Máster Universitario en Ingeniería Informática

Tratamiento Inteligente de Datos

**PRÁCTICA 4: Reglas de Asociación**

Préstamo


UNIVERSIDAD DE GRANADA


Carlos Santiago Sánchez Muñoz

Grupo de prácticas 1 - Jueves

*Email:* carlossamu7@correo.ugr.es

*10 de diciembre de 2020*

# Índice

# 1. Descripción del problema

UniversalBank es un banco relativamente joven que está creciendo rápidamente en términos de captación de clientes. La mayoría de ellos son clientes de pasivo (es decir, clientes con predominio de contratos de productos financieros de ahorro e inversión y, por tanto, aquellos sobre los que el banco tiene una serie de obligaciones de pago) con diferentes tamaños de relación con el banco. La base de clientes de activo (es decir, aquellos con predominio de contratos de productos financieros de financiación y, por tanto, clientes sobre los que el banco tiene una serie de derechos de cobro) es bastante pequeña y el banco quiere crecer esta base rápidamente para traer más negocios de préstamos. En concreto, el banco quiere estudiar la manera de convertir sus clientes de pasivo en clientes de activo.

El año pasado, el banco desarrolló una campaña para clientes de pasivo con la que se consiguió una conversión del 9 %. Esto ha alentado al departamento de marketing para diseñar campañas más inteligentes y con un objetivo mejor definido. En concreto, se persigue responder a la siguiente cuestión: Hay varios productos/servicios que el banco ofrece tales como cuentas de valores (securities accounts), certificados de depósitos, servicios de banca en línea o tarjetas de crédito. ¿Se puede detectar alguna relación entre estos productos para encontrar oportunidades de venta cruzada? Para responder a la pregunta se propone obtener reglas de asociación mediante alguno de los algoritmos de generación de reglas.

Se comenzará aplicando un preprocesamiento de datos ya que existen atributos que no son nominales, o que no parecen interesantes para este problema.

La lectura del conjunto de datos sería la siguiente:

```
""" Lectura de datos. Devuelve el dataframe.
"""
def read_data():
    df = pd.read_excel('prestamo.xls', sheet_name='datos')

    if(IMPRIME_INFO):
        print("\nInformacion del dataframe:")
        print(df.info())
        print("\nDataframe:")
        print(df)
    return df
```

## 2. Preprocesamiento

Esta sección está dedicada al preprocesamiento de los datos. En un primer lugar, se hará un pequeño estudio de la distribución de los datos. A continuación se realizará un tratamiento muy particular para la técnica de obtención de reglas de asociación.

### 2.1. Distribución de los datos

Comenzamos observando los datos y el número de instancias que hay de cada tipo en cada atributo. Para ello se puede usar `df.groupby(["NombreAtributo"]).size()`. Si queremos información más detallada (media, cuantiles, etc.) que también puede ser muy útil de cara a tomar decisiones acerca del tratamiento de los datos se puede usar `df["NombreAtributo"].describe()`.

La base de datos recién leída y sin manipular es la siguiente:

Dataframe:

| | Age | Experience | Income | ZIP Code | Family | CCAvg | Education | Mortgage | Personal Loan | Securities Account | CD Account | Online | CreditCard |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 25 | 1 | 49 | 91107 | 4 | 1.6 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 45 | 19 | 34 | 90089 | 3 | 1.5 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2 | 39 | 15 | 11 | 94720 | 1 | 1.0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 35 | 9 | 100 | 94112 | 1 | 2.7 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 35 | 8 | 45 | 91330 | 4 | 1.0 | 2 | 0 | 0 | 0 | 0 | 0 | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 4995 | 29 | 3 | 40 | 92697 | 1 | 1.9 | 3 | 0 | 0 | 0 | 0 | 1 | 0 |
| 4996 | 30 | 4 | 15 | 92037 | 4 | 0.4 | 1 | 85 | 0 | 0 | 0 | 1 | 0 |
| 4997 | 63 | 39 | 24 | 93023 | 2 | 0.3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4998 | 65 | 40 | 49 | 90034 | 3 | 0.5 | 2 | 0 | 0 | 0 | 0 | 1 | 0 |
| 4999 | 28 | 4 | 83 | 92612 | 3 | 0.8 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

Imagen 1: Conjunto de datos

Sería interesante poder observar todas las variables en conjunto. Por ejemplo podemos mostrar un histograma con todos los atributos. Sea la Imagen 2.

