Ciencia de Datos: Análisis Multivariante y Principios de Aprendizaje de Máquina.

Estadística para las Ciencias de la Computación

Integrantes: Sarmiento Basurto Douglas Bryan, Zhizhpon Tacuri César Eduardo

1. Seleccionar y cargar un dataset

El dataset seleccionado es master_dataset.csv.

link: https://www.covid19survivalcalculator.com/download

Para cargar el dataset, debido al excesivo ruido, se optó por elegir los valores que se considerarían como nulos o vacíos al momento de cargar el Dataset.

In [29]:
```
# Cargar los datasets.
%matplotlib notebook

import pandas as pd
import numpy as np

_PATH = '4.DatasetsCOVID19/'
file = 'master_dataset.csv'
dataset = _PATH + file
df = pd.read_csv(dataset, ',', dtype='str',
                 keep_default_na=False, na_values=['na', '', 'null', 'NaN', 'undefined'])
df
```

Out[29]:

	survey_date	region	country	ip_latitude	ip_longitude	ip_accuracy	sex	age	height	weight	...
0	2020-03-24	NA	CA	43.6023	-79.3058	100	male	20_30	178	88	...
1	2020-03-25	NA	CA	51.0263	-114.0862	5	female	30_40	158	54	...
2	2020-03-25	NA	CA	43.1642	-79.8471	100	male	90_100	184	94	...
3	2020-03-25	NA	CA	45.6605	-73.6724	5	male	60_70	172	96	...
4	2020-03-25	NA	CA	49.2525	-122.9481	1	male	30_40	166	70	...
...	...	...	...	...	...	...	...	...	...	...	...
820575	2020-07-07	NA	US	32.2146	-110.7915	5	female	70_80	166	66	...

820576	2020-07-07	NA	CA	43.5832	-79.391	1	male	70_80	168	70	...
820577	2020-07-08	NA	US	30.5336	-97.7256	1	male	70_80	182	82	...
820578	2020-07-06	NA	US	39.8837	-75.3197	5	male	50_60	184	80	...
820579	2020-07-08	NA	US	47.6375	-122.2305	10	female	30_40	172	66	...

820580 rows × 61 columns

In [30]:
```
# Funciones que nos ayudarán en la verificación del ruido
def printNaN(df):
    columns_with_null = df.columns[df.isnull().any()]
    if len(columns_with_null) > 0:
        for c in columns_with_null:
            print(c, ': ', df[c].isnull().sum())
    else:
        print('None')

def printType(df):
    columns = df.columns
    dataTypeSeries = df.dtypes
    i = 0
    for c in columns:
        print(c, ": ", dataTypeSeries[i])
        i+=1

printNaN(df)
#print('\n\n')
#printType(df)
```

```
region :  36
country :  36
ip_accuracy :  10
sex :  4
age :  4
height :  4
weight :  4
bmi :  4
blood_type :  4
insurance :  688743
income :  688743
race :  688743
immigrant :  688743
smoking :  1857
alcohol :  1897
cannabis :  98458
amphetamines :  116764
cocaine :  118931
lsd :  114484
mdma :  110138
contacts_count :  4247
house_count :  22
public_transport_count :  688743
working :  4236
worried :  688743
rate_reducing_mask :  1857
rate_reducing_mask_type :  706502
prescription_medication :  570031
opinion_infection :  145890
opinion_mortality :  145890
risk_infection :  1
```

```
risk_mortality :  1
Unnamed: 60 :  820580
```

2. Desarrollar procesamiento de datos: limpiar ruido, transformar variables categóricas en numéricas, transformación de datos numéricos (estandarización, MinMax, Scaling)

2.1 Limpiar Ruido

In [172…

```
df_p = df
# 1. Eliminar filas que presentan mucho ruido al igual que la fecha.
df_p = df_p.iloc[:, range(1, len(df_p.columns) - 1)]

df_p.drop([211409, 211676, 352078, 539050, 544425], axis=0,inplace=True)
df_p
```

```
C:\Users\cesc_\anaconda3\envs\estadistica\lib\site-packages\pandas\core\frame.py:3997: S
ettingWithCopyWarning:
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
  errors=errors,
```

Out[172]:

	region	country	ip_latitude	ip_longitude	ip_accuracy	sex	age	height	weight	bmi	...	hiv_pos
0	NA	CA	43.6023	-79.3058	100	male	20_30	178	88	27.7	...	
1	NA	CA	51.0263	-114.0862	5	female	30_40	158	54	21.6	...	
2	NA	CA	43.1642	-79.8471	100	male	90_100	184	94	27.7	...	
3	NA	CA	45.6605	-73.6724	5	male	60_70	172	96	32.4	...	
4	NA	CA	49.2525	-122.9481	1	male	30_40	166	70	25.4	...	
...	...	...	...	...	...	...	...	...	...	...	...	
820575	NA	US	32.2146	-110.7915	5	female	70_80	166	66	23.9	...	
820576	NA	CA	43.5832	-79.391	1	male	70_80	168	70	24.8	...	
820577	NA	US	30.5336	-97.7256	1	male	70_80	182	82	24.7	...	
820578	NA	US	39.8837	-75.3197	5	male	50_60	184	80	23.6	...	
820579	NA	US	47.6375	-122.2305	10	female	30_40	172	66	22.3	...	

820575 rows × 59 columns

2.2. Transformar variables

In [173…

```
# PREPROCESADO: Reemplazar los valores NULL o NaN
# 1 UK = Unknow
df_p['region'].fillna('UK', inplace=True)
df_p['country'].fillna('UK', inplace=True)
df_p['working'].fillna('unknow', inplace=True)
df_p['blood_type'].fillna('unknow', inplace=True)
df_p['sex'].fillna('unknow', inplace=True)

# 2 Cambiar por 0
```

```
df_p['contacts_count'].fillna(0, inplace=True)
df_p['house_count'].fillna(0, inplace=True)
df_p['ip_accuracy'].fillna(0, inplace=True)
df_p['public_transport_count'].fillna(0, inplace=True)
df_p['worried'].fillna(0, inplace=True)
df_p['rate_reducing_mask'].fillna(0, inplace=True)
df_p['rate_reducing_mask_type'].fillna(0, inplace=True)
df_p['opinion_infection'].fillna(0, inplace=True)
df_p['opinion_mortality'].fillna(0, inplace=True)

# 3 "blank"
df_p['insurance'].fillna('blank', inplace=True)
df_p['income'].fillna('blank', inplace=True)
df_p['immigrant'].fillna('blank', inplace=True)

# 4 "other"
df_p['race'].fillna('other', inplace=True)

# 5 "never"
df_p['smoking'].fillna('never', inplace=True)

# 6 Cambiar por -1
df_p['alcohol'].fillna(-1, inplace=True)

# 7 Cambiar por -2
df_p['cannabis'].fillna(-2, inplace=True)
df_p['amphetamines'].fillna(-2, inplace=True)
df_p['cocaine'].fillna(-2, inplace=True)
df_p['lsd'].fillna(-2, inplace=True)
df_p['mdma'].fillna(-2, inplace=True)

# 8 Cambiar por "none"
df_p['prescription_medication'].fillna('none', inplace=True)

# 9 Cambiar por 20_30
df_p['age'].fillna('20_30', inplace=True)

# PREPROCESADO: Transformar variables CATEGORICAS y NUMERICAS
# 1. Categoricas Ordinales

# Para evitar errores en el dataset: De que hay valores nulos.
df_p['region'].replace('NA', 'AN', inplace=True)
df_p['country'].replace('NA', 'AN', inplace=True)

# 2 Variable income: porque si importa el orden de ingresos
df_p['income'].replace('blank', 0, inplace=True)
df_p['income'].replace('gov', 1, inplace=True)
df_p['income'].replace('low', 2, inplace=True)
df_p['income'].replace('med', 3, inplace=True)
df_p['income'].replace('high', 4, inplace=True)

# 3 Variable rate_reducing_mask_type: porque si importa el nivel de proteccion
df_p['rate_reducing_mask_type'].replace('clothhome', 1, inplace=True)
df_p['rate_reducing_mask_type'].replace('clothstore', 2, inplace=True)
df_p['rate_reducing_mask_type'].replace('surgical', 3, inplace=True)
df_p['rate_reducing_mask_type'].replace('level1', 4, inplace=True)
df_p['rate_reducing_mask_type'].replace('level2', 5, inplace=True)
df_p['rate_reducing_mask_type'].replace('level3', 6, inplace=True)

# 4 Introducir valores NaN con su media
risk_infection_mean = pd.to_numeric(df_p['risk_infection'], errors='coerce').mean()
risk_mortality_mean = pd.to_numeric(df_p['risk_mortality'], errors='coerce').mean()
bmi_mean = pd.to_numeric(df_p['bmi'], errors='coerce').mean()
height_mean = pd.to_numeric(df_p['height'], errors='coerce').mean()
weight_mean = pd.to_numeric(df_p['weight'], errors='coerce').mean()
```

```
df_p['risk_infection'].fillna(float(risk_infection_mean), inplace=True)
df_p['risk_mortality'].fillna(float(risk_mortality_mean), inplace=True)
df_p['bmi'].fillna(float(bmi_mean), inplace=True)
df_p['height'].fillna(float(height_mean), inplace=True)
df_p['weight'].fillna(float(weight_mean), inplace=True)

df_p.reset_index(drop=True, inplace=True)
n = len(df_p['prescription_medication'])

# 5 Variable prescription_medication; debido a la cantidad de datos,
# se opto por tomarle como si fuera una categorica ordinal, en donde importa
# la cantidad de medicamentos prescritos
for i in range(n):
    s = str(df_p.at[i, 'prescription_medication']).split(';')
    if (s[0] == "none"):
        df_p.at[i, 'prescription_medication'] = 0
    else:
        df_p.at[i, 'prescription_medication'] = len(s)

# Creación y almacenamiento del Dataset limpio y listo para procesar.
DatasetPreprocesado = pd.DataFrame(data=df_p,columns=df_p.columns)

DatasetPreprocesado.to_csv("recursos/Nexoid-Limpio.csv", sep=";", index = False)
DatasetPreprocesado
```

```
C:\Users\cesc_\anaconda3\envs\estadistica\lib\site-packages\pandas\core\generic.py:6245:
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
  self._update_inplace(new_data)
C:\Users\cesc_\anaconda3\envs\estadistica\lib\site-packages\pandas\core\generic.py:6746:
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
  self._update_inplace(new_data)
```

Out[173]:

	region	country	ip_latitude	ip_longitude	ip_accuracy	sex	age	height	weight	bmi	...	hiv_pos
0	AN	CA	43.6023	-79.3058	100	male	20_30	178	88	27.7	...	
1	AN	CA	51.0263	-114.0862	5	female	30_40	158	54	21.6	...	
2	AN	CA	43.1642	-79.8471	100	male	90_100	184	94	27.7	...	
3	AN	CA	45.6605	-73.6724	5	male	60_70	172	96	32.4	...	
4	AN	CA	49.2525	-122.9481	1	male	30_40	166	70	25.4	...	
...	...	...	...	...	...	...	...	...	...	...	...	
820570	AN	US	32.2146	-110.7915	5	female	70_80	166	66	23.9	...	
820571	AN	CA	43.5832	-79.391	1	male	70_80	168	70	24.8	...	
820572	AN	US	30.5336	-97.7256	1	male	70_80	182	82	24.7	...	
820573	AN	US	39.8837	-75.3197	5	male	50_60	184	80	23.6	...	
820574	AN	US	47.6375	-122.2305	10	female	30_40	172	66	22.3	...	

820575 rows × 59 columns

In [31]:
```
# Cargar el dataset limpio.
import pandas as pd
import numpy as np

#Se carga el DataSet, sin ruido y dtype apropiados.
df_limpio = pd.read_csv("recursos/Nexoid-Limpio.csv", ';', low_memory=False)

printNaN(df_limpio)
df_limpio
```

None

Out[31]:

	region	country	ip_latitude	ip_longitude	ip_accuracy	sex	age	height	weight	bmi	...	hiv_pos
0	AN	CA	43.6023	-79.3058	100	male	20_30	178.0	88.0	27.7	...	
1	AN	CA	51.0263	-114.0862	5	female	30_40	158.0	54.0	21.6	...	
2	AN	CA	43.1642	-79.8471	100	male	90_100	184.0	94.0	27.7	...	
3	AN	CA	45.6605	-73.6724	5	male	60_70	172.0	96.0	32.4	...	
4	AN	CA	49.2525	-122.9481	1	male	30_40	166.0	70.0	25.4	...	
...	...	...	...	...	...	...	...	...	...	...	...	
820570	AN	US	32.2146	-110.7915	5	female	70_80	166.0	66.0	23.9	...	
820571	AN	CA	43.5832	-79.3910	1	male	70_80	168.0	70.0	24.8	...	
820572	AN	US	30.5336	-97.7256	1	male	70_80	182.0	82.0	24.7	...	
820573	AN	US	39.8837	-75.3197	5	male	50_60	184.0	80.0	23.6	...	
820574	AN	US	47.6375	-122.2305	10	female	30_40	172.0	66.0	22.3	...	

820575 rows × 59 columns

2.3. Filtrar los valores

In [65]:
```
from sklearn.model_selection import train_test_split

# Se seleccionan los datos con un rango de edad de 80 a 100
df_d = df_limpio[(((df_limpio['age'] == '60_70') | (df_limpio['age'] == '70_80')
                   | (df_limpio['age'] == '80_90') | (df_limpio['age'] == '90_100')))]

df_d.reset_index(inplace=True)

df_d = df_d.drop('index', 1)

df_d.to_csv("recursos/N-Filtrado-Edades.csv", sep=";", index = False)


df_e = df_d[(df_d['covid19_positive'] == 0)]

print(len(df_e.values))

df_e, df_e_test = train_test_split(df_e, test_size=0.00436, random_state=0)
df_f = df_d[(df_d['covid19_positive'] == 1)]

df_d = pd.concat([df_f, df_e_test])

df_d.sort_index(inplace=True)
df_d.reset_index(inplace=True)
df_d = df_d.drop('index', 1)

df_d
```

74403

Out[65]:

	region	country	ip_latitude	ip_longitude	ip_accuracy	sex	age	height	weight	bmi	...	hiv_positive
0	EU	NL	52.0218	4.5357	5	female	80_90	164.0	84.0	31.2	...	0
1	EU	CH	47.1921	8.1766	20	male	90_100	132.0	124.0	71.1	...	1
2	AN	US	42.7789	-71.1011	1	male	60_70	184.0	104.0	30.7	...	0
3	AN	US	39.8751	-74.2053	1	female	60_70	158.0	136.0	54.4	...	0
4	AN	US	42.2013	-76.3173	20	female	60_70	162.0	54.0	20.5	...	0
...	...	...	...	...	...	...	...	...	...	...	...	...
645	AN	US	39.3627	-82.9006	20	male	60_70	180.0	104.0	32.0	...	0
646	AN	CA	49.4373	-123.0993	5	male	60_70	188.0	96.0	27.1	...	0
647	SA	BR	-22.9799	-43.3934	200	male	70_80	178.0	106.0	33.4	...	0
648	AN	US	44.2191	-88.3356	5	female	60_70	174.0	70.0	23.1	...	0
649	AN	US	33.7024	-84.4555	1000	male	60_70	194.0	88.0	23.3	...	0

650 rows × 59 columns

2.4. Estandarización de datos

Cabe recalcar que, en este punto, se realizó primero los siguientes procesos con el Dataset Limpio completo; posteriormente, este se almacenó como DatasetPreprocesado.csv, sin ser filtrado por el rango de edades, esto con el fin de realizar el análisis.

