**TRƯỜNG ĐẠI HỌC XÂY DỰNG HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🙡🕮🙣**



**BÁO CÁO TỔNG HỢP BÀI TẬP NHÓM**

**Nhóm 10- 67PM4**

**Thành viên nhóm: Nguyễn Đắc Điền – 0004667**

**Nguyễn Minh Sơn – 0234667**

**Nguyễn Anh Quân – 0238867**

**Trần Thế Thái – 0123267**

**Nguyễn Văn Đạo – 0047667**

**Nguyễn Tá Phong – 0183267**

Hà Nội, 2025

Bài tập nhóm số 1:

import json

class Book:

    def \_\_init\_\_(self,bookId,bookName,author,category,quantity):

        self.\_\_bookId = bookId

        self.\_\_bookName = bookName

        self.\_\_author = author

        self.\_\_category = category

        self.setQuantity(quantity)

    def getBookId(self):

        return self.\_\_bookId

    def setBookId(self,bookId):

        self.\_\_bookId = bookId

    def getBookName(self):

        return self.\_\_bookName

    def setBookName(self,bookName):

        self.\_\_bookName = bookName

    def getAuthor(self):

        return self.\_\_author

    def setAuthor(self,author):

        self.\_\_author = author

    def getCategory(self):

        return self.\_\_category

    def setCategory(self,category):

        self.\_\_category = category

    def getQuantity(self):

        return self.\_\_quantity

    def setQuantity(self, quantity):

        if quantity < 0:

            raise ValueError("Quantity cannot be negative")

        self.\_\_quantity = quantity

    def printInformation(self):

        print("=" \* 40)

        print(f"📖 Book ID: {self.getBookId()}")

        print(f"📚 Title: {self.getBookName()}")

        print(f"✍️  Author: {self.getAuthor()}")

        print(f"📂 Category: {self.getCategory()}")

        print(f"📦 Quantity Available: {self.getQuantity()}")

        print("=" \* 40)

    def to\_dict(self):

        return self.\_\_dict\_\_

    @staticmethod

    def from\_dict(data):

        clean\_data = {key.replace("\_Book\_\_", ""): value for key, value in data.items()}

        return Book(\*\*clean\_data )

from datetime import datetime

class BorrowReturnRecord:

    def \_\_init\_\_(self,recordId,memberId,borrowingList,borrowDate,returnDateEst,returnDate=None):

        self.\_\_recordId=recordId

        self.\_\_memberId=memberId

        self.\_\_borrowingList=borrowingList

        self.\_\_borrowDate=borrowDate

        self.\_\_returnDateEst=returnDateEst

        self.\_\_returnDate=returnDate

    # Getter và Setter cho recordId

    def getRecordId(self):

        return self.\_\_recordId

    def setRecordId(self, recordId):

        self.\_\_recordId = recordId

    # Getter và Setter cho memberId

    def getMemberId(self):

        return self.\_\_memberId

    def setMemberId(self, memberId):

        self.\_\_memberId = memberId

    # Getter và Setter cho borrowingList

    def getBorrowingList(self):

        return self.\_\_borrowingList

    def setBorrowingList(self, borrowingList):

        self.\_\_borrowingList = borrowingList

    # Getter và Setter cho borrowDate

    def getBorrowDate(self):

        return self.\_\_borrowDate

    def setBorrowDate(self, borrowDate):

        self.\_\_borrowDate = borrowDate

    # Getter và Setter cho returnDateEst (Ngày trả dự kiến)

    def getReturnDateEst(self):

        return self.\_\_returnDateEst

    def setReturnDateEst(self, returnDateEst):

        self.\_\_returnDateEst = returnDateEst

    # Getter và Setter cho returnDate (Ngày trả thực tế)

    def getReturnDate(self):

        return self.\_\_returnDate

    def setReturnDate(self, returnDate):

        self.\_\_returnDate = returnDate

    def calculate\_late\_fee(self):

        borrow\_date = datetime.strptime(self.getBorrowDate(), "%Y-%m-%d")

        return\_date = datetime.strptime(self.getReturnDate(), "%Y-%m-%d")

        return\_estDate = datetime.strptime(self.getReturnDateEst(), "%Y-%m-%d")

        if return\_date > return\_estDate:

            late\_days = (return\_date - borrow\_date).days

            return max(0, late\_days \* 5000)

        return 0

    def to\_dict(self):

        return self.\_\_dict\_\_

    @staticmethod

    def from\_dict(data):

        clean\_data = {key.replace("\_BorrowReturnRecord\_\_", ""): value for key, value in data.items()}

        return BorrowReturnRecord(\*\*clean\_data)

class Member:

    def \_\_init\_\_(self,memberId,fullName,phoneNumber,identificationNumber,address,borrowingBooks=[]):

        self.\_\_memberId=memberId

        self.\_\_fullName=fullName

        self.\_\_phoneNumber=phoneNumber

        self.\_\_identificationNumber=identificationNumber

        self.\_\_address=address

        self.\_\_borrowingBooks=borrowingBooks

    # Getter và Setter cho memberId

    def getMemberId(self):

        return self.\_\_memberId

    def setMemberId(self, memberId):

        self.\_\_memberId = memberId

    # Getter và Setter cho fullName

    def getFullName(self):

        return self.\_\_fullName

    def setFullName(self, fullName):

        self.\_\_fullName = fullName

    # Getter và Setter cho phoneNumber

    def getPhoneNumber(self):

        return self.\_\_phoneNumber

    def setPhoneNumber(self, phoneNumber):

        self.\_\_phoneNumber = phoneNumber

    # Getter và Setter cho identificationNumber

    def getIdentificationNumber(self):

        return self.\_\_identificationNumber

    def setIdentificationNumber(self, identificationNumber):

        self.\_\_identificationNumber = identificationNumber

    # Getter và Setter cho address

    def getAddress(self):

        return self.\_\_address

    def setAddress(self, address):

        self.\_\_address = address

    # Getter và Setter cho borrowingBooks

    def getBorrowingBooks(self):

        return self.\_\_borrowingBooks

    def setBorrowingBooks(self, borrowingBooks):

        self.\_\_borrowingBooks = borrowingBooks

    def borrow\_book(self, book\_id, quantity):

        if len(self.\_\_borrowingBooks) < 5:

            self.\_\_borrowingBooks.append({"book\_id": book\_id, "quantity": quantity})

            return True

        return False

    def return\_book(self, book\_id):

        if book\_id in self.borrowed\_books:

            self.borrowed\_books.remove(book\_id)

    def printInformation(self):

        print("=" \* 50)

        print(f"🆔 Member ID: {self.getMemberId()}")

        print(f"👤 Name: {self.getFullName()}")

        print(f"📞 Phone: {self.getPhoneNumber()}")

        print(f"🛂 ID Number: {self.getIdentificationNumber()}")

        print(f"🏠 Address: {self.getAddress()}")

        print(f"📚 Borrowed Books: {', '.join(map(str, self.getBorrowingBooks())) if self.getBorrowingBooks() else 'None'}")

        print("=" \* 50)

    def to\_dict(self):

        return self.\_\_dict\_\_

    @staticmethod

    def from\_dict(data):

        clean\_data = {key.replace("\_Member\_\_", ""): value for key, value in data.items()}

        return Member(\*\*clean\_data)

from models.Book import Book

from models.Member import Member

from models.BorrowReturnRecord import BorrowReturnRecord

from datetime import datetime, timedelta

from collections import Counter, defaultdict

import re

class LibraryManagement:

    def \_\_init\_\_(self):

        self.book = self.load\_data("Data/book.json", Book)

        self.member = self.load\_data("Data/member.json", Member)

        self.borrowRecord = self.load\_data("Data/borrowrecord.json", BorrowReturnRecord)

    def load\_data(self, filename, cls):

        try:

            with open(filename, "r", encoding="utf-8") as file:

                data = json.load(file)

            return [cls.from\_dict(item) for item in data]

        except FileNotFoundError:

            return []

    def save\_data(self, filename, data):

        with open(filename, "w", encoding="utf-8") as file:

            json.dump([item.to\_dict() for item in data], file, indent=4)

    # ------BOOK------:Start

    def add\_book(self,bookId=None,bookName=None,author=None,category=None,quantity=None):

        if any(param == '' for param in [bookId,bookName,author,category,quantity]):

            raise ValueError("Không để trống thông tin.")

        if int(quantity) < 0:

            raise ValueError("Quantity cannot be negative")

        if any(b.getBookId() == bookId for b in self.book):

            raise ValueError(f"Book ID {bookId} already exist")

        new\_book = Book(bookId, bookName, author, category, int(quantity))

        self.book.append(new\_book)

        self.save\_data("Data/book.json", self.book)

        return print(f"Book ID {bookId} added successfully.")

    def edit\_book(self,bookId,bookName=None,author=None,category=None,quantity=None):

        for book in self.book:

            if book.getBookId() == bookId:

                if bookName:

                    book.setBookName(bookName)

                if author:

                    book.setAuthor(author)

                if category:

                    book.setCategory(category)

                if quantity is not None:

                    book.setQuantity(quantity)

                self.save\_data("Data/book.json", self.book)  # Save the updated book list

                return print(f"Book ID {bookId} updated successfully.")

        raise ValueError(f"Book ID {bookId} not found.")

    def delete\_book(self, bookId):

        book = next((b for b in self.book if b.getBookId() == bookId), None)

        if not book:

            raise ValueError(f"Book ID {bookId} not found.")

        for record in self.borrowRecord:

            if any(b["book\_id"] == bookId for b in record.getBorrowingList()):

                raise ValueError(f"Book ID {bookId} cannot be deleted because it is currently borrowed.")

        self.book = [b for b in self.book if b.getBookId() != bookId]

        self.save\_data("Data/book.json", self.book)

        return print(f"Book ID {bookId} deleted successfully.")