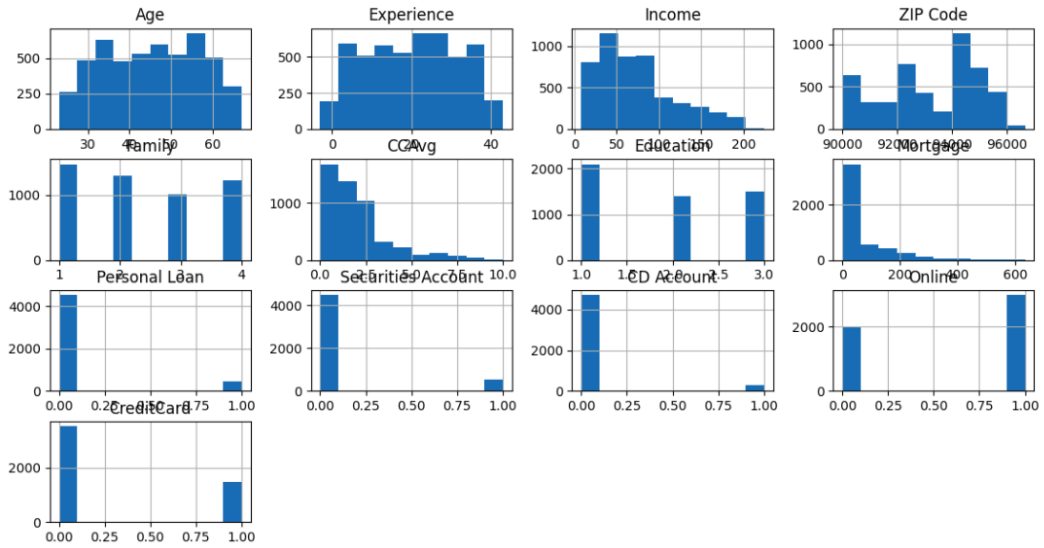


Imagen 2: Histograma

En dicho histograma se aprecia muy bien la existencia de atributos binarios como `Personal Loan`, `Securities Account`, `CD Account`, `Online` y `CreditCard`. Del mismo modeo se puede decir que los datos están más o menos balanceados. Algunos atributos son una excepción y es cierto que existe un desequilibrio en la frecuencia de clase como lo son `CCAvg` y `Mortgage`.

## 2.2. Construyendo la base de datos transaccional

Los algoritmos de extracción de reglas de asociación usan bases de datos transaccionales. Este es el primer paso que hay que hacer en nuestra base de datos. La primera función implementada es para los atributos binarios. Básicamente inserta como cadena de caracteres si posee el atributo o si por el contrario no lo posee.

```
""" Haciendo nominal la variable 'atribute' que contiene 0 y 1. Devuelve el df.
- df: dataframe.
- atribute: nombre del atributo.
"""
def nominalize_0_1(df, atribute):
    for i in range(len(df[atribute])):
        # 0 es que no posee el atributo
        if(df[atribute][i]=="0"):
            df[atribute][i] = "no" + atribute.replace(" ", "")
        # 1 es que sí que posee ese atributo
        else:
            df[atribute][i] = "has" + atribute.replace(" ", "")
    return df
```

A continuación se van a programar una serie de reglas que se encargan de preprocesar un poco cada atributo en particular y dejarlo preparado para dicha base de datos transaccional.

`Education`

Este atributo lo tratamos del siguiente modo: si es 1, `Undergraduate`; si es 2, `Graduate` y si es 3 `Advanced/Professional`.

```
""" Discretizando el atributo 'Education'. Devuelve el df.
- df: dataframe.
"""
def discretize_education(df):
    for i in range(len(df["Education"])):
        # 1 es Undergraduate
        if(df["Education"][i]=="1"):
            df["Education"][i] = "Undergraduate"
        # 2 es Graduate
        elif(df["Education"][i]=="2"):
            df["Education"][i] = "Graduated"
        # 3 es Advanced/Professional
        else:
            df["Education"][i] = "Advanced/Professional"
    return df
```

`Family`

Este atributo lo tratamos del siguiente modo: si es 1, `1Component`; si es 2, `2Component`; si es 3 `3Component` y si es 4, `4Component`.

```
""" Discretizando el atributo 'Family'. Devuelve el df.
- df: dataframe.
"""
def discretize_family(df):
    for i in range(len(df["Family"])):
        # 1 componente
        if(df["Family"][i]=="1"):
            df["Family"][i] = "1Component"
        # 2 componentes
        elif(df["Family"][i]=="2"):
            df["Family"][i] = "2Component"
        # 3 componentes
        elif(df["Family"][i]=="3"):
            df["Family"][i] = "3Component"
        # 4 componentes
        else:
            df["Family"][i] = "4Component"
    return df
```

