**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN HỌC PHẦN**

TÊN HỌC PHẦN: **Lập trình Python (PYPRO)**

MÃ SỐ LỚP HP: **IPPA233277\_05CLC**

Tên chủ đề EDA: **EDA PHÂN TÍCH ELECTRIC VEHICLE POPULATION TRONG QUÝ 2 NĂM 2023**

Chủ đề CV: **LẬP TRÌNH PYTHON NHẬN DIỆN KHUÔN MẶT TRONG VIDEO**

Chủ đề Game: **LẬP TRÌNH PYTHON GAME CAR RACING**

**Họ tên sinh viên: ĐINH ĐẠI HẢI ĐĂNG**

**Mã số sinh viên: 21110164**

**Lớp: 21110CL3**

**Ngày nộp: 05/12/2023**

**Ký tên:**

**TP.HCM, ngày 05 Tháng 12 năm 2023**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA [CÔNG NGHỆ THÔNG TIN|CHẤT LƯỢNG CAO]**

**ĐỀ TÀI ĐỒ ÁN HỌC PHẦN**

TÊN HỌC PHẦN: **Lập trình Python (PYPRO)**

MÃ SỐ LỚP HP: **IPPA233277\_05CLC**

Tên chủ đề EDA: PHÂN TÍCH ELECTRIC VEHICLE POPULATION TRONG QUÝ 2 NĂM 2023

Chủ đề CV: LẬP TRÌNH PYTHON NHẬN DIỆN KHUÔN MẶT TRONG VIDEO

Chủ đề Game: LẬP TRÌNH PYTHON GAME CAR RACING

**Giảng viên giảng dạy: VÕ XUÂN THỂ**

**Họ tên sinh viên: ĐINH ĐẠI HẢI ĐĂNG**

**Mã số sinh viên: 21110164**

**Lớp: 21110CL3**

**TÊN CÁC FILE SẢN PHẨM ĐỀ TÀI:**

Thư mục bài làm: **G5C12DinhDaiHaiDang\_Topic**

Tập tin báo cáo bài làm: **2\_ G5C12DinhDaiHaiDang \_Topic.docx**

Tập tin mã nguồn: **G5C12DinhDaiHaiDang\_FullCode.rar**

Thư mục bài làm phần EDA: **G5C12DinhDaiHaiDang \_ ElectricVehiclePopulation**

Thư mục bài làm phần CV: **G5C12DinhDaiHaiDang\_CV. FaceDetector**

Thư mục bài làm phần Game: **G5C12DinhDaiHaiDang \_Game.CarRacing**

Giới thiệu thư mục bài làm: **1\_ G5C12DinhDaiHaiDang \_Intro.docx**

**Tập các thư viện sử dụng:** # Speech: speech\_recognition, gtts, playsound

# GUI: tkinter, threading, PIL

# EDA: pandas, numpy, scipy, sklearn

# CV: facenet\_pytorch, cv2

**NHẬN XÉT CỦA GIẢNG VIÊN**

**………………………………………………………..**

**………………………………………………………..**

**Ngày……./……../202…..**

**Giảng viên: Ký tên**

**TP.HCM, ngày 05 tháng 12 năm 2023**

# LỜI CẢM ƠN

Đồ án này là kết quả của quá trình học tập, tiếp thu kiến thức tại trường, lớp và cả những tìm tòi, nghiên cứu riêng của bản thân em và sự chỉ dạy tận tình của thầy, người đã trực tiếp hướng dẫn em trong môn học này. Do vậy, qua đây em xin phép được gửi lời cảm ơn chân thành nhất tới quý thầy.

Mặc dù đã dành nhiều thời gian và nỗ lực để hoàn thành bài đồ án này, nhưng do sự hạn chế về mặt kiến thức nên bài làm khó tránh khỏi những thiếu sót. Em kính mong nhận được những lời góp ý của thầy để bài làm ngày càng hoàn thiện hơn. Em xin chân thành cảm ơn!

**Ký tên**

***Đăng***

Đinh Đại Hải Đăng

# DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| CSDL hoặc DB | Cơ sở dữ liệu: DataBase |
| NSD = Users = Account | Người Sử Dụng = là quyền làm việc trên WebApp được đảm bảo bởi tối thiểu là 2 yếu tố: tên đăng nhập (UserName) và mật khẩu (Password) |
|  |  |
| AI | Artificial intelligence: Trí tuệ nhân tạo |
| EDA | Exploratory Data Analysis: Phân tích Khám phá Dữ liệu (Còn gọi là: phân tích dữ liệu thăm dò) |
| GUI | Graphical User Interface: Giao diện đồ hoạ |
| PYPRO | Lập trình Python |

**DANH MỤC CÁC THUẬT NGỮ**

|  |  |
| --- | --- |
| Giao diện người dùng:  User-Interface | Là hệ thống các màn hình giao tiếp cho phép người sử dụng tương tác với các thành phần phần mềm trong HTTT, thường là 1 trong các dong: Win-form, Web-form, Mobile-Form. |
| Tài khoản (Account) | Là một quyền làm việc trên HTTT được cấp phát cho một cá nhân thông qua tên tài khoản (username) và mật khẩu (password). |
| Trợ lý ảo: Voice Assistant | Là các hệ thống có khả năng “nghe” và “nói” với con người, nhờ đó hỗ trợ con người trong một số chức năng như một “trợ lý”. |

**MỤC LỤC**

[LỜI CẢM ƠN 3](#_Toc152278990)

[DANH MỤC CÁC TỪ VIẾT TẮT 4](#_Toc152278991)

[**DANH MỤC CÁC THUẬT NGỮ** 4](#_Toc152278992)

[Chương 1: GIỚI THIỆU VỀ ĐỒ ÁN HỌC PHẦN 9](#_Toc152278993)

[1.1. Tổng quan về Đồ án học phần 9](#_Toc152278994)

[1.2. Nội dung chuyên môn chính của Đồ án học phần 9](#_Toc152278995)

[1.3. Công cụ và nền tảng kỹ thuật thực hiện Đồ án học phần 10](#_Toc152278996)

[1.4. Bố cục của báo cáo 10](#_Toc152278997)

[Chương 2: CƠ SỞ LÝ THUYẾT CỦA HỌC PHẦN 11](#_Toc152278998)

[2.1. GIỚI THIỆU VỀ LẬP TRÌNH PYTHON 11](#_Toc152278999)

[2.1.1. Giới thiệu chung 11](#_Toc152279000)

[2.1.2. Cài đặt “bộ thảo chương” (soạn thảo chương trình IDE) và “bộ dịch” 11](#_Toc152279001)

[2.1.3. Giới thiệu các bộ tương tác lập trình 11](#_Toc152279002)

[2.1.4. Cấu hình mô trường tương tác lập trình 11](#_Toc152279003)

[2.2. TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH PYTHON 11](#_Toc152279004)

[2.2.1. Danh hiệu (Identifier) 11](#_Toc152279005)

[2.2.2. Về viết mã lệnh lập trình (Code) 11](#_Toc152279006)

[2.2.3. Về chú giải (Comment) 11](#_Toc152279007)

[2.2.4. Các phép toán 11](#_Toc152279008)

[2.2.5. Chuỗi 11](#_Toc152279009)

[2.2.6. Lệnh 11](#_Toc152279010)

[2.2.7. Định nghĩa hàm 11](#_Toc152279011)

[2.2.8. Danh sách 11](#_Toc152279012)

[2.2.9. Bộ dữ liệu: Tuple 11](#_Toc152279013)

[2.2.10. Tập hợp: Set 11](#_Toc152279014)

[2.2.11. Từ điển: Dictionary 11](#_Toc152279015)

[2.2.12. Module 11](#_Toc152279016)

[2.2.13. Package 11](#_Toc152279017)

[2.2.14. Lớp 11](#_Toc152279018)

[2.2.15. Xử lý ngoại lệ 11](#_Toc152279019)

[2.3. GIỚI THIỆU CÁC THƯ VIỆN VÀ CÔNG CỤ CĂN BẢN 11](#_Toc152279020)

[2.3.1. Nạp thư viện 11](#_Toc152279021)

[2.3.2. speech\_recognition 11](#_Toc152279022)

[2.3.3. gtts 11](#_Toc152279023)

[2.3.4. playsound 11](#_Toc152279024)

[2.3.5. tkinter 11](#_Toc152279025)

[2.3.6. os 11](#_Toc152279026)

[2.3.7. Pandas 11](#_Toc152279027)

[2.3.8. Tensorflow 12](#_Toc152279028)

[2.3.9. NumPy 12](#_Toc152279029)

[2.3.10. SCIPY 12](#_Toc152279030)

[2.3.11. Theano 12](#_Toc152279031)

[2.3.12. Matplotlib 12](#_Toc152279032)

[2.3.13. scikit-learn = sklearn 12](#_Toc152279033)

[2.3.14. Keras 12](#_Toc152279034)

[2.3.15. NLTK 12](#_Toc152279035)

[2.3.16. PyTorch 12](#_Toc152279036)

[2.3.17. LightGBM 12](#_Toc152279037)

[2.3.18. Eli5 12](#_Toc152279038)

[2.4. KỸ THUẬT LẬP TRÌNH ĐỆ QUY: THÁP HÀ NỘI (HANOI TOWER) 12](#_Toc152279039)

[2.5. LẬP TRÌNH PYTHON XỬ LÝ GIỌNG NÓI (Trợ lý ảo: Voice Assistant) 12](#_Toc152279040)

[2.5.1. Giới thiệu chủ đề 12](#_Toc152279041)

[2.5.2. Tài liệu và nguồn thực nghiệm 12](#_Toc152279042)

[2.5.3. Giới thiệu Các thư viện Python liên quan 12](#_Toc152279043)

[2.5.4. Giới thiệu bài toán 12](#_Toc152279044)

[2.5.5. Giới thiệu code tham khảo 12](#_Toc152279045)

[2.6. LẬP TRÌNH GIAO DIỆN ĐỒ HỌA (GUI) 12](#_Toc152279046)

[2.7. LẬP TRÌNH PHÂN TÍCH KHÁM PHÁ (EDA) [Thăm dò] 12](#_Toc152279047)

[2.8. LẬP TRÌNH ĐỒ THỊ & BIỂU ĐỒ TRỰC QUANG HÓA DỮ LIỆU (VISUALIZAION: PLOT) 12](#_Toc152279048)

[2.9. LẬP TRÌNH THỊ GIÁC MÁY TÍNH (CV) 12](#_Toc152279049)

[2.10. LẬP TRÌNH GAME CĂN BẢN 12](#_Toc152279050)

[Chương 3: PHÂN TÍCH VÀ XÁC ĐINH CÁC CƠ SỞ KỸ THUẬT 13](#_Toc152279051)

[3.1. GIỚI THIỆU CHUNG VỀ CÁC CHỦ ĐỀ CỦA ĐỒ ÁN 13](#_Toc152279052)

[3.2. LẬP TRÌNH XỬ GIỌNG NÓI (TRỢ LÝ ẢO: VOICE ASSISTANT) 13](#_Toc152279053)

