26-11-2024

Práctica 3

Redes Bayesianas usando la herramienta Hugin y Python

Adrián Jiménez Benítez
UNIVERSIDAD DE ALMERIA

CONTENIDO

Ilustraciones 2
Enunciado 3
Ejercicio 1 3
Apartado A: 3
Respuesta a preguntas: 5
Apartado B: 6
Apartado C: 7
Apartado D: 7

ILUSTRACIONES

Ilustración 1: Red Bayesiana 3
Ilustración 2: Tabla Request 4
Ilustración 3: Tabla Firewall 4
Ilustración 4: Tabla Attacks 4
Ilustración 5: Tabla Recovery 4
Ilustración 6: Tabla System 4
Ilustración 7: Probabilidades 5
Ilustración 8: Probabilidad cortafuegos desactivado 5
Ilustración 9: Creación de la red Bayesiana 6
Ilustración 10: Código Python 6
Ilustración 11: Nombres de Variables 7
Ilustración 12: Variables Marginalmente independientes 7

ENUNCIADO

Un servidor web recibe cada hora una media de 1000 peticiones HTTP, de las cuales se estima que 100 son ataques. De éstos, se estima que 50 pueden suponer un fallo grave para la integridad del sistema. Se instala un cortafuegos que, según especificaciones del fabricante, no conseguirá bloquear 1 de cada 4 ataques. Además, dicho cortafuegos se actualizará una vez cada hora, lo cual tardará 3 minutos y durante este tiempo no estará activo. Por otro lado, se instalará un sistema de recuperación que estará en funcionamiento el 75 % del tiempo y que evitará la mitad de los fallos graves.

EJERCICIO 1

APARTADO A:

Utilizando la herramienta Hugin, modela este escenario como una red Bayesiana. ¿Cuál es la probabilidad de que, en cualquier momento, se produzca un fallo grave? ¿Y si se desconecta completamente el cortafuegos?

Para comenzar se va a ver los nodos y la probabilidad que tiene cada nodo.

- Nodo Request: Probabilidad de que la petición sea un ataque. Como muestra el enunciado la probabilidad de una petición sea un ataque es del 10%.
- Nodo Firewall: Probabilidad de que el firewall esté en funcionamiento. Está en funcionamiento el 95% del tiempo, porque el cortafuegos se actualizará una vez cada hora durante 3 minutos $\frac{3}{60} = 0.05$. Por tanto, el funcionamiento del firewall será $1 - 0.05 = 0.95$.
- Nodo Attack: Probabilidad de que el ataque cruce el cruce el firewall o no.
- Nodo Recovery: Probabilidad de que el sistema de recuperación esté activo o no (75%).
- Nodo System: Probabilidad de que el sistema tenga un fallo grave o no.

RED BAYESIANA

En este apartado, se mostrará la red bayesiana implementada en Hugin:

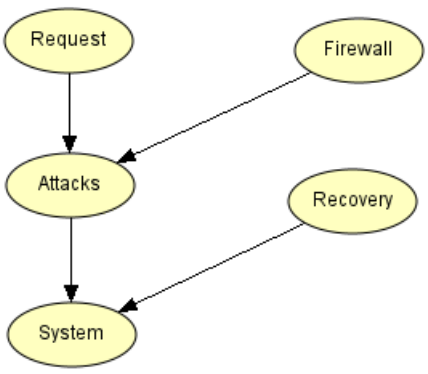


Ilustración 1: Red Bayesiana

TABLA DE PROBABILIDADES

A continuación, se verá como son las tablas generadas por Hugin al ponerle las probabilidades de cada caso.