In [67]:

```
# Estas variables representan a las traducciones de los atributos originales
encoderNames_esp = ['region', 'pais', 'sexo', 'edad', 'tipo_sangre', 'seguro', 'etnia',

otherNames_esp = ['ip_latitud', 'ip_longitud', 'ip_precision','altura', 'peso', 'imc', '
            'alcohol', 'canabis', 'anfetaminas', 'cocaina', 'lsd', 'mdma', 'numero_conta
            'numero_conviviente', 'numero_transporte_publico', 'preocupado', 'tasa_reduc
            'tasa_reduccion_riesgo_personal_distanciamiento_social', 'tasa_reduccion_rie
            'tasa_reduccion_riesgo_hogar', 'tasa_reducción_riesgo_hogar_distanciamiento_
            'tasa_reduccion_riesgo_lavado_manos_hogar', 'tasa_reduccion_riesgo_desinfect
            'tasa_reduccion_mascarilla', 'tasa_reduccion_tipo_mascarilla', 'tasa_ayuda_g
            'tasa_control_gubernamental', 'tasa_gasto_gubernamental', 'covid19_positivo'
            'sintomas_covid19', 'contacto_covid19', 'asma', 'enfemedad_riniones', 'enfem
            'inmunidad_comprometida', 'enfermedad_corazon', 'enfermedad_pulmonar', 'diab
            'hipertension', 'otras_enfermedades_cronicas', 'asilo', 'trabajador_sanitari
            'opinion_infeccion', 'opinion_mortalidad', 'riesgo_infeccion', 'riesgo_morta

atributos = ['region', 'pais', 'ip_latitud', 'ip_longitud', 'ip_precision', 'sexo', 'eda
             'tipo_sangre', 'seguro', 'ingresos', 'etnia', 'inmigrante', 'fumando',
            'alcohol', 'canabis', 'anfetaminas', 'cocaina', 'lsd', 'mdma', 'numero_conta
            'numero_conviviente', 'numero_transporte_publico', 'trabajando', 'preocupado
            'tasa_reduccion_riesgo_personal_distanciamiento_social', 'tasa_reduccion_rie
            'tasa_reduccion_riesgo_hogar', 'tasa_reducción_riesgo_hogar_distanciamiento_
            'tasa_reduccion_riesgo_lavado_manos_hogar', 'tasa_reduccion_riesgo_desinfect
            'tasa_reduccion_mascarilla', 'tasa_reduccion_tipo_mascarilla', 'tasa_ayuda_g
            'tasa_control_gubernamental', 'tasa_gasto_gubernamental', 'covid19_positivo'
            'sintomas_covid19', 'contacto_covid19', 'asma', 'enfemedad_riniones', 'enfem
            'inmunidad_comprometida', 'enfermedad_corazon', 'enfermedad_pulmonar', 'diab
            'hipertension', 'otras_enfermedades_cronicas', 'asilo', 'trabajador_sanitari
            'opinion_infeccion', 'opinion_mortalidad', 'riesgo_infeccion', 'riesgo_morta

df_d.columns = atributos
df_d
```

Out[67]:

	region	pais	ip_latitud	ip_longitud	ip_precision	sexo	edad	altura	peso	imc	...	vih_positivo	hiper
0	EU	NL	52.0218	4.5357	5	female	80_90	164.0	84.0	31.2	...	0	
1	EU	CH	47.1921	8.1766	20	male	90_100	132.0	124.0	71.1	...	1	
2	AN	US	42.7789	-71.1011	1	male	60_70	184.0	104.0	30.7	...	0	
3	AN	US	39.8751	-74.2053	1	female	60_70	158.0	136.0	54.4	...	0	
4	AN	US	42.2013	-76.3173	20	female	60_70	162.0	54.0	20.5	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
645	AN	US	39.3627	-82.9006	20	male	60_70	180.0	104.0	32.0	...	0	
646	AN	CA	49.4373	-123.0993	5	male	60_70	188.0	96.0	27.1	...	0	
647	SA	BR	-22.9799	-43.3934	200	male	70_80	178.0	106.0	33.4	...	0	
648	AN	US	44.2191	-88.3356	5	female	60_70	174.0	70.0	23.1	...	0	
649	AN	US	33.7024	-84.4555	1000	male	60_70	194.0	88.0	23.3	...	0	

650 rows × 59 columns

In [69]:

```
from sklearn.preprocessing import OneHotEncoder, StandardScaler, MinMaxScaler
from sklearn.compose import make_column_transformer, ColumnTransformer
from sklearn.pipeline import Pipeline

#df_d = pd.read_csv("recursos/N-Filtrado-Edades.csv", ';', low_memory=False)
#df_d = pd.read_csv("recursos/Nexoid-Limpio.csv", ';', low_memory=False)

# Se realiza un preprocesamiento de las variables NUMÉRICAS usando StandardScaler.
# Se realiza un preprocesamiento de las variables CATEGÓRICAS NOMINALES usando OneHotEnc

# Array que contiene las columnas que son CATEGóRICAS NOMINALES, según el dataset.
encoderNames = ['region', 'country', 'sex', 'age', 'blood_type', 'insurance', 'race', 'i

otherNames = ['ip_latitude', 'ip_longitude', 'ip_accuracy','height', 'weight', 'bmi', 'i
              'alcohol', 'cannabis', 'amphetamines', 'cocaine', 'lsd', 'mdma', 'contacts_c
              'house_count', 'public_transport_count', 'worried', 'rate_reducing_risk_sing
              'rate_reducing_risk_single_social_distancing', 'rate_reducing_risk_single_wa
              'rate_reducing_risk_house', 'rate_reducing_risk_house_social_distancing',
              'rate_reducing_risk_house_washing_hands', 'rate_reducing_risk_single_sanitiz
              'rate_reducing_mask', 'rate_reducing_mask_type', 'rate_government_action',
              'rate_government_control', 'rate_government_spend', 'covid19_positive',
              'covid19_symptoms', 'covid19_contact', 'asthma', 'kidney_disease', 'liver_di
              'compromised_immune', 'heart_disease', 'lung_disease', 'diabetes', 'hiv_posi
              'hypertension', 'other_chronic', 'nursing_home', 'health_worker', 'prescript
              'opinion_infection', 'opinion_mortality', 'risk_infection', 'risk_mortality'
# Estableces las variables traducidas
encoderNames = encoderNames_esp
otherNames = otherNames_esp

# Aplicación del método OneHotEncoder(Coding) para transformar los datos.
preprocesador1 = make_column_transformer((OneHotEncoder(), encoderNames), remainder='pas
X = preprocesador1.fit_transform(df_d)
pre_features = preprocesador1.transformers_[0][1].get_feature_names(encoderNames)

pre_features = pre_features.tolist()
pre_features.extend(otherNames)

# Usar X.todense() al problema de Dimensiones.
categorical = pd.DataFrame(data=X, columns=pre_features)
```

```
escalerNames = []
salida = 'covid19_positivo'
for col in categorical.columns:
    if(col != salida):
        escalerNames.append(col)

preprocesador2 = make_column_transformer((MinMaxScaler(), escalerNames), remainder='pass
procesado = preprocesador2.fit_transform(categorical)
escalerNames.append(salida);
print(len(escalerNames))
Data = pd.DataFrame(data=procesado,columns=escalerNames)
#Data.to_csv("recursos/Nexoid.csv", sep=";",index = False)
#Data.to_csv("recursos/N-Filtrado.csv", sep=";",index = False)
Data.to_csv("recursos/N-Equilibrado.csv", sep=";",index = False)
Data
```

```
145
```

Out[69]:

	region_AF	region_AN	region_AS	region_EU	region_OC	region_SA	pais_AM	pais_AR	pais_AT	pais_AU	...
0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	..
1	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	..
2	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..
3	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..
4	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..
...	...	...	...	...	...	...	...	...	...	...	..
645	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..
646	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..
647	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	..
648	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..
649	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..

650 rows × 145 columns

3. Realizar estadística descriptiva

In [177...

```
import pandas as pd
import numpy as np

df_a = pd.read_csv("recursos/Nexoid.csv", ';')
df_a
```

Out[177]:

	region_AF	region_AN	region_AS	region_EU	region_OC	region_SA	region_UK	country_AD	country_AE
0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...
820570	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

820571	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
820572	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
820573	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
820574	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

820575 rows × 297 columns

In [181…
```
des = df_a.describe()
des
```

Out[181]:

	region_AF	region_AN	region_AS	region_EU	region_OC	region_SA	region_UK
count	820575.000000	820575.000000	820575.000000	820575.000000	820575.000000	820575.000000	820575.000000
mean	0.002597	0.862159	0.009627	0.061578	0.011421	0.052574	0.000044
std	0.050894	0.344733	0.097646	0.240387	0.106258	0.223182	0.006623
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 297 columns

Varianza

In [182…
```
var = df_a.var()
#i = 0
#for v in var.index:
#    print(v, ":", var[i])
#    i += 1
var
```

Out[182]:
```
region_AF             0.002590
region_AN             0.118841
region_AS             0.009535
region_EU             0.057786
region_OC             0.011291
                          ...
opinion_infection     0.066050
opinion_mortality     0.049977
risk_infection        0.022624
risk_mortality        0.000156
covid19_positive      0.002274
Length: 297, dtype: float64
```

In [184…
```
# Correlación de todas las variables
correlacion = df_a.corr()

# Debido a que es imposible tener una buena perspectiva visual con el
# mapa de calor, se optó por buscar e imprimir los valores más altos.
correlacion
```

Out[184]:

	region_AF	region_AN	region_AS	region_EU	region_OC	region_SA	region_UK	country_AD	c
region_AF	1.000000	-0.127615	-0.005031	-0.013071	-0.005485	-0.012020	-0.000338	-0.000159	

region_AN	-0.127615	1.000000	-0.246581	-0.640643	-0.268816	-0.589139	-0.016566	-0.007809
region_AS	-0.005031	-0.246581	1.000000	-0.025256	-0.010598	-0.023226	-0.000653	-0.000308
region_EU	-0.013071	-0.640643	-0.025256	1.000000	-0.027534	-0.060343	-0.001697	0.012189
region_OC	-0.005485	-0.268816	-0.010598	-0.027534	1.000000	-0.025320	-0.000712	-0.000336
...	...	...	...	...	...	...	...	...
opinion_infection	-0.000291	0.027595	-0.032634	-0.039817	-0.025851	0.026932	-0.000604	-0.001031
opinion_mortality	0.002601	0.039730	-0.013618	-0.024279	-0.007565	-0.026287	0.001233	-0.000889
risk_infection	0.005696	-0.021509	0.012276	0.001603	-0.008039	0.028549	0.003543	-0.000924
risk_mortality	0.007101	-0.009360	-0.000332	0.014879	-0.003499	-0.001348	-0.000964	0.002136
covid19_positive	0.008609	-0.022259	0.015703	0.015187	-0.002251	0.010156	0.003542	-0.000149

297 rows × 297 columns

In [185…

```
print("------ Los atributos con correlación positiva ------\n")

print("\t\t RISK_INFECTION\n")
print(correlacion['risk_infection'].nlargest(10))
print("\n\t\t RISK_MORTALITY\n")
print(correlacion['risk_mortality'].nlargest(10))
print("\n\t\t COVID19_POSITIVE\n")
print(correlacion['covid19_positive'].nlargest(20))

print("\n------ Los atributos con correlación negativa ------\n")

print("\t\t RISK_INFECTION\n")
print(correlacion['risk_infection'].nsmallest(9))
print("\n\t\t RISK_MORTALITY\n")
print(correlacion['risk_mortality'].nsmallest(9))
print("\n\t\t COVID19_POSITIVE\n")
print(correlacion['covid19_positive'].nsmallest(19))
```

```
------ Los atributos con correlación positiva ------

                 RISK_INFECTION

risk_infection                1.000000
covid19_contact               0.694033
covid19_symptoms              0.579462
working_travel critical       0.367492
contacts_count                0.314104
covid19_positive              0.296484
health_worker                 0.253542
opinion_infection             0.172204
heart_disease                 0.103465
working_travel non critical   0.100207
Name: risk_infection, dtype: float64

                 RISK_MORTALITY

risk_mortality    1.000000
age_80_90         0.450191
age_70_80         0.371809
diabetes          0.280225
heart_disease     0.272972
age_60_70         0.236725
age_90_100        0.231796
hypertension      0.201807
working_never     0.177739
```

```
kidney_disease    0.172252
Name: risk_mortality, dtype: float64

                COVID19_POSITIVE

covid19_positive           1.000000
risk_infection             0.296484
covid19_symptoms           0.106810
covid19_contact            0.056084
nursing_home               0.050504
opinion_infection          0.041456
risk_mortality             0.039415
age_90_100                 0.027267
worried                    0.026619
income                     0.025854
health_worker              0.024239
immigrant_native           0.023954
age_80_90                  0.021893
rate_reducing_mask_type    0.021580
kidney_disease             0.021554
country_PK                 0.021466
public_transport_count     0.020526
race_asian                 0.020163
country_IT                 0.019269
insurance_yes              0.018682
Name: covid19_positive, dtype: float64

------ Los atributos con correlación negativa ------

                RISK_INFECTION

working_stopped             -0.240719
working_never               -0.122274
working_home                -0.074286
rate_reducing_risk_single   -0.071865
rate_reducing_risk_house    -0.069150
immigrant_blank             -0.043481
race_other                  -0.042876
insurance_blank             -0.040376
age_60_70                   -0.033336
Name: risk_infection, dtype: float64

                RISK_MORTALITY

age_30_40           -0.206583
age_20_30           -0.168831
sex_female          -0.145226
opinion_infection   -0.101063
working_stopped     -0.076895
race_other          -0.071150
immigrant_blank     -0.071118
insurance_blank     -0.069171
contacts_count      -0.069054
Name: risk_mortality, dtype: float64

                COVID19_POSITIVE

race_other                          -0.027874
immigrant_blank                     -0.027825
rate_reducing_risk_single           -0.026773
insurance_blank                     -0.025510
region_AN                           -0.022259
rate_reducing_risk_house            -0.021935
country_US                          -0.018610
sex_female                          -0.014930
working_stopped                     -0.010956
```

```
lsd                                  -0.010864
cannabis                             -0.008693
alcohol                              -0.007655
prescription_medication              -0.007595
ip_latitude                          -0.007561
mdma                                 -0.007236
rate_government_action               -0.007153
smoking_yesmedium                    -0.005700
cocaine                              -0.005267
rate_reducing_risk_single_sanitizer  -0.004746
Name: covid19_positive, dtype: float64
```

Al realizar la estadística descriptiva, observamos que la correlación respecto al covid19_positive, tiene una correlación positiva con las edades avanzadas, por lo que se optó por realizar el filtrado explicado anterior, y a partir de este dataset, realizar los siguientes procedimientos.

