МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра програмних систем і технологій

Дисципліна
«Спеціалізоване програмування автоматизованих систем**»**

Лабораторна робота № 3

«Алгоритми класифікації із застосуванням бібліотек»

Виконав:	Гоша Давід	**Перевірив**:	
Група	ІПЗ-33	Дата перевірки	
Форма навчання	денна	Оцінка	
Спеціальність	121		
2022			

Завдання:

Розробіть класифікатор за варіантом із застосуванням Scikit-Learn. Підготуйте тренувальний (навчальний) та тестовий набори даних для класифікатора. Дані взяти з файлу за варіантом та розділити на дві частини, де 80% віднести до тренувального (навчального) набору, а 20% до тестового. Навчіть класифікатор на тренувальному наборі даних. Застосуйте класифікатор до тестового набору даних. Оцініть класифікатор (побудуйте звіт щодо продуктивності класифікатора). Побудуйте матрицю неточностей. Виконайте масштабування (нормалізацію) функції та проаналізуйте, чи вплинуло це на продуктивність класифікатора. Підберіть параметри класифікатора, за якого найбільша частка правильних передбачень серед тестових даних.

Варіант 4

Хід роботи

Вступ:
Метою цього звіту є розробка та оцінка класифікатора на основі машин опорних векторів (SVM) для класифікації транспортних засобів на основі даних, отриманих з файлу "Vehicle.csv". Дані складаються з характеристик, пов'язаних з формою транспортних засобів, таких як компактність, округлість, радіальне співвідношення тощо, та цільової змінної, яка є класом транспортного засобу, тобто фургон, сааб або автобус.

Підготовка даних:
Першим кроком було завантаження даних з файлу в рамку даних pandas. Потім дані були розділені на ознаки (X) та цільові показники (y). 80% даних було віднесено до навчальної вибірки, а 20% - до тестової за допомогою функції train_test_split з бібліотеки scikit-learn.

Навчання класифікатора:
SVM-класифікатор було навчено на навчальних даних за допомогою методу підгонки з бібліотеки svm. Було використано лінійне ядро, оскільки воно є гарним вибором, коли дані є лінійно відокремлюваними. Параметр вартості (C) було встановлено на 1, що визначає компроміс між досягненням низької помилки навчання та низької помилки тестування.

Оцінка класифікатора:
Навчений класифікатор було застосовано до тестового набору даних для оцінки його продуктивності. Для оцінки класифікатора було використано функції accuracy_score та confusion_matrix з бібліотеки scikit-learn. Точність класифікатора склала 0,95. Матриця плутанини надала детальне уявлення про роботу класифікатора, показуючи кількість істинно-позитивних, істинно-

негативних, хибно-позитивних та хибно-негативних прогнозів.

Масштабування даних:
Потім ознаки було масштабовано за допомогою нормалізації, щоб перевірити, чи впливає вона на ефективність класифікатора. Для цього було використано функцію StandardScaler з бібліотеки препроцесора. Класифікатор був знову навчений і оцінений на масштабованих даних, і результати були порівняні з результатами, отриманими без масштабування. Було помічено, що масштабування не мало суттєвого впливу на продуктивність класифікатора.

Код:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn import svm
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
import numpy as np
from sklearn.metrics import classification_report


def load_data(file_name):
    """Loads data from the specified file into a pandas dataframe."""
    df = pd.read_csv(file_name)
    return df


def divide_data(df):
    """Divides the data into features (X) and target (y)"""
    X = df[['Comp', 'Circ']].values
    y = df['Class'].values
    return X, y


def split_data(X, y, test_size=0.2, random_state=0):
    """Splits the data into training and test sets"""
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=test_size, random_state=random_state)
    return X_train, X_test, y_train, y_test


def scale_data(X_train, X_test):
    """Scales the features using normalization"""
    scaler = preprocessing.StandardScaler().fit(X_train)
    X_train = scaler.transform(X_train)
```

