Semana 3 - Actividad 1

Ciencia y analítica de datos

Instituto Tecnológico y de Estudios Superiores de Monterrey

Equipo 117

Profesor Jobish Vallikavungal Devassia

José Daniel Camacho Torres - A01793555

Yocks Sandoval Jaik Randy - A01793725

4/10/2022

Parte 1: Fundamentos de bases de datos

Desde un inicio, el manejo y procesamiento de la información se ha convertido en el reto más importante para los especialistas e ingenieros de datos, las herramientas computacionales marcaron una revolución para dicho análisis de datos.

No obstante, toda persona que quiera resolver un problema, directamente se ve implicada a aplicar un análisis de datos, por ello debe conocer las fases que con lleva. Generar información sin un orden lógico y correcta selección significa acumular mucha información importante sin productividad alguna. No es solo generar y acumular, sino que conlleva una recopilación selectiva, limpieza, ordenamiento y depósito de datos.

Ese flujo de colectar de forma organizada los datos, corresponde a una aplicación cíclica que promueve usar los datos de manera efectiva para tomar decisiones. Poder condensar una base de datos y que sea funcional, se debe llevar a cabo un tratamiento y transformación de datos, para poder llevarla al siguiente nivel como lo son los almacenes de datos. Estos conceptos se relacionan entre sí, cada vez que generamos una colección autodescriptiva de registros integrados obtenemos una base de datos; a medida que esta crece, se potencializa y se llega a interconectar con otras bases de datos, se obtiene, por ende, un almacén de datos bien constituido.

> Como se mencionó anteriormente, es necesario que una herramienta de software ayude a manejar grandes volúmenes de datos y otra que se encargue de su

> almacenamiento. Esta información se distribuye en tablas de datos, en filas y columnas de manera estandarizada. Esto permite una emisión clara, redundancia controlada y una mejor integridad de los datos. Los fundamentos de las bases y los almacenes de datos, ponen a nuestra disposición hoy día instrumentos como los metadatos, virtualización de almacenamiento y la alta eficiencia empresarial, para la solución moderna de problemas que con alto beneficio.

Parte 2: Selección y limpieza de los Datos en Python

Importación de las librerias pandas y numpy con las que vamos a trabajar para realizar la limpieza de nuestros datos.

```
import pandas as pd
import numpy as np
```

Creamos un nuevo objeto DataFrame a partir del archivo CSV proporsionado para la actividad

```
# Se utiliza la función de pandas 'read_csv()' a la cual le enviamos como parametro al direcc
dataFrame = pd.read_csv('https://raw.githubusercontent.com/PosgradoMNA/Actividades_Aprendizaj

#Creamos una copia del nuevo DF para realizar comparaciones mas adelante
df = dataFrame.copy()

df # Se muestra el DataFrame creado
```

	ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	X15	X16	X
0	1	20000	2.0	2.0	1.0	24.0	2.0	2.0	-1.0	-1.0	...	0.0	0.0	

Renombramos las columnas de DF segun su documentación:

X1: Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.

X2: Gender (1 = male; 2 = female).

X3: Education (1 = graduate school; 2 = university; 3 = high school; 4 = others).

X4: Marital status (1 = married; 2 = single; 3 = others).

X5: Age (year).

X6 - X11: History of past payment. We tracked the past monthly payment records (from April to September, 2005) as follows: X6 = the repayment status in September, 2005; X7 = the repayment status in August, 2005; . . .;X11 = the repayment status in April, 2005. The measurement scale for the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.

X12-X17: Amount of bill statement (NT dollar). X12 = amount of bill statement in September, 2005; X13 = amount of bill statement in August, 2005; . . .; X17 = amount of bill statement in April, 2005.

