**ESPECIALIZACIÓN EN DESARROLLO DE SOFTWARE**
**LEANDRO JOSÉ GUTIÉRREZ MIRANDA CODIGO: 2019188003**

# MODELO CASCADA

En presente documento se aplicará el modelo de desarrollo en cascada a la campaña o modelo de fidelización utilizado por los bancos para las ofertas de sus servicios para su fidelización. Para ello, se realizará la descripción de cada una de las etapas de dicho modelo Ver figura 1.

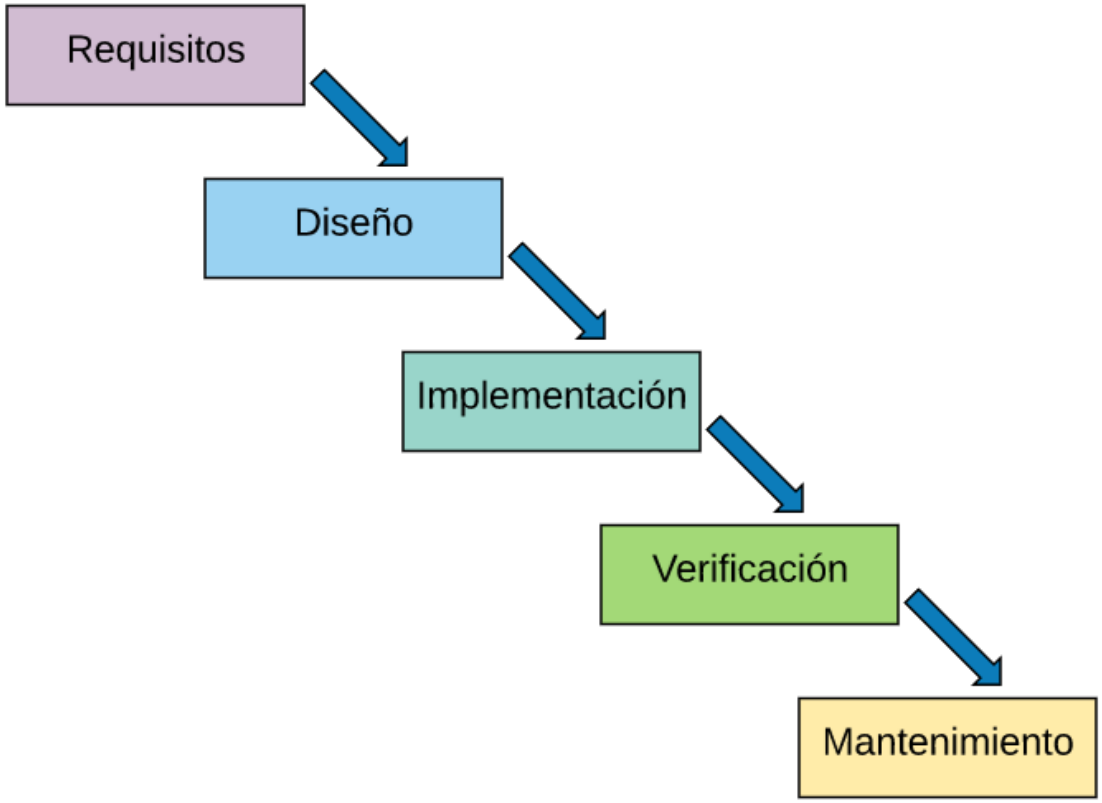


Figura 1. Etapas del Modelo Cascada

1. **Requisitos:** En esta etapa se analizarán las necesidades de los usuarios que interactuarán con el software. Por ende, se relacionan los siguientes requerimientos:
   - El sistema deberá ser capaz de realizar la predicción de la campaña de fidelización a utilizar.
   - El sistema deberá cargar los datos en formato .csv para su análisis.
   - El sistema deberá listar los datos cargados previamente (mostrando las columnas Tipo de Ocupación, si es mayor a una edad establecida, Volumen de compras, Elemento comprados y la campaña a utilizar).

Para una mayor comprensión sobre los requerimientos o del tipo de datos a tratar se muestra con el modelo bayesiano:

| | Tipo de Ocupación | Mayor a 30 | Volumen de Compras | Mayoria de elementos comprados | Campaña |
|---|---|---|---|---|---|
| 0 | Empleado | Si | Alto | Ferretería | Salud |
| 1 | Independiente | Si | Medio | Vesturario | Hogar |
| 2 | Independiente | No | Alto | Vesturario | Viajes |
| 3 | Independiente | Si | Alto | Mercado | Hogar |
| 4 | Empleado | Si | Bajo | Mercado | Viajes |
| 5 | Empleado | Si | Medio | Vesturario | Salud |
| 6 | Empleado | No | Alto | Vesturario | Viajes |

Figura 2. Información de las campañas

## 2. Diseño:

Para realizar el diseño de nuestra solución, se implementará el Teorema de la probabilidad llamado Teoremas de Bayes el cual consiste en lo siguiente: "P*robabilidad condicional de un evento aleatorio A dado B en términos de la distribución de probabilidad condicional del evento B dado A y la distribución de probabilidad marginal de solo A.*

*En términos más generales y menos matemáticos, el teorema de Bayes es de enorme relevancia puesto que vincula la probabilidad de A dado B con la probabilidad de B dado A. Es decir, por ejemplo, que sabiendo la probabilidad de tener un dolor de cabeza dado que se tiene gripe, se podría saber (si se tiene algún dato más), la probabilidad de tener gripe si se tiene un dolor de cabeza. Muestra este sencillo ejemplo la alta relevancia del teorema en cuestión para la ciencia en todas sus ramas, puesto que tiene vinculación íntima con la comprensión de la probabilidad de aspectos causales dados los efectos observados."*[1]

Sea $\{A_1, A_2, \ldots, A_i, \ldots, A_n\}$ un conjunto de sucesos mutuamente excluyentes y exhaustivos, y tales que la probabilidad de cada uno de ellos es distinta de cero (0). Sea B un suceso cualquiera del que se conocen las probabilidades condicionales $P(B|A_i)$. Entonces, la probabilidad $P(A_i|B)$ viene dada por la expresión:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)}$$

donde:

- $P(A_i)$ son las probabilidades a priori,
- $P(B|A_i)$ es la probabilidad de $B$ en la hipótesis $A_i$,
- $P(A_i|B)$ son las probabilidades a posteriori.

Thomas Bayes (1763)

Figura 2. Etapas del Modelo Cascada, https://es.wikipedia.org/wiki/Teorema_de_Bayes

[1] https://es.wikipedia.org/wiki/Teorema_de_Bayes

Lo anterior, se utilizará para predecir la mejor estrategia o campaña de fidelización según los datos analizados mostrados en la figura 2. y de esta forma determinar la mejor campaña a sus usuarios.

## 3. Implementación:

Para realizar la implementación de la predicción de la campaña de fidelización utilizada por el Banco de la alegría, se utilizará las siguientes herramientas:

- ✓ Lenguaje de programación Python
- ✓ Sklearn (librería para Machine Learning en Python)
- ✓ Librería Pandas
- ✓ Jupyter.

```
In [1]: import pandas as pd

In [3]: Datos = pd.read_csv('Datos_Balegria.csv')

In [4]: Datos.head(7)
```

Out[4]:

| | Tipo de Ocupación | Mayor a 30 | Volumen de Compras | Mayoria de elementos comprados | Campaña |
|---|---|---|---|---|---|
| 0 | Empleado | Si | Alto | Ferretería | Salud |
| 1 | Independiente | Si | Medio | Vesturario | Hogar |
| 2 | Independiente | No | Alto | Vesturario | Viajes |
| 3 | Independiente | Si | Alto | Mercado | Hogar |
| 4 | Empleado | Si | Bajo | Mercado | Viajes |
| 5 | Empleado | Si | Medio | Vesturario | Salud |
| 6 | Empleado | No | Alto | Vesturario | Viajes |

```
In [24]: features_train = Datos.iloc[0:7,0:4]
         target_train = Datos.iloc[0:7,4]
         #print(label)

In [27]: #import LabelEnconder

         from sklearn import preprocessing
         le = preprocessing.LabelEncoder()
         f0=le.fit_transform(features_train.iloc[0:7,0])
         f1=le.fit_transform(features_train.iloc[0:7,1])
         f2=le.fit_transform(features_train.iloc[0:7,2])
         f3=le.fit_transform(features_train.iloc[0:7,3])
         label = le.fit_transform(target_train)
         features = list(zip(f0,f1,f2,f3))
         print(features)
         print(label)

[(0, 1, 0, 0), (1, 1, 2, 2), (1, 0, 0, 2), (1, 1, 0, 1), (0, 1, 1, 1), (0, 1, 2, 2), (0, 0, 0, 2)]
[1 0 2 0 2 1 2]

In [31]: from sklearn.naive_bayes import GaussianNB
         model1 = GaussianNB()
         model1.fit(features,label)
         Predicted =model1.predict([[0,0,2,1]])
         print(Predicted)

[2]
```

## 4. Verificación:

### Datos sin categorizar

| | Tipo de Ocupación | Mayor a 30 | Volumen de Compras | Mayoria de elementos comprados | Campaña |
|---|---|---|---|---|---|
| 0 | Empleado | Si | Alto | Ferretería | Salud |
| 1 | Independiente | Si | Medio | Vesturario | Hogar |
| 2 | Independiente | No | Alto | Vesturario | Viajes |
| 3 | Independiente | Si | Alto | Mercado | Hogar |
| 4 | Empleado | Si | Bajo | Mercado | Viajes |
| 5 | Empleado | Si | Medio | Vesturario | Salud |
| 6 | Empleado | No | Alto | Vesturario | Viajes |

Se procede a realizar una prueba unitaria donde, se tienes los valores categorizados:

| Tipo de Ocupación | valor | Mayor a 30 | valor | Volumen de Compras | valor | Mayoría de elementos comprados | valor | Campaña | valor |
|---|---|---|---|---|---|---|---|---|---|
| Empleado | 0 | No | 0 | Alto | 0 | Ferretería | 0 | Hogar | 0 |
| Independiente | 1 | Si | 1 | Bajo | 1 | Mercado | 1 | Salud | 1 |
| | | | | Medio | 2 | Vesturario | 2 | Viajes | 2 |

Para nuestra prueba unitaria, para ellos se utilizará el marco de pruebas propia del lenguaje Python, para ello se ingresarán los siguientes valores en la función **numBayesiano(0,1,1,1) (Empleado,Si,Bajo,Mercado).** Se espera que el resultado de nuestra prueba sea **viajes** según la imagen sin categorizar ó el valor dos (2) según los datos categorizados. Por ende, al realizar la prueba e implementar la línea assert.equals debe dar verdadera, se relaciona el código fuente de la prueba realizada:

```
#algoritmo de prueba utulizando la herramienda unittest
import unittest
from sklearn.naive_bayes import GaussianNB def numBayesiano(col1,col2,col3,col4):
    #modelo = GaussianNB()
    #modelo.fit(features,label)
    Predicted = model1.predict([[col1,col2,col3,col4]])
    return Predicted

class predecir(unittest.TestCase):
    def validarPredecir(self):
        self.assertEquals(numBayesiano(1,1,1,1),2)
        #self.assertEquals(2,3)
unittest.main(argv=['first-arg-is-ignored'],exit=False)
```

## 5. Mantenimiento

Podemos hablar de mantenimiento de software a la mejora correctiva o preventiva de un producto de software ya entregado al cliente final. Por ende, se destina el mayor porcentaje del valor de los recursos a dicha fase por su grado de criticidad. Para nuestro caso, se busca que el software de fidelización desarrollado (basado en el teorema de Bayes) sea funcional en el mayor tiempo posible, para ello, se deberá realizar cambios pertinentes en el sistema para la satisfacción de las necesidades del cliente.