    # ------BOOK------:End

    # ------MEMBER------:Start

    def add\_member(self,memberId,fullName,phoneNumber,identificationNumber,address):

        if any(param == '' for param in [memberId, fullName, phoneNumber, identificationNumber, address]):

            raise ValueError("Không được để trống thông tin.")

        if any(m.getMemberId() == memberId for m in self.member):

            raise ValueError(f"Member ID {memberId} already exist")

        if not re.fullmatch(r"^0\d{9}$", phoneNumber) and not re.fullmatch(r"^\d{12}$", identificationNumber):

            raise ValueError("Sai định dạng số điện thoại và CCCD.")

        elif not re.fullmatch(r"^\d{12}$", identificationNumber):

            raise ValueError("Sai định dạng CCCD.")

        elif not re.fullmatch(r"^0\d{9}$", phoneNumber):

            raise ValueError("Sai định dạng số điện thoại.")

        new\_member = Member(memberId,fullName,phoneNumber,identificationNumber,address)

        self.member.append(new\_member)

        self.save\_data("Data/member.json", self.member)

        return print(f"Member ID {memberId} added successfully.")

    def edit\_member(self,memberId,fullName=None,phoneNumber=None,identificationNumber=None,address=None):

        for member in self.member:

            if member.getMemberId() == memberId:

                if fullName:

                    member.setFullName(fullName)

                if phoneNumber:

                    member.setPhoneNumber(phoneNumber)

                if identificationNumber:

                    member.setIdentificationNumber(identificationNumber)

                if address is not None:

                    member.setAddress(address)

                self.save\_data("Data/member.json", self.member)

                return print(f"Member ID {memberId} updated successfully.")

        raise ValueError(f"Member ID {memberId} not found.")

    def delete\_member(self, memberId):

        member = next((m for m in self.member if m.getMemberId() == memberId), None)

        if not member:

            raise ValueError(f"Member ID {memberId} not found.")

        for record in self.borrowRecord:

            if memberId == record.getMemberId():

                raise ValueError(f"Member ID {memberId} cannot be deleted because it is currently borrowing.")

        self.member = [m for m in self.member if m.getMemberId() != memberId]

        self.save\_data("Data/member.json", self.member)

        return print(f"Member ID {memberId} deleted successfully.")

    # ------MEMBER------:end

    # ------SEARCH------:Start

    def searchBook(self, keyword):

        result = [

            book

            for book in self.book

            if keyword.lower() in book.getBookName().lower()

            or keyword.lower() in book.getAuthor().lower()

            or keyword.lower() in book.getCategory().lower()

        ]

        if result:

            print("Books Found:")

            for book in result:

                book.printInformation()

        else:

            print(f"No books found for '{keyword}'.")

    def searchMember(self, keyword):

        result = [

            member

            for member in self.member

            if keyword.lower() in member.getFullName().lower()

            or str(member.getMemberId()) == keyword

        ]

        if result:

            print("Members Found:")

            for member in result:

                member.printInformation()

        else:

            print(f"No members found for '{keyword}'.")

    # ------SEARCH------:End

    def borrow\_books(self, member\_id, borrowed\_books):

        member = next((m for m in self.member if m.getMemberId() == member\_id), None)

        if not member:

            return "Thành viên không tồn tại"

        borrow\_date = datetime.today().strftime("%Y-%m-%d")

        due\_date = (datetime.today() + timedelta(days=14)).strftime("%Y-%m-%d")

        for book\_info in borrowed\_books:

            found\_member = False

            book = next((b for b in self.book if b.getBookId() == book\_info["book\_id"]), None)

            if not book :

                return "Không tìm thấy sách."

            elif book.getQuantity() < book\_info["quantity"]:

                return "Sách không đủ số lượng."

            book.setQuantity(book.getQuantity() - book\_info["quantity"])

            for i in member.getBorrowingBooks():

                if (book\_info["book\_id"] == i["book\_id"]):

                    i["quantity"]+= book\_info["quantity"]

                    found\_member = True

                    break

            if not found\_member:

                found\_member = False

                member.borrow\_book(book\_info["book\_id"],book\_info["quantity"])

        new\_record = BorrowReturnRecord(len(self.borrowRecord) + 1, member\_id, borrowed\_books, borrow\_date, due\_date)

        self.borrowRecord.append(new\_record)

        self.save\_data("Data/book.json", self.book)

        self.save\_data("Data/member.json", self.member)

        self.save\_data("Data/borrowrecord.json", self.borrowRecord)

        return "Mượn sách thành công."

    def return\_books(self, record\_id, return\_date):

        record = next((r for r in self.borrowRecord if r.getRecordId() == record\_id), None)

        if not record:

            return "Phiếu mượn không tồn tại"

        if datetime.strptime(return\_date, "%Y-%m-%d") >= datetime.strptime(record.getBorrowDate(), "%Y-%m-%d"):

            record.setReturnDate(return\_date)

            fee = record.calculate\_late\_fee()

        else:

            return "Ngày trả không hợp lệ"

        memberBorrowed = next((m for m in self.member if m.getMemberId() == record.getMemberId()), None)

        for book\_info in record.getBorrowingList():

            book = next((b for b in self.book if b.getBookId() == book\_info["book\_id"]), None)

            if book:

                newQuantityBook = book.getQuantity() + book\_info["quantity"]

                book.setQuantity(newQuantityBook)

            for i in memberBorrowed.getBorrowingBooks():

                if(i["book\_id"]==book\_info["book\_id"] ):

                    if(i["quantity"] > book\_info["quantity"]):

                        i["quantity"] -= book\_info["quantity"]

                    else:

                        borrowing\_books = memberBorrowed.getBorrowingBooks()

                        borrowing\_books.remove(i)

                        memberBorrowed.setBorrowingBooks(borrowing\_books)

        self.save\_data("Data/book.json", self.book)

        self.save\_data("Data/borrowrecord.json", self.borrowRecord)

        self.save\_data("Data/member.json", self.member)

        return f"Trả sách thành công. Phí trễ hạn: {fee} VND"

    def sortBook(self):

        sortBook = {str(book\_id.getBookId()):book\_id.getQuantity() for book\_id in self.book}

        return dict(sorted(sortBook.items(), key=lambda x:x[1]))

    def sortBookReverse(self):

        sortBook = {str(book\_id.getBookId()):book\_id.getQuantity() for book\_id in self.book}

        return dict(sorted(sortBook.items(), key=lambda x:x[1], reverse=True))

    def sortMember(self):

        sortMebmber = {str(member\_id.getMemberId()):0 for member\_id in self.member}

        for i in self.member:

            if (i.getBorrowingBooks() is not None):

                for j in i.getBorrowingBooks():

                    sortMebmber[str(i.getMemberId())] += j["quantity"]

        return dict(sorted(sortMebmber.items(), key=lambda x:x[1], reverse=True))

    def statistics(self):

        totalListBook = Counter([i.getCategory() for i in self.book ])

        BorrowedBook = {str(book\_id.getBookId()):0 for book\_id in self.book}

        for i in self.borrowRecord:

            if(i.getReturnDate() is None):

                for j in i.getBorrowingList():

                    book\_id = j["book\_id"]

                    BorrowedBook[str(book\_id)] += j["quantity"]

        mostBorrowedBook = dict(sorted(BorrowedBook.items(), key=lambda x:x[1],reverse=True))

        MemberBorrow = {str(member\_id.getMemberId()):0 for member\_id in self.member}

        for i in self.member:

            if (i.getBorrowingBooks() is not None):

                for j in i.getBorrowingBooks():