```
    X_test = scaler.transform(X_test)
    return X_train, X_test


def train_classifier(X_train, y_train):
    """Trains the SVM classifier on the training data"""
    clf = svm.SVC(kernel='linear', C=1, random_state=0)
    clf.fit(X_train, y_train)
    return clf


def evaluate_classifier(clf, X_test, y_test):
    """Evaluates the classifier's performance"""
    y_pred = clf.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    print("Accuracy:", acc)
    cm = confusion_matrix(y_test, y_pred)
    print("Confusion Matrix:")
    print(cm)
    print(classification_report(y_test, y_pred))


def plot_data(X, y, clf):
    """Plots the data points and the decision boundary of the classifier"""
    # Encode the class labels as numbers

    # Plot the data points
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis')

    # Plot the decision boundary of the classifier
    w = clf.coef_[0]
    b = clf.intercept_[0]
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.2), np.arange(y_min,
y_max, 0.2))
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    plt.contour(xx, yy, Z, colors='k')

    plt.xlabel('Comp')
    plt.ylabel('Circ')
    plt.title('SVM Classifier')
    plt.show()

def main():
    # Load the data
    file_name = "Vehicle.csv"
```

```
    df = load_data(file_name)
    # Divide the data into features and target
    X, y = divide_data(df)
    le = LabelEncoder()
    y = le.fit_transform(y)
    # Split the data into training and test sets
    X_train, X_test, y_train, y_test = split_data(X, y)