X18-X23: Amount of previous payment (NT dollar). X18 = amount paid in September, 2005; X19 = amount paid in August, 2005; . . .;X23 = amount paid in April, 2005.

```
df.columns = ['ID','monto', 'genero', 'educacion', 'estado_civil', 'edad', 'historial_pago_se
              'historial_pago_jun', 'historial_pago_may', 'historial_pago_abr', 'monto_factur
              'monto_factura_jun', 'monto_factura_may', 'monto_factura_abr', 'monto_pagado_se
              'monto_pagado_jun', 'monto_pagado_may', 'monto_pagado_abr', 'Y']
```

Se realiza validación de datos nulos

```
if df.isnull().values.any(): print('El data frame contiene valores nulos (null)')
```

```
El data frame contiene valores nulos (null)
```

Revisamos donde se encuentran los valores nulos

```
# Muestra las columnas que contienen valores nulos
df.isnull().any()
```

```
ID                  False
```

```
monto                False
genero                True
educacion             True
estado_civil          True
edad                  True
historial_pago_sep    True
historial_pago_ago    True
historial_pago_jul    True
historial_pago_jun    True
historial_pago_may    True
historial_pago_abr    True
monto_factura_sep     True
monto_factura_ago     True
monto_factura_jul     True
monto_factura_jun     True
monto_factura_may     True
monto_factura_abr     True
monto_pagado_sep      True
monto_pagado_ago      True
monto_pagado_jul      True
monto_pagado_jun      True
monto_pagado_may      True
monto_pagado_abr      True
Y                     True
dtype: bool
```

Se realiza validación de datos no válidos

```
if df.isna().values.any(): print('El data frame contiene valores no válidos (NaN, NA)')
```

```
El data frame contiene valores no válidos (NaN, NA)
```

Revisamos donde se encuentran los valores no válidos

```
# Muestra las columnas que contienen valores no válidos
df.isna().any()
```

```
ID                   False
monto                False
genero                True
educacion             True
estado_civil          True
edad                  True
historial_pago_sep    True
historial_pago_ago    True
historial_pago_jul    True
historial_pago_jun    True
historial_pago_may    True
historial_pago_abr    True
monto_factura_sep     True
monto_factura_ago     True
monto_factura_jul     True
```

```
monto_factura_jun      True
monto_factura_may      True
monto_factura_abr      True
monto_pagado_sep       True
monto_pagado_ago       True
monto_pagado_jul       True
monto_pagado_jun       True
monto_pagado_may       True
monto_pagado_abr       True
Y                      True
dtype: bool
```

El siguiente paso es decidir si eliminamos las filas con valores nulos y no váidos o imputar valores con la media o mediana de cada columna.

Como primera opción se puede validar cuantas filas con valores nulos o no válidos existen para definir el impacto que tiene eliminarlas.

```
# En este caso, eliminaremos las filas con valores no válidos o nulos para validar cuantas so
# Primero realizamos una copia del DF para trabajar sobre el
df_n = df.copy()
df_n.dropna(inplace = True)

# Validamos el número de columnas eliminadas comparando ambos DF

print('Se elimiaron ' + str(df.shape[0] - df_n.shape[0]) + ' filas')

# Verificamos si el nuevo dataframe se ha limpiado
if not df_n.isna().values.any(): print('El data frame no contiene valores no válidos (NaN, NA
```

```
Se elimiaron 42 filas
El data frame no contiene valores no válidos (NaN, NA)
```

Se observa que solo se eliminaron 42 filas de un tontal de 30000, lo cual es un porcentaje muy pequeño y no eliminar las filas no representa problema de analisis para la base de datos.

Otro método para la limpieza de datos es imputar los valores nulos. Volvemos al DF original y realizamos nuevo procedimiento.

Una forma de rellenar los valores es obteniendo la media de cada columna y poner ese valor donde este mismo sea nulo.

Se debe tomar en cuenta que existen ciertos tipos de datos que no pueden ser promediados ya que son cualitativos, por ejemplo: cuando hablamos de la columna 'genero' sacar la media no es viable ya que solo puede haber dos, representados por el número '1' para el hombre y el número '2' para la mujer, al sacar la media nos da un valor decimal entre 1 y 2, cuando el valor solo puede ser 1 o 2 exactamente.

Para esto es mejor opción utilizar el valor que es mas frecuente en las columnas o en su defecto la mediana.

