**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 \_Topic.py**

Thư mục bài làm phần EDA: **G5C12DinhDaiHaiDang \_EDA.Topic**

Tập dữ liệu thực nghiệm EDA: 12DinhDaiHaiDang\_ElectricVehiclePopulationData (nên sử dụng Keysearch)

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

File Video thực nghiệm CV: 1.mp4 (nên sử dụng Keysearch)

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

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**

H

# 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](#_Toc151453978)

[DANH MỤC CÁC TỪ VIẾT TẮT 4](#_Toc151453979)

[**DANH MỤC CÁC THUẬT NGỮ** 4](#_Toc151453980)

[Chương 1: GIỚI THIỆU VỀ ĐỒ ÁN HỌC PHẦN 8](#_Toc151453981)

[1.1. Tổng quan về Đồ án học phần 8](#_Toc151453982)

[1.2. Nội dung chuyên môn chính của Đồ án học phần 8](#_Toc151453983)

[1.3. Công cụ và nền tảng kỹ thuật thực hiện Đồ án học phần 9](#_Toc151453984)

[1.4. Bố cục của báo cáo 9](#_Toc151453985)

[Chương 2: CƠ SỞ LÝ THUYẾT CỦA HỌC PHẦN 11](#_Toc151453986)

[2.1. GIỚI THIỆU VỀ LẬP TRÌNH PYTHON 11](#_Toc151453987)

[2.1.1. Giới thiệu chung 11](#_Toc151453988)

[2.1.2. Cài đặt “bộ thảo chương” (soạn thảo chương trình IDE) và “bộ dịch” 11](#_Toc151453989)

[2.1.3. Giới thiệu các bộ tương tác lập trình 11](#_Toc151453990)

[2.1.4. Cấu hình mô trường tương tác lập trình 11](#_Toc151453991)

[2.2. TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH PYTHON 11](#_Toc151453992)

[2.2.1. Danh hiệu (Identifier) 11](#_Toc151453993)

[2.2.2. Về viết mã lệnh lập trình (Code) 11](#_Toc151453994)

[2.2.3. Về chú giải (Comment) 11](#_Toc151453995)

[2.2.4. Các phép toán 11](#_Toc151453996)

[2.2.5. Chuỗi 11](#_Toc151453997)

[2.2.6. Lệnh 11](#_Toc151453998)

[2.2.7. Định nghĩa hàm 11](#_Toc151453999)

[2.2.8. Danh sách 11](#_Toc151454000)

[2.2.9. Bộ dữ liệu: Tuple 11](#_Toc151454001)

[2.2.10. Tập hợp: Set 11](#_Toc151454002)

[2.2.11. Từ điển: Dictionary 11](#_Toc151454003)

[2.2.12. Module 11](#_Toc151454004)

[2.2.13. Package 11](#_Toc151454005)

[2.2.14. Lớp 11](#_Toc151454006)

[2.2.15. Xử lý ngoại lệ 11](#_Toc151454007)

[2.3. GIỚI THIỆU CÁC THƯ VIỆN VÀ CÔNG CỤ CĂN BẢN 11](#_Toc151454008)

[2.3.1. Nạp thư viện 11](#_Toc151454009)

[2.3.2. speech\_recognition 11](#_Toc151454010)

[2.3.3. gtts 11](#_Toc151454011)

[2.3.4. playsound 11](#_Toc151454012)

[2.3.5. tkinter 11](#_Toc151454013)

[2.3.6. os 11](#_Toc151454014)

[2.3.7. Pandas 11](#_Toc151454015)

[2.3.8. Tensorflow 11](#_Toc151454016)

[2.3.9. NumPy 12](#_Toc151454017)

[2.3.10. SCIPY 12](#_Toc151454018)

[2.3.11. Theano 12](#_Toc151454019)

[2.3.12. Matplotlib 12](#_Toc151454020)

[2.3.13. scikit-learn = sklearn 12](#_Toc151454021)

[2.3.14. Keras 12](#_Toc151454022)

[2.3.15. NLTK 12](#_Toc151454023)

[2.3.16. PyTorch 12](#_Toc151454024)

[2.3.17. LightGBM 12](#_Toc151454025)

[2.3.18. Eli5 12](#_Toc151454026)

[2.4. KỸ THUẬT LẬP TRÌNH ĐỆ QUY: THÁP HÀ NỘI (HANOI TOWER) 12](#_Toc151454027)

[2.5. LẬP TRÌNH PYTHON XỬ LÝ GIỌNG NÓI (Trợ lý ảo: Voice Assistant) 12](#_Toc151454028)

[2.5.1. Giới thiệu chủ đề 12](#_Toc151454029)

[2.5.2. Tài liệu và nguồn thực nghiệm 12](#_Toc151454030)

[2.5.3. Giới thiệu Các thư viện Python liên quan 12](#_Toc151454031)

[2.5.4. Giới thiệu bài toán 12](#_Toc151454032)

[2.5.5. Giới thiệu code tham khảo 12](#_Toc151454033)

[2.6. LẬP TRÌNH GIAO DIỆN ĐỒ HỌA (GUI) 12](#_Toc151454034)

[2.7. LẬP TRÌNH PHÂN TÍCH KHÁM PHÁ (EDA) [Thăm dò] 12](#_Toc151454035)

[2.8. LẬP TRÌNH ĐỒ THỊ & BIỂU ĐỒ TRỰC QUANG HÓA DỮ LIỆU (VISUALIZAION: PLOT) 12](#_Toc151454036)

[2.9. LẬP TRÌNH THỊ GIÁC MÁY TÍNH (CV) 12](#_Toc151454037)

[2.10. LẬP TRÌNH GAMEs CĂN BẢN 12](#_Toc151454038)

[Chương 3: PHÂN TÍCH VÀ XÁC ĐINH CÁC CƠ SỞ KỸ THUẬT 13](#_Toc151454039)

[3.1. GIỚI THIỆU CHUNG VỀ CÁC CHỦ ĐỀ CỦA ĐỒ ÁN 13](#_Toc151454040)

[3.2. LẬP TRÌNH XỬ GIỌNG NÓI (TRỢ LÝ ẢO: VOICE ASSISTANT) 13](#_Toc151454041)

[3.2.1. Giới thiệu chung về chủ đề 13](#_Toc151454042)

[3.2.1.1. Mô tả yêu cầu của chủ đề 13](#_Toc151454043)

[3.2.1.2. Phân tích yều cầu của chủ đề 13](#_Toc151454044)

[3.2.2. Các nền tảng kỹ thuật 13](#_Toc151454045)

[3.2.2.1. Nền tảng kỹ thuật cơ bản 13](#_Toc151454046)

[3.2.2.2. Hệ thống thư việc sử dụng 13](#_Toc151454047)

[3.2.3. Thiết kế giao diện 14](#_Toc151454048)

[3.2.4. Mã lệnh lập trình (python) 14](#_Toc151454049)

[3.3. LẬP TRÌNH GIAO DIỆN ĐỒ HỌA (GUI) 15](#_Toc151454050)

[3.4. LẬP TRÌNH TRỰC QUANG HÓA DỮ LIỆU (PLOT) 15](#_Toc151454051)

[Chương 4: LẬP TRÌNH PYTHON ELECTRIC VEHICLE POPULATION TRONG QUÝ 2 NĂM 2023 15](#_Toc151454052)

[4.1. Giới thiệu chủ đề EDA ……. 15](#_Toc151454053)

[3.4.1. Mô tả yêu câu của chủ đề 15](#_Toc151454054)

[3.4.2. Phân tích yều cầu của chủ đề 16](#_Toc151454055)

[3.5. CÁC NỀN TẢNG KỸ THUẬT 16](#_Toc151454056)

[3.5.1. Hệ thống thư viện sử dụng 16](#_Toc151454057)

[3.5.2. Tập dữ liệu thực nghiệm 16](#_Toc151454058)

[3.5.3. Mô tả giải thuật 17](#_Toc151454059)

[4.2. Thiết kế giao diện 19](#_Toc151454060)

[4.3. Mã lệnh lập trình (python) 19](#_Toc151454061)

[Chương 5: LẬP TRÌNH PYTHON NHẬN DIỆN KHUÔN MẶT TRONG VIDEO. 23](#_Toc151454062)

[Chương 6: LẬP TRÌNH PYTHON CAR RACING 23](#_Toc151454063)

[Chương 7: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỒ ÁN HỌC PHẦN 23](#_Toc151454064)

[7.1. Kết luận 23](#_Toc151454065)

[7.1.1. Những kết quả đạt được 23](#_Toc151454066)

[7.1.2. Hạn chế 23](#_Toc151454067)

[7.2. Hướng phát triển 23](#_Toc151454068)

[7.2.1. Hướng khắc phục các hạn chế 23](#_Toc151454069)

[7.2.2. Hướng mở rộng ĐỒ ÁN HỌC PHẦN 23](#_Toc151454070)

[DANH MỤC TÀI LIỆU THAM KHẢO 24](#_Toc151454071)

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:

# B1: NẠP THƯ VIỆN

# Speech

import speech\_recognition as sr

from gtts import gTTS

import playsound

# tkinter

import tkinter as tk

from tkinter import messagebox

# thư viện OS (lập thư mục, files)

import os # THƯ VIỆN OS MS. WINDOWS = Lập thư mục & lưu file

# Nạp các thư viện cần thiết EDA

from tkinter import filedialog, simpledialog

import numpy as np

import pandas as pd

from scipy import stats

from sklearn import preprocessing

from sklearn.feature\_selection import SelectKBest, f\_regression

import os

import tkinter as tk  
 # Nạp hàm Thư viện phân tích dữ liệu thăm dò

## 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 GAMEs 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: Speech….: chính là part1 của chủ đề này

Part2: Các kỹ thuật AI

Part2: Voice….: 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

conda install …. Các thư viện sau:

------------------gốc hướng dẫn / dùng python trong visual (pip3)

pip3 install SpeechRecognition

*pip3 install SpeechRecognition pydub # Long audio source*

pip3 install pyaudio

B2 : Nạp thư viện vào App

import speech\_recognition as sr

**import** speech\_recognition **as** sr

**from** gtts **import** gTTS

**import** os

**import** time

**import** playsound

## Thiết kế giao diện

+ Xử lý trong cửa sổ Console Output của PyCharm

## Mã lệnh lập trình (python)

**FULL CODE**

# -\*- coding: utf-8 -\*-

"""

Created on Sat Jul 16 19:59:48 2023

@author: VOXUAN

"""

# B1: NẠP THƯ VIỆN

import speech\_recognition as sr

from gtts import gTTS

import playsound

# B2: CHỌN PHƯƠNG ÁN NHẬP ÂM THANH TỪ MICROPHONE

r = sr.Recognizer()

# B3: XỬ LÝ NHẬN DIỆN MIC

#PHẦN 1: IN RA TEXT THEO LỜI NÓI = Nghe tiếng Việt => Text

with sr.Microphone() as Source:

#hiệu chỉnh mic để chuẩn bị nói

print("Hieu chinh nhieu trươc khi noi!")

r.adjust\_for\_ambient\_noise(Source, duration=1)

#nhận lời nói của người dùng từ MIc mặc định lưu dữ liệu âm thanh vào audio\_data

print("Nói tiếng Việt đi, sau 5s sẽ in ra văn bản!")

audio\_data = r.record(Source, duration = 5)

#In ra văn bản text

print("KẾT QUẢ NHẬN DIỆN ..................")

#chuyển lời nói thành văn bản

try:

text = r.recognize\_google(audio\_data,language="vi")

except:

text = "Quý vị nói gì nghe không rõ...!"

#in kết quả ra

print("Quý vị đã nói là : ",format(text))

#PHẦN 2: XUẤT RA LỜI NÓI THEO VĂN BẢN ĐÃ NHẬP = Trả lời bằng tiếng Việt : Text => Nói tiếng Việt

# ĐỌC CHỮ VIỆT => NÓI RA TIẾNG VIỆT

def VXT(t):

vx = gTTS(text=t, lang = 'vi')

xuan = '06VXT.mp3'

vx.save(xuan)

playsound.playsound(xuan)

VXT(text)

#VXT("số thứ tự 06, họ tên: VÕ XUÂN THỂ là giảng viên viên của TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH")

## LẬP TRÌNH GIAO DIỆN ĐỒ HỌA (GUI)

<kết cấu nội dung bài GUI vào đây>

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

<kết cấu nội dung bài PLOT vào đây>

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 scipy import stats

from sklearn import preprocessing

from sklearn.feature\_selection import SelectKBest, f\_regression

import os

import tkinter as tk

### 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}")

try:

selected\_index = self.get\_user\_input("Nhập số tương ứng với tệp bạn muốn chọn:") - 1

if 0 <= selected\_index < len(csv\_files):

self.selected\_csv\_file = csv\_files[selected\_index]

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 = self.get\_user\_input(

"Nhập số lượng dòng bạn muốn in ra từ DataFrame:")

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.")

except TypeError:

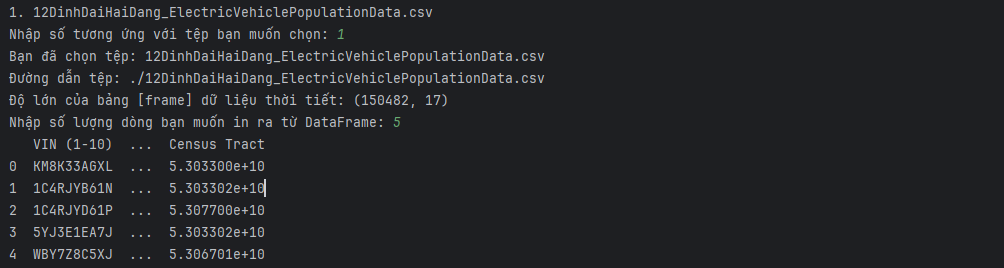
self.update\_textbox("Lựa chọn không hợp lệ. Vui lòng nhập một số nguyên.")

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:



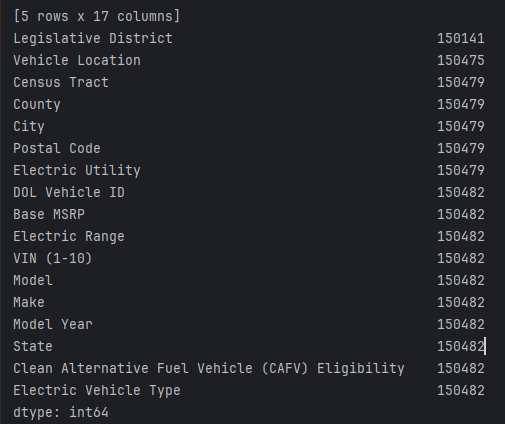
### Mô tả giải thuật

**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 scipy import stats

from sklearn import preprocessing

from sklearn.feature\_selection import SelectKBest, f\_regression

import os

import tkinter as tk

class DataPreprocessing:

def \_\_init\_\_(self, master, directory\_path="./"):

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).grid(row=1, column=0, pady=10)

tk.Button(self.top\_level\_window, text="Processing", command=self.preprocessing).grid(row=1, column=1, pady=10)

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):

return simpledialog.askinteger("Input", prompt, parent=self.top\_level\_window)

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}")

try:

selected\_index = self.get\_user\_input("Nhập số tương ứng với tệp bạn muốn chọn:") - 1

if 0 <= selected\_index < len(csv\_files):

self.selected\_csv\_file = csv\_files[selected\_index]

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 = self.get\_user\_input(

"Nhập số lượng dòng bạn muốn in ra từ DataFrame:")

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.")

except TypeError:

self.update\_textbox("Lựa chọn không hợp lệ. Vui lòng nhập một số nguyên.")

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']]))

while True:

try:

threshold = self.get\_user\_input("Nhập ngưỡng giá trị Z-Score: ")

break

except ValueError:

self.update\_textbox("Vui lòng nhập một số thực hợp lệ.")

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

X = self.df[['Electric Range']]

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)

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

## Giới thiệu chủ đề CV …….

## Phân tích chủ đề CV …….

## Các nền tảng kỹ thuật CV

## Các thư viện liên quan

## Giới thiệu Clip …….thực nghiệm

## Mô tả giải thuật CV nhận diện ………. từ Clip……

## Giao diện CV nhận diện ………. từ Clip……

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

import cv2 #pip install opencv-python

from facenet\_pytorch import MTCNN #pip install facenet-pytorch

class FaceDetector:

def \_\_init\_\_(self, video\_path):

# Load the MTCNN detector

self.mtcnn = MTCNN()

# Open the video file

self.cap = cv2.VideoCapture(video\_path)

def detect\_faces(self):

while True:

# Read the frame

\_, img = self.cap.read()

# Break the loop if the video has ended

if img is None:

break

# Convert to RGB (MTCNN uses RGB format)

rgb\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

# Detect faces using MTCNN

boxes, \_ = self.mtcnn.detect(rgb\_img)

# Draw the rectangle around each detected face

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)

# Display the result

cv2.imshow('Face Detection', img)

# Break the loop if the 'Esc' key is pressed

key = cv2.waitKey(30) & 0xFF

if key == 27:

break

# Release the VideoCapture object

self.cap.release()

# Destroy all OpenCV windows

cv2.destroyAllWindows()

1. **LẬP TRÌNH PYTHON CAR RACING**

## Giới thiệu chủ đề Game …….

## Phân tích chủ đề Game …….

## Các nền tảng kỹ thuật Lập trình Game

## Các thư viện liên quan

## Giới thiệu Các mẫu media…….thực nghiệm (mẫu xe và đường đua)

## Mô tả giải thuật Game….

## Giao diện Game ….

## Mã lệnh lập trình Game Car Racing

import pygame

import sys

import random

from pygame.locals import \*

class RacingGame:

def \_\_init\_\_(self):

pygame.init()

self.WINDOWWIDTH = 400

self.WINDOWHEIGHT = 600

self.FPS = 120

self.fpsClock = pygame.time.Clock()

self.BGSPEED = 1.5

self.BGIMG = pygame.image.load('img/background.png')

self.CARIMG = pygame.image.load('img/car.png')

self.OBSTACLESIMG = pygame.image.load('img/obstacles.png')

self.DISPLAYSURF = pygame.display.set\_mode((self.WINDOWWIDTH, self.WINDOWHEIGHT))

pygame.display.set\_caption('12 Dinh Dai Hai Dang Game ĐUA XE')

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 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

elif 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)

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: ……………………., có Plot

+ CV ……………

+ Game………..

## Hạn chế

Vì thời gian có hạn nên chưa

+ liên kết voice assistant để ra lệnh phân tích khám phá bằng lời nói.

+ ….

## Hướng phát triển

## Hướng khắc phục các hạn chế

Đồ án sẽ tiếp tục thực hiện

+ liên kết voice assistant để ra lệnh phân tích khám phá bằng lời nói

+ …

## Hướng mở rộng ĐỒ ÁN HỌC PHẦN

Sẽ tiếp tục mở rộng sản phẩm đề tài với các ứng dụng thực tiễn.

# 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. http://… : nguồn dataFrame được sử dụng trong đề tài này
4. ….

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