In [1]:
```
%matplotlib notebook
import pandas as pd
import numpy as np

df_b = pd.read_csv("recursos/N-Equilibrado.csv", ';')
df_b
```

Out[1]:

	region_AF	region_AN	region_AS	region_EU	region_OC	region_SA	pais_AM	pais_AR	pais_AT	pais_AU	...
0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	..
1	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	..
2	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..
3	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..
4	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..
...	...	...	...	...	...	...	...	...	...	...	..
645	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..
646	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..
647	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	..
648	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..
649	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..

650 rows × 145 columns

In [72]:
```
df_b_corr = df_b.corr()

print("------ Los atributos con correlación positiva ------\n")

print("\t\t RISK_INFECTION\n")
print(df_b_corr['riesgo_infeccion'].nlargest(20))
print("\n\t\t RISK_MORTALITY\n")
print(df_b_corr['riesgo_mortalidad'].nlargest(20))
print("\n\t\t covid19_positive\n")
print(df_b_corr['covid19_positivo'].nlargest(20))

print("\n------ Los atributos con correlación negativa ------\n")

print("\t\t RISK_INFECTION\n")
print(df_b_corr['riesgo_infeccion'].nsmallest(19))
print("\n\t\t RISK_MORTALITY\n")
print(df_b_corr['riesgo_mortalidad'].nsmallest(19))
```

```
print("\n\t\t covid19_positive\n")
print(df_b_corr['covid19_positivo'].nsmallest(19))
```

```
------ Los atributos con correlación positiva ------

                 RISK_INFECTION

riesgo_infeccion       1.000000
covid19_positivo       0.989362
sintomas_covid19       0.389042
opinion_infeccion      0.316129
numero_conviviente     0.314502
contacto_covid19       0.306065
numero_contactos       0.305052
riesgo_mortalidad      0.289659
inmigrante_native      0.244326
edad_80_90             0.236462
preocupado             0.234544
edad_90_100            0.225131
region_EU              0.218719
asilo                  0.209913
ingresos               0.199412
diabetes               0.193580
sexo_male              0.185530
ip_longitud            0.185490
tipo_sangre_unknown    0.184527
enfemedad_riniones     0.180803
Name: riesgo_infeccion, dtype: float64

                 RISK_MORTALITY

riesgo_mortalidad              1.000000
pais_IT                        0.590926
enfemedad_riniones             0.545788
edad_80_90                     0.485278
enfemedad_higado               0.475916
enfermedad_corazon             0.414269
inmunidad_comprometida         0.394178
vih_positivo                   0.384457
diabetes                       0.338777
region_EU                      0.299095
riesgo_infeccion               0.289659
covid19_positivo               0.288584
sexo_male                      0.264588
edad_90_100                    0.232148
fumando_quit0                  0.228466
ip_longitud                    0.227418
hipertension                   0.220590
otras_enfermedades_cronicas    0.207427
numero_conviviente             0.194872
sintomas_covid19               0.192110
Name: riesgo_mortalidad, dtype: float64

                 covid19_positive

covid19_positivo       1.000000
riesgo_infeccion       0.989362
sintomas_covid19       0.386713
numero_conviviente     0.318980
opinion_infeccion      0.314169
riesgo_mortalidad      0.288584
contacto_covid19       0.286005
numero_contactos       0.281456
inmigrante_native      0.243141
edad_80_90             0.238752
preocupado             0.233474
```

```
edad_90_100            0.227552
region_EU              0.222718
asilo                  0.212170
ingresos               0.199063
ip_longitud            0.191921
sexo_male              0.190770
diabetes               0.184120
tipo_sangre_unknown    0.184111
enfemedad_riniones     0.175977
Name: covid19_positivo, dtype: float64

------ Los atributos con correlación negativa ------

                RISK_INFECTION

tasa_reduccion_riesgo_personal        -0.373983
tasa_reduccion_riesgo_hogar           -0.314912
prescripcion_medica                   -0.282554
edad_60_70                            -0.282289
etnia_other                           -0.265300
inmigrante_blank                      -0.258520
region_AN                             -0.232112
seguro_blank                          -0.222021
pais_US                               -0.218959
sexo_female                           -0.185705
tasa_reduccion_riesgo_desinfectante   -0.168931
tasa_reduccion_mascarilla             -0.161880
alcohol                               -0.158836
trabajando_stopped                    -0.154843
tipo_sangre_op                        -0.148559
canabis                               -0.120810
tasa_ayuda_gubernamental              -0.095325
trabajando_home                       -0.089193
tipo_sangre_an                        -0.073574
Name: riesgo_infeccion, dtype: float64

                RISK_MORTALITY

edad_60_70                                      -0.408177
sexo_female                                     -0.262277
region_AN                                       -0.251065
pais_US                                         -0.206835
tasa_reduccion_riesgo_personal                  -0.204976
tasa_reduccion_riesgo_desinfectante             -0.169715
tasa_reduccion_riesgo_hogar                     -0.165478
fumando_never                                   -0.158780
seguro_blank                                    -0.145444
etnia_other                                     -0.142571
inmigrante_blank                                -0.140015
prescripcion_medica                             -0.121628
tipo_sangre_op                                  -0.108016
tasa_reduccion_mascarilla                       -0.082687
tasa_gasto_gubernamental                        -0.080936
opinion_infeccion                               -0.077432
trabajando_home                                 -0.072160
trabajando_travel critical                      -0.061941
tasa_reduccion_riesgo_personal_lavado_manos     -0.060022
Name: riesgo_mortalidad, dtype: float64

                covid19_positive

tasa_reduccion_riesgo_personal        -0.373807
tasa_reduccion_riesgo_hogar           -0.313073
edad_60_70                            -0.287220
prescripcion_medica                   -0.276676
etnia_other                           -0.265510
```

```
inmigrante_blank                        -0.259551
region_AN                               -0.234734
seguro_blank                            -0.225766
pais_US                                 -0.225438
sexo_female                             -0.190770
tasa_reduccion_riesgo_desinfectante     -0.175901
tasa_reduccion_mascarilla               -0.175460
alcohol                                 -0.158633
tipo_sangre_op                          -0.153362
trabajando_stopped                      -0.147909
canabis                                 -0.117811
tasa_ayuda_gubernamental                -0.098906
trabajando_home                         -0.081969
fumando_quit10                          -0.074465
Name: covid19_positivo, dtype: float64
```

Se realizaron las correlaciones para verificar que los datos no se alteren y mantengan el análisis previo.

In [73]:
```
df_completo = pd.read_csv("recursos/Nexoid-Limpio.csv", ';', low_memory=False)
df_f_edades = pd.read_csv("recursos/N-Filtrado.csv", ';', low_memory=False)
df_f = pd.read_csv("recursos/N-Equilibrado.csv", ';', low_memory=False)

len_limpio = df_completo.groupby(df_completo['covid19_positive'])['covid19_positive'].co
print('Nexoid Total:', df_completo['covid19_positive'].count(), '\n\t', len_limpio)

len_f_edades = df_f_edades.groupby(df_f_edades['covid19_positive'])['covid19_positive'].
print('\nN-Filtrado Total:', df_f_edades['covid19_positive'].count(), '\n\t', len_f_edad

len_f = df_f.groupby(df_f['covid19_positivo'])['covid19_positivo'].count()
print('\nN-Equilibrado Total:', df_f['covid19_positivo'].count(), '\n\t', len_f)
```

```
Nexoid Total: 820575
         covid19_positive
0    818705
1      1870
Name: covid19_positive, dtype: int64

N-Filtrado Total: 74728
         covid19_positive
0.0    74403
1.0      325
Name: covid19_positive, dtype: int64

N-Equilibrado Total: 650
         covid19_positivo
0.0    325
1.0    325
Name: covid19_positivo, dtype: int64
```

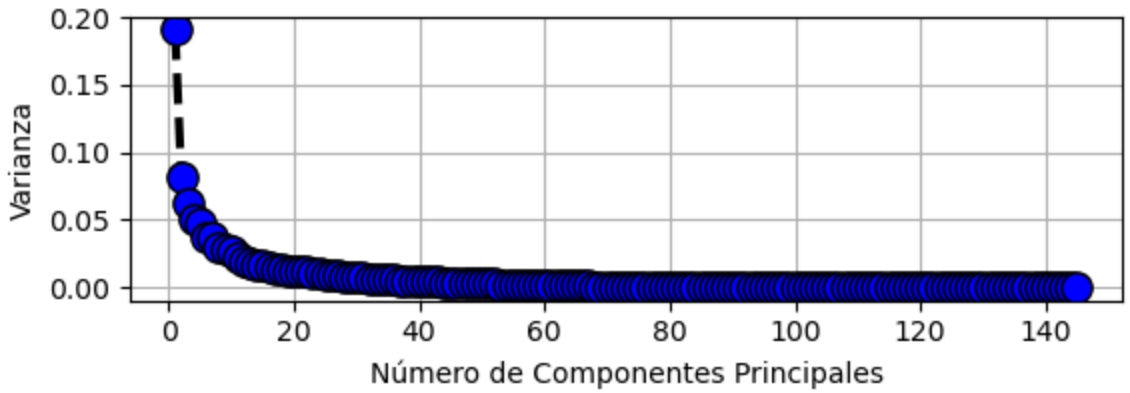
In [2]:
```
# Función para descubrir el mejor K, basado en porcentajes
# respecto al valor anterior. Metodo Elbow
def descubrirXElbow(y, x_min=1, porcentaje=0.8):
    y_len = len(y)
    aux = y[0]
    i = 1
    p_aux = 0
    optimized_x = x_min
    while(i < y_len):
        p = round((y[i] / aux), 4)
        if p_aux != 0:
            if p >= porcentaje:
                optimized_x -= 1
                i = y_len
            else:
                p_aux = 0
```

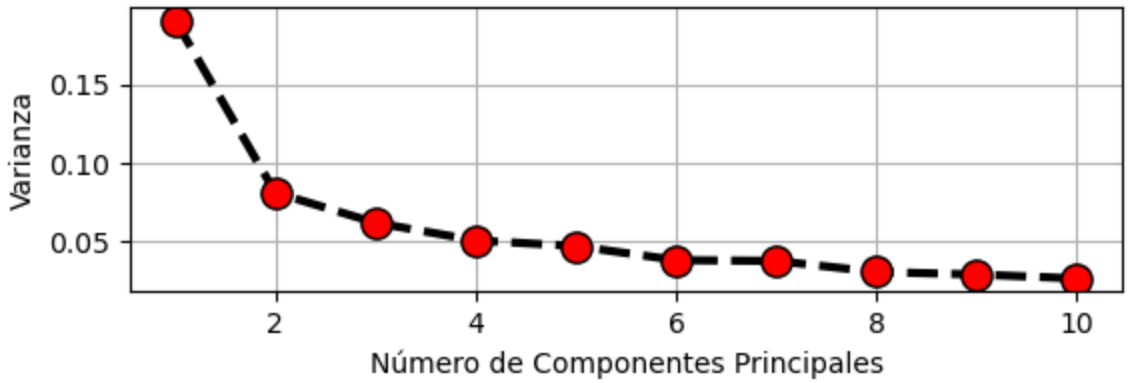
```
        elif p >= porcentaje and p <= 1:
            p_aux = p
        if i < y_len:
            aux = y[i]
            optimized_x += 1
        i += 1

    return optimized_x
```

Reducción de dimensionalidad - PCA

In [3]:

```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

#df_edades = pd.read_csv("recursos/DatasetEdadesPreprocesado.csv", ';', low_memory=False

# Se carga el archivo CSV con los datos preprocesados que creó anteriormente.
df_Preprocesado = df_b

# Cálculo PCA sin componentes.
componentesPrincipales = PCA().fit(df_Preprocesado)
varianza = componentesPrincipales.explained_variance_ratio_

num_CP= range(1, len(varianza) + 1)

# Gráfica de las muestra en 2D.

fig1 = plt.figure('Análisis mediante el gráfico del codo')
fig1.subplots_adjust(hspace=0.6, wspace=0.5)
ax = fig1.add_subplot(2, 1, 1)
ax.plot(num_CP, varianza, color='black', linestyle='dashed', linewidth = 3,
         marker='o', markerfacecolor='blue', markersize=11)

#ax.set_title('Análisis mediante el Método del Codo')
ax.set_xlabel('Número de Componentes Principales')
ax.set_ylabel('Varianza')
ax.grid(True)

num_CP= range(1, 11)
ax = fig1.add_subplot(2, 1, 2)
ax.plot(num_CP, varianza[:10:], color='black', linestyle='dashed', linewidth = 3,
         marker='o', markerfacecolor='red', markersize=11)

#ax.set_title('Análisis mediante el Método del Codo con acercamiento')
ax.set_xlabel('Número de Componentes Principales')
ax.set_ylabel('Varianza')
ax.grid(True)

plt.savefig('recursos/graficas/pcaCodo.png', dpi=300)
plt.show()
```





