Informe

Integrantes

Sebastián Suárez Gómez C411

Héctor Miguel Rodríguez Sosa C411

Introducción

En este proyecto pretendemos simular el tiempo de vida de una aseguradora mediante el uso de diferentes funciones de distribución en los posibles eventos que puedan ocurrir. Estos eventos son:

- la llegada de un nuevo cliente,
- la salida de un cliente de la empresa
- la llegada de una queja para un pago
- el valor de ese pago

Cada cliente de la empresa debe realizar su pago a la aseguradora por unidad de tiempo, este pago es un costo fijo. La empresa puede comenzar la simulación con una determinada cantidad de clientes y con un presupuesto inicial.

Objetivos y Metas

El objetivo de este proyecto es similar a lo descrito anteriormente para comprender su comportamiento y desempeño bajo diferentes condiciones y distribuciones. Las metas específicas incluyen: - Analizar el tiempo de vida de la aseguradora para conocer su rentabilidad en el tiempo - Analizar el tiempo de llegada y salida de los clientes

Varibles que describen el problema

- Cantidad inicial de clientes
- Presupuesto inicial de la empresa
- Distribución de llegada de los clientes a la empresa
- Distribución de salida de los clientes de la empresa
- Distribución de llegada de una nueva queja por parte de algún cliente de la empresa
- Distribución del valor de la queja realizada por un cliente en la empresa

Variables de interés

- Tiempo de vida de la empresa
- Cantidad de clientes que llegaron a la empresa en su tiempo de vida
- Cantidad de clientes que abandoraron a la empresa en su tiempo de vida
- Cantidad de quejas que llegaron a la empresa en su tiempo de vida

Detalles de Implementación

Para la implementación de la simulación se utilizó el lenguaje de programación Python.

Esta simulación contiene diferentes datos, los posibles eventos y los valores iniciales como el costo para el cliente, la cantidad inicial de clientes de la empresa y el presupuesto inicial.

Estos son los datos iniciales:

```
import numpy as np
from heapq import heapify, heappush, heappop

next_client_dist = np.random.poisson
next_complain_dist = np.random.poisson
claim_dist = np.random.binomial
leave_client_dist = np.random.poisson

N0_CLIENTS = 5
A0_INITIAL_BUDGET = 10000
COST = 100
MAX_TIME = 1_000_000
MAX_CLAIM = 5_000
CLIENT_THRESHOLD = 0

lambda_next_client = 1
lambda_leave_client = 2
lambda_next_complaint = 5
amount_tries = 100
amount_tests = 50
```

La similación empieza inicializando los datos

```
def insurance_simulation(next_client_dist, next_complain_dist,
 claim_dist, leave_client_dist, n0_clients, a0_initial_budget,
 cost, client_threshold):
    total_clients = n0_clients
    total_complaints = 0
    time = 0
    num_clients = n0_clients
    budget = a0_initial_budget
    clients = [leave_client_dist(lambda_leave_client)
                for _ in range(num_clients)]

    heapify(clients)
```

```
    next_client, leave_client = generate_client
                      (next_client_dist, leave_client_dist)
    next_complaint, claim = generate_complain
                       (next_complain_dist, claim_dist)
```

Generamos el tiempo de salida de la empresa de los clientes iniciales mediante la distribución propuesta y los valores los almacenamos en un heap de mínimos para siempre tener el menor tiempo del cliente que abandonará la aseguradora.

```
def generate_complain(distribution_function_complain,
  distribution_function_claim):
    u = np.random.random()
    time_to_complain = distribution_function_complain(
                         lambda_next_complaint)
    claim_amount = distribution_function_claim(MAX_CLAIM, u)

    return time_to_complain, claim_amount


def generate_client(distribution_function_client,
  distribution_function_leave):
    next_client = distribution_function_client(lambda_next_client)
    leave_client = distribution_function_leave(lambda_leave_client)

    return next_client, leave_client
```

Luego generamos los valores de los posibles eventos con sus funciones de distribución.