`Age`

Este atributo lo tratamos del siguiente modo: si está en el intervalo $[20, 30)$, `Twenties`; si está en $[30, 40)$, `Thirties`; si está en $[40, 50)$, `Forties`; si está en $[50, 60)$, `Fifties` y si está en $[60, 70)$, `Fifties`. Para esta decisión se ha tenido en cuenta la frecuencia de clase.

```
""" Discretizando el atributo 'Age'. Devuelve el df.
- df: dataframe.
"""
def discretize_age(df):
    for i in range(len(df["Age"])):
        # Twenties
        if(int(df["Age"][i])>=20 and int(df["Age"][i])<30):
            df["Age"][i] = "Twenties"
        # Thirties
        elif(int(df["Age"][i])>=30 and int(df["Age"][i])<40):
            df["Age"][i] = "Thirties"
        # Forties
        elif(int(df["Age"][i])>=40 and int(df["Age"][i])<50):
            df["Age"][i] = "Forties"
        # Fifties
        elif(int(df["Age"][i])>=50 and int(df["Age"][i])<60):
            df["Age"][i] = "Fifties"
        # Sixties
        else:
            df["Age"][i] = "Sixties"
    return df
```

`Mortgage`

Este atributo lo tratamos del siguiente modo: si es 0, `NoMortgage`; si está en $(0, 100]$, `VeryLowMortgage`; si está en $(100, 200]$, `LowMortgage`; si está en $(200, 300]$, `MediumMortgage`; si está en $(300, 400]$, `HighMortgage` y si está en $(400, +\infty]$, `VeryLowMortgage`.

```
""" Discretizando el atributo 'Mortgage'. Devuelve el df.
- df: dataframe.
"""
def discretize_mortgage(df):
    for i in range(len(df["Mortgage"])):
        # NoMortgage
        if(df["Mortgage"][i]=="0"):
            df["Mortgage"][i] = "NoMortgage"
        # VeryLowMortgage
        elif(int(df["Mortgage"][i])>0 and int(df["Mortgage"][i])<=100):
            df["Mortgage"][i] = "VeryLowMortgage"
        # LowMortgage
        elif(int(df["Mortgage"][i])>100 and int(df["Mortgage"][i])<=200):
            df["Mortgage"][i] = "LowMortgage"
        # MediumMortgage
        elif(int(df["Mortgage"][i])>200 and int(df["Mortgage"][i])<=300):
            df["Mortgage"][i] = "MediumMortgage"
        # HighMortgage
        elif(int(df["Mortgage"][i])>300 and int(df["Mortgage"][i])<=400):
            df["Mortgage"][i] = "HighMortgage"
        # VeryHighMortgage
        else:
            df["Mortgage"][i] = "VeryHighMortgage"
    return df
```

`CCAvg`

Este atributo lo tratamos del siguiente modo: si está en $(0, 0{,}75]$, `VeyLowCCAvg`; si está en $(0{,}75, 1{,}5]$, `LowCCAvg`; si está en $(1{,}5, 2{,}5]$, `HighCCAvg` y si está en $(2{,}5, 10]$, `VeryHighCCAvg`. Para seleccionar esos valores concretos y realizar la discretización se ha tenido en cuenta la distribución de la variable. Los cuantiles han sido una buena referencia.

```
""" Discretizando el atributo 'CCAvg'. Devuelve el df.
- df: dataframe.
"""
def discretize_CCAvg(df):
    for i in range(len(df["CCAvg"])):
        # VeryLowCCAvg
        if(float(df["CCAvg"][i])>=0 and float(df["CCAvg"][i])<0.75):
            df["CCAvg"][i] = "VeyLowCCAvg"
        # LowCCAvg
        elif(float(df["CCAvg"][i])>=0.75 and float(df["CCAvg"][i])<1.5):
            df["CCAvg"][i] = "LowCCAvg"
        # HighCCAvg
        elif(float(df["CCAvg"][i])>=1.5 and float(df["CCAvg"][i])<2.5):
            df["CCAvg"][i] = "HighCCAvg"
        # VeryHighCCAvg
        else:
            df["CCAvg"][i] = "VeryHighCCAvg"
    return df
```