A continuación se pone en práctica lo mencionado

```
# Como ejemplo tomamos la columna educación para aplicarle los diferentes procesos
print('Valores de la columna "educacion"')
print('media: ' + str(df.educacion.mean())) # La función mean nos da la media del total de la
print('mediana: ' + str(df.educacion.median())) # La función median nos da la mediana del tot
print('Mayor Valor: ' + str(df.educacion.mode())) # La función mode nos da el valor que se en

# Dependiendo del tipo de columna es la función que se debe aplicar

vGenero = df.genero.mode()[0] # mode regresa un arreglo de dos valores
vEducacion = df.educacion.mode()[0]
vEstadoCivil = df.estado_civil.mode()[0]
vEdad = df.edad.median()

vHistorialPagoAbr = df.historial_pago_abr.mean()
vHistorialPagoMay = df.historial_pago_may.mean()
vHistorialPagoJun = df.historial_pago_jun.mean()
vHistorialPagoJul = df.historial_pago_jul.mean()
vHistorialPagoAgo = df.historial_pago_ago.mean()
vHistorialPagoSep = df.historial_pago_sep.mean()

vMontoFacturAbr = df.monto_factura_abr.mean()
vMontoFacturMay = df.monto_factura_may.mean()
vMontoFacturJun = df.monto_factura_jun.mean()
vMontoFacturJul = df.monto_factura_jul.mean()
vMontoFacturAgo = df.monto_factura_ago.mean()
vMontoFacturSep = df.monto_factura_sep.mean()

vMontoPagadoAbr = df.monto_pagado_abr.mean()
vMontoPagadoMay = df.monto_pagado_may.mean()
vMontoPagadoJun = df.monto_pagado_jun.mean()
vMontoPagadoJul = df.monto_pagado_jul.mean()
vMontoPagadoAgo = df.monto_pagado_ago.mean()
vMontoPagadoSep = df.monto_pagado_sep.mean()
```

```
Valores de la columna "educacion"
media: 1.8530568704580306
mediana: 2.0
Mayor Valor: 0    2.0
dtype: float64
```

Una vez que tenemos los valores correspondientes a cada columna, podemos proceder a sustituitlos en sus respectivas celdas.

```
# Con la siguiente función fillna agregamos el valor calculado a los valores nulos en cada co

df['genero'].fillna(value = vGenero, inplace = True)
df['educacion'].fillna(value = vEducacion, inplace = True)
df['estado_civil'].fillna(value = vEstadoCivil, inplace = True)
df['edad'].fillna(value = vEdad, inplace = True)

df['historial_pago_abr'].fillna(value = vHistorialPagoAbr, inplace = True)
df['historial_pago_may'].fillna(value = vHistorialPagoMay, inplace = True)
df['historial_pago_jun'].fillna(value = vHistorialPagoJun, inplace = True)
df['historial_pago_jul'].fillna(value = vHistorialPagoJul, inplace = True)
df['historial_pago_ago'].fillna(value = vHistorialPagoAgo, inplace = True)
df['historial_pago_sep'].fillna(value = vHistorialPagoSep, inplace = True)

df['monto_factura_abr'].fillna(value = vMontoFacturAbr, inplace = True)
df['monto_factura_may'].fillna(value = vMontoFacturMay, inplace = True)
df['monto_factura_jun'].fillna(value = vMontoFacturJun, inplace = True)
df['monto_factura_jul'].fillna(value = vMontoFacturJul, inplace = True)
df['monto_factura_ago'].fillna(value = vMontoFacturAgo, inplace = True)
df['monto_factura_sep'].fillna(value = vMontoFacturSep, inplace = True)

df['monto_pagado_abr'].fillna(value = vMontoPagadoAbr, inplace = True)
df['monto_pagado_may'].fillna(value = vMontoPagadoMay, inplace = True)
df['monto_pagado_jun'].fillna(value = vMontoPagadoJun, inplace = True)
df['monto_pagado_jul'].fillna(value = vMontoPagadoJul, inplace = True)
df['monto_pagado_ago'].fillna(value = vMontoPagadoAgo, inplace = True)
df['monto_pagado_sep'].fillna(value = vMontoPagadoSep, inplace = True)

df
```