[3.2.1. Giới thiệu chung về chủ đề 13](#_Toc152279054)

[3.2.1.1. Mô tả yêu cầu của chủ đề 13](#_Toc152279055)

[3.2.1.2. Phân tích yều cầu của chủ đề 13](#_Toc152279056)

[3.2.2. Các nền tảng kỹ thuật 13](#_Toc152279057)

[3.2.2.1. Nền tảng kỹ thuật cơ bản 13](#_Toc152279058)

[3.2.2.2. Hệ thống thư việc sử dụng 13](#_Toc152279059)

[3.2.3. Thiết kế giao diện 14](#_Toc152279060)

[3.2.4. Mã lệnh lập trình (python) 14](#_Toc152279061)

[3.3. LẬP TRÌNH GIAO DIỆN ĐỒ HỌA (GUI) 16](#_Toc152279062)

[3.4. LẬP TRÌNH TRỰC QUANG HÓA DỮ LIỆU (PLOT) 18](#_Toc152279063)

[Chương 4: LẬP TRÌNH PYTHON PHÂN TÍCH ELECTRIC VEHICLE POPULATION TRONG QUÝ 2 NĂM 2023 24](#_Toc152279064)

[4.1. Giới thiệu chủ đề EDA ELECTRIC VEHICLE POPULATION 24](#_Toc152279065)

[4.2. Phân tích yều cầu của chủ đề 24](#_Toc152279066)

[4.3. Các nền tảng kỹ thuật liên quan 24](#_Toc152279067)

[4.3.1. Hệ thống thư viện sử dụng 24](#_Toc152279068)

[4.3.2. Tập dữ liệu thực nghiệm 25](#_Toc152279069)

[4.3.3. Mô tả giải thuật 26](#_Toc152279070)

[4.4. Thiết kế giao diện 27](#_Toc152279071)

[4.5. Mã lệnh lập trình (python) 27](#_Toc152279072)

[Chương 5: LẬP TRÌNH PYTHON NHẬN DIỆN KHUÔN MẶT TRONG VIDEO 35](#_Toc152279073)

[5.1. Giới thiệu chủ đề Cv Nhận Diện Khuôn Mặt Trong Video 35](#_Toc152279074)

[5.2. Phân tích chủ đề Cv Nhận Diện Khuôn Mặt Trong Video 35](#_Toc152279075)

[5.3. Các nền tảng kỹ thuật CV 35](#_Toc152279076)

[5.3.1. Các thư viện liên quan 35](#_Toc152279077)

[5.3.2. Giới thiệu Video thực nghiệm 35](#_Toc152279078)

[5.4. Mô Tả Giải Thuật Cv Nhận Diện Khuôn Mặt Trong Video 35](#_Toc152279079)

[5.5. Giao diện CV Nhận Diện Khuôn Mặt Trong Video 36](#_Toc152279080)

[5.6. Mã lệnh lập trình CV nhận diện khuôn mặt từ Video 36](#_Toc152279081)

[Chương 6: LẬP TRÌNH PYTHON CAR RACING 40](#_Toc152279082)

[6.1. Giới Thiệu Chủ Đề Game **Car Racing** 40](#_Toc152279083)

[6.2. Phân Tích Chủ Đề Game **Car Racing** 40](#_Toc152279084)

[6.3. Các Nền Tảng Kỹ Thuật Lập Trình **Car Racing** 40](#_Toc152279085)

[6.3.1. Các Thư Viện Liên Quan 40](#_Toc152279086)

[6.3.2. Giới Thiệu Các Mẫu Media Thực Nghiệm (Mẫu Xe Và Đường Đua) 40](#_Toc152279087)

[6.4. Giao diện CAR RACING 41](#_Toc152279088)

[6.5. Mã lệnh lập trình Game Car Racing 41](#_Toc152279089)

[Chương 7: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỒ ÁN HỌC PHẦN 49](#_Toc152279090)

[7.1. Kết luận 49](#_Toc152279091)

[7.1.1. Những kết quả đạt được 49](#_Toc152279092)

[7.1.2. Hạn chế 49](#_Toc152279093)

[7.1.3. Hướng khắc phục các hạn chế 49](#_Toc152279094)

[DANH MỤC TÀI LIỆU THAM KHẢO 50](#_Toc152279095)

1. **GIỚI THIỆU VỀ ĐỒ ÁN HỌC PHẦN**

## Tổng quan về Đồ án học phần

Đồ án Học phần vận dụng kiến thức và kỹ năng về Lập trình Python để xây dựng giao diện đồ hoạ (GUI) cho phép: tương tác với người dùng bằng giọng nói (Trợ lý ảo: Voice Assistant) với các chủ đề:

+ Phân tích thăm dò (EDA: Epolore Data Analysing) ELECTRIC VEHICLE POPULATION tại Hoa Kỳ trong Quý 2 Năm 2023 đối với tập dữ liệu thực nghiệm theo link: https://www.kaggle.com/datasets/willianoliveiragibin/electric-vehicle-population; có trực quang hóa dữ liệu dùng các kỹ thuật biểu đồ và đồ thị (Plot)

+ CV Nhận Diện Khuôn Mặt Trong Video

+ Games Car Racing

## Nội dung chuyên môn chính của Đồ án học phần

Bao gồm 6 nội dung lập trình chính dùng ngôn ngữ lập trình Python:

+ Trợ lý ảo: . thông tin nhập có 2 phương án (lời nói dạng “chọn” OR nhập ký tự)

. tin xuất “kết quả thăm dò” ở cả 2 dạng: text và lời nói

+ Phân tích dữ liệu khám phá || thăm dò [EDA] ELECTRIC VEHICLE POPULATION tại Hoa Kỳ trong Quý 2 Năm 2023

+ Trực quang hóa dữ liệu thăm dò ELECTRIC VEHICLE POPULATION bằng biểu đồ | đồ thi.

+ Giao diện đồ họa dạng [WinForm or Webform or Mobileform]

+ CV: Thị giác máy tính cho phép nhận diện khuôn mặt trong video

+ Game căn bản Car Racing (“Đua xe”).

Nội dung chuyên môn của Đồ án này là nền tảng cho các lĩnh vực:

1. Lập trình AI (TTNT), ML (HM), Deep Learning (HS)
2. Data Science: Lập trình Khoa học dữ liệu, đặc biệt EDA (…..): Phân tích dữ liệu thăm dò
3. CV: Lập trình Thị giác máy tính
4. Games: Lập trình Games
5. Data Visualization = Plot: Lập trình trực quang hóa dữ liệu, dùng biểu đồ & đồ thi
6. GUI (….): Lập trình giao diện đồ họa

## Công cụ và nền tảng kỹ thuật thực hiện Đồ án học phần

+ Ngôn ngữ lập trình **Python**

+ Công cụ lập trình và biên dịch: **PyCharm**

+ Các thư viện chính:

import tkinter as tk  
from threading import Thread  
from PIL import Image, ImageTk  
import CarRacing  
import CV  
import EDA

import pygame  
import sys  
import random  
from pygame.locals import \*

import os  
import speech\_recognition as sr  
import cv2  
from PIL import Image, ImageTk  
from facenet\_pytorch import MTCNN

## Bố cục của báo cáo

Báo cáo gồm những nội dung như sau:

Chương 1: Giới thiệu đồ án của Học phần

Chương 2: Các cơ sở lý thuyết của Học phần phục việc thực hiện đề tài

Chương 3: Phân tích và xác đinh các cơ sở kỹ thuật thực hiện đề tài

Chương 4: Lập Trình Phân Tích Electric Vehicle Population Trong Quý 2 Năm 2023: 12DinhDaiHaiDang\_ElectricVehiclePopulationData

Chương 5: Lập Trình Python Nhận Diện Khuôn Mặt Trong Video

Chương 6: Lập trình Game Car Racing

Chương 7: Tổng kết các kết quản đạt được và còn hạn chế của đồ án, đồng thời đề xuất hướng khắc phục hạn chế và phát triển Đồ án.

1. **CƠ SỞ LÝ THUYẾT CỦA HỌC PHẦN**

## GIỚI THIỆU VỀ LẬP TRÌNH PYTHON

### Giới thiệu chung

### Cài đặt “bộ thảo chương” (soạn thảo chương trình IDE[[1]](#footnote-1)) và “bộ dịch”

### Giới thiệu các bộ tương tác lập trình

### Cấu hình mô trường tương tác lập trình

## TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH PYTHON

### Danh hiệu (Identifier)

### Về viết mã lệnh lập trình (Code)

### Về chú giải (Comment)

### Các phép toán

### Chuỗi

### Lệnh

### Định nghĩa hàm

### Danh sách

### Bộ dữ liệu: Tuple

### Tập hợp: Set

### Từ điển: Dictionary

### Module

### Package

### Lớp

### Xử lý ngoại lệ

## GIỚI THIỆU CÁC THƯ VIỆN VÀ CÔNG CỤ CĂN BẢN

### Nạp thư viện

### speech\_recognition

### gtts

### playsound

### tkinter

### os

### Pandas

### Tensorflow

### NumPy

### SCIPY

### Theano

### Matplotlib

### scikit-learn = sklearn

### Keras

### NLTK

### PyTorch

### LightGBM

### Eli5

## KỸ THUẬT LẬP TRÌNH ĐỆ QUY: THÁP HÀ NỘI (HANOI TOWER)

## LẬP TRÌNH PYTHON XỬ LÝ GIỌNG NÓI (Trợ lý ảo: Voice Assistant)

### Giới thiệu chủ đề

### Tài liệu và nguồn thực nghiệm

### Giới thiệu Các thư viện Python liên quan

### Giới thiệu bài toán

### Giới thiệu code tham khảo

**[1] Nghe tiếng Việt => Text**

**[2] Text => đọc tiếng Việt**

## LẬP TRÌNH GIAO DIỆN ĐỒ HỌA (GUI)

## LẬP TRÌNH PHÂN TÍCH KHÁM PHÁ (EDA) [Thăm dò]

## LẬP TRÌNH ĐỒ THỊ & BIỂU ĐỒ TRỰC QUANG HÓA DỮ LIỆU (VISUALIZAION: PLOT)

## LẬP TRÌNH THỊ GIÁC MÁY TÍNH (CV)

## LẬP TRÌNH GAME CĂN BẢN

1. **PHÂN TÍCH VÀ XÁC ĐINH CÁC CƠ SỞ KỸ THUẬT**

## GIỚI THIỆU CHUNG VỀ CÁC CHỦ ĐỀ CỦA ĐỒ ÁN

Đồ án này gồm 3 chủ đề chính:

+ Lập trình EDA thăm dò Phân Tích Electric Vehicle Population Trong Quý 2 Năm 2023 [Chương 4:]; có trực quang hóa dữ liệu dùng Plot vẽ đồ thị và biểu đồ [3.4].