`Income`

Este atributo lo tratamos del siguiente modo: si está en $[0, 40)$, `VeryLowIncome`; si está en $[40, 65)$, `LowIncome`; si está en $[65, 100)$, `HighIncome` y en otro caso `VeryHighIncome`. Para seleccionar esos valores concretos y realizar la discretización también se ha tenido en cuenta la distribución de la variable y en particular los cuartiles.

```
""" Discretizando el atributo 'Mortgage'. Devuelve el df.
- df: dataframe.
"""
def discretize_Income(df):
    for i in range(len(df["Income"])):
        # VerLowIncome
        if(int(df["Income"][i])>=0 and int(df["Income"][i])<40):
            df["Income"][i] = "VeryLowIncome"
        # LowIncome
        elif(int(df["Income"][i])>=40 and int(df["Income"][i])<65):
                df["Income"][i] = "LowIncome"
        # HighIncome
        elif(int(df["Income"][i])>=65 and int(df["Income"][i])<100):
                df["Income"][i] = "HighIncome"
        # VeryHighIncome
        else:
            df["Income"][i] = "VeryHighIncome"
    return df
```

El preprocesamiento global quedaría así:

```
""" Todo el preprocesado. Devuelve el df preprocesado.
- df: dataframe.
"""
def preprocesamiento(df):
    correlaciones = df.corr()
    df = df.astype(str)
    print("\n---- PREPROCESAMIENTO ----")
    print("Descarto la variable 'Experience' ya que tiene una correlacion de {}"
        .format(correlaciones['Age'].sort_values(ascending=False)[1]))
    df = df.drop(columns = ['Experience'])
    df = df.drop(columns = ['ZIP Code'])
    print("Nominalizando 'Personal Loan'")
    df = nominalize_0_1(df, "Personal Loan")
    print("Nominalizando 'Securities Account'")
    df = nominalize_0_1(df, "Securities Account")
    print("Nominalizando 'CD Account'")
    df = nominalize_0_1(df, "CD Account")
    print("Nominalizando 'CreditCard'")
    df = nominalize_0_1(df, "CreditCard")
    print("Nominalizando 'Online'")
    df = nominalize_0_1(df, "Online")
    print("Discretizando 'Education'")
    df = discretize_education(df)
    print("Discretizando 'Family'")
    df = discretize_family(df)
    print("Discretizando 'Age'")
    df = discretize_age(df)
    print("Discretizando 'Mortgage'")
    df = discretize_mortgage(df)
    print("Discretizando 'CCAvg'")
    df = discretize_CCAvg(df)
    print("Discretizando 'Income'")
    df = discretize_Income(df)
    print()
    print(df)
    return df
```

La ejecución de todo el preprocesamiento es la siguiente:

```
[5000 rows x 13 columns]

---  PREPROCESAMIENTO  ---
Descarto la variable 'Experience' ya que tiene una correlación de 0.9942148569683271
Nominalizando 'Personal Loan'
Nominalizando 'Securities Account'
Nominalizando 'CD Account'
Nominalizando 'CreditCard'
Nominalizando 'Online'
Discretizando 'Education'
Discretizando 'Family'
Discretizando 'Age'
Discretizando 'Mortgage'
Discretizando 'CCAvg'
Discretizando 'Income'
```

Imagen 3: Preprocesamiento

Los resultados que deja sobre el conjunto de datos, son considerables. Nuestro dataset es el de la Imagen 4 ahora.

| | Age | Income | Family | CCAvg | Education | ... | Personal Loan | Securities Account | CD Account | Online | CreditCard |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Twenties | LowIncome | 4Component | HighCCAvg | Undergraduate | ... | noPersonalLoan | hasSecuritiesAccount | noCDAccount | noOnline | noCreditCard |
| 1 | Forties | VeryLowIncome | 3Component | HighCCAvg | Undergraduate | ... | noPersonalLoan | hasSecuritiesAccount | noCDAccount | noOnline | noCreditCard |
| 2 | Thirties | VeryLowIncome | 1Component | LowCCAvg | Undergraduate | ... | noPersonalLoan | noSecuritiesAccount | noCDAccount | noOnline | noCreditCard |
| 3 | Thirties | VeryHighIncome | 1Component | VeryHighCCAvg | Graduated | ... | noPersonalLoan | noSecuritiesAccount | noCDAccount | noOnline | noCreditCard |
| 4 | Thirties | LowIncome | 4Component | LowCCAvg | Graduated | ... | noPersonalLoan | noSecuritiesAccount | noCDAccount | noOnline | hasCreditCard |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 4995 | Twenties | LowIncome | 1Component | HighCCAvg | Advanced/Professional | ... | noPersonalLoan | noSecuritiesAccount | noCDAccount | hasOnline | noCreditCard |
| 4996 | Thirties | VeryLowIncome | 4Component | VeyLowCCAvg | Undergraduate | ... | noPersonalLoan | noSecuritiesAccount | noCDAccount | hasOnline | noCreditCard |
| 4997 | Sixties | VeryLowIncome | 2Component | VeyLowCCAvg | Advanced/Professional | ... | noPersonalLoan | noSecuritiesAccount | noCDAccount | noOnline | noCreditCard |
| 4998 | Sixties | LowIncome | 3Component | VeyLowCCAvg | Graduated | ... | noPersonalLoan | noSecuritiesAccount | noCDAccount | hasOnline | noCreditCard |
| 4999 | Twenties | HighIncome | 3Component | LowCCAvg | Undergraduate | ... | noPersonalLoan | noSecuritiesAccount | noCDAccount | hasOnline | hasCreditCard |