TABLA REQUEST:

Request	
ataque	0.1
no_ataque	0.9

Ilustración 2: Tabla Request

TABLA FIREWALL:

Firewall	
activo	0.95
no_activo	0.05

Ilustración 3: Tabla Firewall

TABLA ATTACKS:

Attacks				
Firewall	activo		no_activo	
Request	ataque	no_ataque	ataque	no_ataque
pasa	0.25	0	1	0
no_pasa	0.75	1	0	1

Ilustración 4: Tabla Attacks

TABLA RECOVERY:

Recovery	
activo	0.75
no_activo	0.25

Ilustración 5: Tabla Recovery

TABLA SYSTEM:

System				
Recovery	activo		no_activo	
Attacks	pasa	no_pasa	pasa	no_pasa
fallo	0.25	0	0.5	0
no_fallo	0.75	1	0.5	1

Ilustración 6: Tabla System

RESPUESTA A PREGUNTAS:

Para responder a las preguntas, se pulsa el rayo que se encuentra en la parte de arriba del programa, y sale la siguiente pantalla:

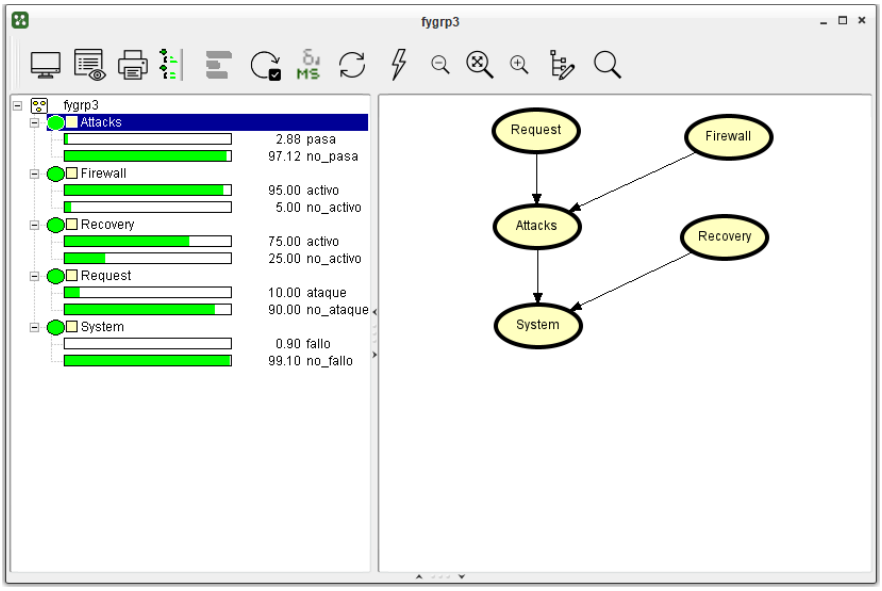


Ilustración 7: Probabilidades

¿CUÁL ES LA PROBABILIDAD DE QUE, EN CUALQUIER MOMENTO, ¿SE PRODUZCA UN FALLO GRAVE?

La probabilidad de que se produzca un fallo grave en cualquier momento es del 0.9%, tal y como muestra la imagen.

¿Y SI SE DESCONECTA COMPLETAMENTE EL CORTAFUEGOS?

