Jose Alfonso Pérez Contreras
Cod:2019188005

Análisis de requerimientos de Software de fidelización del Banco de la Alegría

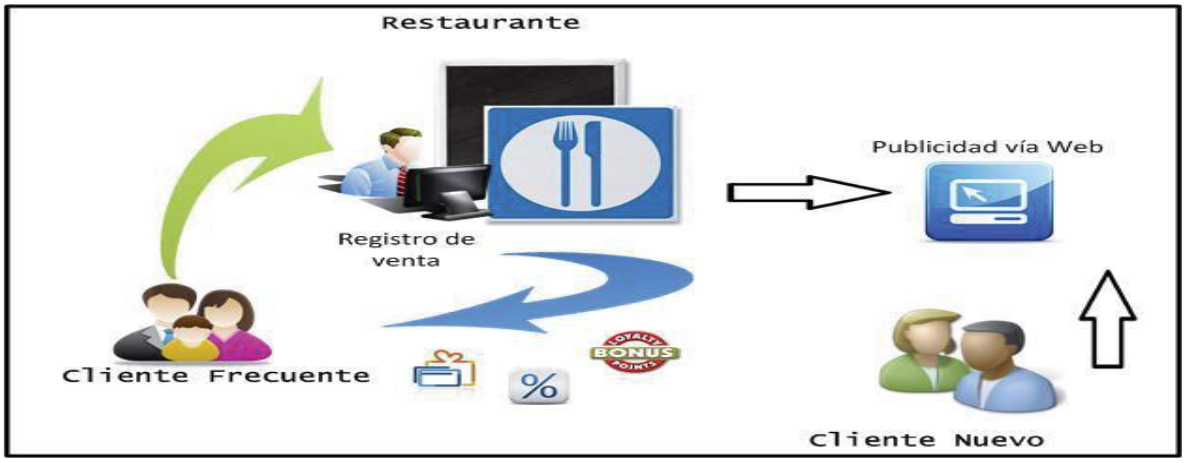
Se requiere un software que realice el estudio de fidelización de un cliente, mediante diferentes premios según los gustos del cliente.

Análisis de modelo cascada

Se debe aplicar el modelo Cascada en software que escoge aleatoriamente el premio que más le gusta al cliente, para esto utilizaremos un modelo bayesiano que realice la elección.

Tipo de Ocupación	Mayor a 30	Volumen de Compras	Mayoría de elementos comprados	Campaña
Empleado	Si	Alto	Ferretería	Salud
Independiente	Si	Medio	Vestuario	Hogar
Independiente	No	Alto	Vestuario	Viajes
Independiente	Si	Alto	Mercado	Hogar
Empleado	Si	Bajo	Mercado	Viajes
Empleado	Si	Medio	Vestuario	Salud
Empleado	No	Alto	Vestuario	Viajes
Empleado	Si	Alto	Mercado	Hogar

Descripción de la Necesidad.



Requisitos del software

Se busca gestionar una campaña de fidelización para que los clientes del banco de Alegría puedan disfrutar sus puntos en los diferentes productos y servicios que se les entregara según el gusto de cada cliente.

Se entregarán diferentes Beneficios:

- (descuentos, regalos, bonos, productos / servicios de socios)

- Esquemas variados de recompensa adaptados al grado de participación
- Gestión de la logística de los premios

1. El sistema debe ser un usuario registrado.
2. El sistema debe verificar que tengas puntos acumulados
3. El sistema debe verificar los gustos del cliente
4. El sistema de asignarle el beneficio que más se acerque a los gustos del cliente (Según campaña de fidelización)

Diseño de la solución

Para esta solución decidimos trabajar en aplicando el teorema Bayes de probabilidad que consiste en:

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.

Sea {\displaystyle \{A_{1},A_{2},...,A_{i},...,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 {\displaystyle P(B|A_{i})} . Entonces, la

probabilidad {\displaystyle P(A_{i}|B)} viene dada por la expresión:

{\displaystyle P(A_{i}|B)={\frac {P(B|A_{i})P(A_{i})}{P(B)}}}

donde:

- {\displaystyle P(A_{i})} son las probabilidades a priori,
- {\displaystyle P(B|A_{i})} es la probabilidad de {\displaystyle B} en la hipótesis {\displaystyle A_{i}} ,
- {\displaystyle P(A_{i}|B)} son las probabilidades a posteriori.

Jose Alfonso Pérez Contreras
Cod:2019188005

Implementación

En esta fase ponemos en ejecución los requisitos especificados haciendo uso de las estructuras de datos diseñadas en la fase anterior. La programación se desarrollara de la siguiente manera.

In [13]:
```
import pandas as pd
import unittest
```

In [2]:
```
Datos = pd.read_csv('Datos_balegria.csv')
Datos.head(7)
```

Out[2]:

	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 [8]:
```
features_train = Datos.iloc[0:7,0:4]
target_train = Datos.iloc[0:7,4]
#print(label)
```

In [9]:
```
#Import  LabelEncoder
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 [12]:
```
#from sklear.naive_bayes import GaussianNB
#model1 = GaussianNB()
#model1.fit(features,label)
#Predicted = model.pedrict([[1,1,0,2]])
#Predicted = model.pedrict([[0,1,2,1]])
#Predicted = model1.predict([[0,0,2,1]])
#print(le.inverse_transform(Predicted))
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-12-e652b2262c00> in <module>
----> 1 from sklear.naive_bayes import GaussianNB
      2 model1 = GaussianNB()
```

Jose Alfonso Pérez Contreras
Cod:2019188005

Verificación

La verificación se realizará mediante pruebas unitarias mediante unittest, consta en programar pequeñas pruebas que validen una función de nuestro programa, es decir, si tenemos un método llamado "divide_entre_dos" el cual recibe un número entero y te devuelve ese número dividido entre dos, nuestra prueba unitaria debería validar que en todo momento nuestro método devuelve el resultado esperado al ingresarle el valor correcto.

Con esta prueba buscamos que mediante la elección del cliente

```
import unittest

class TestMyModule(unittest.TestCase)

    def test_get_prime_numbers(self):

        self.assertEqual(mymodule.get_prime_numbers(10), [2, 3, 5, 7])

    def test_is_prime(self):

        self.assertTrue(mymodule.is_prime(5))

        self.assertFalse(mymodule.is_prime(6))

    def test_sum(self):

        self.assertEqual(mymodule.sum(5, 7), 12)

        with self.assertRaises(TypeError):

            mymodule.sum(5, "Python")
if __name__ == "__main__":

    unittest.main()
```

Instalación y mantenimiento

En esta fase buscamos que prevalezca el software en el tiempo, para esto se programara manteamiento sobre la base de datos que se utilizara para almacenar los datos de los clientes y las transacciones que realice cada cliente.