La simulación tiene lugar mientras el tiempo recorrido sea menor a un tiempo máximo(un valor predefinido para forzar una parada y así evitar un bucle infinito) en un ciclo while.

Usamos la distribución de Poisson para los eventos que ocurren en el tiempo, y usamos la ditribución Binomial alrededor de un valor predefinido en este caso **MAX_CLAIM=5000**, sería el máximo valor a cobrar por una queja, siendo el valor de n y el valor de p, un valor obtenido mediante una distribución uniforme.

Exiten 2 posibilidades por los posibles valores de las distribuciones:

Si **next_complaint** es menor que **next_client** y **num_clients** es distinto de cero se le cobra a los diferentes clientes en el tiempo, se le resta al presupuesto el valor de la queja, eliminamos los clientes que su tiempo de ida sea menor o igual que el tiempo a recorrer y le restamos el tiempo recorrido a los otros clientes, luego aumentamos el tiempo actual y generamos la próxima queja y el valor de la misma. Usamos client_threshold como una cota inferior a la cantidad de clientes que puede tener la empresa, por lo general se considera 0 pero puede variar.

```
    if num_clients != 0 and next_complaint <= next_client:
```

```
        t = next_complaint
        next_client -= t
        budget = budget + num_clients * t * cost
        budget = budget - claim
        total_complaints += 1

        if budget < 0:
             return time, total_clients, total_complaints

        next_complaint, claim =
            generate_complain(next_complain_dist, claim_dist)

        while len(clients) > client_threshold and clients[0] <= t:

             num_clients -= 1
             heappop(clients)

        for i in range(len(clients)):
            clients[i] -= t

        time = time + t
```

Si **next_client** es menor que **next_complaint** realizamos algo similar lo que esta vez solo se recoge dinero de los clientes y se eliminan los que el tiempo sea menor al recorrido y restamos el tiempo de los otros que queden.

```
    else:
        t = next_client

        if next_client < next_complaint:
             next_complaint -= t

        num_clients += 1
        total_clients += 1
        heappush(clients, leave_client)
        budget = budget + num_clients * t * cost
        next_client, leave_client =
            generate_client(next_client_dist, leave_client_dist)

        while len(clients) > client_threshold and clients[0] <= t:
            num_clients -= 1
            heappop(clients)

        for i in range(len(clients)):
            clients[i] -= t

        time = time + t
```

En ambas posibilidades se el resta el tiempo de la menor posibilidad a la mayor para representar el recorrido del tiempo

```
if num_clients != 0 and next_complaint <= next_client:
    t = next_complaint
    next_client -= t

else:
    t = next_client

    if next_client < next_complaint:
        next_complaint -= t
```

Esta simulación se detiene si el presupuesto es menor a 0 (**budget<0**) o si llega al tiempo máximo predefinido (**MAX_TIME = 1_000_000**)

Resultados

Simulación 1 Con los datos actuales de la simulación se realizaron 50 muestras de 2000 simulaciones cada una. Los diferentes resultados en las simulaciones entre las 50 muestras son:

- Media: 1200-1400
- Varianza: 3788649-6674379
- Desviacion Estandar: 1946-2583

A partir de estos datos, la media muestra que la empresa quebrará alrededor de las 1300 unidades de tiempo por otro lado la varianza y la desviación estandar muestran, que es muy incierto conocer con exactitud cuando quebrará la empresa ya que los datos de las simulaciones estan muy dispersos. No hay un resultado muy confiable.

A parte del tiempo de vida de la empresa, la simulación devuelve también la cantidad de clientes y quejas totales que tuvo la empresa en su tiempo de vida. Pero al estos datos ser generados por la distribución de Poisson, no son de mucho interés.

Todos los datos estan almacenados el archivo *data.json*, por si se quiere realizar otros trabajos estadísticos.