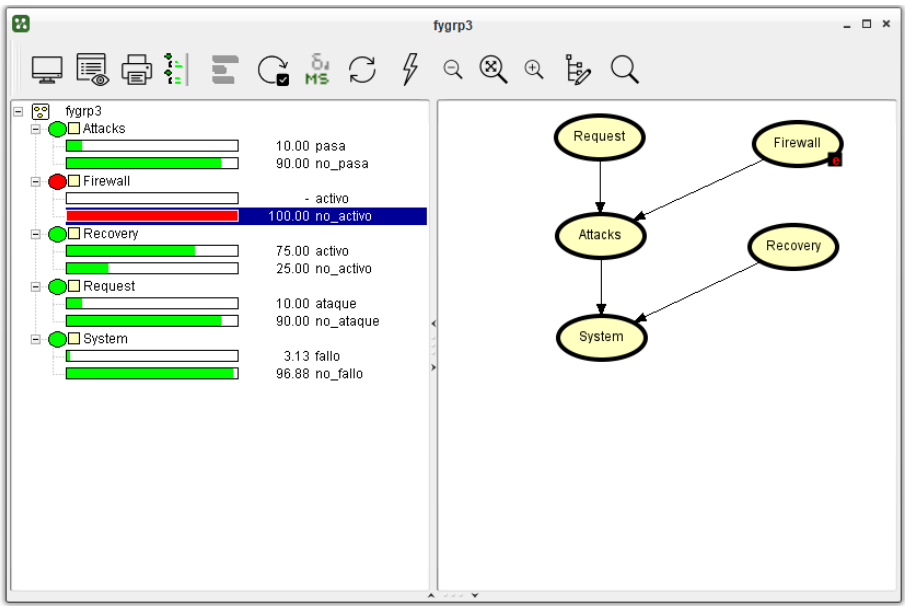
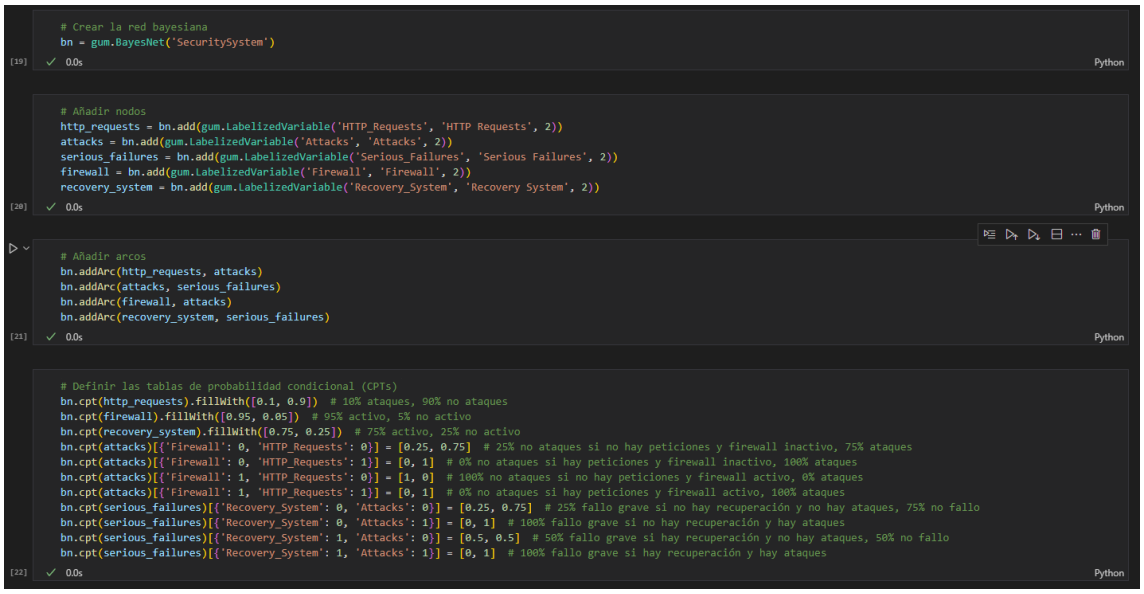


Ilustración 8: Probabilidad cortafuegos desactivado

Cuando no está el firewall activado, la probabilidad de que se produzca un fallo grave es del 3.13%.

APARTADO B:

Utilizando el paquete PyAgrum, especifica el código en Python que permita definir la misma red y calcular las probabilidades del apartado anterior.



```
# Crear la red bayesiana
bn = gum.BayesNet('SecuritySystem')
```
[19] ✓ 0.0s Python