                    MemberBorrow[str(i.getMemberId())] += j["quantity"]

        mostMemberBorrow = dict(sorted(MemberBorrow.items(), key=lambda x:x[1],reverse=True))

        return {"totalListBook":totalListBook, "mostBorrowedBook": mostBorrowedBook, "MemberBorrow":mostMemberBorrow}

from services.LibraryManagement import LibraryManagement

from models.Book import Book

from models.Member import Member

from unidecode import unidecode

def main():

    library = LibraryManagement()

    while True:

        print("\n--- QUẢN LÝ THƯ VIỆN ---")

        print("1. Thêm sách mới")

        print("2. Thêm thành viên")

        print("3. Mượn sách")

        print("4. Trả sách")

        print("5. Chỉnh sửa sách")

        print("6. Xóa sách")

        print("7. Chỉnh sửa thành viên")

        print("8. Xóa thành viên")

        print("9. Tìm kiếm sách")

        print("10. Tìm kiếm thành viên")

        print("11. Sắp xếp sách theo số lượng")

        print("12. Sắp xếp sách theo số lượng (giảm dần)")

        print("13. Sắp xếp thành viên theo số sách đã mượn")

        print("14. Thống kê thư viện")

        print("15. Thoát")

        choice = input("Chọn: ")

        if choice == "1":

            library.add\_book(len(library.book) + 1, input("Tên sách: "), input("Tác giả: "), input("Thể loại: "), input("Số lượng: "))

            print("Đã thêm sách.")

        elif choice == "2":

            library.add\_member(len(library.member) + 1, input("Họ tên: "), input("Số điện thoại: "), input("CCCD: "), input("Địa chỉ: "))

            print("Đã thêm thành viên.")

        elif choice == "3":

            list\_book = []

            optionBorrow = True

            member\_id = int(input("Nhập mã thành viên: "))

            while(optionBorrow):

                book\_id = int(input("Nhập mã sách: "))

                quantity = int(input("Số lượng: "))

                option = input("Có muốn mượn thêm loại sách khác: ")

                list\_book.append({"book\_id": book\_id, "quantity" : quantity})

                if(unidecode(option).lower() in "khong"):

                    optionBorrow = False

                else:optionBorrow=True

            result = library.borrow\_books(member\_id, list\_book)

            print(result)

        elif choice == "4":

            record\_id = int(input("Nhập mã phiếu mượn: "))

            return\_date = input("Nhập ngày trả (YYYY-MM-DD): ")

            result = library.return\_books(record\_id, return\_date)

            print(result)

        elif choice == "5":

            book\_id = int(input("Nhập mã sách cần chỉnh sửa: "))

            book\_name = input("Nhập tên sách mới (bỏ trống nếu không đổi): ")

            author = input("Nhập tác giả mới (bỏ trống nếu không đổi): ")

            category = input("Nhập thể loại mới (bỏ trống nếu không đổi): ")

            quantity = input("Nhập số lượng mới (bỏ trống nếu không đổi): ")

            library.edit\_book(book\_id, book\_name or None, author or None, category or None, int(quantity) if quantity else None)

            print("Đã chỉnh sửa sách.")

        elif choice == "6":

            book\_id = int(input("Nhập mã sách cần xóa: "))

            library.delete\_book(book\_id)

            print("Đã xóa sách.")

        elif choice == "7":

            member\_id = int(input("Nhập mã thành viên cần chỉnh sửa: "))

            name = input("Nhập họ tên mới (bỏ trống nếu không đổi): ")

            phone = input("Nhập số điện thoại mới (bỏ trống nếu không đổi): ")

            cccd = input("Nhập CCCD mới (bỏ trống nếu không đổi): ")

            address = input("Nhập địa chỉ mới (bỏ trống nếu không đổi): ")

            library.edit\_member(member\_id, name or None, phone or None, cccd or None, address or None)

            print("Đã chỉnh sửa thành viên.")

        elif choice == "8":

            member\_id = int(input("Nhập mã thành viên cần xóa: "))

            library.delete\_member(member\_id)

            print("Đã xóa thành viên.")

        elif choice == "9":

            keyword = input("Nhập từ khóa tìm kiếm sách: ")

            print("Kết quả tìm kiếm:")

            library.searchBook(keyword)

        elif choice == "10":

            keyword = input("Nhập từ khóa tìm kiếm thành viên: ")

            print("Kết quả tìm kiếm:")

            library.searchMember(keyword)

        elif choice == "11":

            sorted\_books = library.sortBook()

            print("\n📚 SÁCH ĐƯỢC SẮP XẾP THEO SỐ LƯỢNG (TĂNG DẦN):")

            print("Mã sách".ljust(10) + "Số lượng")

            print("-" \* 20)

            for book\_id, quantity in sorted\_books.items():

                print(book\_id.ljust(10) + str(quantity))

        elif choice == "12":

            sorted\_books = library.sortBookReverse()

            print("\n📚 SÁCH ĐƯỢC SẮP XẾP THEO SỐ LƯỢNG (GIẢM DẦN):")

            print("Mã sách".ljust(10) + "Số lượng")

            print("-" \* 20)

            for book\_id, quantity in sorted\_books.items():

                print(book\_id.ljust(10) + str(quantity))

        elif choice == "13":

            sorted\_members = library.sortMember()

            print("\n👥 THÀNH VIÊN ĐƯỢC SẮP XẾP THEO SỐ SÁCH ĐÃ MƯỢN:")

            print("Mã TV".ljust(10) + "Số sách mượn")

            print("-" \* 25)

            for member\_id, borrowed in sorted\_members.items():

                print(member\_id.ljust(10) + str(borrowed))

        elif choice == "14":

            stats = library.statistics()

            print("\n📊 THỐNG KÊ THƯ VIỆN:")

            print("\n🔹 SỐ LƯỢNG SÁCH THEO THỂ LOẠI:")

            for category, count in stats["totalListBook"].items():

                print(f"  - {category}: {count} sách")

            print("\n📖 SÁCH ĐƯỢC MƯỢN NHIỀU NHẤT (TOP 5):")

            print("Mã sách".ljust(10) + "Số lượng")

            print("-" \* 20)

            for i, (book\_id, count) in enumerate(stats["mostBorrowedBook"].items()):

                if i >= 5:

                    break

                print(book\_id.ljust(10) + str(count))

            print("\n👤 THÀNH VIÊN MƯỢN NHIỀU SÁCH NHẤT (TOP 5):")

            print("Mã TV".ljust(10) + "Số sách mượn")

            print("-" \* 25)

            for i,(member\_id, count) in enumerate(stats["MemberBorrow"].items()):

                print(member\_id.ljust(10) + str(count))

        elif choice == "15":

            print("Đã thoát chương trình.")

            break

        else:

            print("Lựa chọn không hợp lệ, vui lòng thử lại.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Bài tập nhóm số 2:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# b

df\_thong\_tin\_sach = pd.read\_excel('LtrUDchoKT\_DuLieuBTNhomSo2\_final.xlsx',sheet\_name='Thong tin sach')

df\_khach\_hang= pd.read\_excel('LtrUDchoKT\_DuLieuBTNhomSo2\_final.xlsx',sheet\_name='Khach hang')

df\_muon\_tra = pd.read\_excel('LtrUDchoKT\_DuLieuBTNhomSo2\_final.xlsx',sheet\_name='Muon Tra')

# print (df\_thong\_tin\_sach)

# print (df\_khach\_hang)

# print (df\_muon\_tra)

# c

df\_muon\_sach = pd.merge(df\_thong\_tin\_sach,df\_muon\_tra,on='Mã sách')

df\_muon\_sach = pd.merge(df\_muon\_sach, df\_khach\_hang, on='Mã khách')

df\_muon\_sach['Ngày mượn'] = pd.to\_datetime(df\_muon\_sach['Ngày mượn'])

df\_muon\_sach['Ngày trả'] = pd.to\_datetime(df\_muon\_sach['Ngày trả'])

# tinh toan ngay tra tre

df\_muon\_sach['Ngày trả dự kiến'] = df\_muon\_sach['Ngày mượn'] + pd.to\_timedelta(df\_muon\_sach['Số ngày thuê tối đa'], unit='d')

df\_muon\_sach['Số ngày trả trễ'] = (df\_muon\_sach['Ngày trả'] - df\_muon\_sach['Ngày trả dự kiến']).dt.days

df\_muon\_sach['Số ngày trả trễ'] = df\_muon\_sach['Số ngày trả trễ'].apply(lambda x: max(0, x))

df\_muon\_sach = df\_muon\_sach[['Mã sách', 'Tên sách', 'Tác giả', 'Thể loại', 'Giá thuê', 'Mã khách', 'Tên khách', 'Ngày mượn', 'Ngày trả', 'Số ngày trả trễ']]

# print(df\_muon\_sach)

# d

tong\_so\_sach\_da\_muon = len(df\_muon\_sach)

print(f"Tổng số sách đã mượn: {tong\_so\_sach\_da\_muon}")

df\_muon\_sach['Số ngày mượn'] = (df\_muon\_sach['Ngày trả'] - df\_muon\_sach['Ngày mượn']).dt.days

tong\_so\_ngay\_muon = df\_muon\_sach['Số ngày mượn'].sum()

print(f"Tổng số ngày mượn của các sách đã mượn: {tong\_so\_ngay\_muon}")

sach\_tra\_tre = df\_muon\_sach[df\_muon\_sach['Số ngày trả trễ'] > 0]

so\_sach\_tra\_tre = len(sach\_tra\_tre)

ty\_le\_tra\_tre = (so\_sach\_tra\_tre / tong\_so\_sach\_da\_muon) \* 100

print(f"Tỷ lệ sách trả trễ: {ty\_le\_tra\_tre:.2f}%")

# e

sach\_tra\_tre\_hon\_5\_ngay = df\_muon\_sach[df\_muon\_sach['Số ngày trả trễ'] > 5]

print("Danh sách các sách trả trễ hơn 5 ngày:")

print(sach\_tra\_tre\_hon\_5\_ngay[['Mã sách', 'Tên sách', 'Tác giả', 'Thể loại', 'Mã khách', 'Tên khách', 'Ngày mượn', 'Ngày trả', 'Số ngày trả trễ']])

so\_lan\_muon = df\_muon\_sach['Mã khách'].value\_counts()

khach\_hang\_muon\_nhieu = so\_lan\_muon[so\_lan\_muon > 3].reset\_index()

khach\_hang\_muon\_nhieu.columns = ['Mã khách', 'Số lần mượn']

khach\_hang\_muon\_nhieu = pd.merge(khach\_hang\_muon\_nhieu, df\_khach\_hang, on='Mã khách')

print("Danh sách các khách hàng đã mượn hơn 3 sách:")

print(khach\_hang\_muon\_nhieu[['Mã khách', 'Tên khách', 'Địa chỉ', 'Số điện thoại', 'Số lần mượn']])

# f

so\_luong\_tra\_tre\_theo\_the\_loai = sach\_tra\_tre.groupby('Thể loại').size().reset\_index(name='Số lượng trả trễ')

# ve bieu do

plt.figure(figsize=(10, 6))

sns.barplot(x='Thể loại', y='Số lượng trả trễ', data=so\_luong\_tra\_tre\_theo\_the\_loai, palette='viridis')

plt.title('Số lượng sách trả trễ theo từng thể loại', fontsize=16)

plt.xlabel('Thể loại', fontsize=14)

plt.ylabel('Số lượng trả trễ', fontsize=14)

plt.show()

# Vẽ biểu đồ scatter plot

plt.figure(figsize=(10, 6))

sns.scatterplot(x='Giá thuê', y='Số ngày mượn', data=df\_muon\_sach, hue='Thể loại', palette='deep', s=100)

plt.title('Mối quan hệ giữa Giá thuê và Số ngày mượn', fontsize=16)

plt.xlabel('Giá thuê (VND)', fontsize=14)

plt.ylabel('Số ngày mượn', fontsize=14)

plt.legend(title='Thể loại', bbox\_to\_anchor=(1.05, 1), loc='upper left')

plt.grid(True)

plt.show()

# g

with pd.ExcelWriter('DuLieuXuLy.xlsx', datetime\_format='dd/MM/yy') as writer:

    # Xuất df\_muon\_sach vào sheet "MuonSach"

    df\_muon\_sach.to\_excel(writer, sheet\_name='MuonSach', index=False)

    # Xuất sach\_tra\_tre vào sheet "SachTraTre"

    sach\_tra\_tre.to\_excel(writer, sheet\_name='SachTraTre', index=False)

    # Xuất khach\_hang\_muon\_nhieu vào sheet "KhachHangMuonNhieu"

    khach\_hang\_muon\_nhieu.to\_excel(writer, sheet\_name='KhachHangMuonNhieu', index=False)

    # Xuất so\_luong\_tra\_tre\_theo\_the\_loai vào sheet "ThongKeTraTre"

    so\_luong\_tra\_tre\_theo\_the\_loai.to\_excel(writer, sheet\_name='ThongKeTraTre', index=False)

print("Xuất dữ liệu thành công vào file 'DuLieuXuLy.xlsx'")

Bài tập nhóm số 3:

**Phần 1:**

**I. Trình bày tóm tắt các kiến thức, khái niệm cơ bản liên quan đến AI.**

**1. Định Nghĩa và Mục Tiêu**

- Trí tuệ nhân tạo (AI): Là khả năng của máy tính hay hệ thống máy tính thực hiện các nhiệm vụ thông minh, mà trước đây chỉ con người mới có thể làm được. Mục tiêu chính là xây dựng các hệ thống có khả năng học hỏi, suy luận và tự cải tiến qua thời gian.

**2. Các Lĩnh Vực Con Trong AI**

- Machine Learning (Học máy): Là nhánh của AI tập trung vào phát triển các thuật toán cho phép máy tính “học” từ dữ liệu. Các phương pháp chính gồm:

- Học có giám sát (Supervised Learning)

- Học không giám sát (Unsupervised Learning)

- Học tăng cường (Reinforcement Learning)

- Deep Learning (Học sâu): Là tập hợp các thuật toán học máy sử dụng các mạng nơ-ron nhân tạo với nhiều lớp (layers) để trích xuất và học các đặc trưng phức tạp từ dữ liệu.

Natural Language Processing (Xử lý ngôn ngữ tự nhiên): Tập trung vào khả năng máy tính hiểu và xử lý ngôn ngữ của con người.

- Computer Vision (Thị giác máy tính): Cho phép máy tính “nhìn” và hiểu hình ảnh, video.

- Robotics (Người máy): Ứng dụng AI để tạo ra các hệ thống tự động, từ robot công nghiệp đến robot dịch vụ.

**3. Các Thuật Toán và Phương Pháp Cơ Bản**

- Thuật toán học máy: Gồm các thuật toán như Decision Trees, Support Vector Machines, K-Nearest Neighbors, và Naive Bayes.

- Mạng nơ-ron nhân tạo (Artificial Neural Networks): Mô phỏng cấu trúc và cách thức hoạt động của bộ não con người.

- Thuật toán tối ưu và giải thuật tìm kiếm: Như thuật toán di truyền, thuật toán tìm kiếm heuristic… để giải quyết các bài toán phức tạp.

**4. Ứng Dụng của AI**

- Y tế: Chẩn đoán hình ảnh, phát hiện bệnh, hỗ trợ ra quyết định lâm sàng.

- Giao thông: Xe tự lái, quản lý giao thông thông minh.

- Tài chính: Dự báo thị trường, phát hiện gian lận, tư vấn đầu tư tự động.

- Giáo dục: Hệ thống học tập cá nhân hóa, trợ giảng ảo.

- Giải trí và truyền thông: Cá nhân hóa nội dung, chatbot, trợ lý ảo.

**5. Các Thách Thức và Vấn Đề Đạo Đức**

- Độ chính xác và hiệu quả: Xây dựng các mô hình chính xác và có thể mở rộng.

- Đạo đức và sự minh bạch: Đảm bảo các quyết định của AI không bị thiên vị và có thể giải thích được.

- Bảo mật và quyền riêng tư: Quản lý dữ liệu người dùng một cách an toàn.

- Tác động xã hội: Thay đổi cách thức làm việc, ảnh hưởng đến thị trường lao động và đời sống xã hội.

**II. Mô tả quá trình thực hiện các bài toán đã được học và thực hành.**

**1. Bài thực hành về xử lý ảnh và phân loại số viết tay**

**- Quy trình thực hiện:**

**Bước 1: Nhập các thư viện cần thiết**

* **Mục đích:** Hỗ trợ tính toán ma trận, xử lý dữ liệu dạng bảng, trực quan hóa và xây dựng các mô hình machine learning.
* **Lệnh thực hiện:**

import numpy as np # Xử lý mảng, ma trận

import pandas as pd # Xử lý dữ liệu dạng bảng

import seaborn as sns # Vẽ biểu đồ trực quan

import matplotlib.pyplot as plt # Vẽ biểu đồ

from sklearn.preprocessing import MinMaxScaler # Chuẩn hóa dữ liệu

from sklearn.decomposition import PCA # Giảm chiều dữ liệu

from sklearn.pipeline import Pipeline # Xây dựng quy trình xử lý tự động

import plotly.express as px # Vẽ biểu đồ tương tác

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import classification\_report, confusion\_matrix, ConfusionMatrixDisplay

from scipy.ndimage import rotate # Xoay ảnh

**Bước 2: Định nghĩa hàm xoay ảnh**

* **Mục đích:** Chuyển đổi ảnh dạng vector (1D) thành dạng 2D, xoay ảnh theo góc xác định, sau đó chuyển lại về dạng vector.
* **Hàm xử lý:**

def func\_array1d\_rotate2d(Narray1d, rotate\_angle, sizePixel):

N = Narray1d.shape[0]

Narray1d\_rotate = np.zeros(Narray1d.shape)

for i in range(N):

image = np.reshape(Narray1d[i, :], (sizePixel, sizePixel))

rotated\_image = rotate(image, rotate\_angle, reshape=False)

rotated\_array1d = np.reshape(rotated\_image, (1, -1))

Narray1d\_rotate[i, :] = rotated\_array1d

return Narray1d\_rotate

**Bước 3: Đọc dữ liệu từ file CSV và kiểm tra dữ liệu**

* **Mục đích:** Tải dữ liệu ảnh (với các nhãn tương ứng) và xác định xem có thiếu dữ liệu nào không.
* **Lệnh thực hiện:**

df\_train = pd.read\_csv('train.csv')

df\_test = pd.read\_csv('test.csv')

print("Missing values in df\_train: ", df\_train.isna().sum().sum())

print("Missing values in df\_test: ", df\_test.isna().sum().sum())

# Hiển thị 5 bản ghi đầu tiên

df\_train.head()

**Bước 4: Tách dữ liệu thành các biến đầu vào và đầu ra**

* **Mục đích:** Phân chia dữ liệu ảnh và nhãn (label) cho các bước xử lý tiếp theo.
* **Lệnh thực hiện:**

X\_train = df\_train.drop(columns=['label']).values

y\_train = df\_train['label'].values

**Bước 5: Trực quan hóa ảnh mẫu của từng lớp**

* **Mục đích:** Hiển thị một hình ảnh từ mỗi lớp (0 đến 9) để kiểm tra dữ liệu.
* **Lệnh thực hiện:**

fig, axs = plt.subplots(nrows=2, ncols=5, figsize=(10,5))

for i in range(10):

idx = np.where(y\_train == i)[0][2]

img = axs[int(i >= 5), i % 5].imshow(X\_train[idx].reshape(28, 28), cmap='Blues')

img.axes.get\_yaxis().set\_visible(False)

img.axes.get\_xaxis().set\_visible(False)

img.axes.set\_title(f'Label: {i}')

**Bước 6: Thiết lập tham số xoay ảnh và tăng cường dữ liệu**

* **Mục đích:** Định nghĩa số lần xoay, góc xoay ban đầu và phạm vi xoay cho mỗi mẫu ảnh.
* **Lệnh thực hiện:**

numRotate\_prior = 10 # Số lần xoay cho mỗi mẫu

angleOffset\_prior = -20 # Góc xoay ban đầu (độ)

angleRange\_prior = 40 # Phạm vi xoay từ -20 đến 20 độ

* Đồng thời, danh sách các ảnh được chọn để xoay được định nghĩa trong dictionary X\_train\_pick2rotate (trong đó chứa các index và offset cụ thể cho từng lớp). *(Tham khảo )*

**Bước 7: Thực hiện xoay ảnh (Data Augmentation)**

* **Mục đích:** Tạo ra các phiên bản mới của ảnh qua việc xoay theo các góc khác nhau, nhằm tăng kích thước tập huấn luyện.
* **Quy trình và lệnh thực hiện:**

numPickRotate = len(X\_train\_pick2rotate['index'])

print('numPickRotate = ', numPickRotate)

X\_sample\_rotate = np.zeros((10, numRotate\_prior, numPickRotate, X\_train.shape[1]))

y\_sample\_rotate = np.zeros((10, numRotate\_prior, numPickRotate))

for k0 in range(10):

for k1 in range(numRotate\_prior):

for k2 in range(numPickRotate):

idxPick\_X\_train = X\_train\_pick2rotate['index'][k2]

idx = np.where(y\_train == k0)[0][idxPick\_X\_train]

y\_sample\_rotate[k0, k1, k2] = k0 \* numRotate\_prior + k1

angleRotate = (k1 / numRotate\_prior \* angleRange\_prior) + angleOffset\_prior + X\_train\_pick2rotate['offset'][k2][k0]

X\_sample\_rotate[k0, k1, k2, :] = func\_array1d\_rotate2d(X\_train[idx][np.newaxis, :], angleRotate, 28)

* Sau đó, loại bỏ các pixel có giá trị nhỏ (như nhiễu) và chuẩn hóa giá trị pixel:

X\_sample\_rotate[X\_sample\_rotate < 10] = 0

minmax\_scaler = MinMaxScaler()

for k0 in range(10):

for k1 in range(numRotate\_prior):

X\_sample\_rotate[k0, k1, :, :] = minmax\_scaler.fit\_transform(X\_sample\_rotate[k0, k1, :, :]) \* 255

**Bước 8: Trực quan hóa ảnh sau khi tăng cường**

* **Mục đích:** Kiểm tra trực quan các ảnh đã xoay (có thể chọn một góc nhất định).
* **Lệnh thực hiện:**

for k2 in range(numPickRotate):

k1 = 0 # Lấy ảnh với góc xoay đầu tiên (có thể thay đổi)

fig, axs = plt.subplots(nrows=2, ncols=5, figsize=(10,5))

for k0 in range(10):

img = axs[int(k0 >= 5), k0 % 5].imshow(X\_sample\_rotate[k0, k1, k2, :].reshape(28, 28), cmap='Blues')

img.axes.get\_yaxis().set\_visible(False)

img.axes.get\_xaxis().set\_visible(False)

img.axes.set\_title(f'Label: {k0}')

**Bước 9: Chuyển đổi dữ liệu sau augmentation và áp dụng PCA**

* **Mục đích:** Chuyển dữ liệu ảnh đã xoay sang dạng vector và giảm số chiều bằng PCA để giảm độ phức tạp tính toán.
* **Lệnh thực hiện:**

X\_train\_rotate = np.reshape(X\_sample\_rotate, (-1, 784))

y\_train\_rotate = np.reshape(y\_sample\_rotate, (-1, 1)).flatten()

# Giảm chiều với PCA (giữ lại 40 thành phần chính)

pca = Pipeline(steps=[('scaler', MinMaxScaler()), ('pca', PCA(n\_components=40))])

X\_train\_rotate\_pca = pca.fit\_transform(X\_train\_rotate)

**Bước 10: Xây dựng mô hình KNN kết hợp PCA và tối ưu siêu tham số**

* **Mục đích:** Kết hợp quy trình chuẩn hóa, giảm chiều và phân loại bằng KNN; sử dụng GridSearchCV để tìm số láng giềng tối ưu.
* **Lệnh thực hiện:**

knn\_clf = Pipeline(steps=[('scaler', MinMaxScaler()), ('pca', PCA(n\_components=10)),

('knn', KNeighborsClassifier())])

knn\_clf\_params = {'knn\_\_n\_neighbors': [2, 4, 8, 16, 32]}

knn\_clf\_gs = GridSearchCV(estimator=knn\_clf, param\_grid=knn\_clf\_params, scoring='accuracy',

n\_jobs=-1, cv=5, verbose=1, return\_train\_score=True)

knn\_clf\_gs.fit(X\_train\_rotate\_pca, y\_train\_rotate)

print('best PCA+KNN params:', knn\_clf\_gs.best\_params\_)

print('best PCA+KNN cv accuracy:', knn\_clf\_gs.best\_score\_)

**Bước 11: Trực quan hóa kết quả tìm kiếm siêu tham số**

* **Mục đích:** Hiển thị đồ thị biểu diễn sự thay đổi của accuracy (trên tập train và test) theo số lượng láng giềng.
* **Lệnh thực hiện:**

fig, ax = plt.subplots()

ax.plot(range(len(knn\_clf\_params['knn\_\_n\_neighbors'])),knn\_clf\_gs.cv\_results\_['mean\_train\_score'], label='train', color='blue')

ax.plot(range(len(knn\_clf\_params['knn\_\_n\_neighbors'])),knn\_clf\_gs.cv\_results\_['mean\_test\_score'], label='test', color='orange')

ax.set\_xticks(ticks=range(len(knn\_clf\_params['knn\_\_n\_neighbors'])),labels=knn\_clf\_params['knn\_\_n\_neighbors'])

ii=knn\_clf\_params['knn\_\_n\_neighbors'].index(knn\_clf\_gs.best\_params\_['knn\_\_n\_neighbors'])

plt.axvline(x=ii, linestyle='--', color='red', label='best', alpha=0.3)

plt.legend()

plt.xlabel("'n\_neighbors' vals")

plt.ylabel('Accuracy')

plt.title("PCA+KNN GridSearchCV Results")

plt.show()

**Bước 12: Đánh giá mô hình KNN kết hợp PCA và hiển thị kết quả**

* **Mục đích:** Dự đoán trên tập dữ liệu, phân tích kết quả dự đoán (số và góc quay) và trực quan hóa ma trận nhầm lẫn cũng như ảnh tái tạo sau PCA.
* **Lệnh thực hiện:**

from sklearn.metrics import accuracy\_score

X\_val = X\_train\_rotate\_pca

Y\_val = y\_train\_rotate

best\_cv\_clf = knn\_clf\_gs.best\_estimator\_

y\_pred = best\_cv\_clf.predict(X\_val)

# Phân tích dự đoán theo số và góc quay

y\_val\_number = Y\_val // numRotate\_prior

y\_pred\_number = y\_pred // numRotate\_prior

y\_val\_angle = (Y\_val % numRotate\_prior)

y\_pred\_angle = (y\_pred % numRotate\_prior)

print('KNN number + angle, Acc({:.2f}%)'.format(accuracy\_score(Y\_val, y\_pred)\*100))

print('KNN number, Acc({:.2f}%)'.format(accuracy\_score(y\_val\_number, y\_pred\_number)\*100))

print('KNN angle, Acc({:.2f}%)'.format(accuracy\_score(y\_val\_angle, y\_pred\_angle)\*100))

# Hiển thị ma trận nhầm lẫn

conf\_matrix = confusion\_matrix(y\_val\_number, y\_pred\_number)

conf\_matrix\_disp = ConfusionMatrixDisplay(conf\_matrix)

conf\_matrix\_disp.plot(cmap='Blues')

* **Hiển thị ảnh gốc và ảnh tái tạo sau PCA:**

pca\_object = pca.named\_steps['pca']

fig, axs = plt.subplots(nrows=10, ncols=2, figsize=(6,20))

for i in range(10):

idx = np.where(y\_train == i)[0][0]

original\_img = axs[i, 0].imshow(X\_train[idx].reshape(28, 28), cmap='Blues')

original\_img.axes.get\_yaxis().set\_visible(False)

original\_img.axes.get\_xaxis().set\_visible(False)

original\_img.axes.set\_title(f'Original [{i}]:')

approx\_img\_ = pca\_object.inverse\_transform(X\_val[idx])

approx\_img = axs[i, 1].imshow(approx\_img\_.reshape(28, 28), cmap='Blues')

approx\_img.axes.get\_yaxis().set\_visible(False)

approx\_img.axes.get\_xaxis().set\_visible(False)

approx\_img.axes.set\_title(f'Predicted angle [{y\_pred\_angle[idx]}]:')

**2. Bài thực hành KNN với bộ dữ liệu Iris**

**- Quy trình thực hiện:**

**Bước 1: Nhập các thư viện cần thiết**

* **Mục đích:** Sử dụng cho xử lý dữ liệu, trực quan hóa và xây dựng mô hình học máy.
* **Lệnh thực hiện:**

import numpy as np

import pandas as pd

import matplotlib.pylab as plt

import seaborn as sns

from sklearn.model\_selection import KFold, cross\_val\_score, train\_test\_split, GridSearchCV

from sklearn.linear\_model import LogisticRegression

from sklearn import metrics, preprocessing

from sklearn.metrics import classification\_report, confusion\_matrix, precision\_score, recall\_score, auc, roc\_curve, accuracy\_score, f1\_score, roc\_auc\_score

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

from sklearn.ensemble import RandomForestClassifier

from tabulate import tabulate

import warnings as warn

from warnings import filterwarnings

filterwarnings("ignore")

**Bước 2: Đọc dữ liệu và tiền xử lý nhãn**

* **Mục đích:** Đọc file “IRIS.csv” và mã hoá cột species từ dạng chuỗi sang số.
* **Lệnh thực hiện:**

python

CopyEdit

data = pd.read\_csv("IRIS.csv")

df = pd.DataFrame(data)

df.groupby('species').count()

df["species"].replace({"Iris-setosa": 0, "Iris-versicolor": 1, "Iris-virginica": 2}, inplace=True)

df

**Bước 3: Tổng quan và trực quan hóa dữ liệu**

* **Mục đích:** Khám phá các đặc trưng, phân phối dữ liệu và mối quan hệ giữa các đặc trưng.
* **Lệnh thực hiện:**

1. **Thông tin mô tả:**

df.describe(include='all')

1. **Heatmap ma trận tương quan:**

plt.figure(figsize=(16, 6))

mask = np.triu(np.ones\_like(df.corr(method="spearman"), dtype=bool))

sns.heatmap(df.corr(method="spearman"), mask=mask, vmin=-1, vmax=1, annot=True, cmap="BrBG")

plt.title("Triangle Correlation Heatmap", fontsize=18, pad=16)

1. **Pairplot:**

sns.pairplot(df, hue='species', diag\_kind="hist", corner=True, palette='hls')

**Bước 4: Vẽ boxplot và countplot**

* **Mục đích:** Kiểm tra phân phối các đặc trưng và số lượng mẫu của từng lớp.
* **Lệnh thực hiện:**

Num = ['sepal\_length', 'sepal\_width', 'petal\_length', 'petal\_width']

j = 0

while j < 5:

fig = plt.figure(figsize=[20, 4])

plt.subplot(1, 2, 1)

sns.boxplot(x=Num[j], data=df, color='skyblue')

sns.set(font\_scale=1.25)

j += 1

plt.subplot(1, 2, 2)

sns.boxplot(x=Num[j], data=df, color='skyblue')

sns.set(font\_scale=1.25)

j += 1

if j == 4:

break

plt.show()

sns.countplot(x=df['species'], data=df)

**Bước 5: Chuẩn bị dữ liệu cho mô hình**

* **Mục đích:** Tách dữ liệu thành đầu vào (X) và đầu ra (y), sau đó chia thành tập huấn luyện và kiểm thử.
* **Lệnh thực hiện:**

X = df[["sepal\_length", "sepal\_width", "petal\_length", "petal\_width"]]

y = df["species"].values.reshape(-1, 1)

print(y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5, random\_state=0)

**Bước 6: Định nghĩa hàm đánh giá hiệu suất mô hình**

* **Mục đích:** Tính toán các chỉ số như accuracy, precision, recall, F1-score dựa trên mô hình đã huấn luyện.
* **Hàm xử lý:**

def Evaluate\_Performance(Model, Xtrain, Xtest, Ytrain, Ytest):

Model.fit(Xtrain, Ytrain)

overall\_score = cross\_val\_score(Model, Xtrain, Ytrain, cv=10)

model\_score = np.average(overall\_score)

Ypredicted = Model.predict(Xtest)

avg = 'weighted'

print("\n • Training Accuracy Score : ", round(Model.score(Xtrain, Ytrain) \* 100, 2))

print(f" • Cross Validation Score : {round(model\_score \* 100, 2)}")

print(f" • Testing Accuracy Score : {round(accuracy\_score(Ytest, Ypredicted) \* 100, 2)}")

print(f" • Precision Score is : {np.round(precision\_score(Ytest, Ypredicted, average=avg) \* 100, 2)}")

print(f" • Recall Score is : {np.round(recall\_score(Ytest, Ypredicted, average=avg) \* 100, 2)}")

print(f" • F1-Score Score is : {np.round(f1\_score(Ytest, Ypredicted, average=avg) \* 100, 2)}")

**Bước 7: Xây dựng và đánh giá mô hình KNN**

* **Mục đích:** Thử nghiệm với nhiều giá trị số láng giềng, vẽ đồ thị thể hiện độ chính xác trên tập huấn luyện và kiểm thử, và tìm siêu tham số tối ưu.
* **Lệnh thực hiện:**

1. **Lặp qua các giá trị của n\_neighbors và thu thập accuracy:**

training\_acc = []

test\_acc = []

neighbors\_setting = range(1, 30)

for n\_neighbors in neighbors\_setting:

KNN = KNeighborsClassifier(n\_neighbors=n\_neighbors)

KNN.fit(X\_train, y\_train.ravel())

training\_acc.append(KNN.score(X\_train, y\_train))

test\_acc.append(KNN.score(X\_test, y\_test))

1. **Vẽ đồ thị biểu diễn:**

plt.plot(neighbors\_setting, training\_acc, label="Độ chính xác trên tập huấn luyện")

plt.plot(neighbors\_setting, test\_acc, label="Độ chính xác trên tập kiểm thử")

plt.xlabel("Số lượng láng giềng")

plt.ylabel("Độ chính xác")

plt.grid(linestyle='-')

plt.legend()

1. **Tối ưu siêu tham số với GridSearchCV:**

parameters = {"n\_neighbors": range(1, 50)}

grid\_kn = GridSearchCV(estimator=KNN, param\_grid=parameters, scoring="accuracy", cv=5, verbose=1, n\_jobs=-1)

grid\_kn.fit(X\_train, y\_train.ravel())

print(grid\_kn.best\_params\_)

1. **Huấn luyện mô hình KNN với giá trị K tối ưu và dự đoán:**

K = 3 # Ví dụ chọn K = 3

KNN = KNeighborsClassifier(K)

KNN.fit(X\_train, y\_train.ravel())

y\_pred\_KNN = KNN.predict(X\_test)

print("K-Nearest Neighbors : ")

Evaluate\_Performance(KNN, X\_train, X\_test, y\_train, y\_test)

1. **Tính cross-validation và hiển thị ma trận nhầm lẫn:**

KNN\_r = cross\_val\_score(KNN, X, y, cv=10)

std\_dev = np.std(KNN\_r)

print(std\_dev)

cm = confusion\_matrix(y, KNN.predict(X))

fig, ax = plt.subplots(figsize=(8, 8))

ax.imshow(cm)

ax.grid(False)

ax.set\_xlabel('Nhãn dự đoán', fontsize=14, color='black')

ax.set\_ylabel('Nhãn thực tế', fontsize=14, color='black')

ax.xaxis.set(ticks=range(3))

ax.yaxis.set(ticks=range(3))

ax.set\_ylim(2.5, -0.5)

for i in range(3):

for j in range(3):

ax.text(j, i, cm[i, j], ha='center', va='center', color='red')

plt.show()

**III. Kiên thức thu được từ 2 bài thực hành trên.**

**A. Bài thực hành chữ viết tay**

**1.Xử lý dữ liệu hình ảnh**

- Cách biểu diễn hình ảnh trong dạng vector và ma trận

- Kỹ thuật xoay ảnh để tăng cường dữ liệu (data augmentation)

**2. Giảm chiều dữ liệu**

- Áp dụng PCA để giảm kích thước dữ liệu, giữ lại thông tin quan trọng

- Quan sát sự đánh đổi giữa việc giữ lại thông tin và giảm kích thước

**3. Mô hình KNN cho phân loại**

- Hiểu cách hoạt động của thuật toán K-Nearest Neighbors

- Tìm giá trị tối ưu cho tham số n\_neighbors qua GridSearchCV

**4. Kỹ thuật đánh giá mô hình**

- Đánh giá độ chính xác trong việc phân loại chữ số và góc xoay

- Sử dụng ma trận nhầm lẫn để phân tích hiệu suất theo lớp

**5. Chuẩn hóa và loại bỏ nhiễu**

- Sử dụng MinMaxScaler và việc loại bỏ các giá trị pixel nhỏ giúp giảm ảnh hưởng của nhiễu và cải thiện hiệu suất mô hình.

**6. Tối ưu mô hình với GridSearchCV**

- Việc tìm kiếm siêu tham số tối ưu (số lượng láng giềng) là bước quan trọng để đạt được mô hình có hiệu suất cao.

**B. Bài thực hành KNN và bộ dữ liệu Iris**

**1. Phân tích dữ liệu**

- Kỹ thuật trực quan hóa dữ liệu đa chiều

- Phân tích mối quan hệ giữa các đặc trưng thông qua ma trận tương quan

**2. Tinh chỉnh mô hình KNN**

- Hiểu được tác động của tham số K đến hiệu suất mô hình

- Phát hiện hiện tượng overfitting khi K quá nhỏ hoặc underfitting khi K quá lớn

**3. Đánh giá mô hình toàn diện**

- Sử dụng nhiều chỉ số đánh giá khác nhau

- Áp dụng Cross-Validation để đánh giá hiệu suất một cách khách quan

- Phân tích ma trận nhầm lẫn để hiểu chi tiết về lỗi phân loại

**4. Kỹ thuật trực quan hóa kết quả**

- Hiển thị kết quả của GridSearchCV dưới dạng biểu đồ

- Trực quan hóa ma trận nhầm lẫn để dễ dàng đánh giá

**5. Tiền xử lý và mã hoá dữ liệu**

- Việc chuyển đổi các nhãn dạng chuỗi sang số giúp mô hình xử lý dễ dàng hơn

**Phần 2. Tìm hiểu về ChatGPT**

**1. Tool được sử dụng chính trong lĩnh vực nào?**

- ChatGPT được sử dụng chính trong lĩnh vực xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) và trí tuệ nhân tạo (AI). Nó được ứng dụng rộng rãi trong nhiều lĩnh vực như:

+ Hỗ trợ viết nội dung và sáng tạo

+ Hỗ trợ học tập và giáo dục

+ Trả lời câu hỏi và cung cấp thông tin

+ Lập trình và phát triển phần mềm

+ Tự động hóa các tác vụ văn phòng

+ Hỗ trợ dịch thuật và xử lý đa ngôn ngữ

**2. Thông tin chính thức về ChatGPT**

- Tên chính thức: ChatGPT (Chat Generative Pre-trained Transformer)

- Nhà phát triển: OpenAI (CEO là Sam Altman )

- Ngày phát hành: ChatGPT ra mắt lần đầu vào ngày 30 tháng 11 năm 2022

- Nền tảng hỗ trợ: Trình duyệt web (tất cả các nền tảng), ứng dụng di động (iOS và Android) và desktop (MacOS và Window)

- Chi phí:

+ Phiên bản miễn phí: Có giới hạn về lượt truy cập và chỉ hỗ trợ mô hình cũ hơn, người dùng sẽ được những quyền lợi và hạn chế như sau:

* Access to GPT‑4o mini (Có thể truy cập được GPT-4i mini)
* Real-time data from the web with search (Tra cứu dữ liệu thời gian thực từ web)
* Limited access to GPT‑4o and o3‑mini )Bị giới hạn khả năng truy cập GPT-4o và o3-mini)
* Limited access to file uploads, data analysis, image generation, and voice mode (Giới hạn việc upload file, image và voice mode)
* Use custom GPTs

+ ChatGPT Plus: $20/tháng (dành cho cá nhân)

* Everything in Free
* Extended limits on messaging, file uploads, data analysis, and image generation
* Standard and advanced voice mode with video and screensharing
* Access to deep research and multiple reasoning models (o3‑mini, o3‑mini‑high, and o1)
* Create and use projects, tasks, and custom GPTs
* Limited access to Sora video generation
* Opportunities to test new features

+ ChatGPT Pro: $200/tháng (dành cho người dùng chuyên sâu, nghiên cứu, và các tác vụ đòi hỏi hiệu năng cao)

* Everything in Plus
* Unlimited access to all reasoning models and GPT‑4o
* Unlimited access to advanced voice, with higher limits for video and screensharing
* Access to a research preview of GPT‑4.5, our largest model yet
* Access to o1 pro mode, which uses more compute for the best answers to the hardest questions
* Extended access to deep research
* Extended access to Sora video generation
* Access to research preview of Operator (U.S. only)

+ ChatGPT Team: $25-30/người dùng/tháng, tạo sự bảo mật về dữ liệu trong team

* Everything in Plus, including deep research, file uploads, data analysis, canvas, projects, tasks, and search
* Higher message limits than Plus for GPT‑4o
* Access to multiple reasoning models (o3‑mini, o3‑mini‑high, and o1)
* Secure and collaborative workspace with admin console and unified billing
* Team data excluded from training by default. [Learn more](https://openai.com/enterprise-privacy/)
* Create and share custom GPTs with your workspace

+ ChatGPT Enterprise: Giá theo thỏa thuận, cung cấp các tính năng nâng cao cho doanh nghiệp

* Higher message limits than Team for GPT‑4o, with an expanded context window for longer inputs and larger files
* Access to multiple reasoning models (o3‑mini, o3‑mini‑high, o1, and o1 pro mode) for solving hard problems
* Access to deep research, custom workspace GPTs, canvas, projects, file uploads, data analysis, and search
* Enterprise data is excluded from training by default, with custom data retention windows and encryption at rest and in transit. [Learn more⁠](https://openai.com/enterprise-privacy/)
* User access management with SCIM, SSO, domain verification, user analytics, and GPT controls
* Comprehensive data practices aligned with CSA STAR and SOC 2 Type 2 standards, and support for compliance with GDPR, CCPA, and other privacy laws
* Enhanced support and account management

**3. Chức năng và tính năng chính**

**a. Tool này để làm gì?**

- ChatGPT là một mô hình ngôn ngữ lớn (LLM) được thiết kế để tương tác với người dùng thông qua các cuộc hội thoại tự nhiên. Mục đích chính của ChatGPT là:

* Tạo ra văn bản mạch lạc và liên quan đến ngữ cảnh dựa trên đầu vào của người dùng
* Trả lời câu hỏi, giải thích khái niệm, và cung cấp thông tin
* Hỗ trợ các tác vụ sáng tạo như viết, lập trình, và tổng hợp thông tin

**b. Các tính năng nổi bật**

* **Tương tác đa vòng (multi-turn)**: Có thể duy trì ngữ cảnh trong suốt cuộc hội thoại
* **Plugins và tích hợp**: Có thể tích hợp với các công cụ và dịch vụ bên ngoài
* **Khả năng đa phương thức**: GPT-4 có thể xử lý cả hình ảnh và văn bản (GPT-4 Vision)
* **Tùy chỉnh GPTs**: Người dùng có thể tạo phiên bản ChatGPT riêng với hướng dẫn, kiến thức và khả năng cụ thể
* **DALL-E tích hợp**: Có thể tạo hình ảnh từ mô tả văn bản
* **Voice Mode**: Cho phép tương tác bằng giọng nói
* **Trình duyệt web**: Có thể truy cập internet để tìm kiếm thông tin cập nhật (đối với phiên bản Plus và Enterprise)
* **Xử lý tệp**: Có thể phân tích và trích xuất thông tin từ các tệp được tải lên
* **Hỗ trợ đa ngôn ngữ:** Có thể giao tiếp bằng hơn 60 ngôn ngữ khác nhau.
* **Zero-Shot and Few-Shot Learning**: Zero-shot learning cho phép mô hình xử lý nhiệm vụ chưa từng thấy trước đó mà không cần ví dụ. Few-shot learning giúp mô hình học nhanh từ một vài ví dụ

**c. So sánh với các tool tương tự**

- So với các công cụ AI tương tự:

* **So với Claude (Anthropic)**:

+ ChatGPT có nhiều tính năng tích hợp hơn (trình duyệt, plugins, tạo hình ảnh)

+ Claude được đánh giá cao về khả năng hiểu ngữ cảnh dài và tuân thủ hướng dẫn phức tạp nhưng hướng đến sự an toàn khi response với người dùng

+ ChatGPT có nhiều mức giá và tùy chọn đăng ký đa dạng hơn

+ ChatGPT phản hồi nhanh hơn nhưng các câu trả lời thường có tính hallucination.