In [4]:
```
# El mejor componente para realizar reducción de dimensionalidad.
componentes = descubrirXElbow(varianza)
print("La mejor cantidad de componentes principales:", componentes)
```

```
La mejor cantidad de componentes principales: 3
```

In [5]:
```
salida_completo = df_b['covid19_positivo']
df_completo = df_b.drop('covid19_positivo', axis=1)
col_pca = []

for i in range(1, componentes + 1):
    col = "PC" + str(i)
    col_pca.append(col)

pca = PCA(componentes)
df_completo_pca = pca.fit_transform(df_completo)
df_completo_pca = pd.DataFrame(df_completo_pca, columns=col_pca)
com = pd.DataFrame(data=pca.components_, columns=df_completo.columns, index = col_pca)
for idx in com.index:
    print(str(idx) + ':\n', round(com.loc[str(idx)].nlargest(10), 3), '\n')

print("Correlaciónes Componentes-Salida:")
df_completo_pca.corrwith(salida_completo)
```

```
PC1:
 inmigrante_native                0.340
etnia_white                       0.310
ingresos                          0.293
seguro_yes                        0.275
preocupado                        0.261
tasa_reduccion_tipo_mascarilla    0.123
riesgo_infeccion                  0.123
region_EU                         0.090
trabajando_never                  0.077
seguro_no                         0.076
```

```
Name: PC1, dtype: float64

PC2:
 sexo_male              0.416
riesgo_infeccion       0.302
tipo_sangre_unknown    0.187
enfermedad_corazon     0.174
diabetes               0.171
region_EU              0.168
sintomas_covid19       0.157
hipertension           0.145
edad_80_90             0.124
contacto_covid19       0.095
Name: PC2, dtype: float64

PC3:
 sexo_female                   0.430
trabajando_never              0.396
region_EU                     0.162
edad_70_80                    0.137
edad_80_90                    0.135
pais_GB                       0.126
tipo_sangre_unknown           0.122
otras_enfermedades_cronicas   0.098
opinion_mortalidad            0.096
riesgo_infeccion              0.092
Name: PC3, dtype: float64

Correlaciónes Componentes-Salida:
PC1    0.328574
PC2    0.489968
PC3    0.148638
dtype: float64
```

Out[5]:

Modelo Machine Learning orientado a la clasificación - Random Forest

In [6]:

```
# SPLITTING DATASET
from sklearn.model_selection import train_test_split

X = df_b.drop('covid19_positivo', 1)
y = df_b['covid19_positivo']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
#print(X_train.shape)
#print(X_test.shape)
#print(y_train.shape)
#print(y_test.shape)
#print(X_test)
print("Tamaño X_test: ", len(X_test.values))
X_train
```

```
Tamaño X_test:  130
```

Out[6]:

	region_AF	region_AN	region_AS	region_EU	region_OC	region_SA	pais_AM	pais_AR	pais_AT	pais_AU	...
34	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..
432	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..
399	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	..
346	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..
542	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	..
...	...	...	...	...	...	...	...	...	...	...	..

9	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..
359	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..
192	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..
629	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	..
559	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	..

520 rows × 144 columns

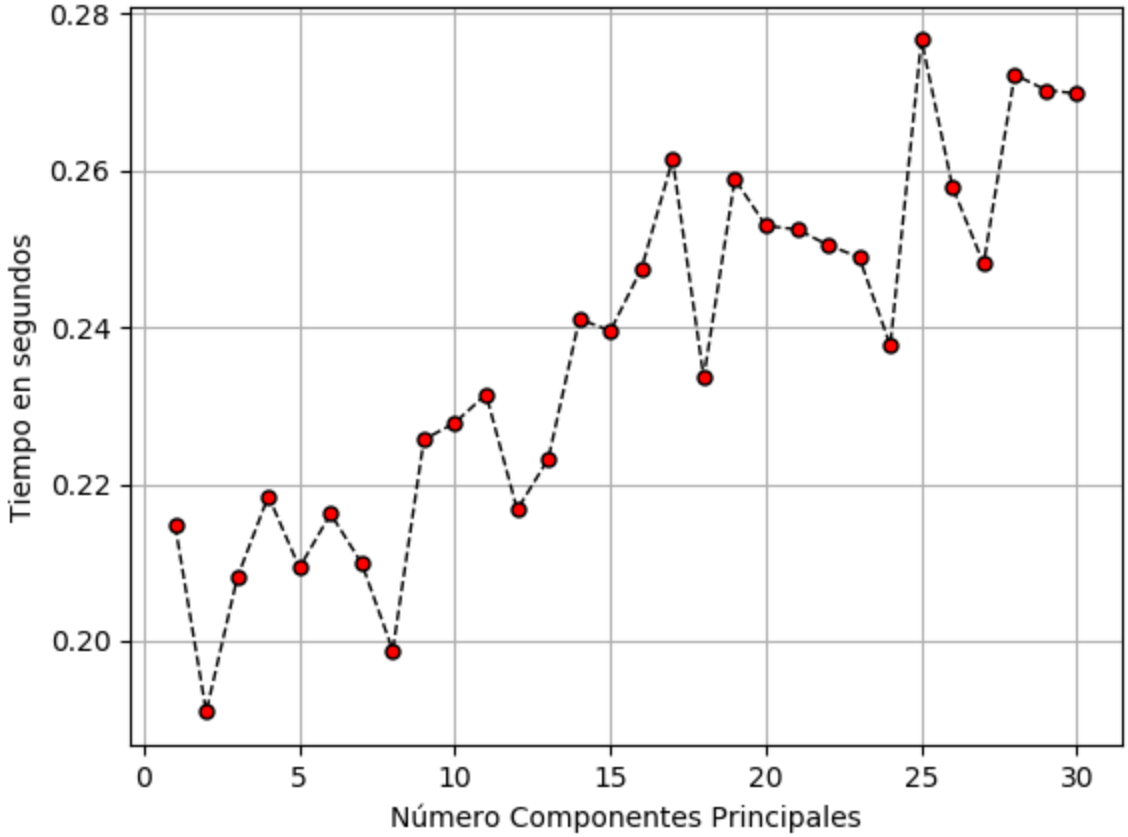
In [12]:

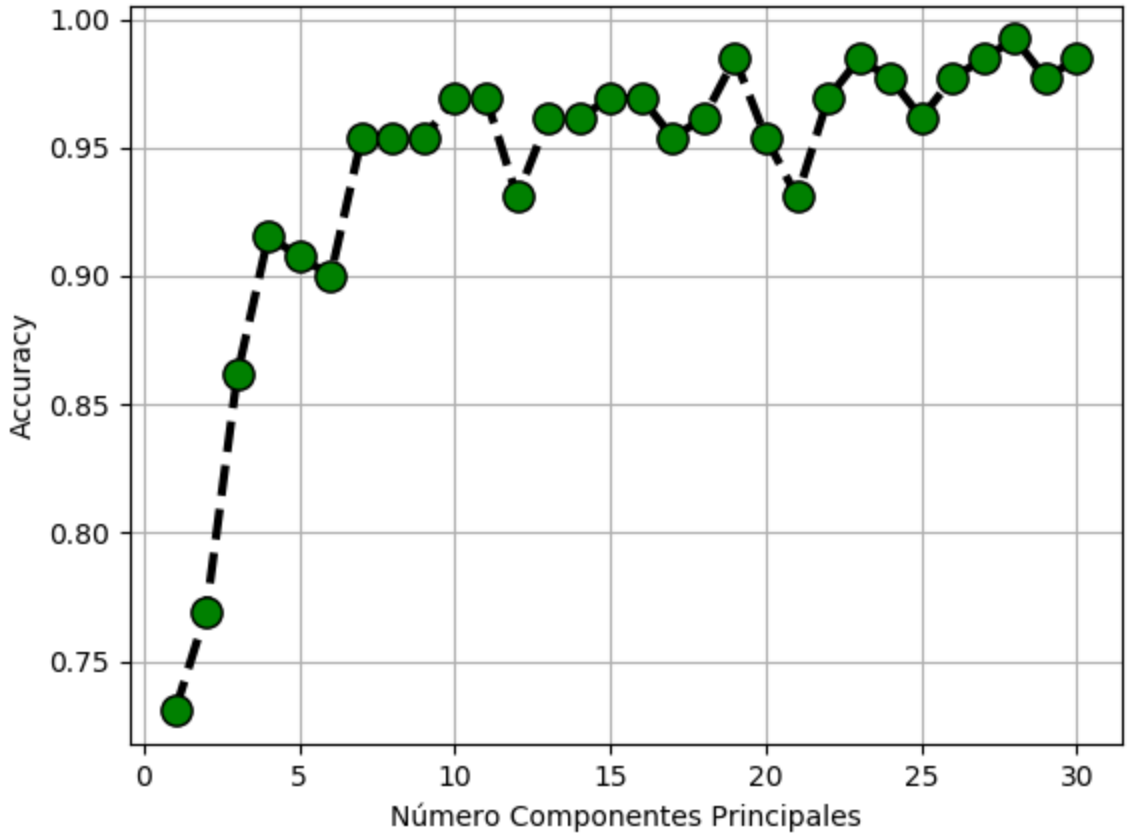
```
from sklearn.preprocessing import StandardScaler, MinMaxScaler, normalize
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
import time
import datetime

x = range(1, min(X_train.shape) + 1)

listaAccuaracy = []
tiempos = []

tiempo_inicial = time.time()

for (comp) in x:
    tiempo_inicial_random = time.time()
    # Aplicamos PCA.
    pca = PCA(n_components=comp)
    X_train_PCA = pca.fit_transform(X_train)
    X_test_PCA = pca.transform(X_test)
    explained_variance = pca.explained_variance_ratio_

     # Se realiza de nuevo el entrenamiento con Random Forest.
    clasificador = RandomForestClassifier(max_depth=15, random_state=0)
    clasificador.fit(X_train_PCA, y_train)

    # Se realiza una nueva predicción en base al Test.
    y_pred = clasificador.predict(X_test_PCA)

    # Finalmente se evalúan los resultados.
    cm = confusion_matrix(y_test, y_pred)
    listaAccuaracy.append(round(accuracy_score(y_test, y_pred,
                                               normalize=True), 5))
    tiempo_final_random = time.time()
    tiempos.append(tiempo_final_random - tiempo_inicial_random)


listaAccuaracy = np.array(listaAccuaracy)

print('Accuracy, '+str(componentes) + ' Componentes principales\n',
     listaAccuaracy[componentes -1:componentes])

maximo = np.amax(listaAccuaracy)
posicion = np.where(listaAccuaracy == np.amax(listaAccuaracy))
print('\nNúmero de componentes: ' + str((posicion[0]+1)))
print('\nMejor Accuracy: ' + str(maximo))

tiempo_final = time.time()
tiempo_ejecucion = (tiempo_final - tiempo_inicial)
print('\nTiempo de ejecución total: ',
      datetime.timedelta(seconds=tiempo_ejecucion))
```

```
Accuracy, 3 Componentes principales
 [0.86154]

Número de componentes: [28 33 34 35 36 40 44 53 55 64 65 74]

Mejor Accuracy: 0.99231

Tiempo de ejecución total:  0:00:44.561071
```

In [80]:
```
# Imprimir arbol en una imágen png
from sklearn.tree import export_graphviz
from sklearn.tree import export_graphviz
import pydot

tree = clasificador.estimators_[50]

export_graphviz(tree, out_file = 'recursos/tree.dot',
                feature_names = X_train.columns, rounded = True, precision = 1)

(graph, ) = pydot.graph_from_dot_file('recursos/tree.dot')
graph.write_png('recursos/tree.png')
```

In [13]:
```
fig6 = plt.figure('Modelo orientado a clasificación (random forest)')
fig6.subplots_adjust(hspace=0.6, wspace=0.5)
ax = fig6.add_subplot(1, 1, 1)
ax.plot(x[:30:], tiempos[:30:], color='black', linestyle='dashed', linewidth = 1,
         marker='o', markerfacecolor='red', markersize=5)

#ax.set_title('Tiempo en base al número de componentes')
ax.set_xlabel('Número Componentes Principales')
ax.set_ylabel('Tiempo en segundos')
ax.grid(True)

plt.savefig('recursos/graficas/tiempo_random-forest.png', dpi=300)
plt.show()

fig8 = plt.figure('Gráfica Accuracy')
fig8.subplots_adjust(hspace=0.6, wspace=0.5)
ax = fig8.add_subplot(1, 1, 1)
ax.plot(x[:30:], listaAccuaracy[:30:], color='black', linestyle='dashed', linewidth = 3,
         marker='o', markerfacecolor='green', markersize=11)

#ax.set_title('Accuracy en base al número de componentes')
ax.set_xlabel('Número Componentes Principales')
ax.set_ylabel('Accuracy')
ax.grid(True)

plt.savefig('recursos/graficas/accuracy-plot.png', dpi=300)
plt.show()
```





In [82]:
```
from sklearn.tree import export_graphviz
from sklearn.tree import export_graphviz
import pydot
```

```
# SPLITTING DATASET
from sklearn.model_selection import train_test_split

X = df_completo_pca
y = salida_completo

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

x = range(1, min(X_train.shape) + 1)

clasificador = RandomForestClassifier(max_depth=15, random_state=0)
clasificador.fit(X_train, y_train)
y_pred = clasificador.predict(X_test)
accuaracy = (round(accuracy_score(y_test, y_pred, normalize=True), 2))
print('Accuracy:', accuaracy)

# Imprimir arbol en una imágen png
tree = clasificador.estimators_[0]

export_graphviz(tree, out_file = 'recursos/tree.dot',
                feature_names = X_train.columns, rounded = True, precision = 1)

(graph, ) = pydot.graph_from_dot_file('recursos/tree.dot')
graph.write_png('recursos/tree.png')
```

```
Accuracy: 0.87
```