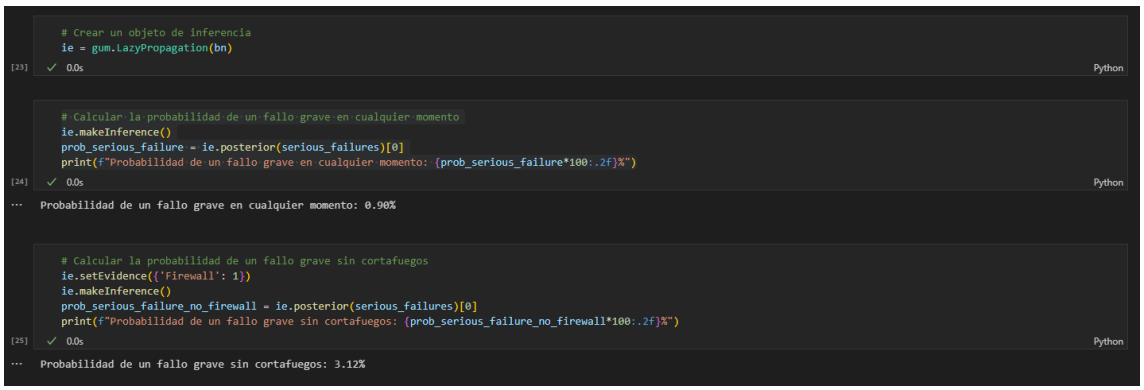
```
# Añadir nodos
http_requests = bn.add(gum.LabelizedVariable('HTTP_Requests', 'HTTP Requests', 2))
attacks = bn.add(gum.LabelizedVariable('Attacks', 'Attacks', 2))
serious_failures = bn.add(gum.LabelizedVariable('Serious_Failures', 'Serious Failures', 2))
firewall = bn.add(gum.LabelizedVariable('Firewall', 'Firewall', 2))
recovery_system = bn.add(gum.LabelizedVariable('Recovery_System', 'Recovery System', 2))
```
[20] ✓ 0.0s Python

```
# Añadir arcos
bn.addArc(http_requests, attacks)
bn.addArc(attacks, serious_failures)
bn.addArc(firewall, attacks)
bn.addArc(recovery_system, serious_failures)
```
[21] ✓ 0.0s Python

```
# Definir las tablas de probabilidad condicional (CPTs)
bn.cpt(http_requests).fillWith([0.1, 0.9])  # 10% ataques, 90% no ataques
bn.cpt(firewall).fillWith([0.95, 0.05])  # 95% activo, 5% no activo
bn.cpt(recovery_system).fillWith([0.75, 0.25])  # 75% activo, 25% no activo
bn.cpt(attacks)[{'Firewall': 0, 'HTTP_Requests': 0}] = [0.25, 0.75]  # 25% no ataques si no hay peticiones y firewall inactivo, 75% ataques
bn.cpt(attacks)[{'Firewall': 0, 'HTTP_Requests': 1}] = [0, 1]  # 0% no ataques si hay peticiones y firewall inactivo, 100% ataques
bn.cpt(attacks)[{'Firewall': 1, 'HTTP_Requests': 0}] = [1, 0]  # 100% no ataques si no hay peticiones y firewall activo, 0% ataques
bn.cpt(attacks)[{'Firewall': 1, 'HTTP_Requests': 1}] = [0, 1]  # 0% no ataques si hay peticiones y firewall activo, 100% ataques
bn.cpt(serious_failures)[{'Recovery_System': 0, 'Attacks': 0}] = [0.25, 0.75]  # 25% fallo grave si no hay recuperación y no hay ataques, 75% no fallo
bn.cpt(serious_failures)[{'Recovery_System': 0, 'Attacks': 1}] = [0, 1]  # 100% fallo grave si no hay recuperación y hay ataques
bn.cpt(serious_failures)[{'Recovery_System': 1, 'Attacks': 0}] = [0.5, 0.5]  # 50% fallo grave si hay recuperación y no hay ataques, 50% no fallo
bn.cpt(serious_failures)[{'Recovery_System': 1, 'Attacks': 1}] = [0, 1]  # 100% fallo grave si hay recuperación y hay ataques
```
[22] ✓ 0.0s Python

Ilustración 9: Creación de la red Bayesiana

Para obtener este resultado, primero se creó la misma red Bayesiana, como se puede observar en la imagen anterior. Posteriormente, el código desarrollado permite calcular las probabilidades que previamente ha generado la aplicación Hugin, tal como se muestra en la siguiente imagen.



```
# Crear un objeto de inferencia
ie = gum.LazyPropagation(bn)
```
[23] ✓ 0.0s Python

```
# Calcular la probabilidad de un fallo grave en cualquier momento
ie.makeInference()
prob_serious_failure = ie.posterior(serious_failures)[0]
print(f"Probabilidad de un fallo grave en cualquier momento: {prob_serious_failure*100:.2f}%")
```
[24] ✓ 0.0s Python

```
Probabilidad de un fallo grave en cualquier momento: 0.90%
```

```
# Calcular la probabilidad de un fallo grave sin cortafuegos
ie.setEvidence({'Firewall': 1})
ie.makeInference()
prob_serious_failure_no_firewall = ie.posterior(serious_failures)[0]
print(f"Probabilidad de un fallo grave sin cortafuegos: {prob_serious_failure_no_firewall*100:.2f}%")
```
[25] ✓ 0.0s Python

```
Probabilidad de un fallo grave sin cortafuegos: 3.12%
```

Ilustración 10: Código Python

APARTADO C:

Define una función que tome como parámetro de entrada una red Bayesiana y devuelva una lista con los nombres de las variables.