+ Lập trình CV hỗ trợ nhận diện Nhận Diện Khuôn Mặt Trong Video [Chương 5:]

+ Lập trình Game căn bản: Car Racing [Chương 6:]

Được thực hiện trên hệ thống giao diện GUI [3.3] với cơ chế nhập thông tin vào (input) và xuất kết quả (output) theo hình thức Text và Voice nhờ xử lý “Trợ lý ảo” [3.2].

## LẬP TRÌNH XỬ GIỌNG NÓI (TRỢ LÝ ẢO: VOICE ASSISTANT)

## Giới thiệu chung về chủ đề

### Mô tả yêu cầu của chủ đề

+ Lập trình App xử lý "lời nới" (Speech): Ứng dụng “Trợ lý ảo” (Voice Assistant)

+ Gồm 2 phần (Part) cơ bản sau

1. Nghe tiếng Việt => Text: alo alo

2. Trả lời bằng tiếng Việt : Text => Nói tiếng Việt: alo alo

### Phân tích yều cầu của chủ đề

+ Chủ đề lập trình python này là cơ sở của các ứng dụng về AI, ML, Deep ML.

+ Các hệ thống này thường có 3 phần :

Part1: SpeechReconigition: chính là part1 của chủ đề này

Part2: Các kỹ thuật AI

Part3: VoiceAssistant: chính là part2 của chủ đề này

## Các nền tảng kỹ thuật

### Nền tảng kỹ thuật cơ bản

### Hệ thống thư việc sử dụng

B1: Cài đặt thư viện

pip install pillow

pip install numpy

pip install pandas

pip install matplotlib

pip install scipy

pip install scikit-learn

pip install opencv-python

pip install SpeechRecognition

pip install facenet-pytorch

pip install pygameB2 : Nạp thư viện vào App

## Thiết kế giao diện

+ Xử lý trong cửa sổ Console Output của PyCharm



## Mã lệnh lập trình (python)

**FULL CODE**

**G5C12DinhDaiHaiDang \_Topic.py**

import tkinter as tk  
from threading import Thread  
from tkinter import Canvas  
from PIL import Image, ImageTk  
import CarRacing  
import CV  
import EDA  
  
  
class MyGUI:  
 def \_\_init\_\_(self, master):  
 self.master = master  
 master.title("Đồ án cuối kỳ")  
 master.geometry("1000x500")  
  
 *# Tạo ba nút* self.button1 = tk.Button(self.master, text="Face Detection", command=self.FaceDetection, width=11, height=2)  
 self.button2 = tk.Button(self.master, text="EDA", command=self.EDA, width=11, height=2)  
 self.button3 = tk.Button(self.master, text="Car Racing", command=self.CarRacingBoy, width=11, height=2)  
 self.exit\_button = tk.Button(self.master, text="Exit", command=self.exit\_application, width=11, height=2)  
  
 *# Đặt ba nút vào cuối cùng và căn đều theo chiều dài* self.button1.place(x=350, y=420)  
 self.button2.place(x=450, y=420)  
 self.button3.place(x=550, y=420)  
 self.exit\_button.place(x=850, y=420)  
  
 screen\_width = root.winfo\_screenwidth()  
 screen\_height = root.winfo\_screenheight()  
  
 *# Background* self.image = Image.open("img/1.png")  
 tk\_image = ImageTk.PhotoImage(self.image)  
  
 canvas = Canvas(root, width=900, height=400)  
 canvas.pack()  
  
 canvas.create\_image(0, 0, anchor="nw", image=tk\_image)  
 canvas.image = tk\_image  
  
 def FaceDetection(self):  
 face\_detector = CV.FaceDetector(self.master)  
 face\_detector.run\_gui()  
  
 def EDA(self):  
 eda = EDA.DataPreprocessing(root)  
  
 def CarRacingBoy(self):  
 *# Ẩn cửa sổ Tkinter* self.master.withdraw()  
  
 *# Tạo một thể hiện của RacingGame* car = CarRacing.RacingGame()  
  
 *# Chạy trò chơi trong một luồng riêng biệt* game\_thread = Thread(target=car.run\_game)  
  
 *# Bắt đầu luồng trò chơi* game\_thread.start()  
  
 *# Đợi cho luồng trò chơi kết thúc và sau đó hiển thị lại cửa sổ Tkinter* self.master.after(0, self.check\_game\_status, game\_thread)  
  
 def check\_game\_status(self, game\_thread):  
 *# Kiểm tra trạng thái của luồng trò chơi* if game\_thread.is\_alive():  
 *# Nếu trò chơi vẫn đang chạy, kiểm tra lại sau một khoảng thời gian nhất định* self.master.after(100, self.check\_game\_status, game\_thread)  
 else:  
 *# Nếu trò chơi đã kết thúc, hiển thị lại cửa sổ Tkinter* self.master.deiconify()  
  
 def exit\_application(self):  
 *# Kết thúc ứng dụng khi nút Exit được nhấn* self.master.destroy()  
  
  
root = tk.Tk()  
app = MyGUI(root)  
root.mainloop()

## LẬP TRÌNH GIAO DIỆN ĐỒ HỌA (GUI)

Trong phần này, chúng ta sẽ thêm giao diện đồ họa (GUI) cho chương trình. Chúng ta sẽ sử dụng thư viện **tkinter** để tạo cửa sổ và các thành phần giao diện như nút và hình ảnh.

**CODE :**

import tkinter as tk  
from threading import Thread  
from tkinter import Canvas  
from PIL import Image, ImageTk  
import CarRacing  
import CV  
import EDA  
  
  
class MyGUI:  
 def \_\_init\_\_(self, master):  
 self.master = master  
 master.title("Đồ án cuối kỳ")  
 master.geometry("1000x500")  
  
 *# Tạo ba nút* self.button1 = tk.Button(self.master, text="Face Detection", command=self.FaceDetection, width=11, height=2)  
 self.button2 = tk.Button(self.master, text="EDA", command=self.EDA, width=11, height=2)  
 self.button3 = tk.Button(self.master, text="Car Racing", command=self.CarRacingBoy, width=11, height=2)  
 self.exit\_button = tk.Button(self.master, text="Exit", command=self.exit\_application, width=11, height=2)  
  
 *# Đặt ba nút vào cuối cùng và căn đều theo chiều dài* self.button1.place(x=350, y=420)  
 self.button2.place(x=450, y=420)  
 self.button3.place(x=550, y=420)  
 self.exit\_button.place(x=850, y=420)  
  
 screen\_width = root.winfo\_screenwidth()  
 screen\_height = root.winfo\_screenheight()  
  
 *# Background* self.image = Image.open("img/1.png")  
 tk\_image = ImageTk.PhotoImage(self.image)  
  
 canvas = Canvas(root, width=900, height=400)  
 canvas.pack()  
  
 canvas.create\_image(0, 0, anchor="nw", image=tk\_image)  
 canvas.image = tk\_image  
  
 def FaceDetection(self):  
 face\_detector = CV.FaceDetector(self.master)  
 face\_detector.run\_gui()  
  
 def EDA(self):  
 eda = EDA.DataPreprocessing(root)  
  
 def CarRacingBoy(self):  
 *# Ẩn cửa sổ Tkinter* self.master.withdraw()  
  
 *# Tạo một thể hiện của RacingGame* car = CarRacing.RacingGame()  
  
 *# Chạy trò chơi trong một luồng riêng biệt* game\_thread = Thread(target=car.run\_game)  
  
 *# Bắt đầu luồng trò chơi* game\_thread.start()  
  
 *# Đợi cho luồng trò chơi kết thúc và sau đó hiển thị lại cửa sổ Tkinter* self.master.after(0, self.check\_game\_status, game\_thread)  
  
 def check\_game\_status(self, game\_thread):  
 *# Kiểm tra trạng thái của luồng trò chơi* if game\_thread.is\_alive():  
 *# Nếu trò chơi vẫn đang chạy, kiểm tra lại sau một khoảng thời gian nhất định* self.master.after(100, self.check\_game\_status, game\_thread)  
 else:  
 *# Nếu trò chơi đã kết thúc, hiển thị lại cửa sổ Tkinter* self.master.deiconify()  
  
 def exit\_application(self):  
 *# Kết thúc ứng dụng khi nút Exit được nhấn* self.master.destroy()  
  
  
root = tk.Tk()  
app = MyGUI(root)  
root.mainloop()

## LẬP TRÌNH TRỰC QUANG HÓA DỮ LIỆU (PLOT)

Trong phần này, chúng ta sẽ thêm phần liên quan đến trực quan hóa dữ liệu (Plot) sử dụng thư viện **matplotlib**.

class DataPreprocessing:  
 def \_\_init\_\_(self, master, directory\_path="./"):  
 self.y = None  
 self.X = None  
 self.master = master  
   
 self.top\_level\_window = tk.Toplevel(self.master)  
   
 self.top\_level\_window.title("Data Loader")  
  
 self.directory\_path = directory\_path  
 self.selected\_csv\_file = None  
 self.file\_path = None  
 self.df = None  
  
 self.create\_widgets()  
  
 def on\_close\_callback(self):  
 *# Close the top-level window* self.top\_level\_window.destroy()  
  
 def create\_widgets(self):  
 *# Label and Entry for directory path* tk.Label(self.top\_level\_window, text="Đường dẫn thư mục:").grid(row=0, column=0, padx=10, pady=10)  
 self.directory\_entry = tk.Entry(self.top\_level\_window, width=50)  
 self.directory\_entry.grid(row=0, column=1, padx=10, pady=10)  
  
 tk.Button(self.top\_level\_window, text="Chọn thư mục", command=self.browse\_directory).grid(row=0, column=2, padx=10, pady=10)  
  
 *# Button to load data* tk.Button(self.top\_level\_window, text="Load Data", command=self.load\_data).place(x=60, y=55)  
  
 tk.Button(self.top\_level\_window, text="Processing", command=self.preprocessing).place(x=180, y=55)  
  
 tk.Button(self.top\_level\_window, text='Biểu Đồ', command=self.show\_plot).place(x=300, y=55)  
  
 tk.Button(self.top\_level\_window, text="Close", command=self.on\_close\_callback).grid(row=1, column=2, pady=10)  
  
