UNIVERSIDAD PRIVADA DE TACNA

Título del informe, práctica, caso de estudio, etc.

Escuela profesional de ingeniería de sistemas

SI-881,Inteligencia Artificial

1. Erick Javier Salinas Condori, 0009-0008-1156-1408

2. Aarón Pedro Paco Ramos, 0009-0004-6789-1175

Docente:
Dr. Oscar J. Jimenez Flores

Orcid:
0000-0002-7981-8467

Semestre:
UPT-EPIS, 2024-I

Resumen: Se realizo un sistema experto de vehiculos se mostraran unas preguntas y el usuario debera seleccionar de acuerdo a las mismas para desvelar el tipo de vehiculo que le tocara.

Palabras clave: experto,vehiculos,preguntas.

1. Introducción

En el contexto automovilístico actual, con la creciente complejidad de los vehículos modernos y la necesidad generalizada de diagnóstico y mantenimiento eficientes, los sistemas expertos se están convirtiendo en una solución prometedora. Diseñadas para simular el pensamiento humano en áreas específicas de especialización, estas aplicaciones de inteligencia artificial pueden identificar errores mecánicos, realizar diagnósticos precisos y proporcionar recomendaciones adecuadas de inmediato. Este trabajo investiga el diseño, desarrollo e implementación de sistemas expertos para vehículos, centrándose en sus componentes centrales, la adquisición de conocimientos y los procesos de gestión, así como sus beneficios potenciales, como la mejora de la eficiencia operativa y la seguridad. En definitiva, los sistemas expertos son herramientas valiosas para mejorar la seguridad, la fiabilidad y la eficiencia de los vehículos modernos, asegurando un futuro lleno de innovación en la industria del automóvil.

```
\section{Título de la sección}
```

La numeración se puede desactivar usando `\section*{}`. Una nueva subsección se crea con el comando

```
\subsection{Título de la subsección}
```

y, de manera similar, la numeración se puede desactivar agregando un asterisco de la siguiente manera

```
\subsection*{}
```

Se recomienda darle una etiqueta a cada sección usando el comando

```
\label{sec:nombre_sección}%
```

donde el argumento es simplemente una cadena de texto que utilizará para hacer referencia a esa parte como sigue: *La* INTRODUCCIÓN se encuentra en la 1 ....

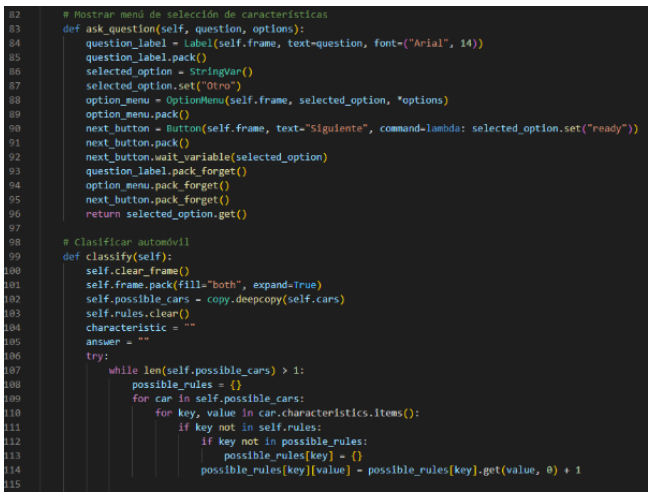
2. Algoritmos

main.py > Car > __init__

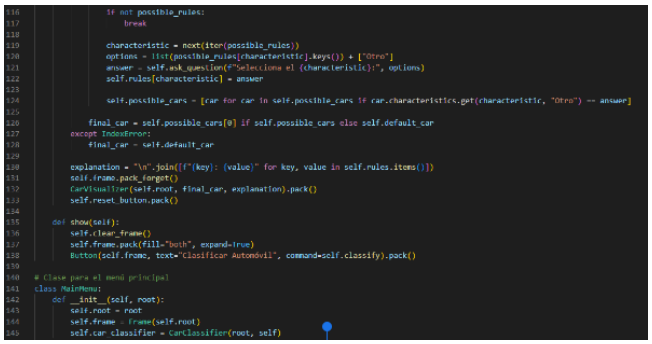
```python
from tkinter import *
from PIL import ImageTk, Image
import copy

# Clase para representar un automóvil
class Car:
    def __init__(self, name="", description="", image="sources/default_car.jpg", characteristics=None):
        self.name = name
        self.description = description
        self.image = image
        self.characteristics = characteristics if characteristics else {}

# Clase para visualizar un automóvil
class CarVisualizer(Frame):
    def __init__(self, root, car, explanation=""):
        super().__init__(root)
        self.car = car
        self.explanation = explanation
        self.pack(fill="both", expand=True)

        # Título
        title_label = Label(self, text=self.car.name, font=("Arial", 25))
        title_label.pack()

        # Descripción
        desc_label = Label(self, text=self.car.description, font=("Arial", 14), wraplength=800)
        desc_label.pack()

        # Explicación
        explanation_label = Label(self, text=self.explanation, font=("Arial", 14), wraplength=800)
        explanation_label.pack()
```

```python
        # Imagen
        img = Image.open(self.car.image)
        img = img.resize((400, 300), Image.LANCZOS)
        self.photo = ImageTk.PhotoImage(img)
        image_label = Label(self, image=self.photo)
        image_label.pack()

# Clase para clasificar automóviles
class CarClassifier:
    def __init__(self, root, main_menu):
        self.root = root
        self.main_menu = main_menu
        self.frame = Frame(self.root)
        self.cars = []
        self.default_car = Car("Desconocido", "Características no identificadas", "sources/default_car.jpg")
        self.load_cars()
        self.rules = {}
        self.possible_cars = []
        self.reset_button = Button(self.root, text="Reiniciar", command=self.main_menu.show)
```

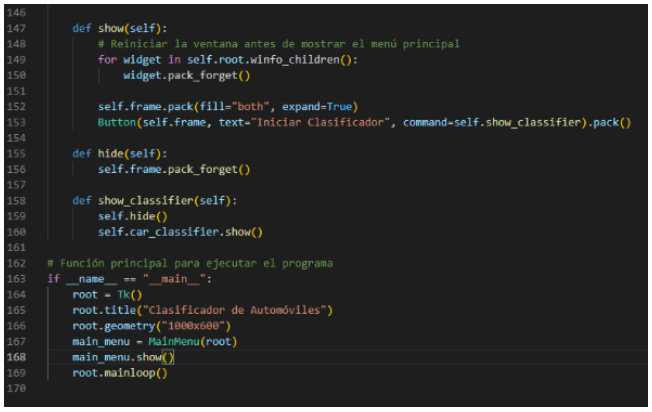
```python
    # Cargar automóviles en la base de datos
    def load_cars(self):
        car1 = Car("Toyota Corolla",
                   "Un sedán confiable y eficiente en combustible.",
                   "sources/Toyota_Corolla.jpg",
                   {"tipo": "sedán", "color": "blanco"})
        car2 = Car("Ford Mustang",
                   "Un clásico coche deportivo.",
                   "sources/ford_mustang.jpg",
                   {"tipo": "deportivo", "color": "rojo"})

        car3 = Car("Honda Civic",
                   "Un sedán compacto y económico.",
                   "sources/honda_civic.jpg",
                   {"tipo": "sedán", "color": "azul"})

        car4 = Car("BMW M5",
                   "Un sedán de lujo con un rendimiento excepcional.",
                   "sources/bmw_m5.jpg",
                   {"tipo": "sedán", "color": "negro"})

        # Agregando todos los autos a la lista
        self.cars.extend([car1, car2, car3, car4])

    def clear_frame(self):
        # Eliminar todos los widgets del frame
        for widget in self.frame.winfo_children():
            widget.pack_forget()
```



```
    # Mostrar menú de selección de características
    def ask_question(self, question, options):
        question_label = Label(self.frame, text=question, font=("Arial", 14))
        question_label.pack()
        selected_option = StringVar()
        selected_option.set("Otro")
        option_menu = OptionMenu(self.frame, selected_option, *options)
        option_menu.pack()
        next_button = Button(self.frame, text="Siguiente", command=lambda: selected_option.set("ready"))
        next_button.pack()
        next_button.wait_variable(selected_option)
        question_label.pack_forget()
        option_menu.pack_forget()
        next_button.pack_forget()
        return selected_option.get()

    # Clasificar automóvil
    def classify(self):
        self.clear_frame()
        self.frame.pack(fill="both", expand=True)
        self.possible_cars = copy.deepcopy(self.cars)
        self.rules.clear()
        characteristic = ""
        answer = ""
        try:
            while len(self.possible_cars) > 1:
                possible_rules = {}
                for car in self.possible_cars:
                    for key, value in car.characteristics.items():
                        if key not in self.rules:
                            if key not in possible_rules:
                                possible_rules[key] = {}
                            possible_rules[key][value] = possible_rules[key].get(value, 0) + 1

```



```
                if not possible_rules:
                    break

                characteristic = next(iter(possible_rules))
                options = list(possible_rules[characteristic].keys()) + ["Otro"]
                answer = self.ask_question(f"Seleccione el {characteristic}:", options)
                self.rules[characteristic] = answer

                self.possible_cars = [car for car in self.possible_cars if car.characteristics.get(characteristic, "Otro") == answer]

            final_car = self.possible_cars[0] if self.possible_cars else self.default_car
        except IndexError:
            final_car = self.default_car

        explanation = "\n".join([f"{key}: {value}" for key, value in self.rules.items()])
        self.frame.pack_forget()
        CarVisualizer(self.root, final_car, explanation).pack()
        self.reset_button.pack()

    def show(self):
        self.clear_frame()
        self.frame.pack(fill="both", expand=True)
        Button(self.frame, text="Clasificar Automóvil", command=self.classify).pack()

# Clase para el menú principal
class MainMenu:
    def __init__(self, root):
        self.root = root
        self.frame = Frame(self.root)
        self.car_classifier = CarClassifier(root, self)

```



```

    def show(self):
        # Reiniciar la ventana antes de mostrar el menú principal
        for widget in self.root.winfo_children():
            widget.pack_forget()

        self.frame.pack(fill="both", expand=True)
        Button(self.frame, text="Iniciar Clasificador", command=self.show_classifier).pack()

    def hide(self):
        self.frame.pack_forget()

    def show_classifier(self):
        self.hide()
        self.car_classifier.show()

# Función principal para ejecutar el programa
if __name__ == "__main__":
    root = Tk()
    root.title("Clasificador de Automóviles")
    root.geometry("1000x600")
    main_menu = MainMenu(root)
    main_menu.show()
    root.mainloop()

```

3. Conclusiones

Como conclusión, los sistemas expertos de vehículos son utiles a través de la integración de datos, reglas y algoritmos, estos sistemas pueden mejorar la eficiencia operativa, reducir costos de mantenimiento y optimizar el rendimiento de los vehículos modernos.

4. Bibliografía y citas

Tu trabajo académico debe contener una bibliografía adecuada que enumere todas las fuentes consultadas para desarrollar el trabajo. La lista de referencias se coloca al final del manuscrito después del capítulo que contiene las conclusiones. Se sugiere usar el paquete BibTeX y guardar las referencias bibliográficas en el archivo `bibliography.bib`. De hecho, este es una base de datos que contiene toda la información sobre las referencias. Para citar en tu manuscrito, usa el comando `\cite{}` de la siguiente manera:
Así es como citas entradas bibliográficas: [2], o varias a la vez: [3, 4].
La bibliografía y la lista de referencias se generan automáticamente ejecutando BibTeX [1].

Referencias

[1] CTAN. BiBTeX documentation.

[2] Donald E. Knuth. Computer programming as an art. *Commun. ACM*, pages 667–673, 1974.

[3] Donald E. Knuth. Two notes on notation. *Amer. Math. Monthly*, 99:403–422, 1992.

[4] Leslie Lamport. *LaTeX: A Document Preparation System*. Pearson Education India, 1994.