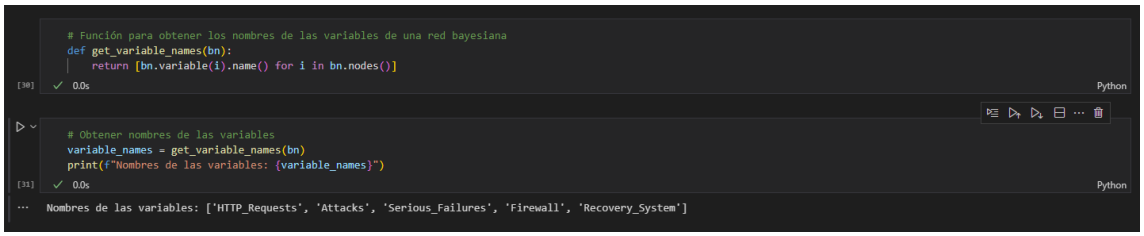


Ilustración 11: Nombres de Variables

La función *"get_variable_names(bn)"* está diseñada para obtener los nombres de las variables que componen una red bayesiana, representada aquí por el objeto bn. Este objeto contiene tanto los nodos de la red como las variables asociadas.

APARTADO D:

Escribe el código que indique, para cada par de variables, si son marginalmente independientes o no.

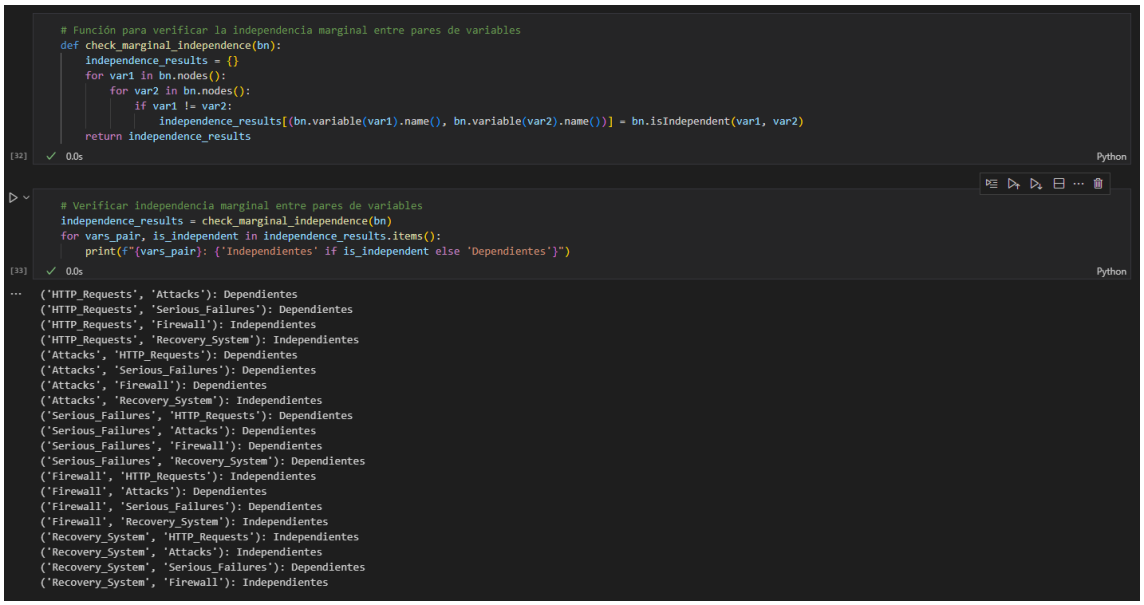


Ilustración 12: Variables Marginalmente independientes

La función *"check_marginal_independence(bn)"* evalúa la independencia marginal entre todos los pares de variables en una red bayesiana, representada por el objeto bn.