohen Kappa Score:", np.round(metrics.cohen_kappa_score(y_test, ypred), 4))
print("Matthews Corrcoef:", np.round(metrics.matthews_corrcoef(y_test, ypred), 4))
print("\t\tClassification Report:\n", metrics.classification_report(ypred, y_test))

from sklearn.metrics import confusion_matrix
from io import BytesIO
import seaborn as sns

sns.set_theme()
import matplotlib.pyplot as plt

mat = confusion_matrix(y_test, 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")

f = BytesIO()
plt.savefig(f, format="svg")
```

Виконання:

```
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
```

Рисунок 5.1 – Виконання програми

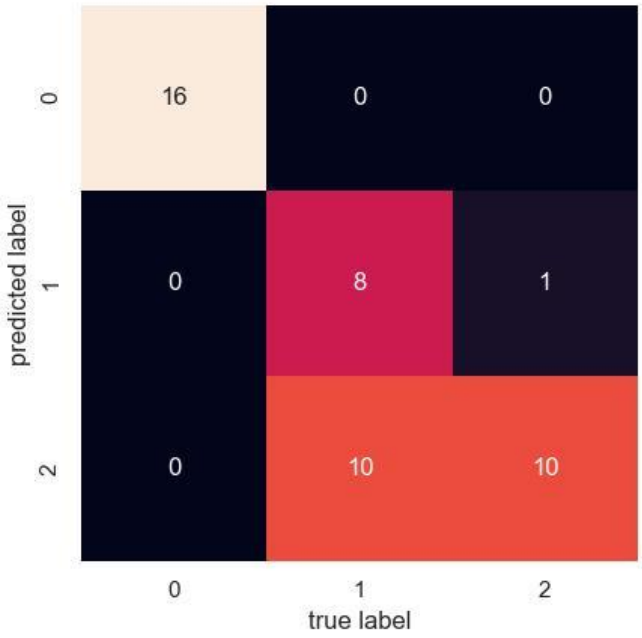

Рисунок 5.2 – Confusion.jpg

Висновок:

- Параметри: tol – допуск для зупинки алгоритму, solver – метод для навчання.
- Показники: accuracy – точність моделі, precision – впевненість у прогнозах, recall – фактичні позитивні зразки, f1 – середнє між precision та recall.
- Зображення: модель ідеально класифікує клас 0, помилково класифікує клас 2 класом 1, плутається у класифікації класів 1 та 2.
- Коефіцієнт Коена Каппа: узгодженість між прогнозованими та фактичними значеннями
- Коефіцієнт кореляції Метьюза: узгодженість між усією матрицею плутанини.

Посилання на GitHub: https://github.com/ipz215kvv/artificial-inteligence-systems