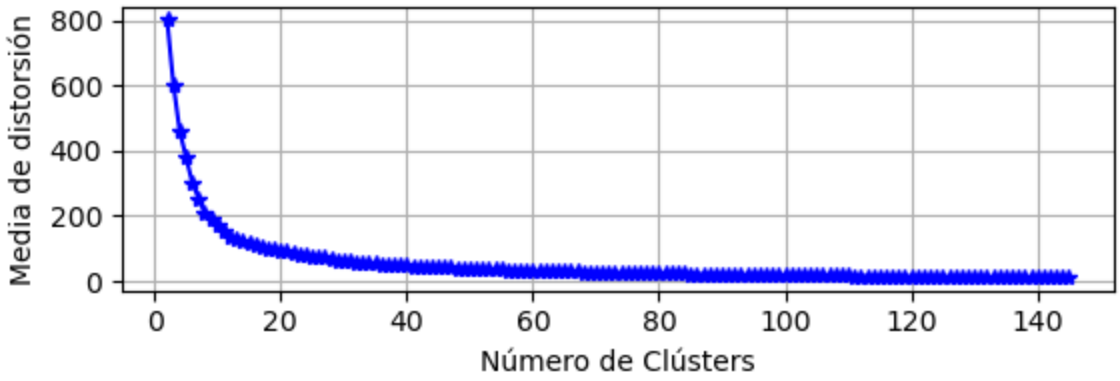
Clustering

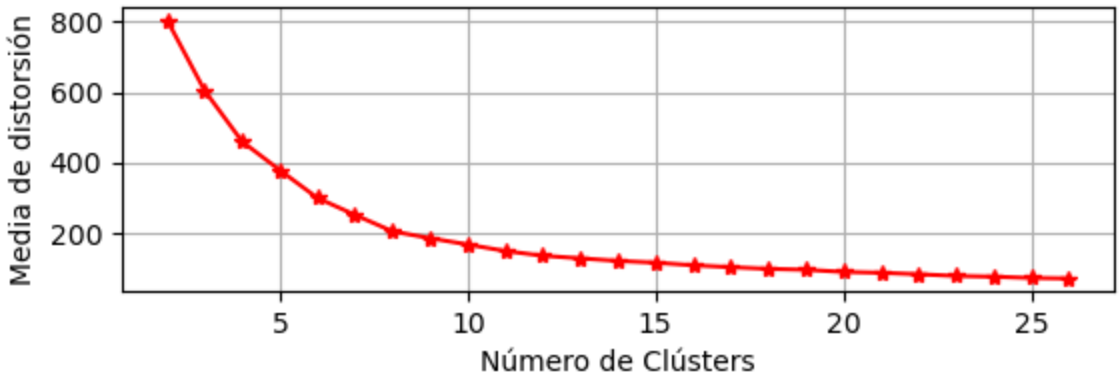
In [8]:
```
from sklearn.decomposition import PCA
from scipy.spatial.distance import cdist
from sklearn.cluster import KMeans
import datetime
import time
from sklearn.utils.testing import ignore_warnings
from sklearn.exceptions import ConvergenceWarning

@ignore_warnings(category=ConvergenceWarning)

def getDistortions(X, k_max):
    K_range=range(2, k_max + 1, 1)
    distortions=[]
    KMeanTime = []
    for i in K_range:
        tiempo_inicial = time.time()
        kmeanModel = KMeans(n_clusters=i)
        kmeanModel.fit(X)
        r = kmeanModel.inertia_
        distortions.append(round(r, 4))
        tiempo_final = time.time()
        KMeanTime.append((tiempo_final - tiempo_inicial))
    return distortions, KMeanTime

#Kmeans Clustering
def doKmeans(X, nclust=2, init='k-means++', max_iter=100,
             tol=0.0001, random_state=0, algorithm='full'):
    model = KMeans(nclust, init=init, random_state=random_state)
    model.fit(X)
    clust_labels = model.predict(X)
    cent = model.cluster_centers_
    return (clust_labels, cent)
```

In [9]:
```
%matplotlib notebook
import matplotlib.pyplot as plt
```

```
k_max = min(df_b.shape)

df_s = df_completo_pca.values

tiempo_inicial = time.time()

distorsiones, distorsionesTime = getDistortions(df_s, k_max)

tiempo_final = time.time()
tiempo_ejecucion = (tiempo_final - tiempo_inicial)
df_s = pd.DataFrame(data=df_s)

print('Tiempo de ejecución total: ', datetime.timedelta(seconds=tiempo_ejecucion))

K_range = np.arange(2, k_max + 1)
fig3=plt.figure('Distorsión-Media')
fig3.subplots_adjust(hspace=0.6, wspace=0.5)

ex = fig3.add_subplot(2,1,1)
ex.plot(K_range, distorsiones[:k_max - 1:1], 'b*-')

ex.grid(True)
ex.set_xlabel('Número de Clústers')
ex.set_ylabel('Media de distorsión')
#ex.set_title('Distorsión Media - Método Elbow')

K_range = K_range[:25:]

ex = fig3.add_subplot(2,1,2)
ex.plot(K_range, distorsiones[:25:], 'r*-')

ex.grid(True)
ex.set_xlabel('Número de Clústers')
ex.set_ylabel('Media de distorsión')
#ex.set_title('Distorsión Media - Método Elbow con acercamiento')

plt.savefig('recursos/graficas/distorsion-media.png', dpi=300)
plt.show()
```

```
Tiempo de ejecución total:  0:00:32.005297
```





In [17]:
```
K = descubrirXElbow(distorsiones, x_min=2, porcentaje=0.81)
#print(distortions)
print('\nEl mejor K: ', K)
```

```
El mejor K:  6
```

In [18]:
```
#pca = PCA(componentes)

#Esta es la matriz de componentes principales
#X_transformed = pca.fit_transform(df_b)

clust_labels, cent = doKmeans(df_completo_pca, K, 'k-means++', random_state=0)
kmeans_pca = pd.DataFrame(clust_labels, columns=['Grupos'])
print('Grupos con ', K, ' Clusters.\n')
kmeans_pca
```

```
Grupos con  6  Clusters.
```

Out[18]:

	Grupos
0	5
1	5
2	2
3	1
4	1
...	...
645	4
646	4

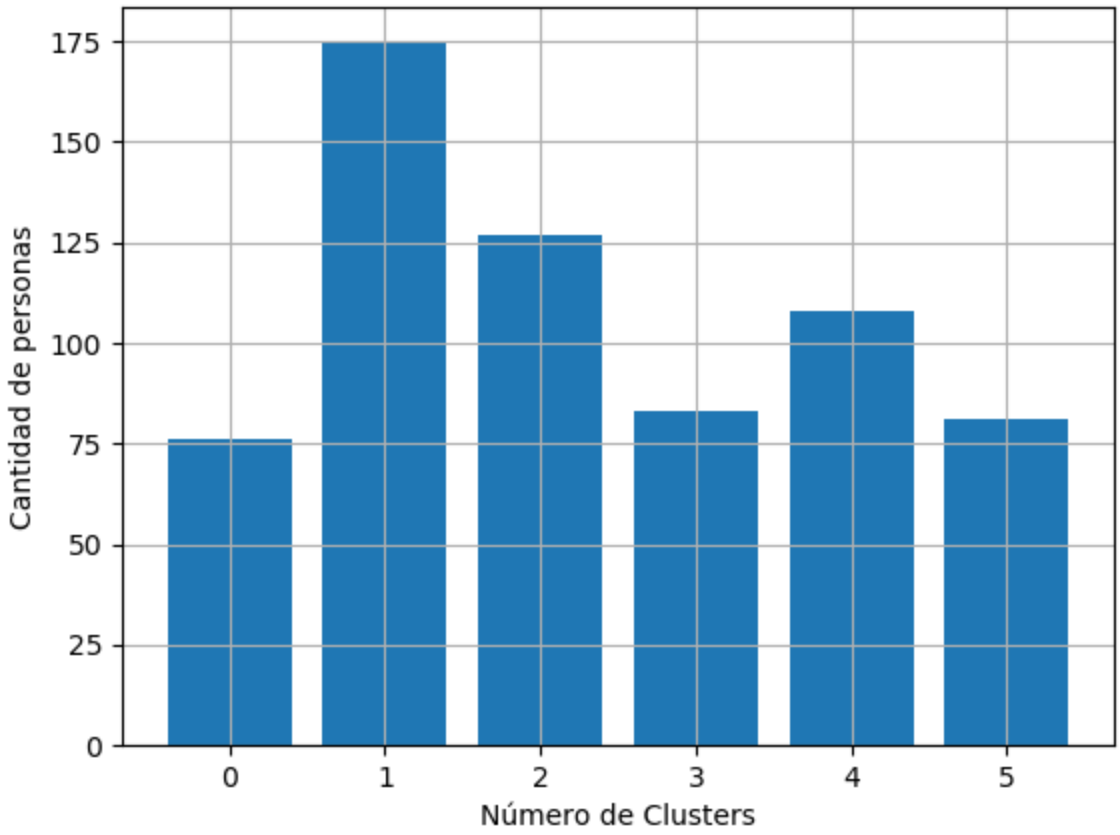
647	3
648	4
649	4

650 rows × 1 columns

Estadística descriptiva con los clusters con mayor cantidad de datos

In [19]:
```
import random
# Realizar estadística descriptiva de cada cluster
df_cluster = pd.concat([df_b, kmeans_pca], axis=1)

UserGrupoK_pca=kmeans_pca.groupby(kmeans_pca.Grupos).Grupos.count()

UserGrupoK_pca=UserGrupoK_pca.sort_values(ascending=False, inplace=False, kind='quicksor
print(UserGrupoK_pca)

x_range = UserGrupoK_pca.index

number_of_colors = len(x_range)

colors = ["#"+''.join([random.choice('0123456789ABCDEF') for j in range(6)])
             for i in range(number_of_colors)]

fig20 = plt.figure('Cantidad de muestras por Cluster')
ex = fig20.add_subplot(1,1,1)
ex.bar(x_range, UserGrupoK_pca)

ex.grid(True)
ex.set_ylabel('Cantidad de personas')
ex.set_xlabel('Número de Clusters')
#ex.set_title('Cantidad de personas por Cluster')

plt.savefig('recursos/graficas/persona-cluster.png', dpi=300)
plt.show()
```

```
Grupos
1    175
2    127
4    108
3     83
5     81
0     76
Name: Grupos, dtype: int64
```



In [88]:
```
df_cluster_1 = df_cluster[(((df_cluster['Grupos'] == 1)))]

des_1 = df_cluster_1.describe()
var = df_cluster_1.var()
print('Media:\n\n', des_1.loc['mean'].nlargest(15))
print('\nDesviación Típica:\n\n', des_1.loc['std'].nlargest(15))
print('\nVarianza:\n\n', var.where(var != 0.0).nsmallest(15))

cov_positivo = df_cluster_1#[(df_cluster_1['covid19_positive'] == 1.0)]

correlacion_c1 = cov_positivo.corr()

print("\n------ Los atributos con correlación positiva ------\n")

print(correlacion_c1['covid19_positivo'].nlargest(20))

print("\n------ Los atributos con correlación negativa ------\n")

print(correlacion_c1['covid19_positivo'].nsmallest(19))
```

```
Media:

 seguro_blank                      1.000000
etnia_other                       1.000000
Grupos                            1.000000
sexo_female                       0.994286
inmigrante_blank                  0.994286
region_AN                         0.954286
pais_US                           0.885714
edad_60_70                        0.777143
ip_latitud                        0.746165
tasa_reduccion_riesgo_personal    0.731429
tasa_reduccion_riesgo_hogar       0.692857
fumando_never                     0.605714
```

```
tasa_reduccion_mascarilla         0.593143
tasa_ayuda_gubernamental          0.565714
trabajando_never                  0.502857
Name: mean, dtype: float64

Desviación Típica:

 trabajando_never               0.501427
fumando_never                   0.490099
tipo_sangre_op                  0.479669
hipertension                    0.467815
trabajando_stopped              0.435079
edad_60_70                      0.417357
tipo_sangre_ap                  0.405383
fumando_quit10                  0.387659
otras_enfermedades_cronicas     0.387659
diabetes                        0.382885
edad_70_80                      0.372891
tipo_sangre_unknown             0.367658
covid19_positivo                0.367658
riesgo_infeccion                0.366613
trabajando_travel critical      0.362259
Name: std, dtype: float64

Varianza:

 tasa_gasto_gubernamental                                        6.074427e-31
mdma                                                            1.381500e-04
lsd                                                             1.526729e-04
cocaina                                                         1.568232e-04
riesgo_mortalidad                                               2.074941e-04
tasa_reduccion_tipo_mascarilla                                  2.285714e-04
anfetaminas                                                     2.658639e-04
ingresos                                                        3.571429e-04
tasa_reduccion_riesgo_personal_distanciamiento_social           3.571429e-04
tasa_reduccion_riesgo_personal_lavado_manos                     3.571429e-04
preocupado                                                      3.657143e-03
tipo_sangre_abn                                                 5.714286e-03
region_EU                                                       5.714286e-03
pais_GB                                                         5.714286e-03
trabajando_unknow                                               5.714286e-03
dtype: float64

------ Los atributos con correlación positiva ------

covid19_positivo              1.000000
riesgo_infeccion              0.955455
opinion_infeccion             0.360339
sintomas_covid19              0.346069
contacto_covid19              0.202986
trabajando_travel critical    0.201945
imc                           0.193144
numero_contactos              0.175080
edad_90_100                   0.173702
fumando_never                 0.160751
peso                          0.151059
ip_latitud                    0.141432
fumando_yeslight              0.110090
region_AN                     0.095523
riesgo_mortalidad             0.084144
diabetes                      0.083285
pais_US                       0.058790
edad_80_90                    0.039520
tipo_sangre_on                0.039193
tasa_reduccion_mascarilla     0.037021
Name: covid19_positivo, dtype: float64
```

```
------ Los atributos con correlación negativa ------

prescripcion_medica              -0.196990
hipertension                     -0.165735
enfermedad_pulmonar              -0.147777
alcohol                          -0.140074
tasa_reduccion_riesgo_personal   -0.133169
tasa_reduccion_riesgo_hogar      -0.130032
fumando_quit10                   -0.125809
canabis                          -0.119233
fumando_yesmedium                -0.107443
anfetaminas                      -0.104816
enfermedad_corazon               -0.101622
region_SA                        -0.082234
pais_BR                          -0.082234
altura                           -0.081718
asma                             -0.077504
trabajando_stopped               -0.073294
fumando_vape                     -0.066750
fumando_quit0                    -0.066750
edad_60_70                       -0.065919
Name: covid19_positivo, dtype: float64
```