Simulación 2 next_client_dist = np.random.poisson next_complain_dist = np.random.poisson claim_dist = np.random.binomial leave_client_dist = np.random.poisson

N0_CLIENTS = 5 A0_INITIAL_BUDGET = 10000 COST = 100 MAX_TIME = 1_000_000 MAX_CLAIM = 5_000 CLIENT_THRESHOLD = 0

lambda_next_client = 1 lambda_leave_client = 5 lambda_next_complaint = 5

Aumentamos en valor de lambda de distribución de salida de los clientes

Los resultados de esta simulación

- Media de Tiempo de vida de la Aseguradora: 82.5
- Media de llegada de nuevos clientes: 88.0
- Media de salida de clientes: 16.41
- Varianza de Tiempo de vida de la Aseguradora: 2277.29
- Varianza de llegada de nuevos clientes: 2546.56
- Varianza de salida de clientes: 73.36
- Desviacion Estándar de Tiempo de vida de la Aseguradora: 1138.64
- Desviacion Estándar de llegada de nuevos clientes: 1273.28
- Desviacion Estándar de salida de clientes: 36.68095

Pudimos observar aqui que al aumentar la probabilidad que los clientes abandonen la empresa en un periodo de tiempo mas corto, el tiempo de vida de la empresa se reduce.

Simulación 3 next_client_dist = np.random.poisson next_complain_dist = np.random.poisson claim_dist = np.random.binomial leave_client_dist = np.random.poisson

N0_CLIENTS = 10 A0_INITIAL_BUDGET = 10000 COST = 100 MAX_TIME = 1_000_000 MAX_CLAIM = 5_000 CLIENT_THRESHOLD = 0

lambda_next_client = 3 lambda_leave_client = 3 lambda_next_complaint = 5

Aumentamos la cantidad de clientes iniciales e igualamos el lambda de salida de los clientes con la llegada de una nueva queja

- Media de Tiempo de vida de la Aseguradora: 400272.3
- Media de llegada de nuevos clientes: 400358.7
- Media de salida de clientes: 79904.3
- Varianza de Tiempo de vida de la Aseguradora: 239782324096.80
- Varianza de llegada de nuevos clientes: 239890072875.61
- Varianza de salida de clientes: 9554658731.81
- Desviacion Estándar de Tiempo de vida de la Aseguradora: 119891162048.40
- Desviacion Estándar de llegada de nuevos clientes: 119945036437.80
- Desviacion Estándar de salida de clientes: 4777329365.90

Podemos observar que el tiempo de vida de la aseguradora aumenta drasticamente, pero estos datos no son muy fiables debido al alto valor de la varianza y la desviacion estándar.

Modelo Matemático

Para esta simulación asumimos un modelo sin pérdida de memoria, generamos de los tiempos de los 3 posibles eventos (llegada de un nuevo cliente, salida de un

cliente y la llegada de una nueva queja con su valor monetario). Almacenamos esos valores en un heap para tener siempre el evento con el menor tiempo a ocurrir. Por cada unidad de tiempo que transcurre se cobra a los clientes de la empresa el valor de la subscripción. El evento de llegada y salida del cliente son parecidos, se recauda el dinero y dependiendo del que sea aumenta o diminuye la cantidad de clientes. Con la llegada de una nueva queja se recauda hasta el momento del evento y despues se descuenta el valor de la queja del presupuesto de la empresa. La simulación termina cuando la empresa se quiebra o cuando pasa un MAX_TIME, devolviendo el tiempo de vida de la empresa.

Supuestos y Restricciones

- La probabilidad de las quejas no depende de la cantidad de clientes que tenga la aseguradora
- Distribución invariable de llegadas de clientes: Se supone que la distribución de llegadas de clientes no cambia con el tiempo. Esto implica que la tasa de llegada de clientes se mantiene constante durante toda la simulación.