	ID	monto	genero	educacion	estado_civil	edad	historial_pago_sep	histor
0	1	20000	2.0	2.0	1.0	24.0	2.0	
1	2	120000	2.0	2.0	2.0	26.0	-1.0	
2	3	90000	2.0	2.0	2.0	34.0	0.0	
3	4	50000	2.0	2.0	1.0	37.0	0.0	
4	5	50000	1.0	2.0	1.0	57.0	-1.0	

Ahora solo queda realizar la validación que nos diga si existen valores nulos en el DF

```
df.isna().any()
```

```
El data frame no contiene valores no válidos (NaN, NA)
ID                   False
monto                False
genero               False
educacion            False
estado_civil         False
edad                 False
historial_pago_sep   False
historial_pago_ago   False
historial_pago_jul   False
historial_pago_jun   False
historial_pago_may   False
historial_pago_abr   False
monto_factura_sep    False
monto_factura_ago    False
monto_factura_jul    False
monto_factura_jun    False
monto_factura_may    False
monto_factura_abr    False
monto_pagado_sep     False
monto_pagado_ago     False
monto_pagado_jul     False
monto_pagado_jun     False
monto_pagado_may     False
monto_pagado_abr     False
Y                     True
dtype: bool
```

Como se puede observar ya no exoisten valores nulos a excepción de la columna 'Y'. Dicha columna es una campo que se debe calcular con modelos mas avanzados de analisis ya que depende de todas las columnas para terminar su valor.

Parte 3: Preparación de los datos

Con base en los resultados se responden detalladamente las siguientes preguntas:

1. ¿Qué datos consideró mas importantes? ¿Por qué?

   Se consideran mas importantes los datos que muestran el historial crediticio de la persona porque este muestra el comportamiento que ha tenido y da un gran indicio de si es viable o no, otorgar un nuevo crédito a dicha persona. Puede que otro dato importante sea el estado civil de esta persona así como su edad pero consideramos que es necesario de un mayor número de herramientas y análisis mas complejos determinar un resultado con estos valores.

2. ¿Se eliminaron o reemplazaron datos nulos? ¿Qué se hizo y por qué?

   Se realizaron ambos ejercicios, en uno se eliminaron columnas con datos nulos y en otro se reemplazaron. Esto dado que queríamos practicar ambos casos considerando que la opción de reemplazar datos es mas viable.

3. ¿Es necesario limpiar los datos para el análisis?

   Definitivamente es necesario limpiar datos porque esto ayuda a evita errores de código y análisis con valor no cuantitativos o cualitativos. Esta práctica es muy importante, nos asegura de cumplir con lineamientos de calidad para lograr los mejores resultados a la hora de realizar un analisis de datos complejo con diversas herramient

4. ¿Existen problemas de formato que deban solucionar antes del proceso de modelado?

   Consideramos que en este caso no pero seguramente los puede haber con otras bases de datos. Es importante solucionarlas para una mayor claridad en los datos a analizar ya que cualquier problema en el formato puede causar resultados incorrectos que den soluciones erróneas.

5. ¿Qué ajustes se realizaron en el proceso de limpieza de datos (agregar, integrar, eliminar, modificar registros (filas), cambiar atributos (columnas)?

   Se modificaron los nombres de las columnas para una mayor comprensión de los datos, se obtuvieron valores como la media, mediana o el valor de mayor repetición por columna para con estos poder sustituir los valores nulos. según el tipo de valor de cada columna fue el valor calculado para estas mismas. Por ejemplo: cuando hablamos de la columna 'genero' sacar la media no es viable ya que solo puede haber dos, representados por el número '1' para el hombre y el número '2' para la mujer, al sacar la media nos da un valor decimal entre 1 y 2, cuando el valor solo puede ser 1 o 2 exactamente. Para esto es mejor opción utilizar el valor que es mas frecuente en las columnas o en su defecto la mediana.

Productos pagados de Colab - Cancela los contratos aquí

✓ 0 s se ejecutó 21:42