Actividad Semanal -- 4

Ciencia y analítica de Datos

Alumno: Alejandro Jesús Vázquez Navarro

Matrícula: A01793146

Parte 1

Ejercicio guiado

In [19]:
```
# Importo las librerías que necesitaré para la extracción, transformación y limpieza de datos (si aplica)

import pandas as pd
import numpy as np

# Agrego este valor para no truncar los resultados:
pd.set_option('display.max_colwidth', None)

```

In [20]:
```
# Realizo la adquisición de datos:
inPath = 'https://raw.githubusercontent.com/PosgradoMNA/Actividades_Aprendizaje-/main/default%20of%20cred
dfPCA = pd.read_csv(inPath, index_col = 0)

#Verifico que estén correctamente bajados

dfPCA.head()
```

Out[20]:

ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X15	X16	X17	X18	X19	X20	X21	X22	X2
1	20000	2.0	2.0	1.0	24.0	2.0	2.0	-1.0	-1.0	-2.0	...	0.0	0.0	0.0	0.0	689.0	0.0	0.0	0.0	0.(
2	120000	2.0	2.0	2.0	26.0	-1.0	2.0	0.0	0.0	0.0	...	3272.0	3455.0	3261.0	0.0	1000.0	1000.0	1000.0	0.0	2000.(
3	90000	2.0	2.0	2.0	34.0	0.0	0.0	0.0	0.0	0.0	...	14331.0	14948.0	15549.0	1518.0	1500.0	1000.0	1000.0	1000.0	5000.(
4	50000	2.0	2.0	1.0	37.0	0.0	0.0	0.0	0.0	0.0	...	28314.0	28959.0	29547.0	2000.0	2019.0	1200.0	1100.0	1069.0	1000.(
5	50000	1.0	2.0	1.0	57.0	-1.0	0.0	-1.0	0.0	0.0	...	20940.0	19146.0	19131.0	2000.0	36681.0	10000.0	9000.0	689.0	679.(

5 rows × 24 columns

In [21]:

```
# Realizo la asignación de cabeceras:

# Asignación de cabeceras de acuerdo a la información proporcionada:

#X1: Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her
#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,
#X12-X17: Amount of bill statement (NT dollar). X12 = amount of bill statement in September, 2005; X13 =
#X18-X23: Amount of previous payment (NT dollar). X18 = amount paid in September, 2005; X19 = amount paid

# Creo un diccionario con los nombres de las columnas
col_dictionary = {'X1':'given_credit',
                  'X2': 'gender',
                  'X3': 'education',
                  'X4': 'marital_status',
                  'X5': 'age',
                  'X6':'sep05_repayment',
                  'X7':'aug05_repayment',
                  'X8':'jul05_repayment',
                  'X9':'jun05_repayment',
                  'X10':'may05_repayment',
                  'X11':'abr05_repayment',
                  'X12': 'sep05_bill_statement',
                  'X13': 'aug05_bill_statement',
                  'X14': 'jul05_bill_statement',
                  'X15': 'jun05_bill_statement',
                  'X16': 'may05_bill_statement',
                  'X17': 'abr05_bill_statement',
                  'X18': 'sep05_previous_payment',
                  'X19': 'aug05_previous_payment',
                  'X20': 'jul05_previous_payment',
                  'X21': 'jun05_previous_payment',
                  'X22': 'mayu05_previous_payment',
                  'X23': 'abr05_previous_payment'
                  }

# Renombro las columnas para tener un mejor entendimiento del dataset
dfPCA.rename(columns = col_dictionary , inplace=True)
```

```
42
43 dfPCA.head()
44 #df.columns
45
```

Out[21]:

ID	given_credit	gender	education	marital_status	age	sep05_repayment	aug05_repayment	jul05_repayment	jun05_repayment	may0
1	20000	2.0	2.0	1.0	24.0	2.0	2.0	-1.0	-1.0	
2	120000	2.0	2.0	2.0	26.0	-1.0	2.0	0.0	0.0	
3	90000	2.0	2.0	2.0	34.0	0.0	0.0	0.0	0.0	
4	50000	2.0	2.0	1.0	37.0	0.0	0.0	0.0	0.0	
5	50000	1.0	2.0	1.0	57.0	-1.0	0.0	-1.0	0.0	

5 rows × 24 columns

Paso 1:

- Determine el número mínimo de componentes principales que representan la mayor parte de la variación en sus datos

* Utilice la proporción acumulada de la varianza que explican los componentes para determinar la cantidad de varianza que explican los componentes principales.

In [22]:

```
1 # Para este paso utilizaré matplotlib, numpy y PCA de sklearn decomposition
2
3 import matplotlib as mpl
4 import matplotlib.pyplot as plt
5 import numpy as np
6 from sklearn.decomposition import PCA
7
8 # Tengo 23 variables (incluyendo una dependiente Y que significa si el cliente cayó en default o no)
```

In [23]:
```
# Antes de comenzar debo eliminar las variables y para ello me basaré en el ejercicio anterior:
# Esto es solo una sencilla validación:

dfPCA.isna().any()

# Tenemos nullos en casi todas las columnas excepto given_credit

```

Out[23]:
```
given_credit               False
gender                      True
education                   True
marital_status              True
age                         True
sep05_repayment             True
aug05_repayment             True
jul05_repayment             True
jun05_repayment             True
may05_repayment             True
abr05_repayment             True
sep05_bill_statement        True
aug05_bill_statement        True
jul05_bill_statement        True
jun05_bill_statement        True
may05_bill_statement        True
abr05_bill_statement        True
sep05_previous_payment      True
aug05_previous_payment      True
jul05_previous_payment      True
jun05_previous_payment      True
mayu05_previous_payment     True
abr05_previous_payment      True
Y                           True
dtype: bool
```

In [24]:
```
# Tengo variables categóricas como gender, education y marital_status, podría pasarles un one hot encodin
# pero no es el objetivo de este ejercicio, así que las eliminaré del dataframe

dfOriginal = dfPCA
dfPCA = dfPCA.loc[:, ~dfPCA.columns.isin(['gender', 'education', 'marital_status', 'Y'])]

```

In [25]:
```
dfPCA.dropna(inplace=True)
print(f'Longitud del dataframe {len(dfPCA)}')

# El dataframe tiene 30k registros, se perdieron solamente 2.
# No haré una análisis de missing data porque esto fue parte de la tarea anterior


```

```
Longitud del dataframe 29958

C:\Users\aleja\AppData\Local\Temp\ipykernel_30632\2450569021.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  dfPCA.dropna(inplace=True)
```

In [26]:
```
dfPCA.head()
```

Out[26]:

ID	given_credit	age	sep05_repayment	aug05_repayment	jul05_repayment	jun05_repayment	may05_repayment	abr05_repayment	sep
1	20000	24.0	2.0	2.0	-1.0	-1.0	-2.0	-2.0	
2	120000	26.0	-1.0	2.0	0.0	0.0	0.0	2.0	
3	90000	34.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	50000	37.0	0.0	0.0	0.0	0.0	0.0	0.0	
5	50000	57.0	-1.0	0.0	-1.0	0.0	0.0	0.0	

In [27]:
```
# Comencemos con un tratamiento desde cero considerando todas las variables:
pcs = PCA(n_components=20)

pcs.fit(dfPCA)


pcsSummary = pd.DataFrame({'Standard deviation': np.sqrt(pcs.explained_variance_),
                           'Proportion of variance':pcs.explained_variance_ratio_,
                           'Cumulative proportion': np.cumsum(pcs.explained_variance_ratio_)#column
                           }
                           )
pcsSummary.round(4)
```

Out[27]:

	Standard deviation	Proportion of variance	Cumulative proportion
0	166585.0753	0.6106	0.6106
1	115824.7017	0.2952	0.9057
2	37256.8166	0.0305	0.9363
3	27747.1030	0.0169	0.9532
4	20698.6823	0.0094	0.9626
5	20278.0072	0.0090	0.9717
6	18520.7019	0.0075	0.9792
7	17040.2350	0.0064	0.9856
8	16294.1101	0.0058	0.9915
9	11615.2249	0.0030	0.9944
10	10413.1020	0.0024	0.9968
11	8786.7540	0.0017	0.9985
12	8197.6489	0.0015	1.0000
13	9.1140	0.0000	1.0000
14	2.0623	0.0000	1.0000
15	0.9386	0.0000	1.0000
16	0.7321	0.0000	1.0000

	Standard deviation	Proportion of variance	Cumulative proportion
17	0.5963	0.0000	1.0000
18	0.5120	0.0000	1.0000

In [28]:
```
# Hagamos una transposición para mejor lectura

pcsSummary = pcsSummary.transpose()
pcsSummary.round(2)
```

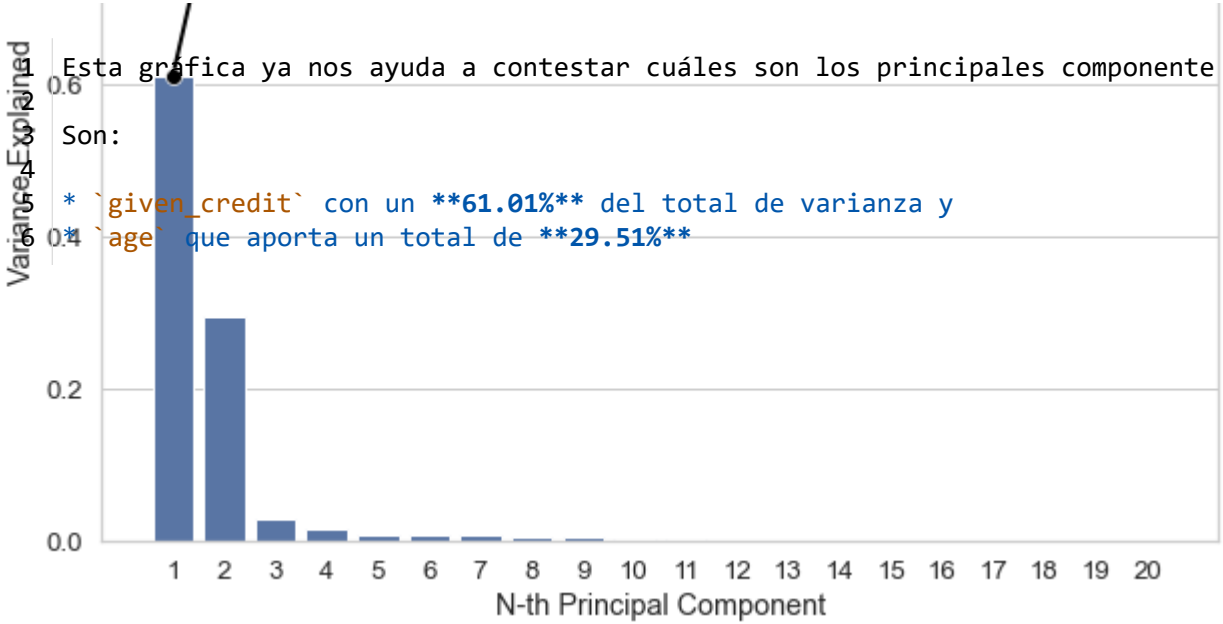
Out[28]:

	0	1	2	3	4	5	6	7	8	9	10	11	
Standard deviation	166585.08	115824.70	37256.82	27747.10	20698.68	20278.01	18520.70	17040.23	16294.11	11615.22	10413.1	8786.75	819
Proportion of variance	0.61	0.30	0.03	0.02	0.01	0.01	0.01	0.01	0.01	0.00	0.0	0.00	
Cumulative proportion	0.61	0.91	0.94	0.95	0.96	0.97	0.98	0.99	0.99	0.99	1.0	1.00	

```
Con esta tabla podemos calcular fácilmente que:

* `z1` accounts for 61.01% del total de varianza y
* `z2` aporta un total de 29.51%

Esto suma un 90.57% del total de varianza.

Significa pues que si removemos el PC2 tendríamos aún 61.01% de varianza. Este componente corresponde al crédito ortorgado

```

In [29]:

```
# Hagamos una gráfica para representarlo y determinar el número mínimo de componentes:

import seaborn as sns

PC_components = np.arange(pcs.n_components_) + 1
#PC_components

_ = sns.set(style = 'whitegrid',
            font_scale = 1.2
            )

fig, ax = plt.subplots(figsize=(10, 7))

_ = sns.barplot(x = PC_components,
                y = pcs.explained_variance_ratio_,
                color = 'b'
                )

_ = sns.lineplot(x = PC_components-1,
                 y = np.cumsum(pcs.explained_variance_ratio_),
                 color = 'black',
                 linestyle = '-',
                 linewidth = 2,
                 marker = 'o',
                 markersize = 8
                 )

plt.title('Scree Plot')
plt.xlabel('N-th Principal Component')
plt.ylabel('Variance Explained')
plt.ylim(0, 1)
plt.show()

```




```
Esta gráfica ya nos ayuda a contestar cuáles son los principales componente del dataset.

Son:

* `given_credit` con un **61.01%** del total de varianza y
* `age` que aporta un total de **29.51%**
```

:s

ginales.

rá la variable correspo

In [30]:

```
# Examinemos los componentes:


pcsSummary.columns = ['PC{}'.format(i) for i in range(1, len(pcsSummary.columns) + 1)]
pcsSummary.round(4)
```

Out[30]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
Standard deviation	166585.0753	115824.7017	37256.8166	27747.1030	20698.6823	20278.0072	18520.7019	17040.2350	16294.1101	11615.2249
Proportion of variance	0.6106	0.2952	0.0305	0.0169	0.0094	0.0090	0.0075	0.0064	0.0058	0.0030
Cumulative proportion	0.6106	0.9057	0.9363	0.9532	0.9626	0.9717	0.9792	0.9856	0.9915	0.9944

In [31]:
```
# Es necesario identificar magnitudes y direcciones de cada componente para determinar importancia de cad
# en cada componente:

pcsComponents_dfX = pd.DataFrame(pcs.components_.transpose(),
                                 columns = pcsSummary.columns,
                                 index = dfPCA.iloc[:, 0:].columns
                                 )

# Redondeo a 4 cifras para una mejor lectura
pcsComponents_dfX.round(4)
```