 *# Textbox to display DataFrame* self.textbox = tk.Text(self.top\_level\_window, width=80, height=20)  
 self.textbox.grid(row=2, column=0, columnspan=3, padx=10, pady=10)  
  
  
 def browse\_directory(self):  
 self.directory\_path = filedialog.askdirectory()  
 self.directory\_entry.delete(0, tk.END)  
 self.directory\_entry.insert(tk.END, self.directory\_path)  
  
 def update\_textbox(self, message):  
 self.textbox.insert(tk.END, message + '\n')  
 self.textbox.update\_idletasks()  
  
 def get\_user\_input(self, prompt):  
 dialog = CustomDialog(self.top\_level\_window, prompt)  
 self.top\_level\_window.wait\_window(dialog)  
 user\_input = dialog.result  
 return user\_input  
  
 def load\_data(self):  
 if self.directory\_path and os.path.exists(self.directory\_path):  
 all\_files = os.listdir(self.directory\_path)  
 csv\_files = [f for f in all\_files if f.endswith(".csv")]  
  
 if csv\_files:  
 self.update\_textbox("Danh sách tệp CSV có sẵn:")  
 for i, csv\_file in enumerate(csv\_files):  
 self.update\_textbox(f"{i + 1}. {csv\_file}")  
  
 user\_input = self.get\_user\_input("Nhập số tương ứng với tệp bạn muốn chọn:")  
 if user\_input is not None:  
 print("Giá trị đã nhập:", user\_input)  
 else:  
 self.update\_textbox("Đã xảy ra lỗi khi nhập giọng nói.")  
 *# selected\_index = self.get\_user\_input("Nhập số tương ứng với tệp bạn muốn chọn:") - 1* if 0 <= user\_input - 1 < len(csv\_files):  
 self.selected\_csv\_file = csv\_files[user\_input - 1]  
 self.file\_path = os.path.join(self.directory\_path, self.selected\_csv\_file)  
 self.update\_textbox(f"Bạn đã chọn tệp: {self.selected\_csv\_file}")  
 self.update\_textbox(f"Đường dẫn tệp: {self.file\_path}")  
  
 *# Đọc dữ liệu từ tệp CSV đã chọn* self.df = pd.read\_csv(self.file\_path)  
 self.update\_textbox(f'Độ lớn của bảng [frame] dữ liệu: {self.df.shape}')  
  
 *# Hiển thị số lượng dòng từ DataFrame* while True:  
 try:  
 num\_rows\_to\_display = 10  
 self.update\_textbox(str(self.df.head(num\_rows\_to\_display)))  
 break  
 except ValueError:  
 self.update\_textbox("Lựa chọn không hợp lệ. Vui lòng nhập một số nguyên.")  
 else:  
 self.update\_textbox("Lựa chọn không hợp lệ. Vui lòng chọn số thứ tự hợp lệ.")  
  
 else:  
 self.update\_textbox("Không có tệp CSV nào trong thư mục.")  
 else:  
 self.update\_textbox(f"Thư mục '{self.directory\_path}' không tồn tại.")  
  
 def preprocessing(self):  
 *# Bước 3: Xử lý CỘT dữ liệu NULL quá nhiều OR không có giá trị phân tích* count\_values = self.df.count().sort\_values()  
 self.update\_textbox("Số lượng giá trị không NULL cho mỗi cột:")  
 self.update\_textbox(count\_values.to\_string()) *# Chuyển đổi thành chuỗi và hiển thị* self.update\_textbox("Danh sách các cột:")  
 for i, column in enumerate(self.df.columns):  
 self.update\_textbox(f"{i}. {column}")  
  
 *# Chọn cột cần xóa* columns\_to\_delete = [0, 1, 3]  
  
 *# Xóa cột đã chọn* self.df = self.df.drop(self.df.columns[columns\_to\_delete], axis=1)  
  
 *# Bước 4: Xử lý DÒNG dữ liệu NULL* self.df = self.df.dropna(how='any')  
 self.update\_textbox(f"Kích thước DataFrame sau khi xóa các dòng có giá trị null: {self.df.shape}")  
  
 *# Bước 5: Xử lý loại bỏ các giá trị ngoại lệ  
 # Tính toán Z-Score* z = np.abs(stats.zscore(self.df[['Postal Code', 'Electric Range', 'Base MSRP', 'DOL Vehicle ID', 'Census Tract']]))  
  
 threshold = 2  
  
 self.update\_textbox(f"Bạn đã chọn ngưỡng Z-Score là {threshold}")  
  
 *# Lọc dữ liệu dựa trên Z-Score* self.df = self.df[(z < threshold).all(axis=1)]  
 self.update\_textbox(f"Kích thước DataFrame sau khi lọc dựa trên Z-Score: {self.df.shape}")  
  
 *# Bước 7: Chuẩn hóa tập dữ liệu Input dùng MinMaxScaler* scaler = preprocessing.MinMaxScaler()  
 selected\_columns = ['Electric Range', 'Base MSRP', 'DOL Vehicle ID', 'Census Tract']  
 scaler.fit(self.df[selected\_columns])  
 self.df[selected\_columns] = scaler.transform(self.df[selected\_columns])  
  
 *# In ra một số dòng của DataFrame đã xử lý* self.update\_textbox(self.df.head().to\_string())  
  
 *# Bước 8: Xác định mô hình trích lọc các thuộc tính đặc trưng: EDA* numeric\_columns = self.df.select\_dtypes(include=['float64', 'int64']).columns  
 if 'Census Tract' in numeric\_columns:  
 numeric\_columns = numeric\_columns.drop('Census Tract')  
  
 X = self.df.loc[:, numeric\_columns]  
 y = self.df['Census Tract']  
 selector = SelectKBest(f\_regression, k=3)  
 selector.fit(X, y)  
 X\_new = selector.transform(X)  
 self.update\_textbox(f"{X\_new}\n")  
 self.update\_textbox(pd.DataFrame(y).to\_string(index=False) + '\n')  
 selected\_columns = X.columns[selector.get\_support(indices=True)]  
 self.update\_textbox("\n".join(selected\_columns))  
  
 *# Bước 9: Xác định mô hình trích lọc các thuộc tính đặc trưng  
 # XĐ data frame = Chiếu lấy các thuộc tính đặc trưng đã xđ trong B8* self.df = self.df[['Electric Range', 'Legislative District', 'DOL Vehicle ID', 'Census Tract']]  
  
 *# Bước 10: EDA theo nhu cầu thực tế => input vào các mô hình AI, ML,...  
 # Đơn giản nhất là lấy 1 thuộc tính đầu vào (Electric Range) để XD Mô hình* self.X = self.df[['Electric Range']]  
 self.y = self.df[['Census Tract']]  
 X\_str = X.to\_string(index=False)  
 y\_str = y.to\_string(index=False)  
  
 self.update\_textbox("Dữ liệu của Electric Range:\n" + X\_str)  
 self.update\_textbox("Dữ liệu của Census Tract:\n" + y\_str)  
  
  
 def show\_plot(self):  
 *# Tạo một đối tượng Figure của Matplotlib* fig, ax = plt.subplots()  
  
 *# Vẽ biểu đồ trên đối tượng Axis* ax.plot(self.X, self.y, 'o', alpha=0.3)  
 ax.set\_title('Mối quan hệ giữa Electric Range và Census Tract')  
 ax.set\_xlabel('Electric Range')  
 ax.set\_ylabel('Census Tract')  
  
 *# Tạo đối tượng FigureCanvasTkAgg để tích hợp vào Tkinter* canvas = FigureCanvasTkAgg(fig, master=self.top\_level\_window)  
 canvas.draw()  
  
 *# Tạo một cửa sổ Toplevel mới để hiển thị biểu đồ* toplevel = tk.Toplevel(self.top\_level\_window)  
 toplevel.title('Biểu Đồ')  
  
 *# Tạo một đối tượng FigureCanvasTkAgg mới cho mỗi cửa sổ Toplevel* canvas\_toplevel = FigureCanvasTkAgg(fig, master=toplevel)  
 canvas\_toplevel.draw()  
  
 *# Hiển thị FigureCanvasTkAgg trong Toplevel* canvas\_toplevel.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

1. **LẬP TRÌNH PYTHON PHÂN TÍCH ELECTRIC VEHICLE POPULATION TRONG QUÝ 2 NĂM 2023**

## Giới thiệu chủ đề EDA ELECTRIC VEHICLE POPULATION

Vận dụng kỹ thuật lập trình EDA kết hợp với GUI và Voice Assistant cho phép dùng tập dữ liệu thực nghiệm về dân số sử đụng để dự đoán và tư vấn các nhà quản lý hoạch định kế hoạch bán xe điện trong các năm tới.

Dữ liệu thực nghiệm gồm dân số sử dụng xe diện trong quá khứ của Hoa Kỳ

12DinhDaiHaiDang\_ElectricVehiclePopulationData.csv : dạng file Excel .csv

. Số records: 150482

. Số thuộc tính input: 17 để đoán thuộc tính thứ 17: target variable:

= “Census Tract”[Y|N]

## Phân tích yều cầu của chủ đề

+ Chủ đề lập trình python này là cơ sở của các ứng dụng về DataWarehouse, Bigdata, Data Analysis, Data Sciences, . . .

+ Đây là xử lý nền tảng cho các nội dung chuyên môn (nêu trên) chủ yếu là giai đoạn tiền xử lý dữ liệu đầu vào (Input Preporcessing), như: xử lý NULL, lọc dữ liệu, tích hợp dữ liệu (Integration),…

**Một số trường hợp tiền xử lý dữ liệu đầu vào (input) của các hệ thống EDA**

[1] Xử lý NULL là 150137

## Các nền tảng kỹ thuật liên quan

### Hệ thống thư viện sử dụng

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

from tkinter import filedialog, simpledialog  
import numpy as np  
import pandas as pd  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
from scipy import stats  
from sklearn import preprocessing  
from sklearn.feature\_selection import SelectKBest, f\_regression  
import os  
import tkinter as tk  
import speech\_recognition as sr  
import matplotlib.pyplot as plt

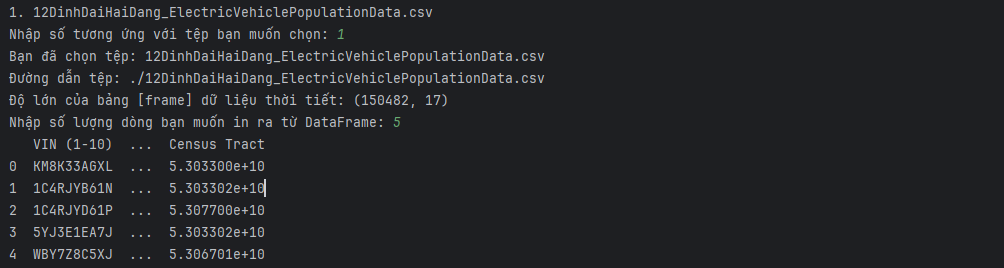
### Tập dữ liệu thực nghiệm

12DinhDaiHaiDang\_ElectricVehiclePopulationData

**Bước 2: Tải tập dữ liệu**

def load\_data(self):  
 if self.directory\_path and os.path.exists(self.directory\_path):  
 all\_files = os.listdir(self.directory\_path)  
 csv\_files = [f for f in all\_files if f.endswith(".csv")]  
  
 if csv\_files:  
 self.update\_textbox("Danh sách tệp CSV có sẵn:")  
 for i, csv\_file in enumerate(csv\_files):  
 self.update\_textbox(f"{i + 1}. {csv\_file}")  
  
 user\_input = self.get\_user\_input("Nhập số tương ứng với tệp bạn muốn chọn:")  
 if user\_input is not None:  
 print("Giá trị đã nhập:", user\_input)  
 else:  
 self.update\_textbox("Đã xảy ra lỗi khi nhập giọng nói.")  
 *# selected\_index = self.get\_user\_input("Nhập số tương ứng với tệp bạn muốn chọn:") - 1* if 0 <= user\_input - 1 < len(csv\_files):  
 self.selected\_csv\_file = csv\_files[user\_input - 1]  
 self.file\_path = os.path.join(self.directory\_path, self.selected\_csv\_file)  
 self.update\_textbox(f"Bạn đã chọn tệp: {self.selected\_csv\_file}")  
 self.update\_textbox(f"Đường dẫn tệp: {self.file\_path}")  
  
 *# Đọc dữ liệu từ tệp CSV đã chọn* self.df = pd.read\_csv(self.file\_path)  
 self.update\_textbox(f'Độ lớn của bảng [frame] dữ liệu: {self.df.shape}')  
  
 *# Hiển thị số lượng dòng từ DataFrame* while True:  
 try:  
 num\_rows\_to\_display = 10  
 self.update\_textbox(str(self.df.head(num\_rows\_to\_display)))  
 break  
 except ValueError:  
 self.update\_textbox("Lựa chọn không hợp lệ. Vui lòng nhập một số nguyên.")  
 else:  
 self.update\_textbox("Lựa chọn không hợp lệ. Vui lòng chọn số thứ tự hợp lệ.")  
  
 else:  
 self.update\_textbox("Không có tệp CSV nào trong thư mục.")  
 else:  
 self.update\_textbox(f"Thư mục '{self.directory\_path}' không tồn tại.")