Imagen 4: Conjunto de datos transaccional

## 3. Reglas de asociación

Esta sección está dedicada a la implementación del modelo de extracción de las reglas de asociación. Se va a usar el conocido algoritmo apriori. En Python existen varias librerías que tienen dicha función. Tras realizar pruebas la que mejor encaja con lo que busco y muestra de manera cómoda las reglas halladas es la del paquete `mlxtend`.

La documentación consultada para el correcto uso de esta función está disponible en este enlace. Del mismo modo en este otro enlace hay algunos ejemplos de su uso. Se ha implementado una función `get_rules` que devuelve las reglas halladas. Obsérvese a continuación:

```
""" Reglas de asociacion. Devuelve las reglas.
- df: dataframe.
"""
def get_rules(df):
    print("\n--- REGLAS DE ASOCIACION ---")
    records = []
    for i in range(df.shape[0]):
        records.append([str(df.values[i,j]) for j in range(df.shape[1])])

    te = TransactionEncoder()
    te_ary = te.fit(records).transform(records)
    df = pd.DataFrame(te_ary, columns=te.columns_)

    print("\nDataframe preparado para extraer las reglas de asociacion:")
    print(df)
    frequent_itemsets = apriori(df, min_support=SUPP, use_colnames=True)
    rules = association_rules(frequent_itemsets, metric="confidence",
        min_threshold=CONF)
    rules = pd.DataFrame(rules)

    to_delete = []
    for i in range(rules.shape[0]):
        if(len(rules["antecedents"][i])>3 or len(rules["consequents"][i])>3):
            to_delete.append(i)

    print("Indices de las reglas que tienen mas de 3 antecedentes o consecuentes: ")
    print(to_delete)
    rules = rules.drop(to_delete, axis=0)
    rules = rules.reset_index()
    del rules['index']

    return rules
```

Tal y como se puede apreciar el primer paso es una manipulación sobre el conjunto de datos. Dicha manipulación separa las variables y ahora el contenido es `True/False`.

```
Dataframe preparado para extraer las reglas de asociación:
      1Component  2Component  3Component  4Component  Advanced/Professional  ...  noCDAccount  noCreditCard  noOnline  noPersonalLoan  noSecuritiesAccount
0          False       False       False        True                  False  ...         True          True      True            True                False
1          False       False        True       False                  False  ...         True          True      True            True                False
2           True       False       False       False                  False  ...         True          True      True            True                 True
3           True       False       False       False                  False  ...         True          True      True            True                 True
4          False       False       False        True                  False  ...         True         False      True            True                 True
...          ...         ...         ...         ...                    ...  ...          ...           ...       ...             ...                  ...
4995        True       False       False       False                   True  ...         True          True     False            True                 True
4996       False       False       False        True                  False  ...         True          True     False            True                 True
4997       False        True       False       False                   True  ...         True          True      True            True                 True
4998       False       False        True       False                  False  ...         True          True     False            True                 True
4999       False       False        True       False                  False  ...         True         False     False            True                 True

[5000 rows x 36 columns]
```

Imagen 5: Conjunto de datos preparado para extraer las reglas

La primera pregunta que hay que hacerse es si tiene sentido que en dicha base de datos esté una característica y su negación como por ejemplo `hasPersonalLoan` y `noPersonalLoan` entre otras. Por el tipo de funcionamiento del algoritmo apriori de dicha librería sí. Eso me permite obtener reglas que involucren a ambos parámetros. No obstante más adelante se realizarán pruebas con uno sólo de ellos.

Para poder observar las reglas de asociación de manera apropiada se han guardado en un fichero `Reglas.csv`, después de la ejecución de dicha función. En la siguiente sección se exponen los resultados y se comentan las reglas obtenidas.

# 4. Resultados

Los primeros resultados que se van a exponer en esta sección son aquellos en los que se usa la totalidad de los atributos. La cantidad de reglas obtenidas depende fundamentalmente de dos parámetros del algoritmo: el soporte mínimo y la confianza mínima. El primero tiene que ver que el porcentaje de veces que se exige que aparezca el antecedente y el segundo con la confianza de la regla.

