Herencia, Clase Abstracta y Template

Pedro Shiguihara

Febrero, 2021

1 Introducción

Se presentan 2 casos. El objetivo es implementar estos casos usando programación orientada a objetos. En la medida de lo posible, usar principalmente objetos y punteros.

2 Caso COVID

El COVID-19 es un virus que viene afectando a muchas ciudades alrededor del mundo. Los datos obtenidos en algunos estados como: Alabama, Illinois o Georgia son informativas de cómo este virus ha ido afectando gradualmente la vida de sus ciudadanos. Algunos datos están disponibles[1] y parte de ellos se tomaron para que un algoritmo de Inteligencia Artificial llamado *árbol de decisión* pueda obtener ciertas conclusiones.

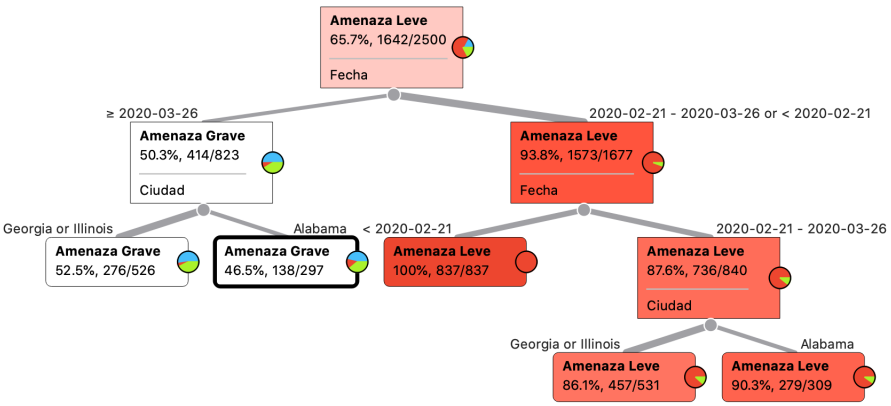

Figure 1: Árbol de decisión obtenido a partir del análisis de datos usando una muestra de un dataset sobre el COVID-19.

[1] https://www.kaggle.com/roche-data-science-coalition/uncover?

En la Figura 1 se muestra el árbol de de decisión que nos indica algunos hallazgos después de analizar 2500 casos sobre COVID-19 en Alabama, Georgia e Illinois:

- Desde el 26 de marzo del 2020 en adelante, hubo dos situaciones:
  - En ciudades como Georgia e Illinois, el *status* fue: **Amenaza Grave: 52.5%, 276/526**.
  - En la ciudad de Alabama, el *status* fue: **Amenaza Grave: 46.5%, 138/297**.
- Antes del 21 de febrero del 2020, para cualquiera de las tres ciudades, el *status* fue clasificado como: **Amenaza Leve: 100%, 837/837**.
- Desde el 21 de febrero del 2020 hasta el 25 de marzo del 2020, hubo dos situaciones detectadas:
  - En Georgia e Illinois, el *status* fue: **Amenaza Leve: 86.1%, 457/837**.
  - En la ciudad de Alabama, el *status* fue: **Amenaza Leve: 90.3%, 279/309**.
- En cualquier otra ciudad, no existen datos para poder analizar la situación según el dataset, por lo que el *status* es: **No hay datos**.
- Escriba un programa que detecte el *status* generado por el árbol de decisión. El usuario debe ingresar:
  - La ciudad (Tipo de dato en Python: str. Ejemplo: Illinois, Alabama, Lima, Georgia, Bogotá, etc.)
  - La fecha (Tipo de dato en Python: int. Ejemplo:*20200221* para denotar el 21 de febrero del 2020, *20200326* para denotar el 26 de marzo del 2020.)
- El programa debe reportar uno de los siguientes *status*, acorde con las condiciones mencionadas previamente, brindado por el árbol de decisión:
  - **Amenaza Grave: 52.5%, 276/526**
  - **Amenaza Grave: 46.5%, 138/297**
  - **Amenaza Leve: 100%, 837/837**
  - **Amenaza Leve: 86.1%, 457/837**
  - **Amenaza Leve: 90.3%, 279/309**
  - **No hay datos**

Implementar un menú para permitir las consultas de parte del usuario.

2.1 Ejemplo 1

Ingrese fecha: 20200321
Ingrese ciudad: Alabama
Amenaza Leve: 90.3%, 279/309

2.2 Ejemplo 2

Ingrese fecha: 20200120
Ingrese ciudad: Illinois
Amenaza Leve: 100%, 837/837

2.3 Ejemplo 3

Ingrese fecha: 20200514
Ingrese ciudad: Georgia
Amenaza Grave: 52.5%, 276/526

2.4 Ejemplo 4

Ingrese fecha: 20200219
Ingrese ciudad: Lima
No hay datos

3 Listas

Se dice que una lista está vacía cuando no contiene elementos, denotado por <>. El número de elementos actualmente almacenado es llamado longitud de la lista. El inicio de la lista es llamado cabeza y el fin de la lista es llamado cola.

Una lista puede o no tener elementos relacionados entre sí. Existen operaciones relevantes para gestionar una lista como:

- insertar un elemento en cualquier parte de la lista
- eliminar un elemento desde cualquier parte de la lista
- tener acceso a cualquier elemento de la lista para leerlo o modificarlo
- crear o limpiar (reinicializar) las listas
- accesar al elemento siguiente o al anterior desde un elemento *actual*

3.1 Clase Abstracta

Una clase abstracta es una cuyas funciones miembros son todas declaradas para ser **puras virtuales** como se indica con la notación = 0 al fin de la declaración de las funciones miembro de la clase.

La clase **Lista** define las funciones miembro que cualquier implementación de lista heredándola de ella debería tener, tomando también en cuenta sus parámetros y tipos de retorno. La lista puede ser incluso más flexible si incluímos el concepto de **template**.

Una clase abstracta no especifica cómo deben ser implementadas las operaciones.

3.2 Clase Abstracta Lista

La clase abstracta **Lista** es una clase template de un parámetro, llamado **E** (de elemento) que sirve para sustituir cualquier tipo de dato que el usuario desea almacenar.

agregar permite agregar un elemento al final de la lista

insertar permite insertar en cualquier parte de la lista un elemento

moverAPosicion permite cambiar la posición actual hacia otra posición

posicionActual permite retornar la posición actual en la lista

moverAInicio permite actualizar la posición actual al primer elemento de la lista

siguiente permite posicionarnos en el siguiente elemento desde el actual

anterior permite actualizar la posición actual al previo elemento desde el actual

limpiar permite eliminar los elementos de la lista y reinicializarla

getValor permite retornar el valor del elemento de la posición actual en la lista

eliminar permite retornar y eliminar un elemento de la lista

moverAInicio permite cambiar la posición actual al primer elemento de la lista

moverAlFinal permite cambiar la posición actual al último elemento de la lista

longitud permite retornar el tamaño de la lista

Implementa la clase abstracta plantilla **Lista**

3.3 Clase ALista

Implementa una lista en forma de arreglo con un tamaño fijo para almacenar datos. Debe heredar de la clase abstracta **Lista** sus métodos correspondientes.

3.4 Clase ListaEnlazada

Implementa una lista con un tamaño variable para almacenar datos. Debe heredar de la clase abstracta **Lista** sus métodos correspondientes.