Chạy python G5C12DinhDaiHaiDang \_Topic.py sẽ hiển thị kết quả:



### Mô tả giải thuật

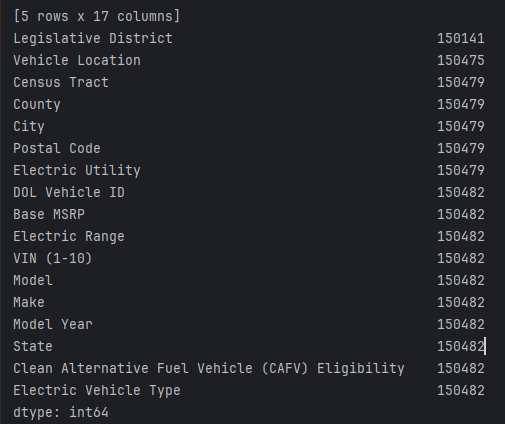
- Z-Score là một phương pháp thống kê được sử dụng để đo lường mức độ chênh lệch của một giá trị so với trung bình của một tập dữ liệu và đo lường nó theo đơn vị độ lệch chuẩn

**Bước 3: Xử lý dữ liệu**

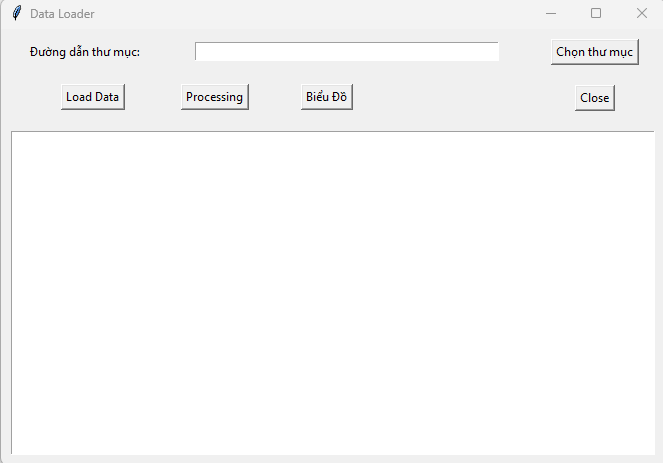
#Xử lý CỘT dữ liệu NULL quá nhiều OR không có giá trị phân tích

print(df.count().sort\_values())

**Kết quả:**



## Thiết kế giao diện



## Mã lệnh lập trình (python)

from tkinter import filedialog, simpledialog  
import numpy as np  
import pandas as pd  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
from scipy import stats  
from sklearn import preprocessing  
from sklearn.feature\_selection import SelectKBest, f\_regression  
import os  
import tkinter as tk  
import speech\_recognition as sr  
import matplotlib.pyplot as plt  
  
  
class CustomDialog(tk.Toplevel):  
 def \_\_init\_\_(self, parent, prompt):  
 super().\_\_init\_\_(parent)  
 self.title("Chọn phương thức nhập")  
 self.prompt\_label = tk.Label(self, text=prompt)  
 self.prompt\_label.pack(pady=10)  
  
 self.voice\_button = tk.Button(self, text="Voice", command=self.get\_voice\_input)  
 self.voice\_button.pack(pady=5)  
  
 self.manual\_button = tk.Button(self, text="Input", command=self.get\_manual\_input)  
 self.manual\_button.pack(pady=5)  
  
 self.result = None  
  
 def get\_voice\_input(self):  
 recognizer = sr.Recognizer()  
 with sr.Microphone() as source:  
 print("Hãy nói gì đó...")  
 audio = recognizer.record(source, duration=3)  
  
 try:  
 text = recognizer.recognize\_google(audio, language="vi-VN")  
 return text  
 except sr.UnknownValueError:  
 return "Không thể nhận dạng giọng nói."  
 except sr.RequestError as e:  
 return "Lỗi trong việc kết nối tới Google API: {0}".format(e)  
  
 def get\_manual\_input(self):  
 *# Hiển thị hộp thoại nhập bằng tay* manual\_input = simpledialog.askinteger("Input", "Nhập số tương ứng với tệp bạn muốn chọn:")  
 if manual\_input is not None:  
 self.result = manual\_input  
 self.destroy() *# Đóng hộp thoại sau khi nhập bằng tay* def get\_user\_input\_with\_voice(self):  
 voice\_input = self.get\_voice\_input()  
 if voice\_input:  
 *# Hiển thị giọng nói đã nhận dạng và chuyển đổi thành số (nếu có thể)* print("Giọng nói đã nhận dạng: ", voice\_input)  
 try:  
 user\_input = self.convert(voice\_input)  
 self.result = user\_input  
 self.destroy()  
 except ValueError:  
 print("Không thể chuyển đổi thành số.")  
 else:  
 print("Đã xảy ra lỗi khi nhập giọng nói.")  
  
 def convert(self, str\_number):  
 so\_dict = {  
 "một": 1, "hai": 2, "ba": 3, "bốn": 4, "năm": 5,  
 "sáu": 6, "bảy": 7, "tám": 8, "chín": 9,  
 "mười": 10, "mười một": 11, "mười hai": 12,  
 "mười ba": 13, "mười bốn": 14, "mười lăm": 15,  
 "mười sáu": 16, "mười bảy": 17, "mười tám": 18,  
 "mười chín": 19, "hai mươi": 20, "hai mươi một": 21,  
 "1": 1, "2": 2, "3": 3, "4": 4, "5": 5  
 }  
 return so\_dict.get(str\_number, "none")  
  
  
class DataPreprocessing:  
 def \_\_init\_\_(self, master, directory\_path="./"):  
 self.y = None  
 self.X = None  
 self.master = master  
   
 self.top\_level\_window = tk.Toplevel(self.master)  
   
 self.top\_level\_window.title("Data Loader")  
  
 self.directory\_path = directory\_path  
 self.selected\_csv\_file = None  
 self.file\_path = None  
 self.df = None  
  
 self.create\_widgets()  
  
 def on\_close\_callback(self):  
 *# Close the top-level window* self.top\_level\_window.destroy()  
  
 def create\_widgets(self):  
 *# Label and Entry for directory path* tk.Label(self.top\_level\_window, text="Đường dẫn thư mục:").grid(row=0, column=0, padx=10, pady=10)  
 self.directory\_entry = tk.Entry(self.top\_level\_window, width=50)  
 self.directory\_entry.grid(row=0, column=1, padx=10, pady=10)  
  
 tk.Button(self.top\_level\_window, text="Chọn thư mục", command=self.browse\_directory).grid(row=0, column=2, padx=10, pady=10)  
  
 *# Button to load data* tk.Button(self.top\_level\_window, text="Load Data", command=self.load\_data).place(x=60, y=55)  
  
 tk.Button(self.top\_level\_window, text="Processing", command=self.preprocessing).place(x=180, y=55)  
  
 tk.Button(self.top\_level\_window, text='Biểu Đồ', command=self.show\_plot).place(x=300, y=55)  
  
 tk.Button(self.top\_level\_window, text="Close", command=self.on\_close\_callback).grid(row=1, column=2, pady=10)  
  
 *# Textbox to display DataFrame* self.textbox = tk.Text(self.top\_level\_window, width=80, height=20)  
 self.textbox.grid(row=2, column=0, columnspan=3, padx=10, pady=10)  
  
  
 def browse\_directory(self):  
 self.directory\_path = filedialog.askdirectory()  
 self.directory\_entry.delete(0, tk.END)  
 self.directory\_entry.insert(tk.END, self.directory\_path)  
  
 def update\_textbox(self, message):  
 self.textbox.insert(tk.END, message + '\n')  
 self.textbox.update\_idletasks()  
  
 def get\_user\_input(self, prompt):  
 dialog = CustomDialog(self.top\_level\_window, prompt)  
 self.top\_level\_window.wait\_window(dialog)  
 user\_input = dialog.result  
 return user\_input  
  
 def load\_data(self):  
 if self.directory\_path and os.path.exists(self.directory\_path):  
 all\_files = os.listdir(self.directory\_path)  
 csv\_files = [f for f in all\_files if f.endswith(".csv")]  
  
 if csv\_files:  
 self.update\_textbox("Danh sách tệp CSV có sẵn:")  
 for i, csv\_file in enumerate(csv\_files):  
 self.update\_textbox(f"{i + 1}. {csv\_file}")  
  
 user\_input = self.get\_user\_input("Nhập số tương ứng với tệp bạn muốn chọn:")  
 if user\_input is not None:  
 print("Giá trị đã nhập:", user\_input)  
 else:  
 self.update\_textbox("Đã xảy ra lỗi khi nhập giọng nói.")  
 *# selected\_index = self.get\_user\_input("Nhập số tương ứng với tệp bạn muốn chọn:") - 1* if 0 <= user\_input - 1 < len(csv\_files):  
 self.selected\_csv\_file = csv\_files[user\_input - 1]  
 self.file\_path = os.path.join(self.directory\_path, self.selected\_csv\_file)  
 self.update\_textbox(f"Bạn đã chọn tệp: {self.selected\_csv\_file}")  
 self.update\_textbox(f"Đường dẫn tệp: {self.file\_path}")  
  
 *# Đọc dữ liệu từ tệp CSV đã chọn* self.df = pd.read\_csv(self.file\_path)  
 self.update\_textbox(f'Độ lớn của bảng [frame] dữ liệu: {self.df.shape}')  
  
 *# Hiển thị số lượng dòng từ DataFrame* while True:  
 try:  
 num\_rows\_to\_display = 10  
 self.update\_textbox(str(self.df.head(num\_rows\_to\_display)))  
 break  
 except ValueError:  
 self.update\_textbox("Lựa chọn không hợp lệ. Vui lòng nhập một số nguyên.")  
 else:  
 self.update\_textbox("Lựa chọn không hợp lệ. Vui lòng chọn số thứ tự hợp lệ.")  
  
 else:  
 self.update\_textbox("Không có tệp CSV nào trong thư mục.")  
 else:  
 self.update\_textbox(f"Thư mục '{self.directory\_path}' không tồn tại.")  
  
 def preprocessing(self):  
 *# Bước 3: Xử lý CỘT dữ liệu NULL quá nhiều OR không có giá trị phân tích* count\_values = self.df.count().sort\_values()  
 self.update\_textbox("Số lượng giá trị không NULL cho mỗi cột:")  
 self.update\_textbox(count\_values.to\_string()) *# Chuyển đổi thành chuỗi và hiển thị* self.update\_textbox("Danh sách các cột:")  
 for i, column in enumerate(self.df.columns):  
 self.update\_textbox(f"{i}. {column}")  
  
 *# Chọn cột cần xóa* columns\_to\_delete = [0, 1, 3]  
  
 *# Xóa cột đã chọn* self.df = self.df.drop(self.df.columns[columns\_to\_delete], axis=1)  
  
 *# Bước 4: Xử lý DÒNG dữ liệu NULL* self.df = self.df.dropna(how='any')  
 self.update\_textbox(f"Kích thước DataFrame sau khi xóa các dòng có giá trị null: {self.df.shape}")  
  
 *# Bước 5: Xử lý loại bỏ các giá trị ngoại lệ  
 # Tính toán Z-Score* z = np.abs(stats.zscore(self.df[['Postal Code', 'Electric Range', 'Base MSRP', 'DOL Vehicle ID', 'Census Tract']]))  
  
 threshold = 2  
  
 self.update\_textbox(f"Bạn đã chọn ngưỡng Z-Score là {threshold}")  
  
 *# Lọc dữ liệu dựa trên Z-Score* self.df = self.df[(z < threshold).all(axis=1)]  
 self.update\_textbox(f"Kích thước DataFrame sau khi lọc dựa trên Z-Score: {self.df.shape}")  
  
 *# Bước 7: Chuẩn hóa tập dữ liệu Input dùng MinMaxScaler* scaler = preprocessing.MinMaxScaler()  
 selected\_columns = ['Electric Range', 'Base MSRP', 'DOL Vehicle ID', 'Census Tract']  
 scaler.fit(self.df[selected\_columns])  
 self.df[selected\_columns] = scaler.transform(self.df[selected\_columns])  
  
 *# In ra một số dòng của DataFrame đã xử lý* self.update\_textbox(self.df.head().to\_string())  
  
 *# Bước 8: Xác định mô hình trích lọc các thuộc tính đặc trưng: EDA* numeric\_columns = self.df.select\_dtypes(include=['float64', 'int64']).columns  
 if 'Census Tract' in numeric\_columns:  
 numeric\_columns = numeric\_columns.drop('Census Tract')  
  
 X = self.df.loc[:, numeric\_columns]  
 y = self.df['Census Tract']  
 selector = SelectKBest(f\_regression, k=3)  
 selector.fit(X, y)  
 X\_new = selector.transform(X)  
 self.update\_textbox(f"{X\_new}\n")  
 self.update\_textbox(pd.DataFrame(y).to\_string(index=False) + '\n')  
 selected\_columns = X.columns[selector.get\_support(indices=True)]  
 self.update\_textbox("\n".join(selected\_columns))  
  
 *# Bước 9: Xác định mô hình trích lọc các thuộc tính đặc trưng  
 # XĐ data frame = Chiếu lấy các thuộc tính đặc trưng đã xđ trong B8* self.df = self.df[['Electric Range', 'Legislative District', 'DOL Vehicle ID', 'Census Tract']]  
  
 *# Bước 10: EDA theo nhu cầu thực tế => input vào các mô hình AI, ML,...  
 # Đơn giản nhất là lấy 1 thuộc tính đầu vào (Electric Range) để XD Mô hình* self.X = self.df[['Electric Range']]  
 self.y = self.df[['Census Tract']]  
 X\_str = X.to\_string(index=False)  
 y\_str = y.to\_string(index=False)  
  
 self.update\_textbox("Dữ liệu của Electric Range:\n" + X\_str)  
 self.update\_textbox("Dữ liệu của Census Tract:\n" + y\_str)  
  
  
 def show\_plot(self):  
 *# Tạo một đối tượng Figure của Matplotlib* fig, ax = plt.subplots()  
  
 *# Vẽ biểu đồ trên đối tượng Axis* ax.plot(self.X, self.y, 'o', alpha=0.3)  
 ax.set\_title('Mối quan hệ giữa Electric Range và Census Tract')  
 ax.set\_xlabel('Electric Range')  
 ax.set\_ylabel('Census Tract')  
  
 *# Tạo đối tượng FigureCanvasTkAgg để tích hợp vào Tkinter* canvas = FigureCanvasTkAgg(fig, master=self.top\_level\_window)  
 canvas.draw()  
  
 *# Tạo một cửa sổ Toplevel mới để hiển thị biểu đồ* toplevel = tk.Toplevel(self.top\_level\_window)  
 toplevel.title('Biểu Đồ')  
  
 *# Tạo một đối tượng FigureCanvasTkAgg mới cho mỗi cửa sổ Toplevel* canvas\_toplevel = FigureCanvasTkAgg(fig, master=toplevel)  
 canvas\_toplevel.draw()  
  
 *# Hiển thị FigureCanvasTkAgg trong Toplevel* canvas\_toplevel.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 *# Khởi tạo đối tượng DataPreprocessing* root = tk.Tk()  
 data\_processor = DataPreprocessing(root)  
 *# data\_processor.preprocessing()* root.mainloop()

1. **LẬP TRÌNH PYTHON NHẬN DIỆN KHUÔN MẶT TRONG VIDEO**

## Giới thiệu chủ đề Cv Nhận Diện Khuôn Mặt Trong Video

Trong lĩnh vực Computer Vision (CV), nhận diện khuôn mặt trong video đang trở thành một lĩnh vực nghiên cứu quan trọng. Việc này không chỉ có ứng dụng trong bảo mật mà còn mở ra nhiều tiềm năng trong các lĩnh vực như giải trí, y tế, và tự động hóa. Chủ đề này tập trung vào việc phát triển một ứng dụng sử dụng thư viện MTCNN để nhận diện và theo dõi khuôn mặt trong các đoạn video.

## Phân tích chủ đề Cv Nhận Diện Khuôn Mặt Trong Video

Chủ đề nhận diện khuôn mặt trong video đặt ra những thách thức đáng kể, bao gồm việc xử lý dữ liệu video liên tục và đồng thời giữ cho độ chính xác của mô hình cao. Phương pháp sử dụng MTCNN trong đoạn mã cho thấy một cách tiếp cận linh hoạt và hiệu quả.

## Các nền tảng kỹ thuật CV

## Các thư viện liên quan

**OpenCV**: Sử dụng cho xử lý ảnh và video.

**facenet\_pytorch**: Thư viện cung cấp mô hình MTCNN cho nhận diện khuôn mặt.

**speech\_recognition**: Sử dụng để nhận dạng lệnh giọng nói.

**tkinter**: Được sử dụng để tạo giao diện đồ họa người dùng.

## Giới thiệu Video thực nghiệm

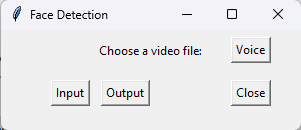
Việc giới thiệu video thực nghiệm sẽ cung cấp ví dụ cụ thể về cách ứng dụng hoạt động trên dữ liệu thực tế, giúp người đọc hiểu rõ hơn về khả năng và hiệu suất của hệ thống.

## Mô Tả Giải Thuật Cv Nhận Diện Khuôn Mặt Trong Video

Mô tả giải thuật tập trung vào việc sử dụng MTCNN để phát hiện và vẽ hộp giới hạn quanh khuôn mặt trong mỗi frame của video. Đồng thời, còn mô tả việc cắt và lưu các khung hình chứa khuôn mặt đã nhận diện.

## Giao diện CV Nhận Diện Khuôn Mặt Trong Video

Giao diện người dùng được xây dựng bằng thư viện tkinter, cung cấp các chức năng như chọn file video, hiển thị kết quả và thực hiện nhận diện khuôn mặt thông qua lệnh giọng nói.



## Mã lệnh lập trình CV nhận diện khuôn mặt từ Video

import os  
import speech\_recognition as sr  
import cv2  
from PIL import Image, ImageTk  
from facenet\_pytorch import MTCNN  
import tkinter as tk  
from tkinter import filedialog  
  
class FaceDetector:  
 def \_\_init\_\_(self, root):  
 self.mtcnn = MTCNN()  
 self.root = root  
 self.cap = None  
  
 self.frames\_to\_cut = 5 *# Số lượng frame cần cắt* self.detected\_frames = [] *# Danh sách các frame đã nhận diện* self.frame\_count = 0  
  
 self.top\_level\_window = tk.Toplevel(self.root)  
 self.top\_level\_window.title("Face Detection")  
 self.top\_level\_window.geometry("300x100")  
 self.output\_folder = "./"  
  
 self.recognizer = sr.Recognizer()  
  
 def voice\_command(self, command):  
 if "input" in command:  
 self.choose\_file()  
 elif "output" in command:  
 self.imageDetect()  
  
 def listen\_voice\_command(self):  
 with sr.Microphone() as source:  
 print("Say something...")  
 self.recognizer.adjust\_for\_ambient\_noise(source)  
 audio = self.recognizer.listen(source, timeout=3)  
  
 try:  
 command = self.recognizer.recognize\_google(audio).lower()  
 print(f"You said: {command}")  
 self.voice\_command(command)  
 except sr.UnknownValueError:  
 print("Could not understand audio.")  
 except sr.RequestError as e:  
 print(f"Could not request results from Google Speech Recognition service; {e}")  
  
 def choose\_file(self):  
 file\_path = filedialog.askopenfilename(filetypes=[("Video files", "\*.mp4")])  
 if file\_path:  
 self.cap = cv2.VideoCapture(file\_path)  
 self.detect\_faces()  
  
 def imageDetect(self):  
  
 for i, frame in enumerate(self.detected\_frames):  
 cv2.imshow(f'Detected Face {i + 1}', frame)  
 cv2.waitKey(0)  
 cv2.destroyAllWindows()  
  
 pil\_img = Image.fromarray(cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB))  
  
 if self.output\_folder:  
 save\_path = os.path.join(self.output\_folder, f"detected\_face\_{i + 1}.png")  
 pil\_img.save(save\_path)  
  
 def detect\_faces(self):  
 while True:  
 \_, img = self.cap.read()  
 if img is None:  
 break  
  
 rgb\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  
 boxes, \_ = self.mtcnn.detect(rgb\_img)  
  
 if boxes is not None:  
 for box in boxes:  
 cv2.rectangle(img, tuple(map(int, box[:2])), tuple(map(int, box[2:])), (255, 0, 0), 2)  
  
 if self.frame\_count != self.frames\_to\_cut:  
 self.detected\_frames.append(img.copy())  
 self.frame\_count += 1  
  
 cv2.imshow('Face Detection', img)  
  
 key = cv2.waitKey(30) & 0xFF  
 if key == 27:  
 break  
  
 self.cap.release()  
 cv2.destroyAllWindows()  
  
 def run\_gui(self):  
 label = tk.Label(self.top\_level\_window, text="Choose a video file:")  
 label.pack(pady=10)  
  
 tk.Button(self.top\_level\_window, text="Close", command=self.on\_close\_callback).place(x=230, y=50)  
  
 button = tk.Button(self.top\_level\_window, text="Input", command=self.choose\_file)  
 button.place(x=50, y=50)  
  
 button = tk.Button(self.top\_level\_window, text="Output", command=self.imageDetect)  
 button.place(x=100, y=50)  
  
 button = tk.Button(self.top\_level\_window, text="Voice", command=self.listen\_voice\_command)  
 button.place(x=230, y=7)  
  
 self.top\_level\_window.mainloop()  
  
 def on\_close\_callback(self):  
 *# Close the top-level window* self.top\_level\_window.destroy()

1. **LẬP TRÌNH PYTHON CAR RACING**

## Giới Thiệu Chủ Đề Game **Car Racing**

Trò chơi CAR RACING là một trò chơi đua xe, trong đó người chơi có nhiệm vụ điều khiển một chiếc xe để vượt qua các rào cản và đạt được điểm số cao. Trò chơi thường được thiết kế với đồ họa đẹp, âm thanh sống động và tăng cường trải nghiệm người chơi qua các cấp độ khác nhau.

