Documentación de la Aplicación de Diagnóstico de Enfermedades en Plantas

Visión General

La Aplicación de Diagnóstico de Enfermedades en Plantas es una herramienta de software diseñada para ayudar a los usuarios a diagnosticar enfermedades en plantas basadas en los síntomas proporcionados en un archivo de Excel. La aplicación utiliza un conjunto de reglas predefinidas para diagnosticar varias enfermedades de las plantas y presenta los resultados del diagnóstico en una interfaz gráfica de usuario (GUI) para facilitar la interpretación.

Funcionalidad

La aplicación ofrece las siguientes características clave:

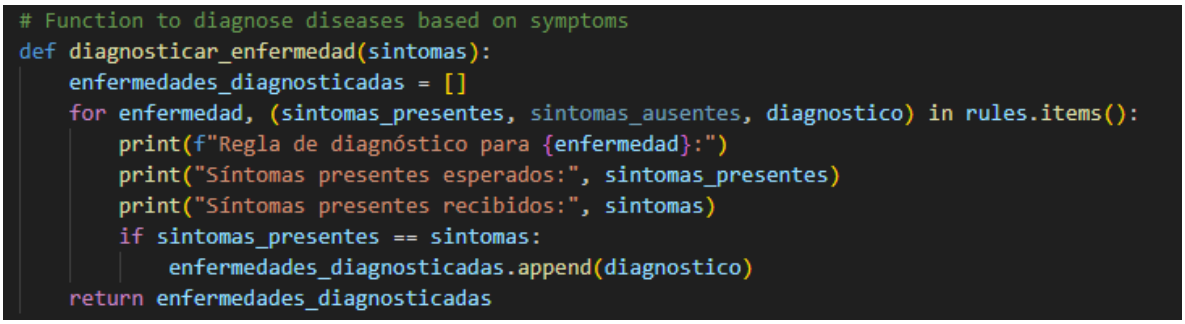
- Diagnóstico Basado en Síntomas: Los usuarios pueden cargar un archivo de Excel que contiene los síntomas de las plantas. La aplicación luego lee los datos del archivo, itera sobre cada fila (que representa una planta), extrae los síntomas, realiza el diagnóstico utilizando reglas predefinidas y almacena los resultados.
- Interfaz Gráfica de Usuario (GUI): Los resultados del diagnóstico se muestran en una ventana de GUI fácil de usar. Cada fila en la GUI representa una planta, con columnas que indican el número de planta y las enfermedades diagnosticadas, si las hay.

Decisiones de Diseño

1. Diagnóstico Basado en Reglas

Decisión: La aplicación emplea un enfoque de diagnóstico basado en reglas para diagnosticar enfermedades de las plantas. Cada enfermedad está asociada con un conjunto de síntomas que indican su presencia.

Justificación: El diagnóstico basado en reglas proporciona una forma estructurada y transparente de determinar enfermedades basadas en síntomas. Permite un mantenimiento y expansión fáciles de las reglas de diagnóstico a medida que se identifican nuevas enfermedades o síntomas.



```python
# Function to diagnose diseases based on symptoms
def diagnosticar_enfermedad(sintomas):
    enfermedades_diagnosticadas = []
    for enfermedad, (sintomas_presentes, sintomas_ausentes, diagnostico) in rules.items():
        print(f"Regla de diagnóstico para {enfermedad}:")
        print("Síntomas presentes esperados:", sintomas_presentes)
        print("Síntomas presentes recibidos:", sintomas)
        if sintomas_presentes == sintomas:
            enfermedades_diagnosticadas.append(diagnostico)
    return enfermedades_diagnosticadas
```

2. Diseño de la Interfaz de Usuario

Decisión: La GUI presenta los resultados del diagnóstico en un widget Treeview, con columnas para el número de planta y las enfermedades diagnosticadas.

Justificación: El widget Treeview ofrece una visualización jerárquica de datos, lo que lo hace adecuado para presentar múltiples resultados de diagnóstico. Permite a los usuarios navegar fácilmente a través de los diagnósticos y comprender las relaciones entre las plantas y las enfermedades.

```python
# Function to display diagnoses in a GUI window
def mostrar_diagnosticos(diagnosticos):
    root = tk.Tk()
    root.title("Diagnósticos de Enfermedades en Plantas")

    frame = ttk.Frame(root)
    frame.pack(padx=10, pady=10, expand=True, fill=tk.BOTH)

    tree = ttk.Treeview(frame, columns=("Planta", "Diagnóstico"), show="headings")
    tree.heading("Planta", text="Planta")
    tree.heading("Diagnóstico", text="Diagnóstico")

    for idx, planta_diagnostico in enumerate(diagnosticos, start=1):
        planta = f"Planta {idx}"
        diagnóstico = ", ".join(planta_diagnostico) if planta_diagnostico else "No se encontraron diagnósticos"
        tree.insert("", "end", values=(planta, diagnóstico))

    tree.column("Planta", width=100)
    tree.column("Diagnóstico", width=800)
    tree.pack(side="left", expand=True, fill=tk.BOTH)

    scrollbar = ttk.Scrollbar(frame, orient="vertical", command=tree.yview)
    scrollbar.pack(side="right", fill="y")
    tree.configure(yscrollcommand=scrollbar.set)

    root.mainloop()
```

3. Manejo de Errores

Decisión: La aplicación incluye mecanismos de manejo de errores para manejar excepciones durante la lectura de archivos o el diagnóstico.

Justificación: El manejo de errores garantiza un manejo elegante de situaciones inesperadas, como formatos de archivo no válidos o datos faltantes. Proporciona una mejor experiencia al usuario al informarles sobre el problema y evitar bloqueos

4. Uso de Bibliotecas Externas

Decisión: La aplicación utiliza la biblioteca pandas para la manipulación de datos y la biblioteca tkinter para el desarrollo de la GUI.

Justificación: pandas ofrece herramientas poderosas para leer y manipular archivos de Excel, lo que la hace ideal para manejar los datos de entrada. tkinter es una herramienta de interfaz gráfica estándar para Python, que proporciona componentes esenciales para crear interfaces de usuario con dependencias mínimas.

5. Mecanismos de Retroalimentación

Decisión: La aplicación proporciona retroalimentación al usuario durante el proceso de diagnóstico imprimiendo pasos de diagnóstico en la consola.

Justificación: La retroalimentación ayuda a los usuarios a entender el progreso del diagnóstico y proporciona transparencia en el proceso de toma de decisiones. Permite a los usuarios verificar que el diagnóstico se alinee con sus expectativas.