Out[31]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13
given_credit	0.4911	0.8693	-0.0206	-0.0184	-0.0441	0.0169	-0.0011	0.0010	0.0095	0.0038	0.0058	0.0000	0.001
age	0.0000	0.0000	-0.0000	-0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0000	0.0000	-0.0000	-0.000
sep05_repayment	0.0000	-0.0000	-0.0000	-0.0000	-0.0000	0.0000	-0.0000	0.0000	-0.0000	0.0000	0.0000	0.0000	-0.000
aug05_repayment	0.0000	-0.0000	-0.0000	-0.0000	-0.0000	0.0000	-0.0000	0.0000	-0.0000	0.0000	0.0000	0.0000	0.000
jul05_repayment	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	0.0000	0.0000	-0.0000	0.0000	0.0000	0.0000	0.000
jun05_repayment	0.0000	-0.0000	0.0000	0.0000	-0.0000	0.0000	0.0000	0.0000	-0.0000	0.0000	-0.0000	0.0000	0.000
may05_repayment	0.0000	-0.0000	0.0000	-0.0000	-0.0000	-0.0000	-0.0000	-0.0000	-0.0000	0.0000	0.0000	0.0000	-0.000
abr05_repayment	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-0.0000	0.000
sep05_bill_statement	0.3886	-0.2211	-0.5489	-0.2049	0.3169	0.0026	-0.4563	0.0073	0.0703	0.1727	-0.2674	-0.2046	-0.005
aug05_bill_statement	0.3815	-0.2262	-0.3910	-0.0378	0.0115	-0.0805	0.5269	-0.0660	0.0993	0.0079	0.3511	0.4765	0.016
jul05_bill_statement	0.3723	-0.2163	-0.0481	0.5534	-0.2476	0.2958	0.0287	0.0813	-0.1080	-0.3365	0.1515	-0.4416	-0.082
jun05_bill_statement	0.3465	-0.1938	0.2610	0.0791	-0.3679	-0.4732	-0.1782	-0.2277	-0.0974	-0.1369	-0.3749	0.2335	0.318
may05_bill_statement	0.3230	-0.1766	0.4281	-0.2001	-0.0158	-0.0783	-0.0134	0.4801	0.1265	0.1010	-0.0525	0.0924	-0.605
abr05_bill_statement	0.3087	-0.1672	0.4837	-0.3084	0.2269	0.3918	0.0768	-0.2629	-0.1055	0.2368	0.2031	-0.1585	0.366
sep05_previous_payment	0.0266	0.0057	0.0375	0.1935	0.2025	-0.2280	0.6117	-0.0826	0.1709	0.2792	-0.4709	-0.3965	-0.017
aug05_previous_payment	0.0313	0.0108	0.1601	0.6733	0.3373	0.1076	-0.2548	-0.0047	0.2264	0.3581	0.0235	0.3840	0.047
jul05_previous_payment	0.0268	0.0110	0.1351	0.0283	0.3106	-0.5666	-0.1526	-0.3082	0.2279	-0.2016	0.4972	-0.2858	-0.151
jun05_previous_payment	0.0222	0.0104	0.0944	-0.0403	0.4364	0.0203	0.0854	0.4481	0.2273	-0.5907	-0.1465	0.0790	0.402

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13
may_05_previous_payment	0.0222	0.0117	0.0559	0.0137	0.3551	0.2087	0.0602	0.4816	0.3075	0.3928	0.2999	0.2346	0.444(

```
Con esta tabla podemos interpretar fácilmente lo siguiente:

- El primer componente principal (PC1) está dominado por dos variables `given_credit` con una magnitud positiva de 0.4911 y por `sep05_bill_statement` con un peso de **0.3886**

- El segundo componente está completamente dominado por la variable `given_credit` con un peso de **0.8693**

- No estoy agregando el PC3 debido a su poca aportación al total de varianza acumulada: **3.05**
```

Paso 3:

- Identifique valores atípicos

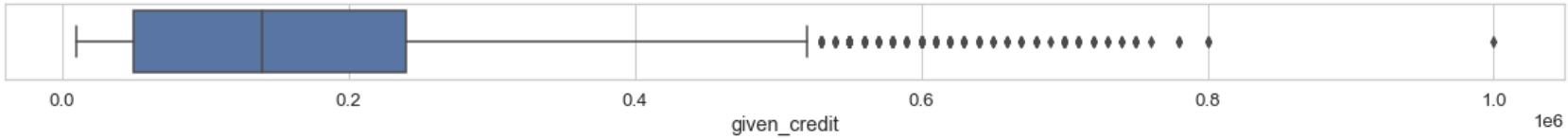
- Realice alguna gráfica de valores atípicos o boxplot para identificar los valores atípicos.
- Cualquier punto que esté más alejado de la línea de referencia es un valor atípico.

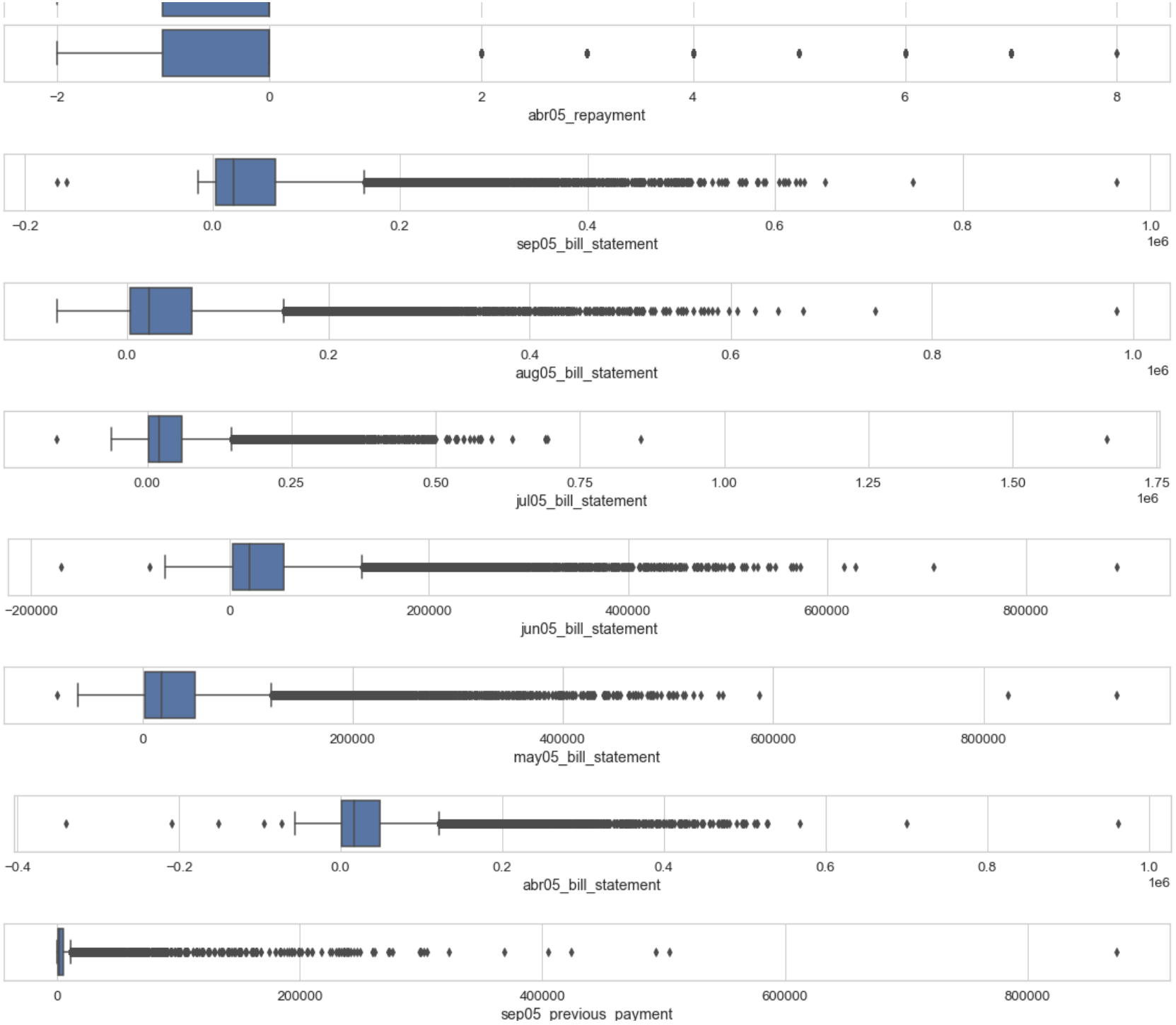
In [32]:
```
# Para este cometido utilizaré el dataframe generado : pcsComponents_dfX

pcsComponents_dfX['feature'] = pcsComponents_dfX.index

dfOutLiers=pcsComponents_dfX.reset_index()
dfOutLiers.reset_index(inplace=True)

# Elimino una variable que surgió
dfOutLiers.pop("level_0")
dfOutLiers.pop("index")


# Verifico mi dataframe
dfOutLiers.head()
dfOutLiers.columns

```

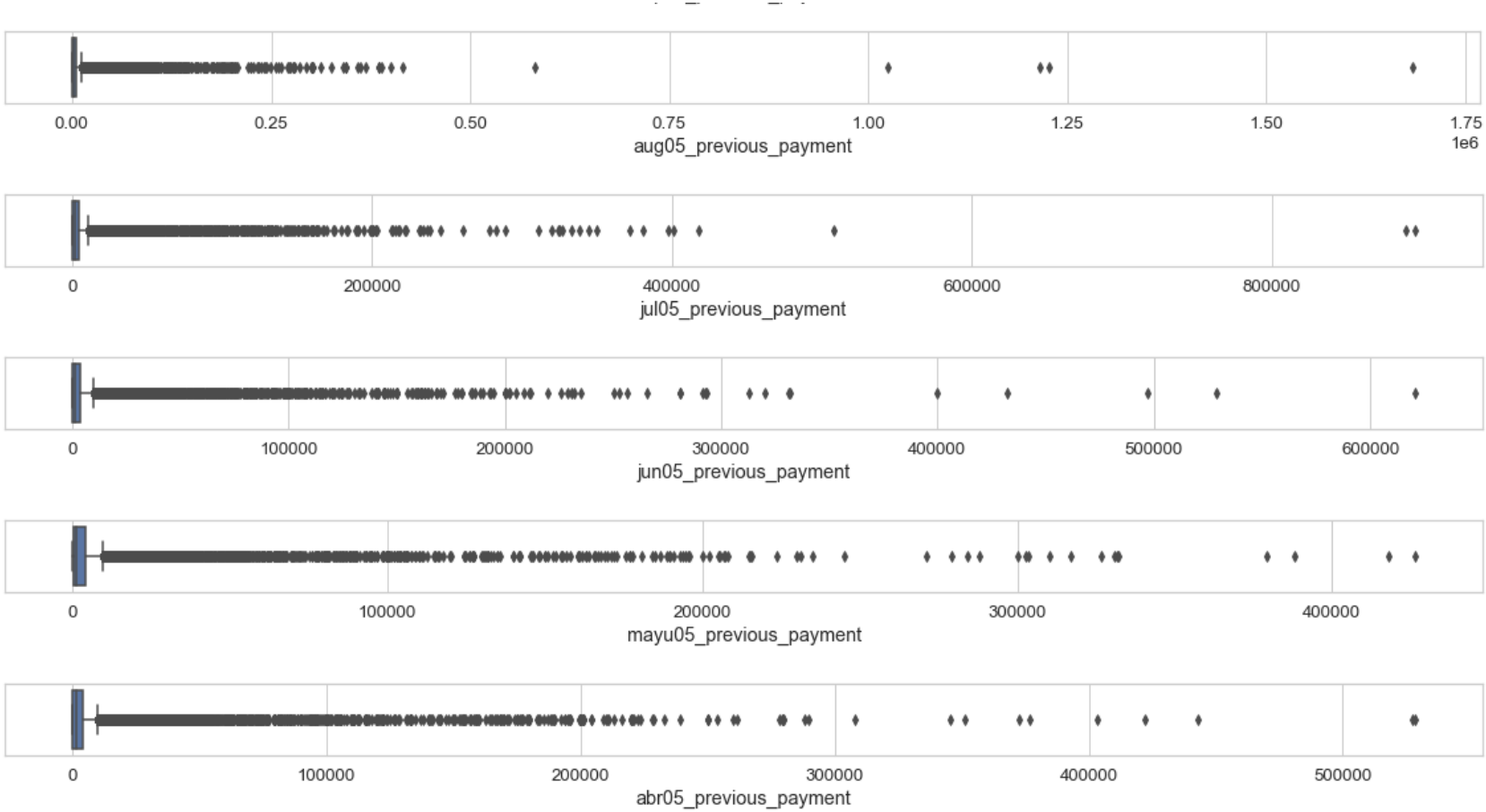
Out[32]:
```
Index(['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10',
       'PC11', 'PC12', 'PC13', 'PC14', 'PC15', 'PC16', 'PC17', 'PC18', 'PC19',
       'PC20', 'feature'],
      dtype='object')
```

In [44]:
```
# Esta línea es solo cosmética para no hacer scroll down

%%javascript
IPython.OutputArea.prototype._should_scroll = function(lines) {
    return false;
}

```

In [45]:
```
1
2 plt.rcParams["figure.figsize"] = (12,10)
3
4
5 for column in dfPCA:
6         plt.figure(figsize=(20,1))
7         sns.boxplot(data=dfOriginal, x=column)
```




−2
0
2
4
6
8
abr05_repayment
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1e6
sep05_bill_statement
0.0
0.2
0.4
0.6
0.8
1.0
1e6
aug05_bill_statement
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e6
jul05_bill_statement
−200000
0
200000
400000
600000
800000
jun05_bill_statement
0
200000
400000
600000
800000
may05_bill_statement
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1e6
abr05_bill_statement
0
200000
400000
600000
800000
sep05_previous_payment



Interesante porque podemos encontrar outliers en `given credit` , `age'`, sep05_bill_statement , aug05_bill_statement', `jun05_bill_statement` , `may05_bill_statement` , y en general en todas las columnas de pago previo

Parte 2

Responde las siguientes preguntas en una celda de texto en Jupyter Notebook

- **¿Cuál es el número de componentes mínimo y por qué?**

Como el análisis de varianza fue posible encontrar que:

- `z1` aporta el 61.01% del total de varianza y
- `z2` aporta un total de 29.51%

Esto suma un 90.57% del total de varianza. No es necesario incluir los demás componentes pues tienen una aportación muy baja.

```
Significa pues que si removemos el PC2 tendríamos aún 61.01% de varianza. Este componente correspond
e al crédito ortorgado.

Estos componentes son:
    - given_credit
    - age
```

- **¿Cuál es la variación de los datos que representan esos componentes?**

Con el cálculo de varianza se pudo calcular que:

- `z1` aporta el 61.01% del total de varianza y
- `z2` aporta un total de 29.51%

- **¿Cuál es la pérdida de información después de realizar PCA?**

Podemos quedarnos con el 90.57% de la información y perder 9.43%

- **De las variables originales, ¿Cuál tiene mayor y cuál tiene menor importancia en los componentes principales?**

Después de calcular el peso de cada variable para cada componente principal pude calcular que:

- El primer componente principal (PC1) está dominado por dos variables `given_credit` con una magnitud positiva de 0.4911 y por `sep05_bill_statement` con un peso de **0.3886**
- El segundo componente está completamente dominado por la variable `given_credit` con un peso de **0.8693**
- No estoy agregando el PC3 debido a su poca aportación al total de varianza acumulada: **3.05**

- **¿Cuándo se recomienda realizar un PCA y qué beneficios ofrece para Machine Learning?**

La reducción de dimensionalidad es una técnica necesaria cuando tenemos un número alto de características en nuestro

dataset. Además de tener un nivel alto de correlación entre diferentes variables, eso lo podemos conocer como multicolinealidad. PCA aplica un principio de parsimonia donde obtenemos el máximo de información con el mínimo de características.

Como beneficios principales, nos ofrece la reduccion de la matriz del dataset reteniendo la máxima información posible, esto deriva en lo siguiente:

- Entendimiento más sencillo de las variables que aportan información al modelo.
- Reducción de procesamiento pues tendremos matrices más pequeñas.
- Naturalmente ayuda a reducir el overfitting pues nuestro modelo tomará solamente las variables necesarias.