## Phân Tích Chủ Đề Game **Car Racing**

***Gameplay:***

Người chơi điều khiển chiếc xe để tránh va chạm với các vật cản và duy trì sự sống.

Điểm số tăng dần theo thời gian hoặc theo khoảng cách đã đi được.

***Tương tác người dùng:***

Sử dụng bàn phím để điều khiển hành động của xe.

Tích hợp các phản hồi âm thanh hoặc hình ảnh khi người chơi đạt được mục tiêu hoặc gặp trở ngại.

## Các Nền Tảng Kỹ Thuật Lập Trình **Car Racing**

## Các Thư Viện Liên Quan

pygame: Thư viện Python được sử dụng để phát triển trò chơi đơn giản với đồ họa.

## Giới Thiệu Các Mẫu Media Thực Nghiệm (Mẫu Xe Và Đường Đua)

Mẫu xe và đường đua được tích hợp để tạo ra một môi trường đua xe hấp dẫn.

## Giao diện CAR RACING





## Mã lệnh lập trình Game Car Racing

import pygame  
import sys  
import random  
from PIL import ImageTk, Image  
from pygame.locals import \*  
import tkinter as tk  
  
class RacingGame:  
 def \_\_init\_\_(self, selected\_fps, selected\_bg, selected\_car, selected\_obstacles):  
 pygame.init()  
 self.WINDOWWIDTH = 400  
 self.WINDOWHEIGHT = 600  
  
 self.FPS = selected\_fps.get()  
 self.fpsClock = pygame.time.Clock()  
  
 self.BGSPEED = 1.5  
 self.BGIMG = pygame.image.load(selected\_bg.get())  
  
 self.CARIMG = pygame.image.load(selected\_car.get())  
 self.OBSTACLESIMG = pygame.image.load(selected\_obstacles.get())  
  
 self.DISPLAYSURF = pygame.display.set\_mode((self.WINDOWWIDTH, self.WINDOWHEIGHT))  
  
  
 pygame.display.set\_caption('12 Dinh Dai Hai Dang Game ĐUA XE')  
  
 self.show\_settings\_interface()  
 self.bg = Background(self.BGSPEED, self.BGIMG, self.DISPLAYSURF)  
 self.car = Car(self.WINDOWWIDTH, self.WINDOWHEIGHT, self.CARIMG, self.DISPLAYSURF)  
 self.obstacles = Obstacles(self.WINDOWHEIGHT, self.OBSTACLESIMG,self.DISPLAYSURF)  
 self.score = Score(self.DISPLAYSURF, self.WINDOWWIDTH, self.WINDOWHEIGHT, self.fpsClock, self.FPS, self.BGSPEED, self.BGIMG, self.CARIMG, self.OBSTACLESIMG)  
  
 def show\_settings\_interface(self):  
 *# Hiển thị giao diện tùy chỉnh thông số  
 # Các ô nhập liệu, nút và các phần tử giao diện người dùng khác có thể được thêm vào đây* pass  
  
 def run\_game(self):  
 self.score.game\_start(self.bg)  
 while True:  
 self.score.game\_play(self.bg, self.car, self.obstacles, self.score)  
 self.score.game\_over(self.bg, self.car, self.obstacles, self.score)  
  
  
class Background:  
 def \_\_init\_\_(self, BGSPEED, BGIMG, DISPLAYSURF):  
 self.DISPLAYSURF = DISPLAYSURF  
 self.BGIMG = BGIMG  
 self.BGSPEED = BGSPEED  
 self.x = 0  
 self.y = 0  
 self.speed = self.BGSPEED  
 self.img = self.BGIMG  
 self.width = self.img.get\_width()  
 self.height = self.img.get\_height()  
  
 def draw(self):  
 self.DISPLAYSURF.blit(self.img, (int(self.x), int(self.y)))  
 self.DISPLAYSURF.blit(self.img, (int(self.x), int(self.y - self.height)))  
  
 def update(self):  
 self.y += self.speed  
  
 if self.y > self.height:  
 self.y -= self.height  
  
class Car:  
 def \_\_init\_\_(self, WINDOWWIDTH, WINDOWHEIGHT, CARIMG, DISPLAYSURF):  
 self.DISPLAYSURF = DISPLAYSURF  
 self.CARIMG = CARIMG  
 self.WINDOWHEIGHT = WINDOWHEIGHT  
 self.WINDOWWIDTH = WINDOWWIDTH  
 self.width = 40  
 self.height = 60  
 self.x = (self.WINDOWWIDTH - self.width) / 2  
 self.y = (self.WINDOWHEIGHT - self.height) / 2  
 self.speed = 3  
  
 def draw(self):  
 self.DISPLAYSURF.blit(self.CARIMG, (int(self.x), int(self.y)))  
  
 def update(self, moveLeft, moveRight, moveUp, moveDown):  
 if moveLeft:  
 self.x -= self.speed  
 if moveRight:  
 self.x += self.speed  
 if moveUp:  
 self.y -= self.speed  
 if moveDown:  
 self.y += self.speed  
 if self.x < 80:  
 self.x = 80  
 if self.x + self.width > self.WINDOWWIDTH - 80:  
 self.x = self.WINDOWWIDTH - 80 - self.width  
 if self.y < 0:  
 self.y = 0  
 if self.y + self.height > self.WINDOWHEIGHT:  
 self.y = self.WINDOWHEIGHT - self.height  
  
class Obstacles:  
 def \_\_init\_\_(self, WINDOWHEIGHT, OBSTACLESIMG, DISPLAYSURF):  
 self.OBSTACLESIMG = OBSTACLESIMG  
 self.DISPLAYSURF = DISPLAYSURF  
 self.WINDOWHEIGHT = WINDOWHEIGHT  
 self.width = 40  
 self.height = 60  
 self.distance = 200  
 self.speed = 2  
 self.changeSpeed = 0.001  
 self.ls = []  
 for i in range(5):  
 y = -self.height - i \* self.distance  
 lane = random.randint(0, 3)  
 self.ls.append([lane, y])  
  
 def draw(self):  
 for i in range(5):  
 x = int(80 + self.ls[i][0] \* 60 + (60 - self.width) / 2)  
 y = int(self.ls[i][1])  
 self.DISPLAYSURF.blit(self.OBSTACLESIMG, (x, y))  
  
 def update(self):  
 for i in range(5):  
 self.ls[i][1] += self.speed  
 self.speed += self.changeSpeed  
 if self.ls[0][1] > self.WINDOWHEIGHT:  
 self.ls.pop(0)  
 y = self.ls[3][1] - self.distance  
 lane = random.randint(0, 3)  
 self.ls.append([lane, y])  
  
class Score:  
 def \_\_init\_\_(self, DISPLAYSURF, WINDOWWIDTH, WINDOWHEIGHT, fpsClock, FPS, BGSPEED, BGIMG, CARIMG, OBSTACLESIMG):  
 self.OBSTACLESIMG = OBSTACLESIMG  
 self.CARIMG = CARIMG  
 self.WINDOWHEIGHT = WINDOWHEIGHT  
 self.BGIMG = BGIMG  
 self.BGSPEED = BGSPEED  
 self.FPS = FPS  
 self.fpsClock = fpsClock  
 self.WINDOWWIDTH = WINDOWWIDTH  
 self.DISPLAYSURF = DISPLAYSURF  
 self.score = 0  
  
 def draw(self):  
 font = pygame.font.SysFont('consolas', 30)  
 score\_surface = font.render('Score: ' + str(int(self.score)), True, (0, 0, 0))  
 self.DISPLAYSURF.blit(score\_surface, (10, 10))  
  
 def update(self):  
 self.score += 0.02  
  
 def rect\_collision(self, rect1, rect2):  
 if rect1[0] <= rect2[0] + rect2[2] and rect2[0] <= rect1[0] + rect1[2] and rect1[1] <= rect2[1] + rect2[3] and \  
 rect2[1] <= rect1[1] + rect1[3]:  
 return True  
 return False  
  
 def is\_gameover(self, car, obstacles):  
 car\_rect = [car.x, car.y, car.width, car.height]  
 for i in range(5):  
 x = int(80 + obstacles.ls[i][0] \* 60 + (60 - obstacles.width) / 2)  
 y = int(obstacles.ls[i][1])  
 obstacles\_rect = [x, y, obstacles.width, obstacles.height]  
 if self.rect\_collision(car\_rect, obstacles\_rect):  
 return True  
 return False  
  
 def game\_over(self, bg, car, obstacles, score):  
 font = pygame.font.SysFont('consolas', 60)  
 heading\_surface = font.render('GAMEOVER', True, (255, 0, 0))  
 heading\_size = heading\_surface.get\_size()  
 font = pygame.font.SysFont('consolas', 20)  
 comment\_surface = font.render('Press "space" to replay', True, (0, 0, 0))  
 comment\_size = comment\_surface.get\_size()  
 while True:  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
 if event.type == pygame.KEYUP:  
 if event.key == K\_SPACE:  
 return  
 elif event.key == K\_q:  
 pygame.quit()  
 quit()  
 bg.draw()  
 car.draw()  
 obstacles.draw()  
 score.draw()  
 self.DISPLAYSURF.blit(heading\_surface, (int((self.WINDOWWIDTH - heading\_size[0]) / 2), 100))  
 self.DISPLAYSURF.blit(comment\_surface, (int((self.WINDOWWIDTH - comment\_size[0]) / 2), 400))  
 pygame.display.update()  
 self.fpsClock.tick(self.FPS)  
  
 def game\_start(self, bg, ):  
 bg.\_\_init\_\_(self.BGSPEED, self.BGIMG, self.DISPLAYSURF)  
 font = pygame.font.SysFont('consolas', 60)  
 heading\_surface = font.render('RACING', True, (255, 0, 0))  
 heading\_size = heading\_surface.get\_size()  
 font = pygame.font.SysFont('consolas', 20)  
 comment\_surface = font.render('Press "space" to play', True, (0, 0, 0))  
 comment\_size = comment\_surface.get\_size()  
 while True:  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
 if event.type == pygame.KEYUP:  
 if event.key == K\_SPACE:  
 return  
 bg.draw()  
 self.DISPLAYSURF.blit(heading\_surface, (int((self.WINDOWWIDTH - heading\_size[0]) / 2), 100))  
 self.DISPLAYSURF.blit(comment\_surface, (int((self.WINDOWWIDTH - comment\_size[0]) / 2), 400))  
 pygame.display.update()  
 self.fpsClock.tick(self.FPS)  
  
 def game\_play(self, bg, car, obstacles, score):  
 car.\_\_init\_\_(self.WINDOWWIDTH, self.WINDOWHEIGHT, self.CARIMG, self.DISPLAYSURF)  
 obstacles.\_\_init\_\_(self.WINDOWHEIGHT, self.OBSTACLESIMG, self.DISPLAYSURF)  
 bg.\_\_init\_\_(self.BGSPEED, self.BGIMG, self.DISPLAYSURF)  
 score.\_\_init\_\_(self.DISPLAYSURF, self.WINDOWWIDTH, self.WINDOWHEIGHT, self.fpsClock, self.FPS, self.BGSPEED, self.BGIMG, self.CARIMG, self.OBSTACLESIMG)  
 move\_left = False  
 move\_right = False  
 move\_up = False  
 move\_down = False  
 while True:  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 quit()  
 if event.type == KEYDOWN:  
 if event.key == K\_LEFT:  
 move\_left = True  
 if event.key == K\_RIGHT:  
 move\_right = True  
 if event.key == K\_UP:  
 move\_up = True  
 if event.key == K\_DOWN:  
 move\_down = True  
 if event.key == K\_q:  
 pygame.quit()  
 quit()  
 if event.type == KEYUP:  
 if event.key == K\_LEFT:  
 move\_left = False  
 if event.key == K\_RIGHT:  
 move\_right = False  
 if event.key == K\_UP:  
 move\_up = False  
 if event.key == K\_DOWN:  
 move\_down = False  
 if self.is\_gameover(car, obstacles):  
 return  
 bg.draw()  
 bg.update()  
 car.draw()  
 car.update(move\_left, move\_right, move\_up, move\_down)  
 obstacles.draw()  
 obstacles.update()  
 self.draw()  
 self.update()  
 pygame.display.update()  
 self.fpsClock.tick(self.FPS)  
  
  
class GameOptionsForm():  
 def \_\_init\_\_(self, master):  
 self.master = master  
 master.title("Game Options")  
 self.master.geometry("400x600")  
  
 self.selected\_fps = tk.IntVar(value=60)  
  
 *# Tạo các Radiobutton với giá trị tương ứng là số làn xe* tk.Radiobutton(master, text="60 fps", variable=self.selected\_fps, value=60).grid(row=1, column=1)  
 tk.Radiobutton(master, text="120 fps", variable=self.selected\_fps, value=120).grid(row=1, column=2)  
  
 self.fps\_label = tk.Label(master, text="Frames per second:")  
 self.fps\_label.grid(row=1, column=0, padx=10, pady=5)  
  
 self.selected\_bg = tk.StringVar(value='img/background.png')  
  
 *# Tạo các Radiobutton với giá trị tương ứng là số làn xe* tk.Radiobutton(master, text="background 1", variable=self.selected\_bg, value='img/background.png').place(x=100,y=50)  
 tk.Radiobutton(master, text="background 2", variable=self.selected\_bg, value='img/background1.png').place(x=210,y=50)  
  
 self.background\_image\_label = tk.Label(master, text="Background")  
 self.background\_image\_label.place(x=20,y=50)  
  
 self.selected\_car = tk.StringVar(value='img/car.png')  
  
 tk.Radiobutton(master, text="car", variable=self.selected\_car, value='img/car.png').place(x=100,y=190)  
 tk.Radiobutton(master, text="car1", variable=self.selected\_car, value='img/car1.png').place(x=200,y=190)  
 tk.Radiobutton(master, text="car2", variable=self.selected\_car, value='img/car2.png').place(x=300, y=190)  
  
 self.car\_type\_label = tk.Label(master, text="Car:")  
 self.car\_type\_label.place(x=20,y=190)  
  
 self.selected\_obstacles= tk.StringVar(value='img/obstacles.png')  
  
 tk.Radiobutton(master, text="obstacles", variable=self.selected\_obstacles, value='img/obstacles.png').place(x=100, y=310)  
 tk.Radiobutton(master, text="obstacles1", variable=self.selected\_obstacles, value='img/obstacles1.png').place(x=200, y=310)  
 tk.Radiobutton(master, text="obstacles2", variable=self.selected\_obstacles, value='img/obstacles2.png').place(x=300, y=310)  
  
 self.car\_type\_label = tk.Label(master, text="Obstacles:")  
 self.car\_type\_label.place(x=20, y=310)  
  
 self.ok\_button = tk.Button(master, text="Start Game", command=self.start\_game)  
 self.ok\_button.place(x=100, y=500)  
 self.cancel\_button = tk.Button(master, text="Cancel", command=self.cancel\_button\_handler)  
 self.cancel\_button.place(x=250, y=500)  
  
 img = Image.open('img/background.png')  
 img = img.resize((100, 100), resample=Image.LANCZOS)  
  
 self.image = ImageTk.PhotoImage(img)  
  
 *# Hiển thị hình ảnh trên form* self.label = tk.Label(master, image=self.image)  
 self.label.place(x=100,y=70)  
  
 img2 = Image.open('img/background1.png')  
 img2 = img2.resize((100, 100), Image.LANCZOS)  
  
 *# Chuyển ảnh sang định dạng PhotoImage* self.image2 = ImageTk.PhotoImage(img2)  
  
 *# Hiển thị hình ảnh trên form* self.label2 = tk.Label(master, image=self.image2)  
 self.label2.place(x=210,y=70)  
  
 img\_car = Image.open('img/car.png')  
  
 self.image\_car = ImageTk.PhotoImage(img\_car)  
  
 self.lb1 = tk.Label(master, image=self.image\_car)  
 self.lb1.place(x=100, y=220)  
  
 img\_car2 = Image.open('img/car1.png')  
  
 self.image\_car2 = ImageTk.PhotoImage(img\_car2)  
  
 self.lb2 = tk.Label(master, image=self.image\_car2)  
 self.lb2.place(x=200, y=220)  
  
 img\_car3 = Image.open('img/car2.png')  
  
 self.image\_car3 = ImageTk.PhotoImage(img\_car3)  
  
 self.lb3 = tk.Label(master, image=self.image\_car3)  
 self.lb3.place(x=300, y=220)  
  
 img\_obstacles = Image.open('img/obstacles.png')  
  
 self.image\_obstacles = ImageTk.PhotoImage(img\_obstacles)  
  
 self.lbo1 = tk.Label(master, image=self.image\_obstacles)  
 self.lbo1.place(x=120, y=350)  
  
 img\_obstacles1 = Image.open('img/obstacles1.png')  
  
 self.image\_obstacles1 = ImageTk.PhotoImage(img\_obstacles1)  
  
 self.lbo2 = tk.Label(master, image=self.image\_obstacles1)  
 self.lbo2.place(x=220, y=350)  
  
 img\_obstacles2 = Image.open('img/obstacles2.png')  
  
 self.image\_obstacles2 = ImageTk.PhotoImage(img\_obstacles2)  
  
 self.lbo3 = tk.Label(master, image=self.image\_obstacles2)  
 self.lbo3.place(x=320, y=350)  
  
 def cancel\_button\_handler(self):  
 self.master.destroy()  
  
 def start\_game(self):  
 self.master.destroy()  
 fps = self.selected\_fps  
 background\_image = self.selected\_bg  
 car\_type = self.selected\_car  
 obstacles\_type = self.selected\_obstacles  
  
 run\_game = RacingGame(fps, background\_image, car\_type, obstacles\_type)  
 run\_game.run\_game()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 root = tk.Tk()  
 game = GameOptionsForm(root)  
 root.mainloop()

1. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỒ ÁN HỌC PHẦN

## Kết luận

## Những kết quả đạt được

ĐỒ ÁN HỌC PHẦN đã thực hiện việc Lập trình Python; bao gồm:

+ Giao diện đồ họa: GUI

+ Xử lý giọng nói trợ lý ảo: Voice Assistant

+ Phân tích khám phá trên tập dữ liệu thực nghiệm: Phân Tích Electric Vehicle Population Trong Quý 2 Năm 2023, có Plot

+ CV Nhận Diện Khuôn Mặt Trong Video

+ Game Car Racing

## Hạn chế

Vì thời gian có hạn nên chưa

+ Chưa liên kết Voice Assistant để thực hiện phân tích khám phá dữ liệu bằng lời nói.

+ Có thể còn một số chức năng chưa được triển khai do hạn chế thời gian và phức tạp của dự án.

## Hướng khắc phục các hạn chế

Đồ án sẽ tiếp tục thực hiện

+ Kiểm tra và hoàn thiện các chức năng chưa triển khai hoặc chưa đạt đến mức độ mong muốn.Hướng mở rộng ĐỒ ÁN HỌC PHẦN

***Tối ưu hóa hiệu suất:*** Tối ưu hóa mã nguồn và hiệu suất của các chức năng để đảm bảo ứng dụng chạy mượt mà và hiệu quả.

***Mở rộng tính năng và giao diện:*** Thêm vào các tính năng mới và cải thiện giao diện người dùng để tăng cường trải nghiệm người dùng.

Đồ án học phần không chỉ là một bước khởi đầu mà còn là cơ hội để phát triển và mở rộng thành một dự án lớn hơn trong tương lai.

# DANH MỤC TÀI LIỆU THAM KHẢO

1. Võ Xuân Thể (2023), Tài liệu học tập Lập trình Python, Trường Đại học Sư phạm kỹ thuật Tp.HCM, Tài liệu lưu hành nội bộ được cung cấp theo từng buổi học
2. Võ Xuân Thể (2023), Bài tập thực hành Lập trình Python, Trường Đại học Sư phạm kỹ thuật Tp.HCM, Tài liệu lưu hành nội bộ được cung cấp theo từng buổi học
3. https://www.kaggle.com/datasets/willianoliveiragibin/electric-vehicle-population: nguồn dataFrame được sử dụng trong đề tài này

1. IDE (Integrated Development Environment) là môi trường tích hợp dùng để viết code để phát triển ứng dụng [↑](#footnote-ref-1)