Normalmente, el valor del soporte suele estar entre el 5 % y el 15 % por ciento, y la confianza suele ser superior al 80 %. Se han definido dos contantes globales `SUPP = 0.15` y `CONF=0.99`. En este caso las condiciones son más exigentes debido a la multitud de reglas existentes.

REGLAS DE ASOCIACIÓN (support>0.15, confidence>0.99):

| | antecedents | consequents | antecedent support | ... | lift | leverage | conviction |
|---|---|---|---|---|---|---|---|
| 0 | (LowIncome) | (noPersonalLoan) | 0.2476 | ... | 1.104408 | 0.023370 | 59.424000 |
| 1 | (VeryLowIncome) | (noPersonalLoan) | 0.2624 | ... | 1.106195 | 0.025190 | inf |
| 2 | (noOnline) | (noCDAccount) | 0.4032 | ... | 1.054252 | 0.020553 | 6.408758 |
| 3 | (noCreditCard, HighCCAvg) | (noCDAccount) | 0.1848 | ... | 1.057372 | 0.009962 | 9.301600 |
| 4 | (LowIncome, NoMortgage) | (noPersonalLoan) | 0.1706 | ... | 1.106195 | 0.016378 | inf |
| .. | ... | ... | ... | ... | ... | ... | ... |
| 74 | (noCreditCard, noPersonalLoan, noSecuritiesAcc... | (noCDAccount) | 0.5714 | ... | 1.064283 | 0.034513 | inf |
| 75 | (noOnline, noPersonalLoan, noSecuritiesAccount) | (noCDAccount) | 0.3290 | ... | 1.064283 | 0.019872 | inf |
| 76 | (LowIncome, noCreditCard, noSecuritiesAccount) | (noPersonalLoan, noCDAccount) | 0.1590 | ... | 1.144429 | 0.020016 | 51.039000 |
| 77 | (NoMortgage, VeryLowIncome, noSecuritiesAccount) | (noPersonalLoan, noCDAccount) | 0.1596 | ... | 1.137251 | 0.019093 | 14.637600 |
| 78 | (noCreditCard, VeryLowIncome, noSecuritiesAcco... | (noPersonalLoan, noCDAccount) | 0.1644 | ... | 1.147315 | 0.021109 | inf |

[79 rows x 9 columns]

Imagen 6: Reglas de asociación extraídas por el algoritmo

Tal y como se observa hay 78 reglas. A partir de la tabla `csv` se van a exponer aquí algunas de las más relevantes para hacer comentarios diversos. Se ha filtrado de manera que se descartaron aquellas reglas en las que había más de tres antecedentes. Como el número de reglas sigue siendo muy extenso, vamos a seleccionar aquellas con un máximo de dos antecedentes.

En la Tabla 1 se observan algunas reglas que tienen mucha lógica. Por ejemplo, algunos comentarios pueden ser:

- Las dos primeras dicen que las personas que tienen bajos o muy bajos ingresos no tienen una cuenta personal.
- Del mismo modo aquellas personas que no tienen tarjeta de crédito y tienen bajos ingresos no poseen un certificado de depósito.
- Aquellos que no tienen hipoteca y tienen ingresos bajos no tienen cuenta personal.
- Los que no tienen hipoteca y no usan el online no tienen certificado de depósito.

Hay que hacer la lectura y contrapartida a estas reglas de la que el banco podría beneficiarse. Una propuesta en función de la última regla explicada sería que si se quisiera aumentar el número de certificados de depósito debería fomentarse el online y facilitar las hipotecas.

| Antecedentes | Consecuentes | Soporte | Confianza |
|---|---|---|---|
| {'LowIncome'} | {'noPersonalLoan'} | 0,248 | 0,998 |
| {'VeryLowIncome'} | {'noPersonalLoan'} | 0,262 | 1,000 |
| {'noOnline'} | {'noCDAccount'} | 0,403 | 0,991 |
| {'noCreditCard', 'HighCCAvg'} | {'noCDAccount'} | 0,185 | 0,994 |
| {'NoMortgage', 'LowIncome'} | {'noPersonalLoan'} | 0,171 | 1,000 |
| {'noCreditCard', 'LowIncome'} | {'noCDAccount'} | 0,178 | 0,996 |
| {'noCDAccount', 'LowIncome'} | {'noPersonalLoan'} | 0,238 | 0,998 |
| {'noCreditCard', 'LowIncome'} | {'noPersonalLoan'} | 0,178 | 0,998 |
| {'noSecuritiesAccount', 'LowIncome'} | {'noPersonalLoan'} | 0,221 | 0,998 |
| {'NoMortgage', 'VeryLowIncome'} | {'noPersonalLoan'} | 0,178 | 1,000 |
| {'NoMortgage', 'noOnline'} | {'noCDAccount'} | 0,278 | 0,992 |
| {'noOnline', 'Undergraduate'} | {'noCDAccount'} | 0,168 | 0,992 |
| {'hasOnline', 'VeryLowIncome'} | {'noPersonalLoan'} | 0,154 | 1,000 |
| {'noCreditCard', 'VeryLowIncome'} | {'noCDAccount'} | 0,183 | 0,993 |
| {'noCDAccount', 'VeryLowIncome'} | {'noPersonalLoan'} | 0,253 | 1,000 |
| {'noCreditCard', 'VeryLowIncome'} | {'noPersonalLoan'} | 0,183 | 1,000 |
| {'noSecuritiesAccount', 'VeryLowIncome'} | {'noPersonalLoan'} | 0,237 | 1,000 |
| {'noOnline', 'noCreditCard'} | {'noCDAccount'} | 0,286 | 0,998 |
| {'noPersonalLoan', 'noCreditCard'} | {'noCDAccount'} | 0,639 | 0,995 |
| {'noSecuritiesAccount', 'noCreditCard'} | {'noCDAccount'} | 0,630 | 0,996 |
| {'noPersonalLoan', 'noOnline'} | {'noCDAccount'} | 0,365 | 0,997 |
| {'noOnline', 'noSecuritiesAccount'} | {'noCDAccount'} | 0,363 | 0,998 |
| {'noCreditCard', 'LowIncome'} | {'noPersonalLoan', 'noCDAccount'} | 0,178 | 0,993 |
| {'noCreditCard', 'VeryLowIncome'} | {'noPersonalLoan', 'noCDAccount'} | 0,183 | 0,993 |

Tabla 1: Resultados todas las reglas con máximo de dos atributos

Ahora se va a realizar un nuevo experimento. Se van a borrar los atributos que comienzan por `'no'` y se deja sólo su versión positiva. Se baja el soporte mínimo a $0,07$ y la confianza a $0,7$. Las reglas son:

REGLAS DE ASOCIACIÓN (support>0.07, confidence>0.7):

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction |
|---|---|---|---|---|---|---|---|---|---|
| 0 | (1Component) | (NoMortgage) | 0.2944 | 0.6924 | 0.2078 | 0.705842 | 1.019414 | 0.003957 | 1.045698 |
| 1 | (3Component) | (NoMortgage) | 0.2020 | 0.6924 | 0.1416 | 0.700990 | 1.012406 | 0.001735 | 1.028728 |
| 2 | (Forties) | (NoMortgage) | 0.2514 | 0.6924 | 0.1768 | 0.703262 | 1.015687 | 0.002731 | 1.036604 |
| 3 | (HighCCAvg) | (NoMortgage) | 0.2616 | 0.6924 | 0.1850 | 0.707187 | 1.021355 | 0.003868 | 1.050498 |
| 4 | (HighIncome) | (NoMortgage) | 0.2456 | 0.6924 | 0.1734 | 0.706026 | 1.019679 | 0.003347 | 1.046351 |
| 5 | (Sixties) | (NoMortgage) | 0.1348 | 0.6924 | 0.0944 | 0.700297 | 1.011405 | 0.001064 | 1.026349 |
| 6 | (VeryHighCCAvg) | (NoMortgage) | 0.2710 | 0.6924 | 0.1906 | 0.703321 | 1.015773 | 0.002960 | 1.036811 |
| 7 | (hasPersonalLoan) | (VeryHighCCAvg) | 0.0960 | 0.2710 | 0.0728 | 0.758333 | 2.798278 | 0.046784 | 3.016552 |
| 8 | (hasPersonalLoan) | (VeryHighIncome) | 0.0960 | 0.2444 | 0.0878 | 0.914583 | 3.742158 | 0.064338 | 8.846049 |
| 9 | (1Component, Undergraduate) | (NoMortgage) | 0.1356 | 0.6924 | 0.0978 | 0.721239 | 1.041651 | 0.003911 | 1.103454 |
| 10 | (1Component, hasOnline) | (NoMortgage) | 0.1766 | 0.6924 | 0.1254 | 0.710079 | 1.025533 | 0.003122 | 1.060980 |
| 11 | (Forties, hasOnline) | (NoMortgage) | 0.1464 | 0.6924 | 0.1050 | 0.717213 | 1.035836 | 0.003633 | 1.087745 |
| 12 | (HighCCAvg, hasOnline) | (NoMortgage) | 0.1512 | 0.6924 | 0.1066 | 0.705026 | 1.018236 | 0.001909 | 1.042805 |
| 13 | (HighIncome, hasOnline) | (NoMortgage) | 0.1482 | 0.6924 | 0.1064 | 0.717949 | 1.036899 | 0.003786 | 1.090582 |
| 14 | (VeryHighCCAvg, Undergraduate) | (NoMortgage) | 0.1542 | 0.6924 | 0.1096 | 0.710765 | 1.026524 | 0.002832 | 1.063496 |
| 15 | (VeryHighIncome, Undergraduate) | (NoMortgage) | 0.1512 | 0.6924 | 0.1068 | 0.706349 | 1.020146 | 0.002109 | 1.047503 |
| 16 | (hasCreditCard, Undergraduate) | (NoMortgage) | 0.1266 | 0.6924 | 0.0898 | 0.709321 | 1.024438 | 0.002142 | 1.058211 |
| 17 | (VeryHighCCAvg, hasOnline) | (NoMortgage) | 0.1624 | 0.6924 | 0.1144 | 0.704433 | 1.017379 | 0.001954 | 1.040713 |
| 18 | (VeryHighCCAvg, VeryHighIncome, Undergraduate) | (NoMortgage) | 0.0992 | 0.6924 | 0.0712 | 0.717742 | 1.036600 | 0.002514 | 1.089783 |

Imagen 7: Reglas de asociación extraídas por el algoritmo después de borrar atributos

En la Tabla 2 se pueden observar mejor las nuevas reglas obtenidas:

| **Antecedentes** | **Consecuentes** | **Soporte** | **Confianza** |
|---|---|---|---|
| {'1Component'} | {'NoMortgage'} | 0,294 | 0,706 |
| {'3Component'} | {'NoMortgage'} | 0,202 | 0,701 |
| {'Forties'} | {'NoMortgage'} | 0,251 | 0,703 |
| {'HighCCAvg'} | {'NoMortgage'} | 0,262 | 0,707 |
| {'HighIncome'} | {'NoMortgage'} | 0,246 | 0,706 |
| {'Sixties'} | {'NoMortgage'} | 0,135 | 0,700 |
| {'VeryHighCCAvg'} | {'NoMortgage'} | 0,271 | 0,703 |
| {'hasPersonalLoan'} | {'VeryHighCCAvg'} | 0,960 | 0,758 |
| {'hasPersonalLoan'} | {'VeryHighIncome'} | 0,960 | 0,915 |
| {'1Component', 'Undergraduate'} | {'NoMortgage'} | 0,136 | 0,721 |
| {'hasOnline', '1Component'} | {'NoMortgage'} | 0,177 | 0,710 |
| {'hasOnline', 'Forties'} | {'NoMortgage'} | 0,146 | 0,717 |
| {'hasOnline', 'HighCCAvg'} | {'NoMortgage'} | 0,151 | 0,705 |
| {'HighIncome', 'hasOnline'} | {'NoMortgage'} | 0,148 | 0,718 |
| {'VeryHighCCAvg', 'Undergraduate'} | {'NoMortgage'} | 0,154 | 0,711 |
| {'VeryHighIncome', 'Undergraduate'} | {'NoMortgage'} | 0,151 | 0,706 |
| {'Undergraduate', 'hasCreditCard'} | {'NoMortgage'} | 0,127 | 0,709 |
| {'VeryHighCCAvg', 'hasOnline'} | {'NoMortgage'} | 0,162 | 0,704 |

Tabla 2: Resultados reglas filtradas con máximo de dos atributos

Encontramos reglas muy interesantes y muy distintas a las ya vistas por lo que hacer el filtrado y volver a ejecutar el algoritmo ha sido muy provechoso. Se analizan a continuación algunas de ellas:

- Las familias que solo tienen un componente no tienen hipoteca. Igual con las de tres componentes.
- Las familias que gastan en media mucho con la tarjeta de crédito no tienen hipoteca, lo cual es coherente ya que si estás hipotecado sueles gastar menos.
- Los que tienen una cuenta personal tienen ingresos muy altos y también tienen un gasto medio de tarjeta muy alto.

Si nos fijamos la mayoría de los consecuentes de estas reglas están relacionados con las hipotecas. Concluimos resaltando la buena funcionalidad de los algoritmos de extracción de reglas de asociación, los cuales necesitan trabajar sobre bases de datos transaccionales pero son relativamente eficientes en el cálculo. Aunque el filtrado y manipulación de las reglas para hacerlas accesibles al humano es una desventaja ya que es tedioso la provechosidad de las mismas le otorga a esta técnica un gran potencial. Se ha visto en esta práctica que para el caso concreto de un banco nos proporciona información relevante y de calidad.