* **So với Gemini (Google)**:

+ ChatGPT có cộng đồng người dùng lớn hơn và hệ sinh thái phát triển hơn

+ Gemini tích hợp tốt hơn với các dịch vụ Google

+ ChatGPT thường được đánh giá cao hơn về khả năng lập trình và tạo nội dung

* **So với Copilot (Microsoft)**:

+ Copilot tích hợp sâu với hệ sinh thái Microsoft (Office, Windows)

+ ChatGPT có các tùy chọn tích hợp đa dạng hơn thông qua API và plugins

+ Hai công cụ đều dựa trên công nghệ OpenAI

**4. Công nghệ sử dụng**

**a. Mô hình AI nền tảng**

- ChatGPT sử dụng kiến trúc Transformer, đặc biệt là mô hình GPT (Generative Pre-trained Transformer). Đây là kiến trúc deep learning tiên tiến được thiết kế đặc biệt cho xử lý ngôn ngữ tự nhiên.

- Các mô hình như GPT-3.5, GPT-4, GPT-4o, GPT-4o mini, và các phiên bản o1, o3 đều được huấn luyện qua kỹ thuật học không giám sát (unsupervised learning) và sau đó được tinh chỉnh bằng **Reinforcement Learning from Human Feedback (RLHF)**.

- Cụ thể:

* Sử dụng Deep Learning với kiến trúc Transformer
* Dựa trên mô hình chỉ có bộ giải mã (decoder-only)
* Các phiên bản hiện tại bao gồm GPT-3.5 Turbo và GPT-4

**b. Công nghệ lõi**

* **Học sâu và kiến trúc Transformer**: Là nền tảng cho mô hình ngôn ngữ lớn. Transformer sử dụng cơ chế **Self-Attention** và **Feed-Forward Neural Networks** để xử lý dữ liệu một cách song song, giúp mô hình học được mối quan hệ giữa các từ trong câu tốt hơn so với các mô hình trước đây như RNN hay LSTM.
* **Pre-training và Fine-tuning**: Được huấn luyện trước trên lượng lớn văn bản từ internet, sau đó tinh chỉnh. Huấn luyện mô hình sử dụng thuật toán **PPO (Proximal Policy Optimization)** để tối ưu phản hồi dựa trên đánh giá của con người.
* **RLHF (Reinforcement Learning from Human Feedback)**: Học tăng cường từ phản hồi của con người để cải thiện tính hữu ích, trung thực và an toàn. Huấn luyện mô hình sử dụng thuật toán **PPO (Proximal Policy Optimization)** để tối ưu phản hồi dựa trên đánh giá của con người.
* **Kỹ thuật Điều chỉnh Suy luận (Inference Optimization):** Khi sinh văn bản, ChatGPT sử dụng các thuật toán như:
  + Beam Search: Tìm kiếm nhiều khả năng khác nhau để chọn câu trả lời tốt nhất.
  + Top-k Sampling: Giới hạn mô hình chỉ chọn từ một tập hợp nhỏ các từ có xác suất cao nhất.
  + Temperature Scaling: Điều chỉnh độ sáng tạo của mô hình bằng cách làm cho xác suất chọn từ ngẫu nhiên hơn hoặc chắc chắn hơn.

**5. Ưu, nhược điểm**

**- Ưu điểm:**

* **Đa năng và linh hoạt**: Có thể thực hiện nhiều loại nhiệm vụ ngôn ngữ khác nhau
* **Tạo văn bản chất lượng cao**: Văn bản mạch lạc, tự nhiên và phù hợp với ngữ cảnh
* **Hỗ trợ đa ngôn ngữ**: Hoạt động hiệu quả với nhiều ngôn ngữ khác nhau
* **Khả năng học tập**: Có thể hiểu và điều chỉnh theo ngữ cảnh cụ thể của cuộc hội thoại
* **Hệ sinh thái phong phú**: Nhiều tính năng, plugins và tùy chọn tích hợp
* **Cập nhật thường xuyên**: OpenAI liên tục cải thiện mô hình và thêm tính năng mới
* **Giao diện thân thiện**: Dễ sử dụng cho cả người dùng kỹ thuật và không kỹ thuật

**- Nhược điểm:**

* **Giới hạn kiến thức**: Thông tin giới hạn đến thời điểm cắt huấn luyện
* **Hallucinations**: Đôi khi tạo ra thông tin không chính xác hoặc hư cấu
* **Chi phí các gói cao cấp**: Giá $20/tháng cho Plus có thể cao đối với một số người dùng
* **Phụ thuộc vào chất lượng prompt**: Hiệu quả của phản hồi phụ thuộc vào cách câu hỏi được đặt
* **Giới hạn độ dài ngữ cảnh**: GPT-4 có giới hạn khoảng 32,000 token (~48,000 từ)
* **Vấn đề quyền riêng tư**: Dữ liệu người dùng có thể được sử dụng để cải thiện mô hình (tùy thuộc vào cài đặt)
* **Khả năng toán học và lập luận hạn chế**: Đôi khi gặp khó khăn với các bài toán phức tạp

**6. Hướng dẫn sử dụng**

**a. Cài đặt & truy cập**

**-** Truy cập qua trình duyệt web:

* Truy cập chat.openai.com
* Đăng ký tài khoản với email hoặc tài khoản Google/Microsoft/Apple
* Không cần cài đặt, có thể sử dụng trực tiếp

- Ứng dụng di động:

* Tải ứng dụng ChatGPT từ App Store (iOS) hoặc Google Play Store (Android)
* Đăng nhập bằng tài khoản OpenAI đã tạo

- Truy cập qua API (cho nhà phát triển):

* Đăng ký tài khoản OpenAI Platform
* Tạo khóa API từ trang platform.openai.com
* Tích hợp vào ứng dụng bằng thư viện chính thức hoặc gọi API trực tiếp

**b. Cách sử dụng cơ bản**

- Bắt đầu với ChatGPT:

1. Đăng nhập vào tài khoản ChatGPT
2. Nhấp vào nút "New chat" (Cuộc trò chuyện mới) ở góc trên bên trái
3. Nhập câu hỏi hoặc yêu cầu vào ô văn bản ở dưới cùng
4. Nhấn Enter hoặc biểu tượng giấy máy bay để gửi tin nhắn
5. Chờ ChatGPT phản hồi
6. Tiếp tục cuộc hội thoại bằng cách gửi thêm tin nhắn

- Sử dụng tính năng nâng cao (cho người dùng Plus/Team/Enterprise và bị giới hạn khi dùng bản miễn phí):

* Tải lên tệp: Nhấp vào biểu tượng kẹp giấy để tải lên và phân tích tệp
* Browsing (Duyệt web): Bật chế độ này từ dropdown mô hình để ChatGPT có thể truy cập internet
* DALL-E: Yêu cầu ChatGPT tạo hình ảnh bằng cách nhập "Tạo một hình ảnh về..."
* Voice Mode: Nhấp vào biểu tượng micrô để bắt đầu tương tác bằng giọng nóic. Tài liệu & hỗ trợ

- Tài liệu chính thức:

* Hướng dẫn người dùng: help.openai.com
* Tài liệu cho nhà phát triển: platform.openai.com/docs
* Hướng dẫn Prompt Engineering: platform.openai.com/docs/guides/prompt-engineering

- Cộng đồng hỗ trợ:

* **OpenAI Community Forum**: community.openai.com - Diễn đàn chính thức nơi người dùng có thể đặt câu hỏi, chia sẻ kinh nghiệm và tìm kiếm hỗ trợ
* **Reddit**: r/ChatGPT với hơn 2 triệu thành viên, cung cấp tin tức, mẹo và hỗ trợ
* **Discord OpenAI**: Cộng đồng Discord chính thức với các kênh dành riêng cho ChatGPT
* **YouTube**: Nhiều hướng dẫn và video hướng dẫn từ cả OpenAI và người dùng cộng đồng
* **OpenAI Blog**: openai.com/blog - Thông tin cập nhật về các tính năng và nghiên cứu mới

- Hỗ trợ khách hàng:

* Trung tâm trợ giúp với các bài viết và hướng dẫn khắc phục sự cố
* Hỗ trợ qua email cho người dùng trả phí
* ChatGPT có tính năng giúp đỡ tích hợp thông qua menu trợ giúp trong ứng dụng