In [89]:
```
df_cluster_2 = df_cluster[(((df_cluster['Grupos'] == 2)))]
des_2 = df_cluster_2.describe()

print('Media:\n\n', des_2.loc['mean'].nlargest(15))
print('\nDesviación Típica:\n\n', des_2.loc['std'].nlargest(15))
print('\nVarianza:\n\n', df_cluster_2.var().nlargest(15))

correlacion_c2 = df_cluster_2.corr()

print("\n------ Los atributos con correlación positiva ------\n")

print(correlacion_c2['covid19_positivo'].nlargest(20))

print("\n------ Los atributos con correlación negativa ------\n")

print(correlacion_c2['covid19_positivo'].nsmallest(19))
```

```
Media:

 Grupos                                                 2.000000
seguro_blank                                            1.000000
inmigrante_blank                                        1.000000
sexo_male                                               0.992126
etnia_other                                             0.992126
region_AN                                               0.913386
pais_US                                                 0.866142
ip_latitud                                              0.737303
edad_60_70                                              0.732283
tasa_reduccion_riesgo_hogar                             0.677165
tasa_reduccion_riesgo_personal                          0.671260
tasa_ayuda_gubernamental                                0.562992
fumando_never                                           0.503937
tasa_control_gubernamental                              0.501969
tasa_reduccion_riesgo_personal_distanciamiento_social   0.500000
Name: mean, dtype: float64

Desviación Típica:

 fumando_never                0.501965
covid19_positivo             0.496331
hipertension                 0.493680
trabajando_stopped           0.486796
```

```
riesgo_infeccion              0.480253
tipo_sangre_unknown           0.452465
edad_60_70                    0.444523
trabajando_never              0.444523
edad_70_80                    0.421429
fumando_quit10                0.421429
tipo_sangre_ap                0.410766
diabetes                      0.405098
tipo_sangre_op                0.405098
trabajando_travel critical    0.393040
enfermedad_corazon            0.365696
Name: std, dtype: float64

Varianza:

 fumando_never                 0.251969
covid19_positivo              0.246344
hipertension                  0.243720
trabajando_stopped            0.236970
riesgo_infeccion              0.230643
tipo_sangre_unknown           0.204724
edad_60_70                    0.197600
trabajando_never              0.197600
edad_70_80                    0.177603
fumando_quit10                0.177603
tipo_sangre_ap                0.168729
diabetes                      0.164104
tipo_sangre_op                0.164104
trabajando_travel critical    0.154481
enfermedad_corazon            0.133733
dtype: float64

------ Los atributos con correlación positiva ------

covid19_positivo              1.000000
riesgo_infeccion              0.993390
sintomas_covid19              0.487674
contacto_covid19              0.472484
numero_contactos              0.414663
trabajando_travel critical    0.398509
inmunidad_comprometida        0.244787
opinion_infeccion             0.237568
numero_conviviente            0.232299
enfemedad_riniones            0.209673
ip_latitud                    0.184196
tipo_sangre_bn                0.147070
riesgo_mortalidad             0.143603
otras_enfermedades_cronicas   0.134893
tipo_sangre_unknown           0.130510
fumando_never                 0.120650
mdma                          0.112431
cocaina                       0.112417
region_AF                     0.103581
pais_EG                       0.103581
Name: covid19_positivo, dtype: float64

------ Los atributos con correlación negativa ------

tasa_reduccion_riesgo_hogar          -0.420791
tasa_reduccion_riesgo_personal       -0.392002
tasa_ayuda_gubernamental             -0.332623
prescripcion_medica                  -0.239261
trabajando_never                     -0.232260
alcohol                              -0.193375
region_SA                            -0.174117
pais_BR                              -0.174117
```

```
fumando_quit10                          -0.164321
tasa_reduccion_mascarilla               -0.160900
tipo_sangre_op                          -0.160067
trabajando_home                         -0.157443
tipo_sangre_bp                          -0.133173
edad_60_70                              -0.127460
opinion_mortalidad                      -0.117890
tipo_sangre_on                          -0.116450
etnia_other                             -0.103581
tasa_reduccion_riesgo_desinfectante     -0.103581
sexo_male                               -0.103581
Name: covid19_positivo, dtype: float64
```

In [90]:

```
df_cluster_4 = df_cluster[(((df_cluster['Grupos'] == 4)))]
des_4 = df_cluster_4.describe()

print('Media:\n\n', des_4.loc['mean'].nlargest(15))
print('\nDesviación Típica:\n\n', des_4.loc['std'].nlargest(15))
print('\nVarianza:\n\n', df_cluster_4.var().nlargest(15))


correlacion_c4 = df_cluster_4.corr()

print("\n------ Los atributos con correlación positiva ------\n")

print(correlacion_c4['covid19_positivo'].nlargest(20))

print("\n------ Los atributos con correlación negativa ------\n")

print(correlacion_c4['covid19_positivo'].nsmallest(19))
```

```
Media:

 Grupos                                                 4.000000
region_AN                                               0.907407
seguro_yes                                              0.870370
inmigrante_native                                       0.870370
ingresos                                                0.844907
pais_US                                                 0.842593
edad_60_70                                              0.842593
etnia_white                                             0.833333
tasa_reduccion_mascarilla                               0.748148
tasa_reduccion_riesgo_personal_lavado_manos             0.736111
tasa_reduccion_riesgo_personal_distanciamiento_social   0.726852
ip_latitud                                              0.716728
tasa_reducción_riesgo_hogar_distanciamiento_social      0.712963
preocupado                                              0.709259
tasa_reduccion_riesgo_lavado_manos_hogar                0.706019
Name: mean, dtype: float64

Desviación Típica:

 trabajando_never              0.493643
covid19_positivo              0.489771
fumando_never                 0.487557
sexo_female                   0.479774
sexo_male                     0.479774
tipo_sangre_ap                0.476789
riesgo_infeccion              0.468807
hipertension                  0.445255
tipo_sangre_op                0.435031
trabajando_stopped            0.423746
trabajando_travel critical    0.417678
diabetes                      0.411310
tipo_sangre_unknown           0.404629
fumando_quit10                0.382532
```

```
etnia_white                    0.374415
Name: std, dtype: float64

Varianza:

 trabajando_never              0.243683
covid19_positivo               0.239875
fumando_never                  0.237712
sexo_female                    0.230183
sexo_male                      0.230183
tipo_sangre_ap                 0.227328
riesgo_infeccion               0.219780
hipertension                   0.198252
tipo_sangre_op                 0.189252
trabajando_stopped             0.179560
trabajando_travel critical     0.174455
diabetes                       0.169176
tipo_sangre_unknown            0.163724
fumando_quit10                 0.146331
etnia_white                    0.140187
dtype: float64

------ Los atributos con correlación positiva ------

covid19_positivo               1.000000
riesgo_infeccion               0.986399
opinion_infeccion              0.448733
trabajando_travel critical     0.350258
numero_contactos               0.343896
contacto_covid19               0.330017
sintomas_covid19               0.304034
imc                            0.245257
etnia_hispanic                 0.240523
diabetes                       0.234543
tipo_sangre_abp                0.211891
numero_transporte_publico      0.194463
peso                           0.184967
edad_60_70                     0.136178
region_AN                      0.123771
etnia_blank                    0.121187
pais_CL                        0.121187
pais_DO                        0.121187
tipo_sangre_bn                 0.121187
hipertension                   0.116664
Name: covid19_positivo, dtype: float64

------ Los atributos con correlación negativa ------

tasa_reduccion_riesgo_personal_distanciamiento_social   -0.280610
fumando_quit10                                          -0.268816
tasa_reducción_riesgo_hogar_distanciamiento_social      -0.241950
trabajando_never                                        -0.236228
prescripcion_medica                                     -0.229788
tasa_reduccion_mascarilla                               -0.227167
etnia_white                                             -0.203859
alcohol                                                 -0.195454
otras_enfermedades_cronicas                             -0.193476
tasa_reduccion_riesgo_personal_lavado_manos             -0.182249
ingresos                                                -0.166882
ip_longitud                                             -0.154699
anfetaminas                                             -0.148771
lsd                                                     -0.148771
tasa_gasto_gubernamental                                -0.147393
cocaina                                                 -0.146460
mdma                                                    -0.140473
edad_70_80                                              -0.136178
```

```
tasa_reduccion_riesgo_lavado_manos_hogar                -0.134699
Name: covid19_positivo, dtype: float64
```

In [92]:

```
df_cluster_0 = df_cluster[(((df_cluster['Grupos'] == 0)))]
des_0 = df_cluster_0.describe()

print('Media:\n\n', des_0.loc['mean'].nlargest(15))
print('\nDesviación Típica:\n\n', des_0.loc['std'].nlargest(15))
print('\nVarianza:\n\n', df_cluster_0.var().nlargest(15))


correlacion_c0 = df_cluster_0.corr()

print("\n------ Los atributos con correlación positiva ------\n")

print(correlacion_c0['covid19_positivo'].nlargest(20))

print("\n", correlacion_c0['riesgo_mortalidad'].nlargest(20))

print("\n------ Los atributos con correlación negativa ------\n")

print(correlacion_c0['covid19_positivo'].nsmallest(19))
```

```
Media:

 sexo_female                                              1.000000
trabajando_never                                         0.894737
inmigrante_native                                        0.881579
etnia_white                                              0.828947
riesgo_infeccion                                         0.809418
covid19_positivo                                         0.802632
ip_latitud                                               0.720779
fumando_never                                            0.697368
ingresos                                                 0.690789
preocupado                                               0.689474
tasa_reduccion_riesgo_personal_lavado_manos              0.644737
tasa_reduccion_riesgo_personal_distanciamiento_social    0.631579
tasa_reduccion_riesgo_lavado_manos_hogar                 0.631579
tasa_reducción_riesgo_hogar_distanciamiento_social       0.621711
seguro_yes                                               0.578947
Name: mean, dtype: float64

Desviación Típica:

 region_AN              0.501751
pais_US                0.500526
tipo_sangre_unknown    0.497009
seguro_yes             0.497009
edad_70_80             0.477567
seguro_no              0.477567
hipertension           0.477567
region_EU              0.462450
fumando_never          0.462450
asilo                  0.450146
edad_60_70             0.435890
edad_80_90             0.427970
pais_GB                0.419482
tipo_sangre_op         0.410391
covid19_positivo       0.400657
Name: std, dtype: float64

Varianza:

 region_AN              0.251754
pais_US                0.250526
tipo_sangre_unknown    0.247018
```

```
seguro_yes             0.247018
edad_70_80             0.228070
seguro_no              0.228070
hipertension           0.228070
region_EU              0.213860
fumando_never          0.213860
asilo                  0.202632
edad_60_70             0.190000
edad_80_90             0.183158
pais_GB                0.175965
tipo_sangre_op         0.168421
covid19_positivo       0.160526
dtype: float64

------ Los atributos con correlación positiva ------

covid19_positivo       1.000000
riesgo_infeccion       0.998892
numero_conviviente     0.380472
opinion_infeccion      0.338875
numero_contactos       0.334190
asilo                  0.306414
fumando_never          0.249025
sintomas_covid19       0.235640
edad_90_100            0.225259
tipo_sangre_unknown    0.222018
riesgo_mortalidad      0.213998
edad_80_90             0.198491
contacto_covid19       0.170087
trabajando_never       0.153078
diabetes               0.150361
enfermedad_pulmonar    0.145180
seguro_blank           0.145180
inmigrante_native      0.125203
pais_CA                0.116881
tipo_sangre_ap         0.110034
Name: covid19_positivo, dtype: float64

 riesgo_mortalidad             1.000000
enfemedad_higado               0.650288
edad_80_90                     0.569879
pais_IT                        0.490674
enfemedad_riniones             0.484986
inmunidad_comprometida         0.380181
otras_enfermedades_cronicas    0.369782
diabetes                       0.352484
enfermedad_corazon             0.322648
asilo                          0.267872
ip_latitud                     0.225612
riesgo_infeccion               0.219325
covid19_positivo               0.213998
tipo_sangre_unknown            0.182863
region_EU                      0.182504
etnia_black                    0.164123
opinion_mortalidad             0.163670
numero_conviviente             0.159544
pais_HU                        0.153365
canabis                        0.144166
Name: riesgo_mortalidad, dtype: float64

------ Los atributos con correlación negativa ------

prescripcion_medica                                  -0.466257
tasa_gasto_gubernamental                             -0.381422
tasa_reduccion_riesgo_personal_distanciamiento_social   -0.356447
edad_70_80                                           -0.339250
```

```
numero_transporte_publico                              -0.331527
pais_CR                                                -0.331527
fumando_quit10                                         -0.329244
tasa_reduccion_riesgo_personal_lavado_manos            -0.301176
altura                                                 -0.285064
tasa_reduccion_riesgo_desinfectante                    -0.259425
tasa_reducción_riesgo_hogar_distanciamiento_social     -0.244809
tasa_control_gubernamental                             -0.239030
tipo_sangre_bp                                         -0.239029
ip_precision                                           -0.237486
pais_NL                                                -0.232857
pais_DK                                                -0.232857
tipo_sangre_op                                         -0.230466
tasa_reduccion_riesgo_lavado_manos_hogar               -0.200138
alcohol                                                -0.196352
Name: covid19_positivo, dtype: float64
```

In [93]:

```
salida_cluster_1 = df_cluster_1['covid19_positivo']
df_cluster_1_tmp = df_cluster_1.drop('covid19_positivo', axis=1)
df_cluster_1_tmp = df_cluster_1_tmp.drop('Grupos', axis=1)

pca = PCA(3)
df_cluster_1_pca = pca.fit_transform(df_cluster_1_tmp)
df_cluster_1_pca = pd.DataFrame(df_cluster_1_pca, columns = ['PC1', 'PC2', 'PC3'])
com = pd.DataFrame(data=pca.components_, columns=df_cluster_1_tmp.columns, index = ['PC1
print('PC1:\n', round(com.loc['PC1'].nlargest(10), 3))
print('\nPC2:\n', round(com.loc['PC2'].nlargest(10), 3))
print('\nPC3:\n', round(com.loc['PC3'].nlargest(10), 3))

df_cluster_1_pca.corrwith(salida_cluster_1)
```

```
PC1:
 trabajando_never       0.659
hipertension            0.368
edad_70_80              0.198
diabetes                0.146
tipo_sangre_ap          0.091
enfermedad_pulmonar     0.090
pais_US                 0.079
enfermedad_corazon      0.074
fumando_quit10          0.073
fumando_quit0           0.044
Name: PC1, dtype: float64

PC2:
 tipo_sangre_op          0.532
edad_70_80               0.381
fumando_quit10           0.231
opinion_mortalidad       0.136
opinion_infeccion        0.124
sintomas_covid19         0.080
tasa_reduccion_mascarilla  0.070
enfermedad_pulmonar      0.069
trabajando_never         0.063
inmunidad_comprometida   0.060
Name: PC2, dtype: float64

PC3:
 fumando_never          0.746
edad_70_80             0.181
riesgo_infeccion       0.162
tipo_sangre_op         0.154
sintomas_covid19       0.083
edad_80_90             0.073
trabajando_never       0.049
contacto_covid19       0.047
```

```
opinion_infeccion     0.037
trabajando_stopped    0.031
Name: PC3, dtype: float64
```