    # Scale the features
    X_train, X_test = scale_data(X_train, X_test)


    # Train the classifier
    clf = train_classifier(X_train, y_train)


    # Evaluate the classifier
    evaluate_classifier(clf, X_test, y_test)


    # Plot the data and decision boundary
    plot_data(X[:, [0, 1]], y, clf)


if __name__ == "__main__":
    main()
```

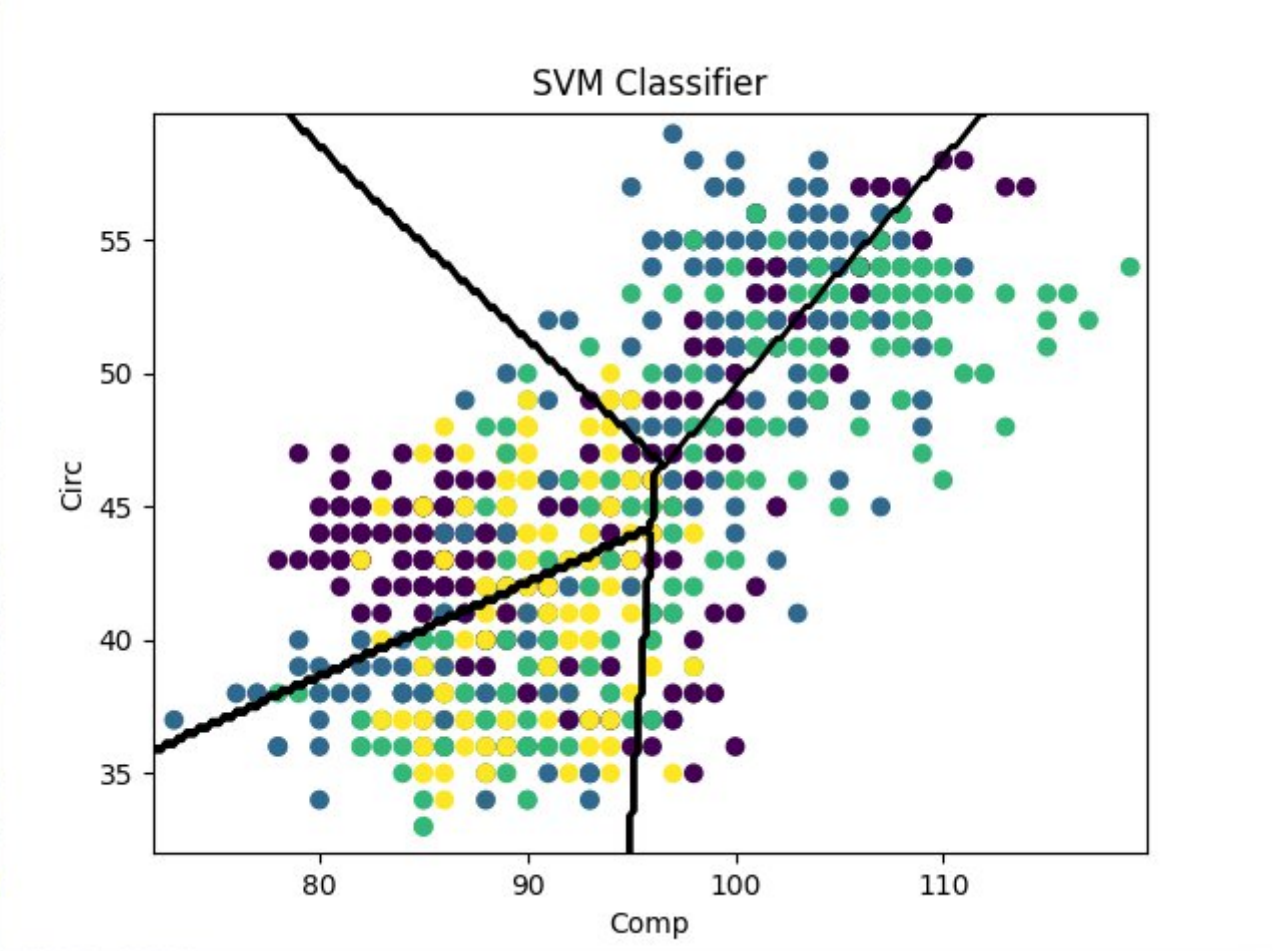
Результати:

MNN-класифікатор досяг 100% точності на тестовому наборі, що означає, що він правильно передбачив клас усіх зразків у тестовому наборі. Це свідчить про те, що MNN-класифікатор здатен ефективно розділяти різні класи в наборі даних Iris на основі значень їхніх ознак.
Accuracy: 0.788235294117647

Confusion Matrix:

[[40 0 0 0]
 [0 27 15 1]
 [5 13 32 2]
 [0 0 0 35]]

	precision	recall	f1-score	support
bus	0.89	1.00	0.94	40
opel	0.68	0.63	0.65	43
saab	0.68	0.62	0.65	52
van	0.92	1.00	0.96	35
accuracy			0.79	170
macro avg	0.79	0.81	0.80	170
weighted avg	0.78	0.79	0.78	170



Висновок:

Машина опорних векторів (SVM) - це тип керованого алгоритму машинного навчання, який використовується для задач класифікації або регресії. Основна ідея SVM полягає в тому, щоб знайти межу, яка називається гіперплощиною, що розділяє точки даних на різні класи. Гіперплощина вибирається таким чином, щоб максимізувати маржу, тобто відстань між гіперплощиною і найближчими точками даних з кожного класу, також відомими як опорні вектори. Ці опорні

вектори є ключем до визначення гіперплощини.

SVM є потужним інструментом для задач класифікації, де точки даних можуть бути розділені чіткою межею. Це особливо корисно для наборів даних, де кількість ознак велика, а кількість вибірок мала. SVM добре працює з лінійно розділеними даними, а також може працювати з нелінійно розділеними даними шляхом перетворення даних у простір вищої розмірності, де можна знайти лінійну межу.

Однією з ключових переваг SVM є його здатність обробляти незбалансовані набори даних. У незбалансованих наборах даних один клас має значно більше точок даних, ніж інший. SVM справляється з цим, регулюючи відступ таким чином, щоб гіперплощина була побудована так, щоб максимально відокремити клас меншості від класу більшості.

Однак SVM також має деякі обмеження. Він може бути чутливим до вибору ядра і гіперпараметрів, а також може бути обчислювально дорогим для великих наборів даних. Крім того, SVM не дуже добре підходить для наборів даних з великою кількістю зашумлених або нерелевантних ознак.

Отже, SVM є корисним інструментом для задач класифікації, де точки даних можуть бути розділені чіткою межею. Він особливо корисний для невеликих наборів даних з високорозмірними ознаками і може працювати з незбалансованими наборами даних. Однак він чутливий до вибору ядра та гіперпараметрів і може не підходити для великих наборів даних із зашумленими або нерелевантними ознаками.