Out[93]:

```
PC1   -0.184376
PC2    0.313054
PC3   -0.106722
dtype: float64
```

In [94]:

```
salida_cluster_2 = df_cluster_2['covid19_positivo']
df_cluster_2_tmp = df_cluster_2.drop('covid19_positivo', axis=1)
df_cluster_2_tmp = df_cluster_2_tmp.drop('Grupos', axis=1)

pca = PCA(componentes)
df_cluster_2_pca = pca.fit_transform(df_cluster_2_tmp)
df_cluster_2_pca = pd.DataFrame(df_cluster_2_pca, columns=['PC1', 'PC2', 'PC3'])
com = pd.DataFrame(data=pca.components_, columns=df_cluster_2_tmp.columns, index = ['PC1
print('PC1:\n', round(com.loc['PC1'].nlargest(10), 3))
print('\nPC2:\n', round(com.loc['PC2'].nlargest(10), 3))
print('\nPC3:\n', round(com.loc['PC3'].nlargest(10), 3))

df_cluster_2_pca.corrwith(salida_cluster_2)
```

```
PC1:
 riesgo_infeccion              0.533
trabajando_travel critical    0.352
contacto_covid19              0.305
sintomas_covid19              0.297
numero_contactos              0.239
fumando_never                 0.170
tipo_sangre_unknown           0.116
inmunidad_comprometida        0.115
diabetes                      0.107
opinion_infeccion             0.105
Name: PC1, dtype: float64

PC2:
 edad_70_80             0.484
fumando_quit10         0.272
trabajando_stopped     0.235
enfermedad_corazon     0.190
diabetes               0.161
pais_US                0.153
tipo_sangre_op         0.093
region_AN              0.091
enfermedad_pulmonar    0.063
fumando_yeslight       0.059
Name: PC2, dtype: float64

PC3:
 tipo_sangre_op                0.329
trabajando_travel critical    0.244
trabajando_never              0.166
numero_contactos              0.159
fumando_quit10                0.144
asma                          0.141
alcohol                       0.128
edad_60_70                    0.127
fumando_quit5                 0.120
sintomas_covid19              0.113
Name: PC3, dtype: float64
```

Out[94]:

```
PC1    0.123711
PC2   -0.082749
PC3    0.222545
dtype: float64
```

In [95]:

```
salida_cluster_4 = df_cluster_4['covid19_positivo']
```

```
df_cluster_4_tmp = df_cluster_4.drop('covid19_positivo', axis=1)
df_cluster_4_tmp = df_cluster_4_tmp.drop('Grupos', axis=1)

pca = PCA(componentes)
df_cluster_4_pca = pca.fit_transform(df_cluster_4_tmp)
df_cluster_4_pca = pd.DataFrame(df_cluster_4_pca, columns=['PC1', 'PC2', 'PC3'])
com = pd.DataFrame(data=pca.components_, columns=df_cluster_4_tmp.columns, index = ['PC1
print('PC1:\n', round(com.loc['PC1'].nlargest(10), 3))
print('\nPC2:\n', round(com.loc['PC2'].nlargest(10), 3))
print('\nPC3:\n', round(com.loc['PC3'].nlargest(10), 3))

df_cluster_4_pca.corrwith(salida_cluster_4)
```

```
PC1:
 sexo_female                                            0.486
trabajando_never                                        0.209
fumando_quit10                                          0.204
etnia_white                                             0.144
asma                                                    0.142
tasa_gasto_gubernamental                                0.110
tasa_reducción_riesgo_hogar_distanciamiento_social      0.100
tipo_sangre_ap                                          0.099
seguro_yes                                              0.086
inmigrante_native                                       0.085
Name: PC1, dtype: float64

PC2:
 riesgo_infeccion              0.373
sexo_female                    0.372
trabajando_travel critical     0.303
fumando_never                  0.186
opinion_infeccion              0.182
asma                           0.150
edad_60_70                     0.143
numero_contactos               0.131
trabajador_sanitario           0.130
tipo_sangre_ap                 0.119
Name: PC2, dtype: float64

PC3:
 edad_60_70                              0.258
region_SA                               0.208
tipo_sangre_op                          0.203
fumando_quit5                           0.178
etnia_hispanic                          0.171
inmigrante_immigrant                    0.168
tasa_reduccion_riesgo_desinfectante     0.167
fumando_quit10                          0.158
seguro_no                               0.150
tasa_reduccion_mascarilla               0.098
Name: PC3, dtype: float64
```

Out[95]:
```
PC1    0.447842
PC2   -0.211013
PC3    0.335562
dtype: float64
```

In [96]:
```
salida_cluster_0 = df_cluster_0['covid19_positivo']
df_cluster_0_tmp = df_cluster_0.drop('covid19_positivo', axis=1)
df_cluster_0_tmp = df_cluster_0_tmp.drop('Grupos', axis=1)

pca = PCA(componentes)
df_cluster_0_pca = pca.fit_transform(df_cluster_0_tmp)
df_cluster_0_pca = pd.DataFrame(df_cluster_0_pca, columns=['PC1', 'PC2', 'PC3'])
com = pd.DataFrame(data=pca.components_, columns=df_cluster_0_tmp.columns, index = ['PC1
print('PC1:\n', round(com.loc['PC1'].nlargest(10), 3))
print('\nPC2:\n', round(com.loc['PC2'].nlargest(10), 3))
```

```
print('\nPC3:\n', round(com.loc['PC3'].nlargest(10), 3))

df_cluster_0_pca.corrwith(salida_cluster_0)
```

```
PC1:
 region_EU              0.388
pais_GB                0.346
seguro_no              0.346
asilo                  0.148
edad_80_90             0.141
tipo_sangre_unknown    0.130
ip_longitud            0.103
opinion_infeccion      0.086
ip_latitud             0.074
enfermedad_pulmonar    0.070
Name: PC1, dtype: float64

PC2:
 asilo                  0.368
fumando_never          0.294
tipo_sangre_unknown    0.293
pais_US                0.288
region_AN              0.252
numero_conviviente     0.230
edad_90_100            0.200
riesgo_infeccion       0.197
sintomas_covid19       0.150
contacto_covid19       0.145
Name: PC2, dtype: float64

PC3:
 fumando_never           0.324
edad_70_80              0.224
etnia_asian             0.186
diabetes                0.159
inmigrante_immigrant    0.143
sintomas_covid19        0.136
edad_60_70              0.126
tipo_sangre_unknown     0.112
etnia_hispanic          0.101
seguro_no               0.099
Name: PC3, dtype: float64
```

Out[96]:

```
PC1    0.118750
PC2    0.351826
PC3   -0.510864
dtype: float64
```

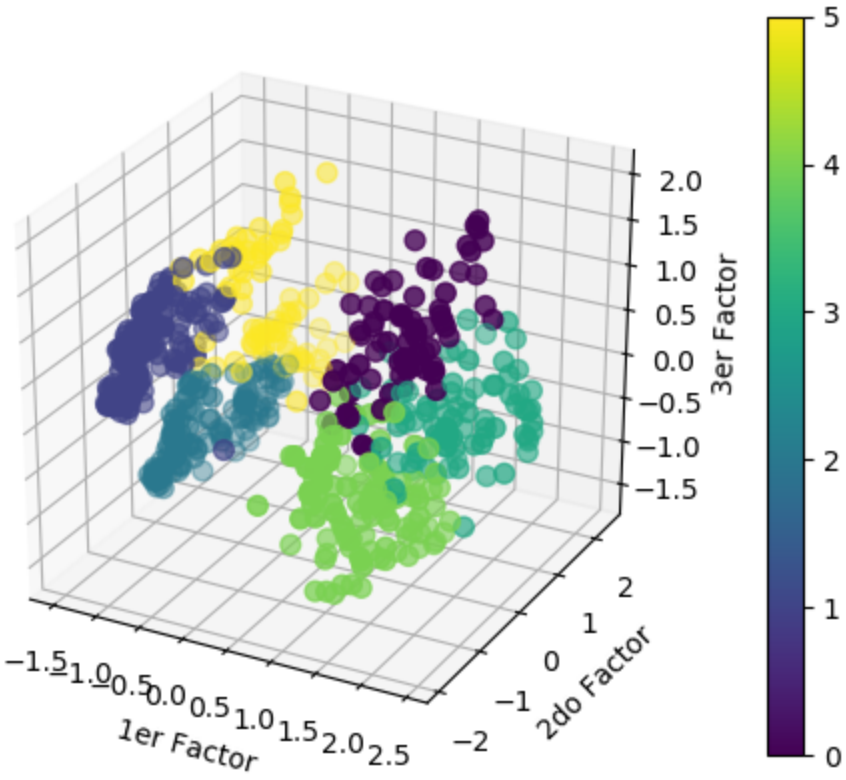
Visualización en 3D

In [24]:

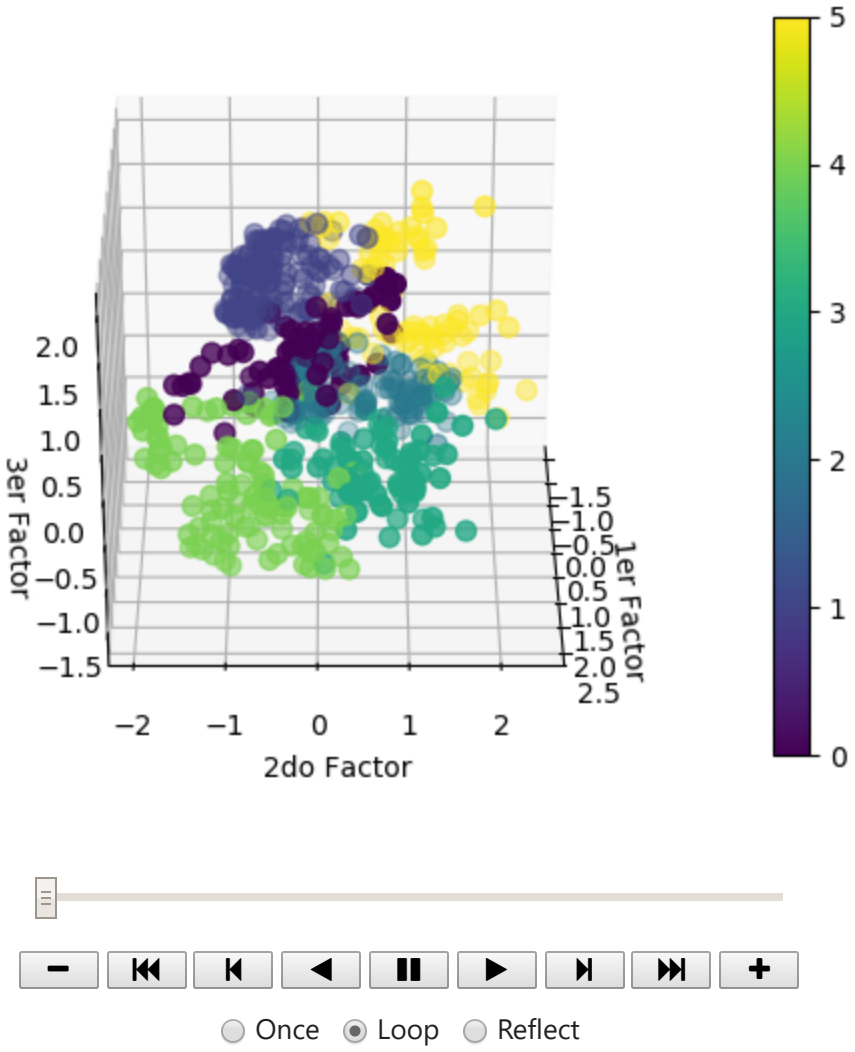
```
%matplotlib notebook

from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt
from IPython.display import HTML
import matplotlib.animation as animation

num_components=3
pca = PCA(num_components)

#Esta es la matriz de componentes principales
X_transformed = pca.fit_transform(df_b)
eigenvalues = pca.explained_variance_

df_c = pd.DataFrame(data=X_transformed, columns = range(num_components))
```

```
print('Varianzas:')
print(eigenvalues.round(3))

explained_variance_ratio_=pca.explained_variance_ratio_

print('\nPorcentaje de varianza de cada dimensión con respecto a la varianza total:')
print(explained_variance_ratio_.round(3))

#Se obtiene el porcentaje acumulado de varianza
print('\nPorcentaje acumulado de varianza:')
explained_variance_ratio_cumsum=explained_variance_ratio_.cumsum()

print(explained_variance_ratio_cumsum.round(3))

correl=df_c.corr()
correl=round(correl,5)
print('\nCorrelación\n', correl)

fig4 = plt.figure("K-Means Clustering 3D")
ax = fig4.add_subplot(1,1,1, projection='3d')

def init():
    scatter = ax.scatter(df_c[0], df_c[1], df_c[2],
                         c=kmeans_pca['Grupos'], s=50)
    #ax.set_title('K-Means Clustering')
    ax.set_xlabel('1er Factor')
    ax.set_ylabel('2do Factor')
    ax.set_zlabel('3er Factor')
    plt.colorbar(scatter)
    plt.savefig('recursos/graficas/k-means-clustering-pca-3D-PCA.png', dpi=300)
    return fig4,

def animate(i):
    ax.view_init(elev=30., azim=3.6*i)
    return fig4

def start():
    ani = animation.FuncAnimation(fig4, animate, init_func=init,
                                  frames=100, interval=100, blit=True)
    return ani

HTML(start().to_jshtml())
```

```
Varianzas:
[1.689 0.717 0.547]

Porcentaje de varianza de cada dimensión con respecto a la varianza total:
[0.191 0.081 0.062]

Porcentaje acumulado de varianza:
[0.191 0.272 0.334]

Correlación
      0    1    2
0  1.0 -0.0  0.0
1 -0.0  1.0 -0.0
2  0.0 -0.0  1.0
```



Out[24]:

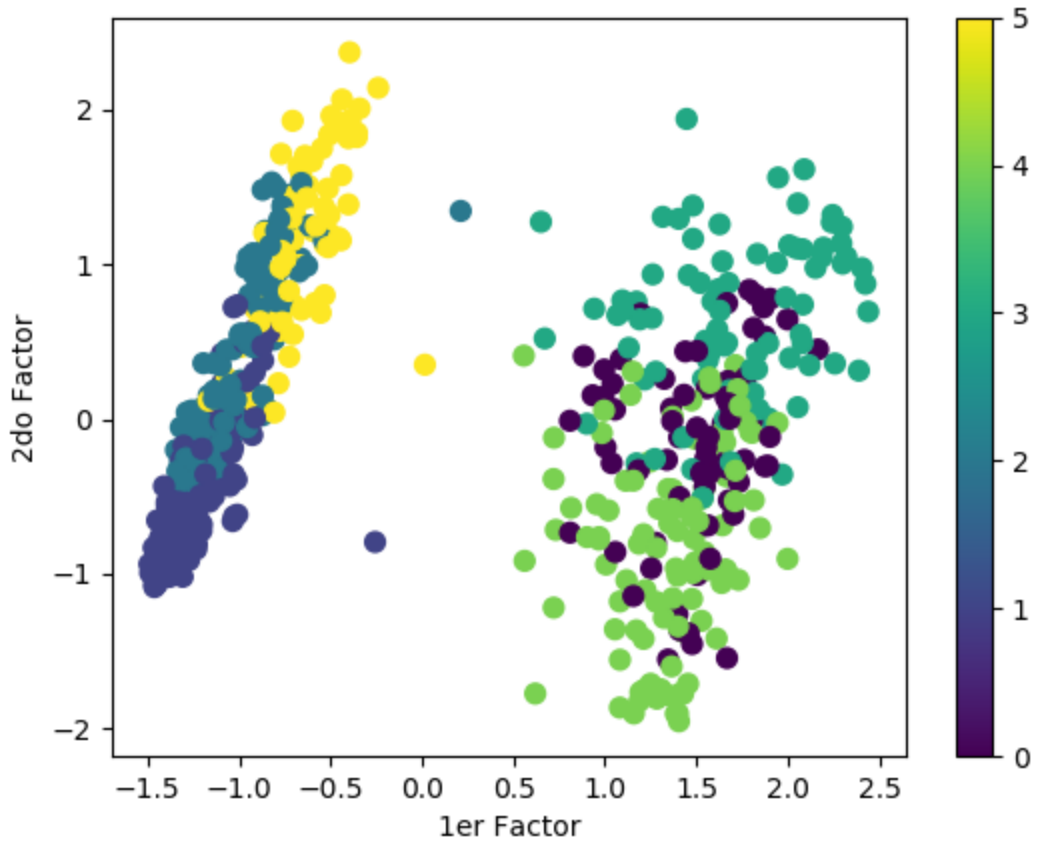


Visualización en 2D

In [21]:
```
num_components=2
pca = PCA(num_components)

#Esta es la matriz de componentes principales
X_transformed = pca.fit_transform(df_b)
eigenvalues = pca.explained_variance_

df_c = pd.DataFrame(data=X_transformed, columns = range(num_components))

print('Varianzas:')
print(eigenvalues.round(3))

explained_variance_ratio_=pca.explained_variance_ratio_

print('\nPorcentaje de varianza de cada dimensión con respecto a la varianza total:')
print(explained_variance_ratio_.round(3))

#Se obtiene el porcentaje acumulado de varianza
print('\nPorcentaje acumulado de varianza:')
explained_variance_ratio_cumsum=explained_variance_ratio_.cumsum()

print(explained_variance_ratio_cumsum.round(3))

correl=df_c.corr()
correl=round(correl, 5)
print('\nCorrelación\n', correl)
```

```
fig5 = plt.figure("K-Means Clustering 2D")
ax = fig5.add_subplot(1,1,1)

scatter1 = ax.scatter(df_c[0], df_c[1],
                      c=kmeans_pca['Grupos'], s=50)
#ax.set_title('K-Means Clustering')
ax.set_xlabel('1er Factor')
ax.set_ylabel('2do Factor')
plt.colorbar(scatter1)
plt.savefig('recursos/graficas/k-means-clustering-pca-2D.png', dpi=300)
plt.show()
```

```
Varianzas:
[1.689 0.717]

Porcentaje de varianza de cada dimensión con respecto a la varianza total:
[0.191 0.081]

Porcentaje acumulado de varianza:
[0.191 0.272]

Correlación
      0    1
0  1.0  0.0
1  0.0  1.0
```

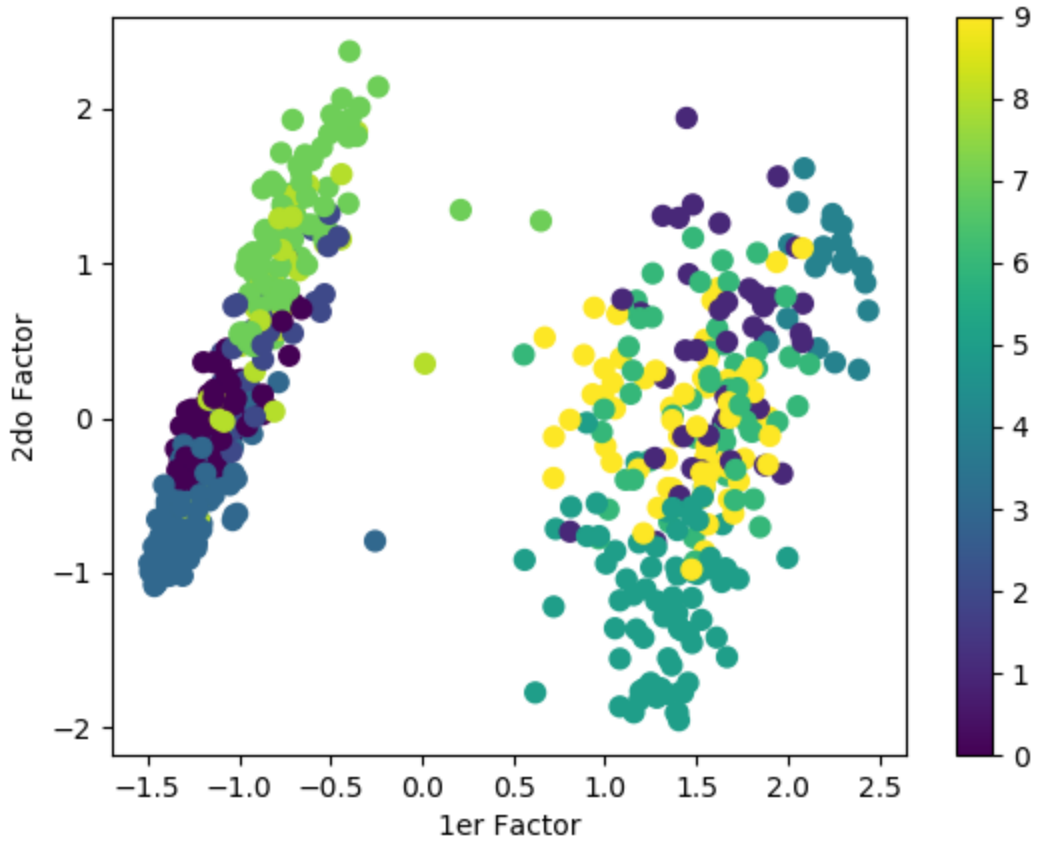


Extra: Visualización en 2D con 10 Clusters

In [22]:

```
K_d = 10
num_components=2
pca = PCA(num_components)

#Esta es la matriz de componentes principales
```

```
X_transformed = pca.fit_transform(df_b)
eigenvalues = pca.explained_variance_

df_c = pd.DataFrame(data=X_transformed, columns = range(num_components))

clust_labels, cent = doKmeans(df_b, K_d, 'k-means++', random_state=0)
kmeans = pd.DataFrame(clust_labels, columns=['Grupos'])

fig7 = plt.figure("K-Means Clustering 2D Disperso")
ax = fig7.add_subplot(1,1,1)

scatter1 = ax.scatter(df_c[0], df_c[1],
                      c=kmeans['Grupos'], s=50)
#ax.set_title('K-Means Clustering con ' +  str(K_d) + ' Clusters')
ax.set_xlabel('1er Factor')
ax.set_ylabel('2do Factor')
plt.colorbar(scatter1)
plt.savefig('recursos/graficas/k-means-clustering-pca-2D-10-clusters.png', dpi=300)
plt.show()
```

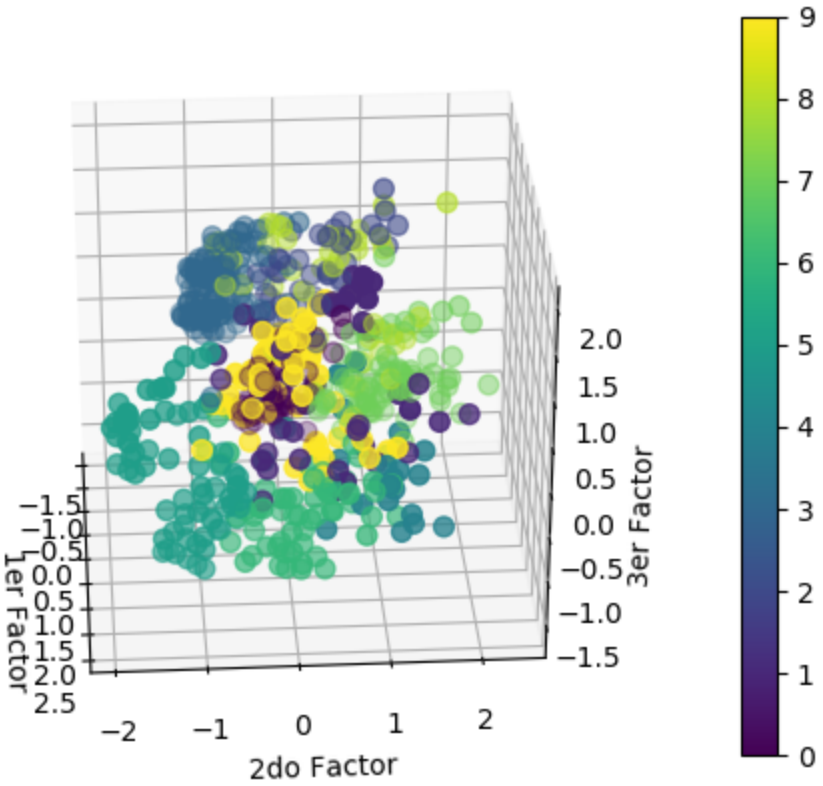


Extra: Visualización en 3D con 10 Clusters

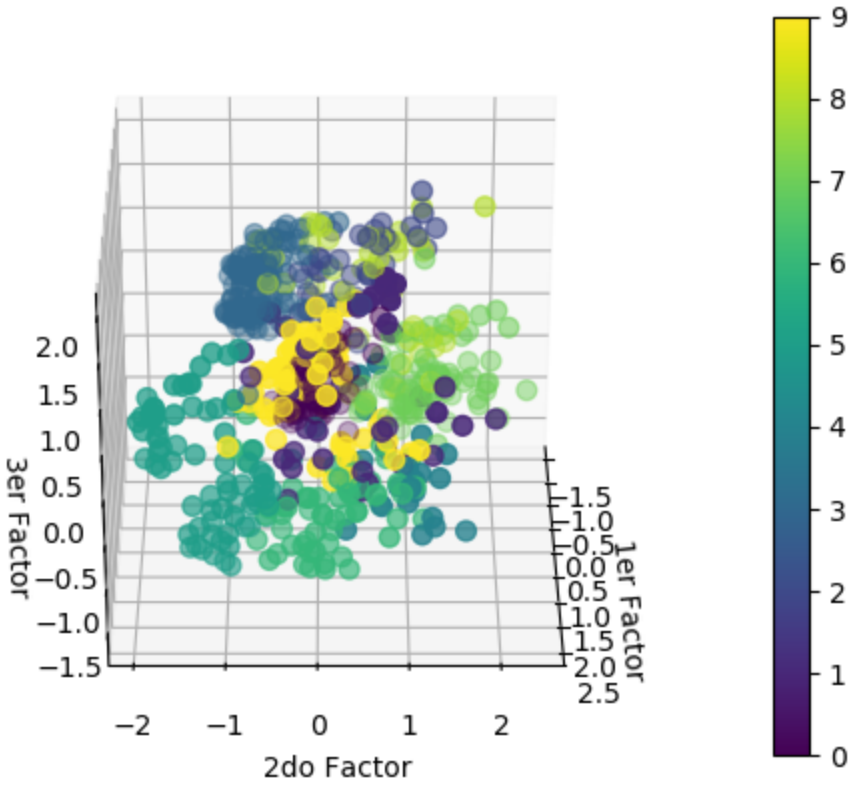
In [23]:

```
%matplotlib notebook

from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt
from IPython.display import HTML
import matplotlib.animation as animation

K_d = 10
num_components=3
pca = PCA(num_components)
```

```
#Esta es la matriz de componentes principales
X_transformed = pca.fit_transform(df_b)
eigenvalues = pca.explained_variance_

df_c = pd.DataFrame(data=X_transformed, columns = range(num_components))

clust_labels, cent = doKmeans(df_b, K_d, 'k-means++', random_state=0)
kmeans = pd.DataFrame(clust_labels, columns=['Grupos'])

figA10 = plt.figure("K-Means Clustering 3D con 10 Clusters")
ax = figA10.add_subplot(1,1,1, projection='3d')

def init():
    scatter = ax.scatter(df_c[0], df_c[1], df_c[2],
                         c=kmeans['Grupos'], s=50)
    #ax.set_title('K-Means Clustering')
    ax.set_xlabel('1er Factor')
    ax.set_ylabel('2do Factor')
    ax.set_zlabel('3er Factor')
    plt.colorbar(scatter)
    plt.savefig('recursos/graficas/k-means-clustering-pca-3D-PCA-10-clusters.png', dpi=3
    return figA10,

def animate(i):
    ax.view_init(elev=30., azim=3.6*i)
    return figA10

def start():
    ani = animation.FuncAnimation(figA10, animate, init_func=init,
                                  frames=100, interval=100, blit=True)
    return ani

HTML(start().to_jshtml())
```



Out[23]:



Once Loop Reflect

In [ ]: