act3_draft

March 18, 2024

[70]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import math
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, DBSCAN
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_samples, silhouette_score
import matplotlib.cm as cm
```

0.1 Información sobre el Dataset

Los cardiotocogramas (CTG) son una manera de evaluar la salud del feto, a través del monitoreo de la frecuencia cardiaca de éste y las contracciones uterinas de la madre, permite el tomar acción y prevenir la mortalidad infantil. Esto, a traves de leer sensores de ultrasonido para conocer el ritmo cardiaco del feto, y un tocógrafo para registrar las contracciones uterinas. [2]

El dataste esta compuesto de 40 variables y 2126 registros extraidos de exámenes de cardiotogramas, que fueron clasificados de dos maneras, una con respecto al patrón morfológico (A, B, C. …) y de acuerdo al estado fetal (N, S, P):

Cabe mencionar que cuando se mencione "SisPorto", se refiere al sistema y evaluación de registros CTG desarrollado por la Universidad de Oporto en Portugal.

0.1.1 Variables: Datos numéricos.

Lista de variables (omitidas las primero cinco columnas para este estudio).

- *LBE: valor base Frecuencia Cardiaca (Médico Experto)
- *LB: valor base Frecuencia Cardiaca (Generado por SisPorto[3])
- *AC: aceleraciones (SisPorto)
- *FM: movimiento fetal (SisPorto)
- *UC: contracciones uterinas (SisPorto)
- *DL: desaceleraciones leves
- *DS: desaceleraciones severas
- *DP: desaceleraciones prolongadas
- *DR: desaceleraciones repetitivas

Producto de un pre'análisis por parte de SisPorto

- *ASTV: porcentaje de tiempo con variabilidad a corto plazo anormal (SisPorto)
- *mSTV: valor medio de la variabilidad a corto plazo (SisPorto)
- *ALTV: porcentaje de tiempo con variabilidad a largo plazo anormal (SisPorto)
- *mLTV: valor medio de la variabilidad a largo plazo (SisPorto)

Números del Histograma

- *Ancho: anchura del histograma
- *Mín: frecuencia baja del histograma
- *Máx: frecuencia alta del histograma
- *Nmax: número de picos del histograma
- *Nzeros: número de ceros del histograma
- *Moda: moda del histograma
- *Media: media del histograma
- *Mediana: mediana del histograma
- *Varianza: varianza del histograma
- *Tendencia: tendencia del histograma: -1=asimétrica a la izquierda; 0=simétrica; 1=asimétrica a la derecha

0.1.2 Clasificación por Patrón Morfológico de 10 Clases, resumida en la columna CLASS

- A: Patrón de sueño tranquilo
- B: Patrón de sueño REM
- C: vigilia tranquila
- D: vigilia activa
- SH: patrón de cambio (A o Susp con cambios)
- AD: patrón acelerativo/decelerativo (situación de estrés)
- DE: patrón decelerativo (estimulación vagal)
- LD: patrón ampliamente decelerativo
- FS: patrón plano-sinusoidal (estado patológico)
- SUSP: patrón sospechoso
- CLASS: código de clase (de 1 a 10) para las clases de A a SUSP

0.1.3 Clasificación por Estado del Feto de 3 Clases, resumida en la columna NSP

- NSP: Normal=1; Sospechoso=2; Patológico=3

[2]:
```
#utilitario para ignorar Warnings, no tomar en cuenta para el reporte final
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

[3]:
```
#utilitario para dar formato a las tablas al estilo 'ggplot' de R
plt.style.use('ggplot')
```

0.2 Análisis Exploratorio de los Datos.

Sin caer en trivialiadades (todos los datos de la tabla son numéricos), haremos una diferención entre ellos de acuerdo a su origen, así, tenemos los datos producto de la Toma de datos de los instrumentos, otros derivados de un cálculo, por ejemplo, un valor promedio, que al provenir de los datos tomados, poseen cierta correlación y por ende podrían inducir a un bias, otros relacionados a lo que parece cierto pre'análsis estadístico a modo de histograma y por último dos tipos de clases. La naturaleza del origen de los datos nos puedan dar indicios acerca de su correlación.

[4]:
```
###Se carga la base de datos completa a ser utilizada para el estudio
data = pd.read_csv('CTG.csv')

###Removemos las primeras cinco columnas que poseen la información␣
 ↪Administrativa del proceso de estudio.
data = data.drop(columns=['FileName', 'Date', 'SegFile', 'b', 'e'])

###Dejaremos a éste punto el resto de los datos, tener en cuenta lo que se␣
 ↪comentó anteriormente en relación aquellos que son producto de la ´Toma de␣
 ↪Datos´

data.head()
```

[4]:

	LBE	LB	AC	FM	UC	ASTV	MSTV	ALTV	MLTV	DL	…	C	D	\
0	120.0	120.0	0.0	0.0	0.0	73.0	0.5	43.0	2.4	0.0	…	0.0	0.0	
1	132.0	132.0	4.0	0.0	4.0	17.0	2.1	0.0	10.4	2.0	…	0.0	0.0	
2	133.0	133.0	2.0	0.0	5.0	16.0	2.1	0.0	13.4	2.0	…	0.0	0.0	
3	134.0	134.0	2.0	0.0	6.0	16.0	2.4	0.0	23.0	2.0	…	0.0	0.0	
4	132.0	132.0	4.0	0.0	5.0	16.0	2.4	0.0	19.9	0.0	…	0.0	0.0	

	E	AD	DE	LD	FS	SUSP	CLASS	NSP
0	0.0	0.0	0.0	0.0	1.0	0.0	9.0	2.0
1	0.0	1.0	0.0	0.0	0.0	0.0	6.0	1.0
2	0.0	1.0	0.0	0.0	0.0	0.0	6.0	1.0
3	0.0	1.0	0.0	0.0	0.0	0.0	6.0	1.0
4	0.0	0.0	0.0	0.0	0.0	0.0	2.0	1.0

[5 rows x 35 columns]

[5]:
```
###Verificamos rápidamente la información general de los datos
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2129 entries, 0 to 2128
Data columns (total 35 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   LBE       2126 non-null   float64
 1   LB        2126 non-null   float64
 2   AC        2126 non-null   float64
```

```
 3   FM          2127 non-null   float64
 4   UC          2127 non-null   float64
 5   ASTV        2127 non-null   float64
 6   MSTV        2127 non-null   float64
 7   ALTV        2127 non-null   float64
 8   MLTV        2127 non-null   float64
 9   DL          2128 non-null   float64
 10  DS          2128 non-null   float64
 11  DP          2128 non-null   float64
 12  DR          2128 non-null   float64
 13  Width       2126 non-null   float64
 14  Min         2126 non-null   float64
 15  Max         2126 non-null   float64
 16  Nmax        2126 non-null   float64
 17  Nzeros      2126 non-null   float64
 18  Mode        2126 non-null   float64
 19  Mean        2126 non-null   float64
 20  Median      2126 non-null   float64
 21  Variance    2126 non-null   float64
 22  Tendency    2126 non-null   float64
 23  A           2126 non-null   float64
 24  B           2126 non-null   float64
 25  C           2126 non-null   float64
 26  D           2126 non-null   float64
 27  E           2126 non-null   float64
 28  AD          2126 non-null   float64
 29  DE          2126 non-null   float64
 30  LD          2126 non-null   float64
 31  FS          2126 non-null   float64
 32  SUSP        2126 non-null   float64
 33  CLASS       2126 non-null   float64
 34  NSP         2126 non-null   float64
dtypes: float64(35)
memory usage: 582.3 KB
```

0.2.1 Descripción de lo visto

Se notan que el número de filas no es uniforme, en su mayoría es de 2126, pero hay columnas con 2127, 2128. Es necesario entender a que se debe, veamos el final de la tabla.

[6]:
```
## Viendo el fondo de la tabla para identificar posibles datos extra.
data.tail()
```

[6]:

	LBE	LB	AC	FM	UC	ASTV	MSTV	ALTV	MLTV	DL	…	C	\
2124	140.0	140.0	1.0	0.0	9.0	78.0	0.4	27.0	7.0	0.0	…	0.0	
2125	142.0	142.0	1.0	1.0	5.0	74.0	0.4	36.0	5.0	0.0	…	0.0	
2126	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	…	NaN	
2127	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	…	NaN	

```
2128    NaN    NaN  NaN  564.0  23.0  87.0   7.0  91.0  50.7  16.0  …  NaN

        D    E   AD   DE   LD   FS  SUSP  CLASS  NSP
2124  0.0  1.0  0.0  0.0  0.0  0.0   0.0    5.0  2.0
2125  0.0  0.0  0.0  0.0  0.0  0.0   0.0    1.0  1.0
2126  NaN  NaN  NaN  NaN  NaN  NaN   NaN    NaN  NaN
2127  NaN  NaN  NaN  NaN  NaN  NaN   NaN    NaN  NaN
2128  NaN  NaN  NaN  NaN  NaN  NaN   NaN    NaN  NaN

[5 rows x 35 columns]
```

Por la naturaleza del estudio hay que darse cuenta que hay una diferencia entre datos nulos NaN y ceros, en algunas columnas, en especial la de ´Toma de Datos´, cero, poseen un valor a ser considerado, lo anterior, de nuevo, depende del tipo de dato, podemos apreciar que las filas extra podrían interpretarse como datos faltantes y por ello aparecen como NaN, de ser ese el caso, es mejor eliminarlos. La razon por la cual no utilizamos algun metodo de inputacion del valor de los datos es porque las filas con valores faltantes representan un porcentaje muy chico del dataset regular, menos del 0.2%

[7]:
```
# Elimina las últimas 3 filas del Dataset
data = data.dropna()
```

[8]:
```
## Reinspeccionando la tabla.
data.tail()
```

[8]:
```
        LBE     LB   AC   FM   UC  ASTV  MSTV  ALTV  MLTV   DL  …    C    D  \
2121  140.0  140.0  0.0  0.0  6.0  79.0   0.2  25.0   7.2  0.0  …  0.0  0.0
2122  140.0  140.0  1.0  0.0  9.0  78.0   0.4  22.0   7.1  0.0  …  0.0  0.0
2123  140.0  140.0  1.0  0.0  7.0  79.0   0.4  20.0   6.1  0.0  …  0.0  0.0
2124  140.0  140.0  1.0  0.0  9.0  78.0   0.4  27.0   7.0  0.0  …  0.0  0.0
2125  142.0  142.0  1.0  1.0  5.0  74.0   0.4  36.0   5.0  0.0  …  0.0  0.0

        E   AD   DE   LD   FS  SUSP  CLASS  NSP
2121  1.0  0.0  0.0  0.0  0.0   0.0    5.0  2.0
2122  1.0  0.0  0.0  0.0  0.0   0.0    5.0  2.0
2123  1.0  0.0  0.0  0.0  0.0   0.0    5.0  2.0
2124  1.0  0.0  0.0  0.0  0.0   0.0    5.0  2.0
2125  0.0  0.0  0.0  0.0  0.0   0.0    1.0  1.0

[5 rows x 35 columns]
```

[9]:
```
###Confirmamos cuantos valores nulos existen por columna
data.isnull().sum()
```

[9]:
```
LBE         0
LB          0
AC          0
FM          0
```

```
UC          0
ASTV        0
MSTV        0
ALTV        0
MLTV        0
DL          0
DS          0
DP          0
DR          0
Width       0
Min         0
Max         0
Nmax        0
Nzeros      0
Mode        0
Mean        0
Median      0
Variance    0
Tendency    0
A           0
B           0
C           0
D           0
E           0
AD          0
DE          0
LD          0
FS          0
SUSP        0
CLASS       0
NSP         0
dtype: int64
```

[10]:
```
###Optenemos una descripción General del Dataset
data.describe().T
```

[10]:

	count	mean	std	min	25%	50%	75%	max
LBE	2126.0	133.303857	9.840844	106.0	126.0	133.0	140.0	160.0
LB	2126.0	133.303857	9.840844	106.0	126.0	133.0	140.0	160.0
AC	2126.0	2.722484	3.560850	0.0	0.0	1.0	4.0	26.0
FM	2126.0	7.241298	37.125309	0.0	0.0	0.0	2.0	564.0
UC	2126.0	3.659925	2.847094	0.0	1.0	3.0	5.0	23.0
ASTV	2126.0	46.990122	17.192814	12.0	32.0	49.0	61.0	87.0
MSTV	2126.0	1.332785	0.883241	0.2	0.7	1.2	1.7	7.0
ALTV	2126.0	9.846660	18.396880	0.0	0.0	0.0	11.0	91.0
MLTV	2126.0	8.187629	5.628247	0.0	4.6	7.4	10.8	50.7
DL	2126.0	1.570085	2.499229	0.0	0.0	0.0	3.0	16.0

DS	2126.0	0.003293	0.057300	0.0	0.0	0.0	0.0	1.0
DP	2126.0	0.126058	0.464361	0.0	0.0	0.0	0.0	4.0
DR	2126.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0
Width	2126.0	70.445908	38.955693	3.0	37.0	67.5	100.0	180.0
Min	2126.0	93.579492	29.560212	50.0	67.0	93.0	120.0	159.0
Max	2126.0	164.025400	17.944183	122.0	152.0	162.0	174.0	238.0
Nmax	2126.0	4.068203	2.949386	0.0	2.0	3.0	6.0	18.0
Nzeros	2126.0	0.323612	0.706059	0.0	0.0	0.0	0.0	10.0
Mode	2126.0	137.452023	16.381289	60.0	129.0	139.0	148.0	187.0
Mean	2126.0	134.610536	15.593596	73.0	125.0	136.0	145.0	182.0
Median	2126.0	138.090310	14.466589	77.0	129.0	139.0	148.0	186.0
Variance	2126.0	18.808090	28.977636	0.0	2.0	7.0	24.0	269.0
Tendency	2126.0	0.320320	0.610829	-1.0	0.0	0.0	1.0	1.0
A	2126.0	0.180621	0.384794	0.0	0.0	0.0	0.0	1.0
B	2126.0	0.272342	0.445270	0.0	0.0	0.0	1.0	1.0
C	2126.0	0.024929	0.155947	0.0	0.0	0.0	0.0	1.0
D	2126.0	0.038100	0.191482	0.0	0.0	0.0	0.0	1.0
E	2126.0	0.033866	0.180928	0.0	0.0	0.0	0.0	1.0
AD	2126.0	0.156162	0.363094	0.0	0.0	0.0	0.0	1.0
DE	2126.0	0.118532	0.323314	0.0	0.0	0.0	0.0	1.0
LD	2126.0	0.050329	0.218675	0.0	0.0	0.0	0.0	1.0
FS	2126.0	0.032455	0.177248	0.0	0.0	0.0	0.0	1.0
SUSP	2126.0	0.092662	0.290027	0.0	0.0	0.0	0.0	1.0
CLASS	2126.0	4.509878	3.026883	1.0	2.0	4.0	7.0	10.0
NSP	2126.0	1.304327	0.614377	1.0	1.0	1.0	1.0	3.0

A continuación se puede ver la diferencia por dicho detalle, nótese que estamos usando los datos producto del proceso de Toma de Datos, detallado anteriormente, en esa diferencia se nota como se han afectado ciertos valores estadísticos, como lo es en el caso de FM o Movimiento Fetal.

Datos Originales

	count	mean	std	min	25%	50%	75%	max
LBE	2126.0	133.303857	9.840844	106.0	126.0	133.0	140.0	160.0
LB	2126.0	133.303857	9.840844	106.0	126.0	133.0	140.0	160.0
AC	2126.0	2.722484	3.560850	0.0	0.0	1.0	4.0	26.0
FM	2127.0	7.503056	39.030452	0.0	0.0	0.0	2.0	564.0
UC	2127.0	3.669017	2.877148	0.0	1.0	3.0	5.0	23.0
ASTV	2127.0	47.008933	17.210648	12.0	32.0	49.0	61.0	87.0
MSTV	2127.0	1.335449	0.891543	0.2	0.7	1.2	1.7	7.0
ALTV	2127.0	9.884814	18.476534	0.0	0.0	0.0	11.0	91.0
MLTV	2127.0	8.207616	5.701926	0.0	4.6	7.4	10.8	50.7
DL	2128.0	1.576128	2.517794	0.0	0.0	0.0	3.0	16.0
DS	2128.0	0.003759	0.061213	0.0	0.0	0.0	0.0	1.0
DP	2128.0	0.127820	0.471687	0.0	0.0	0.0	0.0	4.0
DR	2128.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0

Datos SIN NaN

	count	
LBE	2126.0	133.30
LB	2126.0	133.30
AC	2126.0	2.72
FM	2126.0	7.24
UC	2126.0	3.65
ASTV	2126.0	46.99
MSTV	2126.0	1.33
ALTV	2126.0	9.84
MLTV	2126.0	8.18
DL	2126.0	1.57
DS	2126.0	0.00
DP	2126.0	0.12
DR	2126.0	0.00

Claro está, que esas filas eliminadas son hasta cierto punto obvias, lo cierto es que las tablas en general son tan grandes que una inspección visual es inútil, por ello vamos a seguir estudiando los datos para encontrar mas inconsistencias o anomalías.

Pero, iniciaremos con la obvia, que corresponde a las columnas LBE y LB, ambas corresponden a la Frecuencia Cardiaca del Feto, sólo que una es por parte del médico y la otra por parte del equipo del SisPorto, ambas columnas son mutuamente redundantes, ello se puede notar al ver los datos en la tabla anterior, por lo que se elimnará una de ellas.

De las imágenes anteriores podemos darnos cuenta que a pesar de que los NaN han sido eliminados, hay ciertos valores que dan evidencia de posibles anomalías. ese sigue siendo el caso de FM, si bien su promedio y desviación estándar fueron mejorados al remover los datos extra, todavía apreciamos que, mientras el valor máximo es de 564.0, su promedio es de tan sólo 7.50, de igual manera se podría sospechar de las columnas AC y UC, aunque no parece tan evidente como para FM, por eso aplicaremos técnicas de visualición para terminar de confirmar posibles outliers, y de ser el caso afirmativo, aplicar lo necesario para eliminarlos.

	count	mean	std	min	25%	50%	75%	max
LBE	2126.0	133.303857	9.840844	106.0	126.0	133.0	140.0	160.0
LB	2126.0	133.303857	9.840844	106.0	126.0	133.0	140.0	160.0
AC	2126.0	2.722484	3.560850	0.0	0.0	1.0	4.0	26.0
FM	2126.0	7.241298	37.125309	0.0	0.0	0.0	2.0	564.0
UC	2126.0	3.659925	2.847094	0.0	1.0	3.0	5.0	23.0
ASTV	2126.0	46.990122	17.192814	12.0	32.0	49.0	61.0	87.0
MSTV	2126.0	1.332785	0.883241	0.2	0.7	1.2	1.7	7.0
ALTV	2126.0	9.846660	18.396880	0.0	0.0	0.0	11.0	91.0
MLTV	2126.0	8.187629	5.628247	0.0	4.6	7.4	10.8	50.7
DL	2126.0	1.570085	2.499229	0.0	0.0	0.0	3.0	16.0
DS	2126.0	0.003293	0.057300	0.0	0.0	0.0	0.0	1.0
DP	2126.0	0.126058	0.464361	0.0	0.0	0.0	0.0	4.0
DR	2126.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0

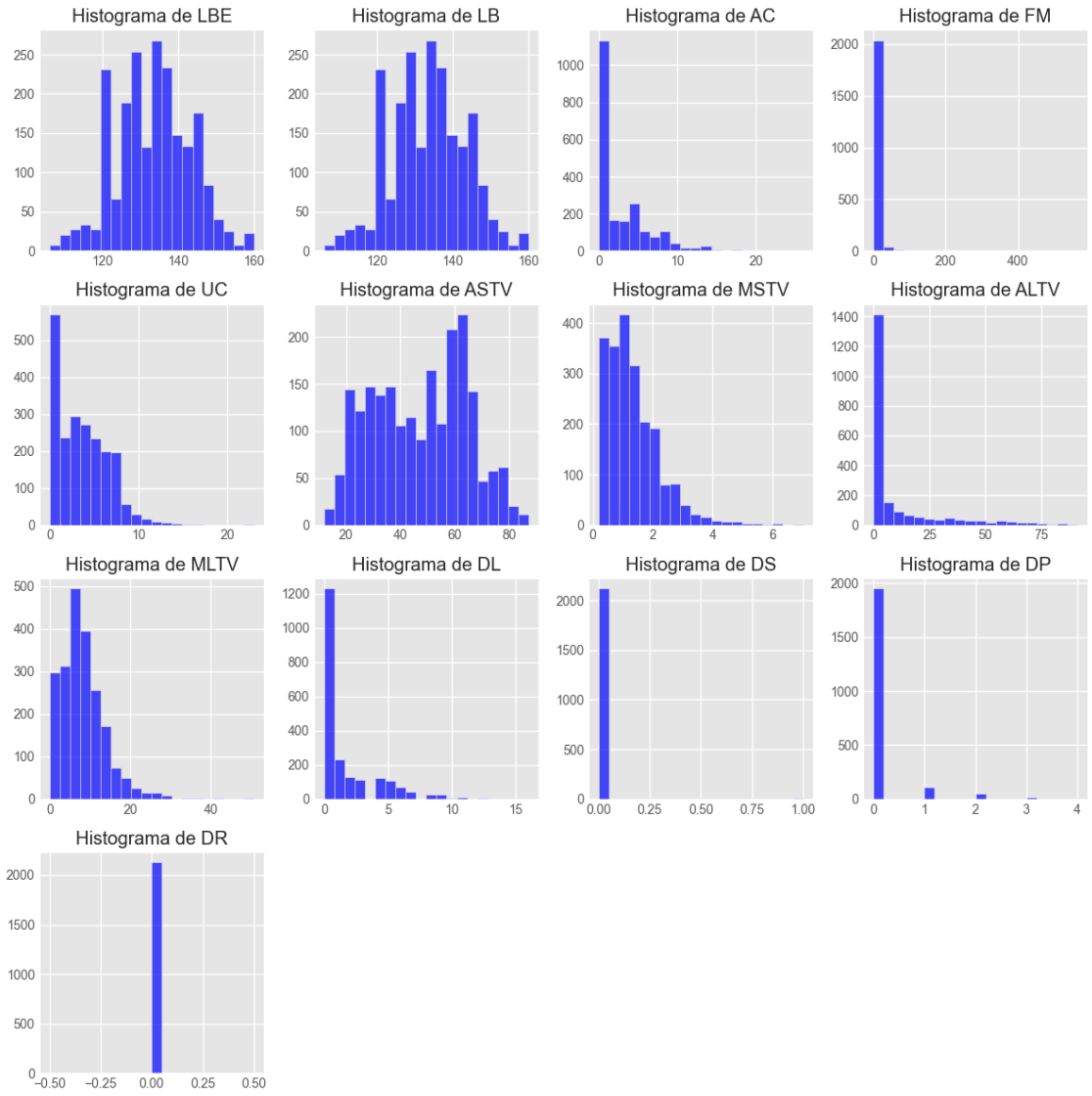
Por otro lado, es interesante analizar los valores de las columnas de Desaceleraciones en general, DL, DS, DP y DR. Se nota que fuera de datos NaN, la cantidad de valores iguales a cero es tal que afecta las estadísticas, acá también es mejor aplicar técnicas de visualización para terminar de entender sus efectos o si son anomalías a eliminar.

Una razón por la cuál aplicando estadística simple no podemos determinar o concluir con certeza si estamos en la presencia de una anomalía es si la distribución de los datos para esa columna en particular no sigue una distribución normal, por ende, empezaremos por ello, viendo cada uno de los histogramas de los datos.

[11]:
```
# Vamos a tener varias tablas para analizar

# La siguiente es para los Datos Tomados
data_tom = data.copy()

data_tom = data_tom.
 ↪drop(columns=['A','B','C','D','E','AD','DE','LD','FS','SUSP','CLASS','NSP',␣
 ↪'Width','Min','Max','Nmax','Nzeros','Mode','Mean','Median','Variance','Tendency'])
```

[12]:
```
# Creamos una visualizacion de histograma de todas las columnas seleccionadas
# Tomamos el total de columnas

# Creamos y obtenemos la Dimension y figura para la visualizacion
n_cols = len(data_tom.columns)
rows = math.ceil(math.sqrt(n_cols))
```

```
cols = math.ceil(n_cols / rows)

fig, axes = plt.subplots(nrows=rows, ncols=cols, figsize=(3*cols, 3*rows))
axes_flat = axes.flatten()

# Graficamos un histograma para cada columna
for i, col in enumerate(data_tom.columns):
    ax = axes_flat[i]
    ax.hist(data_tom[col], bins=20, color='blue', alpha=0.7)
    ax.set_title(f'Histograma de {col}')

# Removemos subplots no utilizados
for ax in axes_flat[i+1:]:
    ax.set_visible(False)

plt.tight_layout()
plt.show()
```

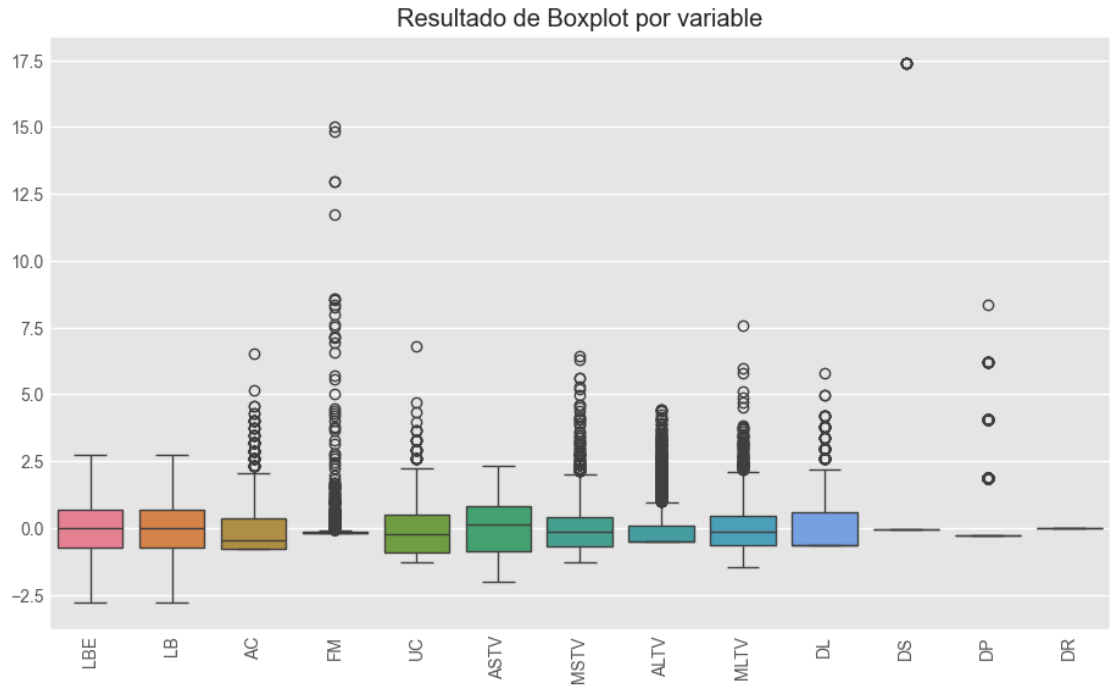


Al ver los histogramas confirmamos que LB y LBE son lo mismo, además que es la única columna que sigue una distribución normal. Para FM se confirma que hay uno o varios outliers los cuales tenemos que identificar puntualmente y arreglar. Para el resto de las columnas se confirma que debemos usar otras técnicas estadíticas y/o de visualización para datos no normales.

[13]:
```
#Los siguientes son las columnas que utilizaremos para analizar cuales poseen␣
 ↪outliers que han de ser eliminados

data_2FindOutl = data.copy()

data_2FindOutl = data_2FindOutl.drop(columns=['LB',␣
 ↪'ASTV','MSTV','ALTV','MLTV','A','B','C','D','E','AD','DE','LD','FS','SUSP','CLASS','NSP',␣
 ↪'Width','Min','Max','Nmax','Nzeros','Mode','Mean','Median','Variance','Tendency'])
```

Utilizaremos un Box-Plot, ésta es una de las mejores maneras de rápidamente identificar si los datos poseen anomalías, de encontrarse, en el siguiente apartado se hará lo necesario para elimnarlos.

[14]:
```
# Estandaricemos los valores
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data_tom)
data_scaled_df = pd.DataFrame(data_scaled, columns=data_tom.columns)

plt.figure(figsize=(10, 6))
sns.boxplot(data=data_scaled_df)
plt.xticks(rotation=90)
plt.title("Resultado de Boxplot por variable")

plt.show()
```



Podemos observar un par de cosas que vale la pena analizar mas a fondo:

- LBE y LB tienen los mismos valores y distribucion
- FM es la variable que mas anomalías parece presentar
- DS parece tener un par de observaciones muy por encima de su promedio
- Los datos de DP estan distribudos en 4 grupos muy marcados

Ahora, después de haber aplicado un análisis estadítico, haber graficado los hitogramas y Box-Plots, es recomendable ahora con certeza, hacer lo necesario para eliminar los datos anómalos del Dataset, sabemos que hay, lo que no tenemos noción en éste momento es de cuales son, pultualmente hablando. Por ende, exploraremos varias técnicas de eliminación de anomalías en el apartado

correspondiente.

0.3 Análisis de variables Categóricas

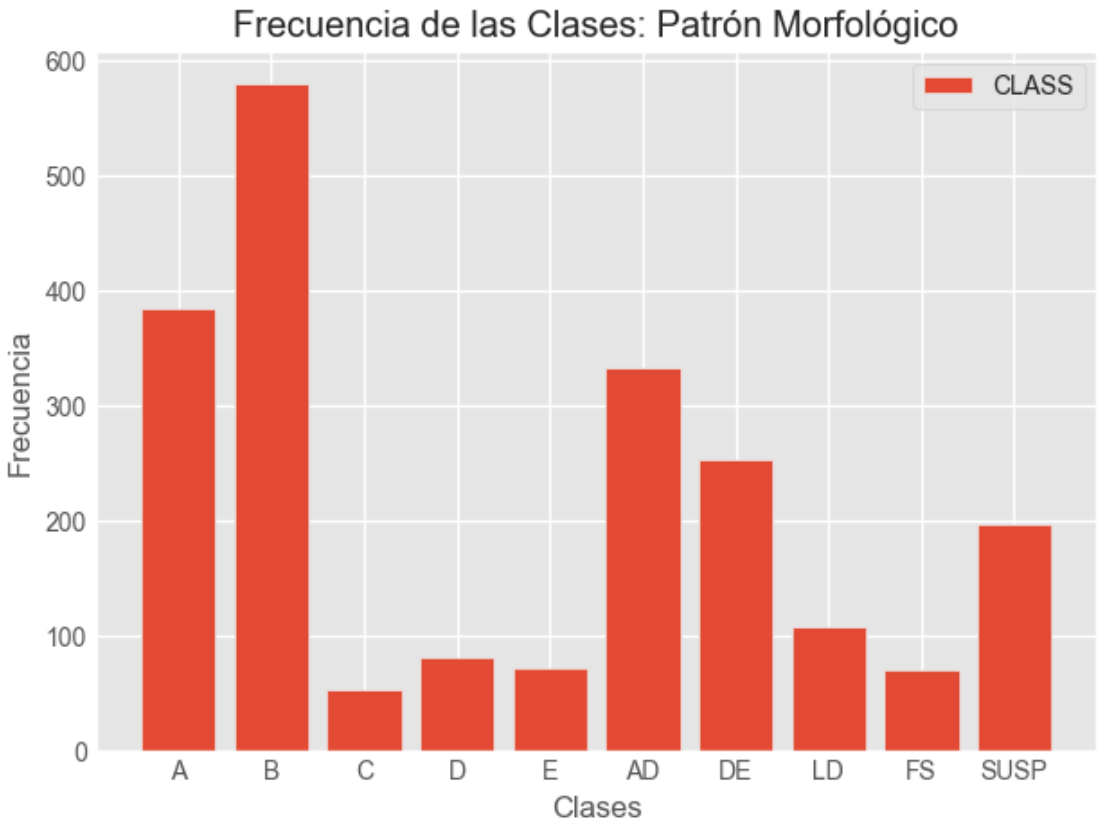
Al inicio se identificó que la tabla posee dos Clases, una que realiza una clasificación con respecto a patrones morfológicos como A: Patrón de sueño calmo, B: Patrón de sueño REM, que se agrupan en la columna CLASS: Código de Clases (1 a 10) paraa las clases de la A a SUSP. Por otro lado se encuentra la clasificación por estado fetal, que está en la columna NSP y se codifica como: 1 = Normal; 2 = Sospechoso; 3 = Patológico, veamos cómo lucen esos datos.

0.3.1 Patrones Morfológicos

Clases de la A a la SUSP, las frecuencias se verán reflejadas en la columna CLASS

[15]:
```
#Los siguientes son las columnas que utilizaremos para analizar las Categorias␣
 ↪existentes

data_Clases = data.copy()

data_Clases = data_Clases.
 ↪drop(columns=['LBE','LB','AC','FM','UC','ASTV','MSTV','ALTV','MLTV','DL','DS','DP','DR','Wi
```

[16]:
```
#El siguiente código es para poder observar la distribución de frecuencias de␣
 ↪la Categoría

class_frequencies = data['CLASS'].value_counts().sort_index()

# Crear una lista con las etiquetas de las clases
classes = ['A','B','C','D','E','AD','DE','LD','FS','SUSP']
# Crear el gráfico de barras
plt.bar(classes, class_frequencies)

# Agregar etiquetas y título
plt.xlabel('Clases')
plt.ylabel('Frecuencia')
plt.title('Frecuencia de las Clases: Patrón Morfológico')

# Agregar leyenda con el nombre de la columna
plt.legend(['CLASS'])

# Mostrar el gráfico
plt.show()
```



Lo que podemos pareciar de la distribución es que las 4 clases predominantes son, por orden de mayor a menor: - B: Patrón de Sueño REM, , con casi 600 casos. - A: Patrón de Sueño Calmo, cercano a 400. - AD: Patrón de situación de Stress, cerca a 350 casos. - DE: Patrón de Desaceleración (Estimulación del nervio vago), con un poco mas de 250 casos.

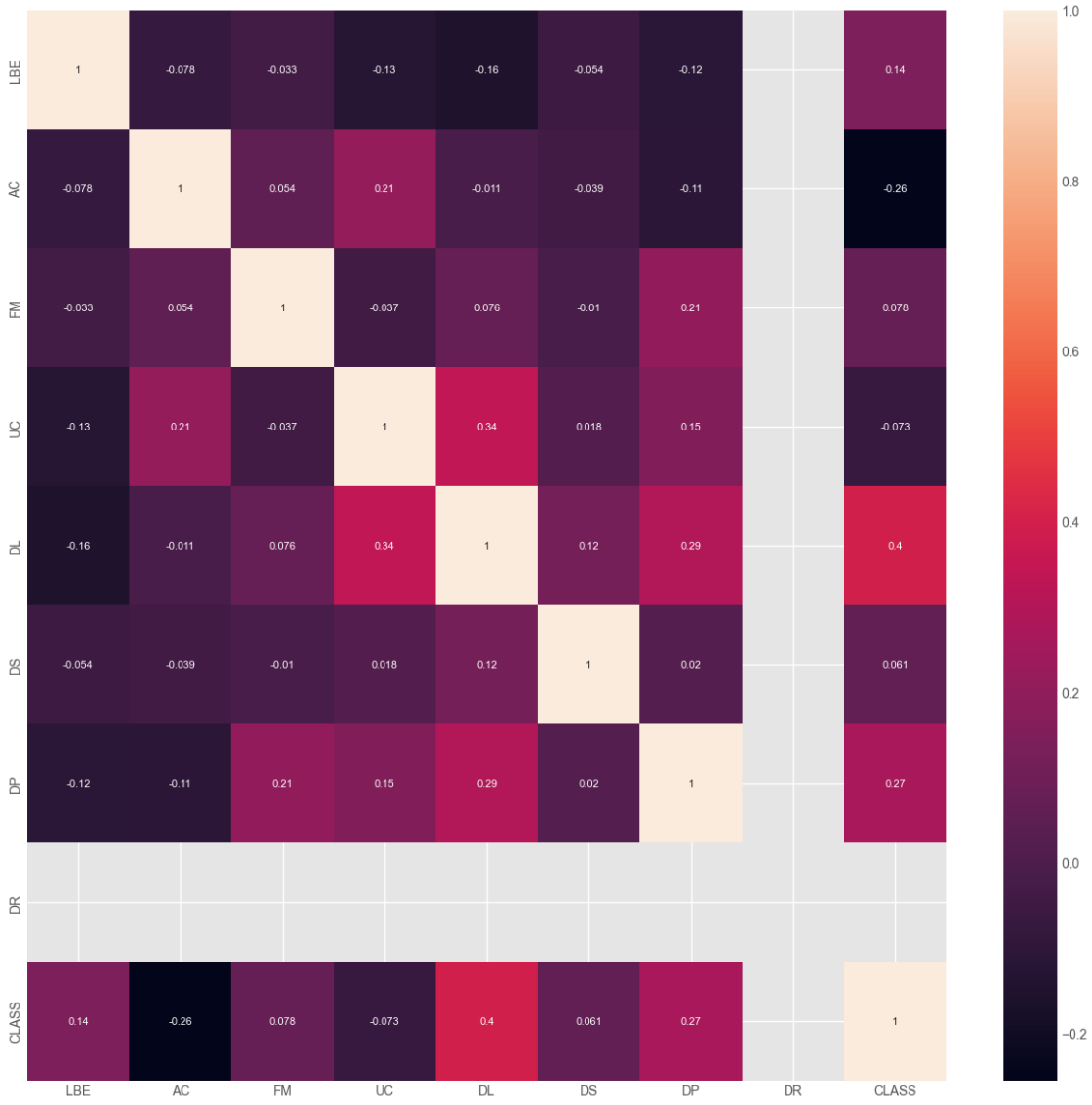
La pregunta acá sería, como se relacionan éstas clases con los datos, bueno, tendremos que hacer una matriz de correlación con el fin de poner en evidencia los casos.

Veamos como correlacionan los datos con sólo éstas categorías.

[17]:
```
#Iniciamos con algunos estudios para entender como se comportan los datos

data_corr10 = data.copy()

data_corr10 = data_corr10.drop(columns=['LB','ASTV','MSTV','ALTV','MLTV',␣
 ↪'NSP','Width','Min','Max','Nmax','Nzeros','Mode','Mean','Median','Variance','Tendency'])
data_corr10 = data_corr10.
 ↪drop(columns=['A','B','C','D','E','AD','DE','LD','FS','SUSP'])
```

[18]:
```
fig, ax = plt.subplots(figsize=(14,14))
sns.heatmap(data_corr10.corr(), annot=True, annot_kws={"size": 8})
```

[18]: <Axes: >



[19]:
```
#Como correlaciona con CLASS

data_corr10.corr()['CLASS'].sort_values(ascending=False)
```

[19]:
```
CLASS    1.000000
DL       0.395887
DP       0.269300
LBE      0.143001
FM       0.077805
DS       0.060861
```

```
UC      -0.073465
AC      -0.255205
DR            NaN
Name: CLASS, dtype: float64
```

El Mapa de calor es bastante oscuro, el nivel de correlación entre los datos tomados y las categorías morfológicas, muestra que:

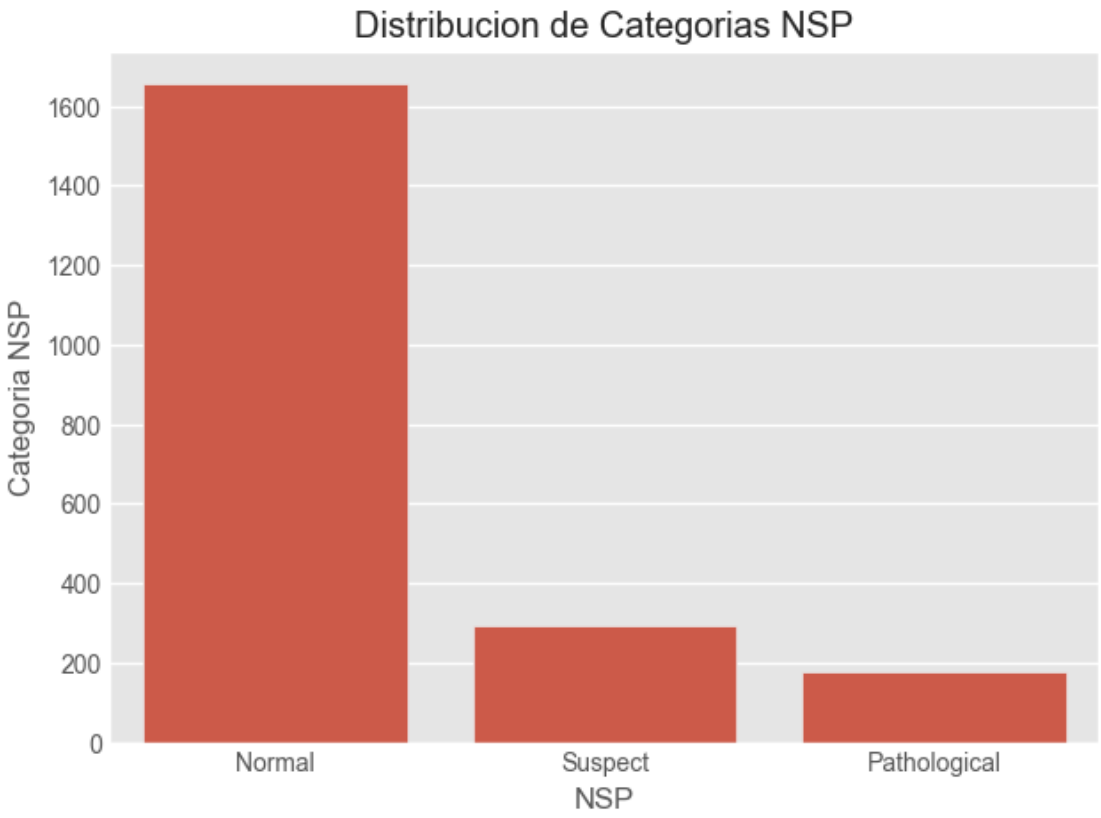
- La variable mas cercana posee un valor de 0.395 que corresponde a las Desaceleraciones Livianas DL.
- Le sigue con 0.269 DP: Desacelereaciones Prolongadas.
- Luego con -0.255 AC: Aceleraciones.
- Y la cuarta sería LBE: Frecuencia Cardiaca con 0.143.

Además DR demuestra no contribuir en lo absoluto al análisis, por lo tanto dejaremos de lado del análisis a las Desaceleraciones Repetitivas.

A continuación, veremos los datos correspondientes al Estado del Feto NSP, detallado anteriormente.

[20]:
```
#Iniciamos con algunos estudios para entender como se comportan los datos

data_corr = data.copy()

data_corr = data_corr.
 ↪drop(columns=['LB','ASTV','MSTV','ALTV','MLTV','Width','Min','Max','Nmax','Nzeros','Mode','
data_corr = data_corr.
 ↪drop(columns=['A','B','C','D','E','AD','DE','LD','FS','SUSP', 'CLASS'])
```

[21]:
```
###Con el siguiente código se graficará la columna 'NSP' que corresponde a la␣
 ↪clasificación por Estado del Feto
plt.figure()
data_chart = data_corr.copy()
data_chart['NSP'] = data['NSP'].replace({1: 'Normal', 2:'Suspect', 3:
 ↪'Pathological'})
ax = sns.countplot(data=data_chart, x="NSP", order=["Normal", "Suspect",␣
 ↪"Pathological"])
ax.set_title('Distribucion de Categorias NSP')
ax.set_ylabel('Casos')
ax.set_ylabel('Categoria NSP')
plt.show()
```

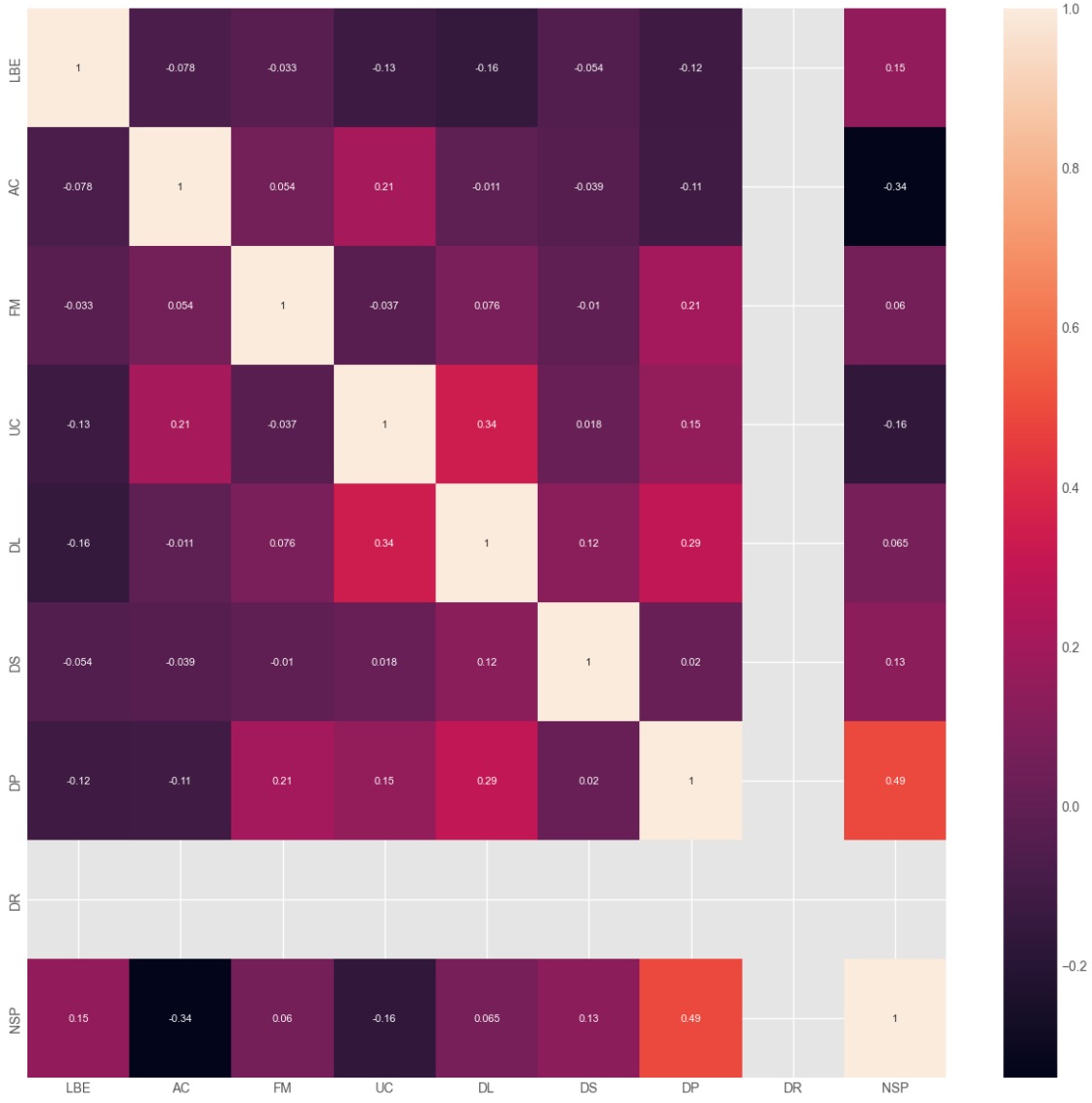


Para éste caso, vemos que la gran mayoría de los casos corresponden al estado Normal, por encima de los 1600, contra apenas cerca a 200 casos Patológicos y alrededor de 300 Sospechosos. De nuevo, cómo ser relacionan éstos resultados con los datos.

A continuación crearemos una matriz de correlación para intentar discernir cada relación.

[22]:
```
fig, ax = plt.subplots(figsize=(14,14))
sns.heatmap(data_corr.corr(), annot=True, annot_kws={"size": 8})
```

[22]: <Axes: >



[23]:
```
#Como correlaciona con NSP

data_corr.corr()['NSP'].sort_values(ascending=False)
```

[23]:
```
NSP    1.000000
DP     0.490626
LBE    0.148151
DS     0.131934
DL     0.065328
FM     0.060366
UC    -0.163295
AC    -0.340394
```

```
DR          NaN
Name: NSP, dtype: float64
```

En éste caso: - DP: Desaceleraciones Prolongadas como la principal variable correlacionada, con 0.49 tenemos. - AC: Aceleraciones, con -0.34. - UC: Contracciones uterinas, con -0.16. - LBE: Frecuencia Cardiaca con 0.148.

Es interesante que ambas clases posean en común las siguientes variables:

- *DP: desaceleraciones prolongadas (primera en NSP, segunda en CLASS)
- *AC: aceleraciones (segunda en NSP, tercera en CLASS)
- *LBE: valor base Frecuencia Cardiaca (cuarta en ambas)

Para las clases morfológicas la primera es - *DL: desaceleraciones leves

Para el Estado del Feto, la tercera es - *UC: contracciones uterinas

Y DR vuelve a aportar nada.

Morfológicas		Estado Feto	
CLASS	1.000000	NSP	1.000000
DL	0.395887	DP	0.490626
DP	0.269300	LBE	0.148151
LBE	0.143001	DS	0.131934
FM	0.077805	DL	0.065328
DS	0.060861	FM	0.060366
UC	-0.073465	UC	-0.163295
AC	-0.255205	AC	-0.340394
DR	NaN	DR	NaN

Definitamente se requiere de un criterio técnico para evaluar si dichas correlaciones hacen sentido en el contexto de una cardiotocografía, de acuerdo a la Guía de monitorización fetal intraparto basada en fisiopatología, tenemos las siguientes definiciones:

DP: Desaceleraciones prolongadas: Aquellas que duran más de 3 minutos. podrían inndicar hipoxia (disminución de oxígeno en un tejido). Las que superen los 5 minutos con una Frecuencia Cardiaca Fetal de menos de 80lpm (latidos por minuto), están frecuentemente asociados con hipoxia/acidosis fetal aguda y requieren una intervención urgente.

AC: Aceleraciones: Es un incremento abrupto de la Frecuencia Cardiaca Fetal, pueden denotar estado de vigilia (desvelo) en el feto. La precencia de aceleraciones suele considerarse un signo tranquilizador, la ausencia por otro lado es un signo, entre otros, de Hipoxia crónica.

LBE: Frecuencia Cardiaca Fetal: un valor normal está entre 110 y 160 latidos por minuto, por encima de 160 lpm durante más de 10min es considerado Taquicardia, por debajo de 110lpm durante más de 10min es considerado Bradicardia.

DL: Desaceleraciones Leves: Son un descenso de la Frecuencia Cardiaca Fetal por debajo de la línea basal de más de 15lpm y que dura más de 15 segundos. Se consideran que son una respouesta

refleja para disminuir el gasto cardiaco cuando el feto es expuesto a un estrés hipóxico o mecánico. Estas son un tipo de desaceleración tardía, igualmente relacionada con hipoxia.

UC: Contracciones Uterinas: una frecuencia excesiva, 5 contracciones en 10 minutos se le llama Taquisistolia y la hiperestimulación es debida a una respuesta exagerada a estimulantes uterinos.

0.4 Acciones a seguir:

Para nuestros efectos, aunque hemos determinado cierto nivel de correlación entre algunas variables y las clases exitentes, también encontramos que a excepción de la frecuencia cardiaca fetal, el resto de las variables poseen anomalías, por ende, y cómo sabemos que dichas anomalías podrían estar impactando negativamente dichas correlaciones, necesitamos corregirlos.

Entre los dos grupos iniciaremos con el análisis para el estado fetal NSP.

Acá las columnas a ser analizadas ['LBE', 'LB', 'AC', 'FM', 'UC', 'DL', 'DS', 'DP', 'DR'] a fin de determinar ['NSP'].

0.5 Tratamiento de los Datos. Detección de Anomalías

[24]:
```
#Confirmando que datos existen en esta copia
data_corr.columns
```

[24]:
```
Index(['LBE', 'AC', 'FM', 'UC', 'DL', 'DS', 'DP', 'DR', 'NSP'], dtype='object')
```

[25]:
```
data_corr.info
```

[25]:
```
<bound method DataFrame.info of         LBE   AC   FM   UC   DL   DS   DP   DR
NSP
0     120.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  2.0
1     132.0  4.0  0.0  4.0  2.0  0.0  0.0  0.0  1.0
2     133.0  2.0  0.0  5.0  2.0  0.0  0.0  0.0  1.0
3     134.0  2.0  0.0  6.0  2.0  0.0  0.0  0.0  1.0
4     132.0  4.0  0.0  5.0  0.0  0.0  0.0  0.0  1.0
…       …    …    …    …    …    …    …    …    …
2121  140.0  0.0  0.0  6.0  0.0  0.0  0.0  0.0  2.0
2122  140.0  1.0  0.0  9.0  0.0  0.0  0.0  0.0  2.0
2123  140.0  1.0  0.0  7.0  0.0  0.0  0.0  0.0  2.0
2124  140.0  1.0  0.0  9.0  0.0  0.0  0.0  0.0  2.0
2125  142.0  1.0  1.0  5.0  0.0  0.0  0.0  0.0  1.0

[2126 rows x 9 columns]>
```

Como ya sabemos que los datos no siguen una distribución normal (excepto LBE o LB), aplicaremos un método estadístico llamado Rango Intercuantil (IQR), con ello, encontraremos los puntos anómalos que sean menores al primer cuartil menos 1.5 veces el IQR y que sean mayores al tercer cuartil mas 1.5 veces el IQR, donde IQR es la diferencia entre los cuatiles tres y uno. Veamos a continuación.

[26]:
```
#Función que Calcula el IQR
def find_outliers_IQR(data_corr):
    q1 = data_corr.quantile(0.25)
    q3 = data_corr.quantile(0.75)
    IQR = q3 - q1
    outliers = data_corr[((data_corr < (q1 - 1.5 * IQR)) | (data_corr > (q3 + 1.
 ↪5 * IQR)))]
    return outliers
```

[27]:
```
#Función para visualizar los resultados de los cálculos
def print_outlier_stats(colOut):
    for column in colOut:
        outliers = find_outliers_IQR(data_corr[column])
        print(f"Columna: {column}")
        print("Número de outliers:", len(outliers))
        if not outliers.empty:
            print("Máximo valor outlier:", outliers.max())
            print("Mínimo valor outlier:", outliers.min())
        print("\n")
```

[28]:
```
#Pasamos solo las columnas que queremos investigar
colOut = ['AC', 'FM', 'UC', 'DL', 'DS', 'DP', 'DR']
print_outlier_stats(colOut)
```

```
Columna: AC
Número de outliers: 83
Máximo valor outlier: 26.0
Mínimo valor outlier: 11.0


Columna: FM
Número de outliers: 310
Máximo valor outlier: 564.0
Mínimo valor outlier: 6.0


Columna: UC
Número de outliers: 22
Máximo valor outlier: 23.0
Mínimo valor outlier: 12.0


Columna: DL
Número de outliers: 81
Máximo valor outlier: 16.0
Mínimo valor outlier: 8.0
```

```
Columna: DS
Número de outliers: 7
Máximo valor outlier: 1.0
Mínimo valor outlier: 1.0


Columna: DP
Número de outliers: 178
Máximo valor outlier: 4.0
Mínimo valor outlier: 1.0


Columna: DR
Número de outliers: 0
```

De los resultados anteriores es necesario hacer ciertas salvedades:

- Como se esperaba LBE no posee anomalías, al menos con éste método, ya que si sigue una distribución normal.
- NSP es una clase y no una variable, lo que se ve en los resultados es realmente el efecto de las frecuencias, por lo que no lo tomaremos en cuenta.

En cuanto a los demás, definitivamente DR no posee datos significativos, por lo que será eliminada del estudio.

Para el resto, haremos dos cosas, el total de datos por variable es del 2126 datos, la que mas anomalías presenta es FM con 310, lo que representa un 14.5% del total, con ese valor, vamos a probar hacerle drop a los datos, para ello, reescribiremos los datos anómalos como NaN y luego eliminamos las filas con ellos.

[29]:
```
#ELiminamos DR del análisis.
data_corr = data_corr.drop(columns='DR')
```

[30]:
```
#Función que detecta outliers y los cambia por un valor NaN
#Se pasan solo las columnas que se deben inspeccionar
def mask_outliers_IQR(data_corr, columns=None, inspect=True):
    if columns is None:
        columns = data_corr.columns  # Si no se especifican columnas, se␣
 ↪inspeccionan todas

    outliers_df = data_corr.copy()  # Copiamos el DataFrame original para no␣
 ↪modificarlo directamente

    for column in columns:
        if inspect:
            q1 = data_corr[column].quantile(0.25)
            q3 = data_corr[column].quantile(0.75)
            IQR = q3 - q1
```

```
            outliers_mask = ((data_corr[column] < (q1 - 1.5 * IQR)) |␣
 ↪(data_corr[column] > (q3 + 1.5 * IQR)))
            outliers_df.loc[outliers_mask, column] = np.nan  # Reemplazamos los␣
 ↪outliers con NaN en el nuevo DataFrame
        else:
            # Si se especifica ignorar, simplemente dejamos la columna intacta
            pass

    return outliers_df
```

[31]:
```
columns_to_inspect = ['AC', 'FM', 'UC', 'DL', 'DS', 'DP']
outliers_df = mask_outliers_IQR(data_corr, columns=columns_to_inspect,␣
 ↪inspect=True)

# outliers_df ahora contendrá tu DataFrame original con los outliers␣
 ↪reemplazados por NaN en las columnas especificadas
```

[32]:
```
#Confirmando que mantenemos el DataFrame original
data_corr.info
```

[32]:
```
<bound method DataFrame.info of         LBE   AC   FM   UC   DL   DS   DP  NSP
0     120.0  0.0  0.0  0.0  0.0  0.0  0.0  2.0
1     132.0  4.0  0.0  4.0  2.0  0.0  0.0  1.0
2     133.0  2.0  0.0  5.0  2.0  0.0  0.0  1.0
3     134.0  2.0  0.0  6.0  2.0  0.0  0.0  1.0
4     132.0  4.0  0.0  5.0  0.0  0.0  0.0  1.0
…       …    …    …    …    …    …    …    …
2121  140.0  0.0  0.0  6.0  0.0  0.0  0.0  2.0
2122  140.0  1.0  0.0  9.0  0.0  0.0  0.0  2.0
2123  140.0  1.0  0.0  7.0  0.0  0.0  0.0  2.0
2124  140.0  1.0  0.0  9.0  0.0  0.0  0.0  2.0
2125  142.0  1.0  1.0  5.0  0.0  0.0  0.0  1.0

[2126 rows x 8 columns]>
```

[33]:
```
#Nuevo DataFrame con TODOS las anomalías NaN
outliers_df.info
```

[33]:
```
<bound method DataFrame.info of         LBE   AC   FM   UC   DL   DS   DP  NSP
0     120.0  0.0  0.0  0.0  0.0  0.0  0.0  2.0
1     132.0  4.0  0.0  4.0  2.0  0.0  0.0  1.0
2     133.0  2.0  0.0  5.0  2.0  0.0  0.0  1.0
3     134.0  2.0  0.0  6.0  2.0  0.0  0.0  1.0
4     132.0  4.0  0.0  5.0  0.0  0.0  0.0  1.0
…       …    …    …    …    …    …    …    …
2121  140.0  0.0  0.0  6.0  0.0  0.0  0.0  2.0
```

```
2122  140.0  1.0  0.0  9.0  0.0  0.0  0.0  2.0
2123  140.0  1.0  0.0  7.0  0.0  0.0  0.0  2.0
2124  140.0  1.0  0.0  9.0  0.0  0.0  0.0  2.0
2125  142.0  1.0  1.0  5.0  0.0  0.0  0.0  1.0

[2126 rows x 8 columns]>
```

En éste momento mantenemos la misma cantidad de filas por columna 2126, si contamos la cantidad de NaN corresponderá a la cantidad de anomalías por cada columna.

[34]:
```
#Cantidad de anomalías por Variable
outliers_df.isnull().sum()
```

[34]:
```
LBE      0
AC      83
FM     310
UC      22
DL      81
DS       7
DP     178
NSP      0
dtype: int64
```

[35]:
```
# Eliminar filas con valores NaN
outliers_df = outliers_df.dropna()
```

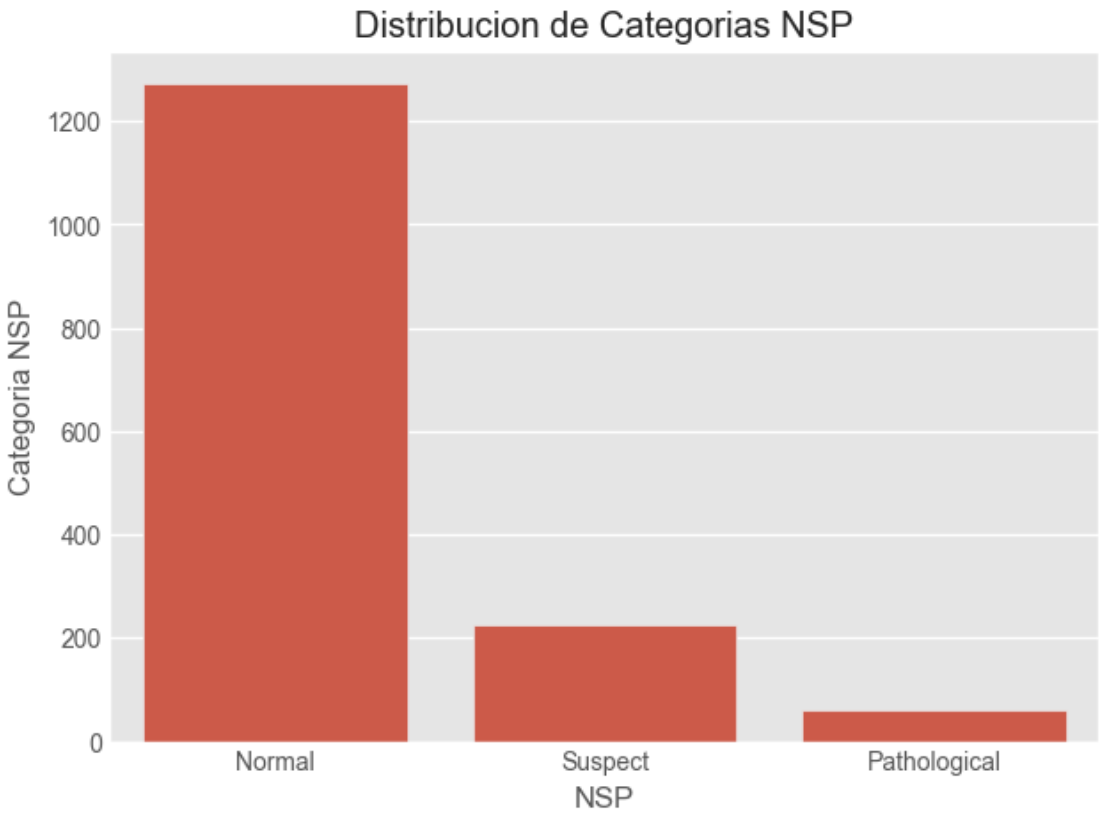
[36]:
```
outliers_df.describe().T
```

[36]:

	count	mean	std	min	25%	50%	75%	max
LBE	1558.0	134.037869	10.204800	106.0	126.0	134.0	142.0	160.0
AC	1558.0	2.227214	2.756153	0.0	0.0	1.0	4.0	10.0
FM	1558.0	0.584082	1.189357	0.0	0.0	0.0	1.0	5.0
UC	1558.0	3.519897	2.615954	0.0	1.0	3.0	5.0	11.0
DL	1558.0	1.087291	1.828576	0.0	0.0	0.0	1.0	7.0
DS	1558.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0
DP	1558.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0
NSP	1558.0	1.221438	0.498298	1.0	1.0	1.0	1.0	3.0

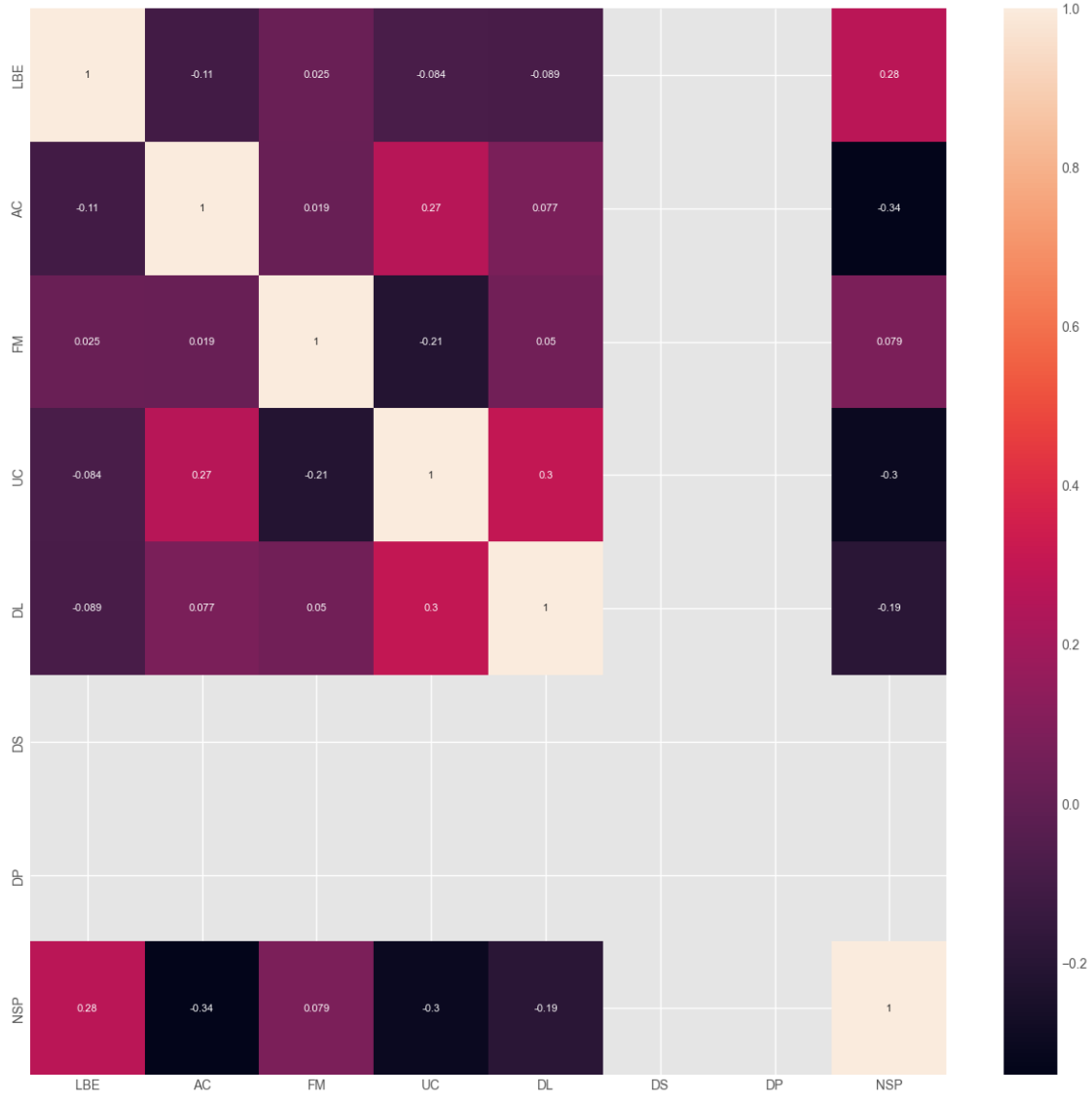
[37]:
```
###Con el siguiente código se graficará la columna 'NSP' que corresponde a la␣
 ↪clasificación por Estado del Feto
plt.figure()
data_chart = outliers_df.copy()
data_chart['NSP'] = data['NSP'].replace({1: 'Normal', 2:'Suspect', 3:
 ↪'Pathological'})
ax = sns.countplot(data=data_chart, x="NSP", order=["Normal", "Suspect",␣
 ↪"Pathological"])
ax.set_title('Distribucion de Categorias NSP')
ax.set_ylabel('Casos')
```

```
ax.set_ylabel('Categoria NSP')
plt.show()
```



[38]:
```
fig, ax = plt.subplots(figsize=(14,14))
sns.heatmap(outliers_df.corr(), annot=True, annot_kws={"size": 8})
```

[38]: <Axes: >



[39]:
```
#Como correlaciona con NSP

outliers_df.corr()['NSP'].sort_values(ascending=False)
```

[39]:
```
NSP    1.000000
LBE    0.276851
FM     0.078559
DL    -0.189692
UC    -0.300241
AC    -0.341097
DS          NaN
DP          NaN
```

```
Name: NSP, dtype: float64
```

Si repetimos lo mismo pero en vez de eliminarlos los imputamos con la media.

[40]:
```python
def mean_outliers_IQR(data_corr, columns=None, inspect=True):
    if columns is None:
        columns = data_corr.columns  # Si no se especifican columnas, se inspeccionan todas

    outliers_df = data_corr.copy()  # Copiamos el DataFrame original para no modificarlo directamente

    for column in columns:
        if inspect:
            q1 = data_corr[column].quantile(0.25)
            q3 = data_corr[column].quantile(0.75)
            IQR = q3 - q1

            lower_bound = q1 - 1.5 * IQR
            upper_bound = q3 + 1.5 * IQR

            # Reemplazar los outliers con la media de la columna
            outliers_df[column] = np.where((data_corr[column] < lower_bound) | (data_corr[column] > upper_bound),
                                           data_corr[column].mean(), outliers_df[column])
        else:
            # Si se especifica ignorar, simplemente dejamos la columna intacta
            pass

    return outliers_df
```

[41]:
```python
columns_to_inspect = ['AC', 'FM', 'UC', 'DL', 'DS', 'DP']
outliers_df = mean_outliers_IQR(data_corr, columns=columns_to_inspect, inspect=True)

# outliers_df ahora contendrá tu DataFrame original con los outliers reemplazados por la media de las columnas especificadas
```

[42]:
```python
#Cantidad de anomalías por Variable
outliers_df.isnull().sum()
```

[42]:
```
LBE    0
AC     0
FM     0
UC     0
DL     0
```

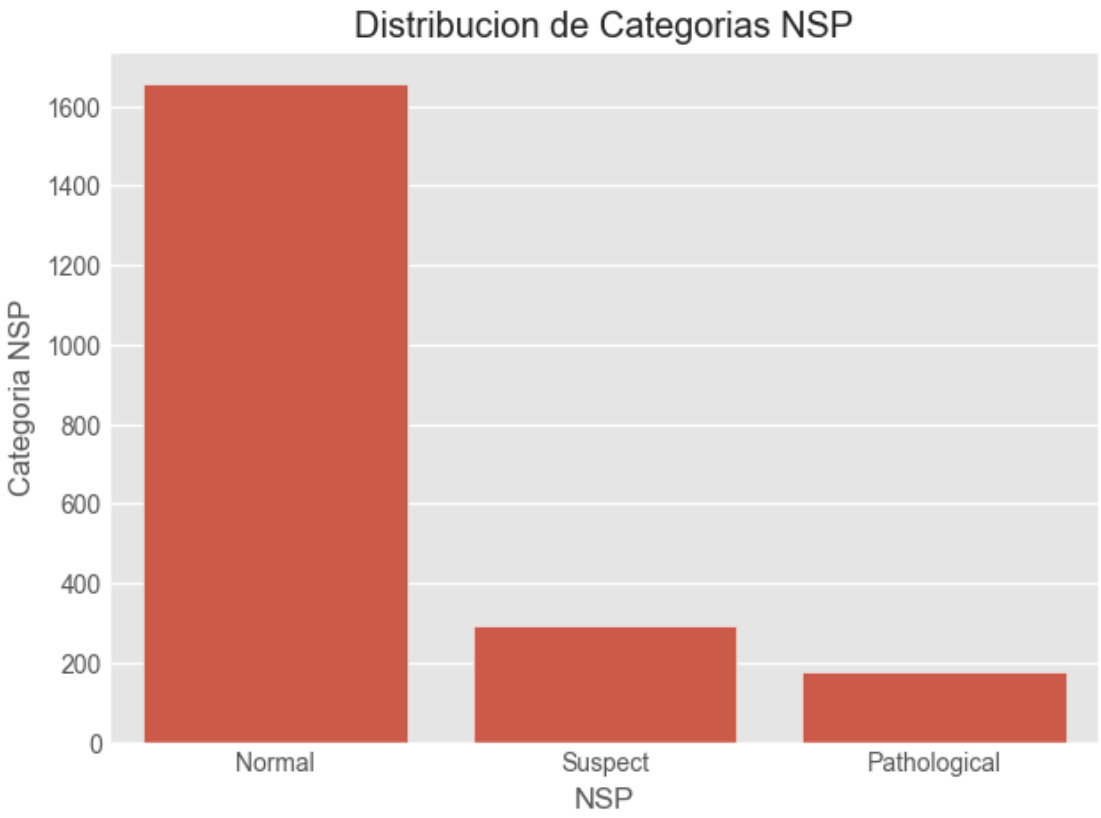
```
DS      0
DP      0
NSP     0
dtype: int64
```

[43]:
```
outliers_df.describe().T
```

[43]:

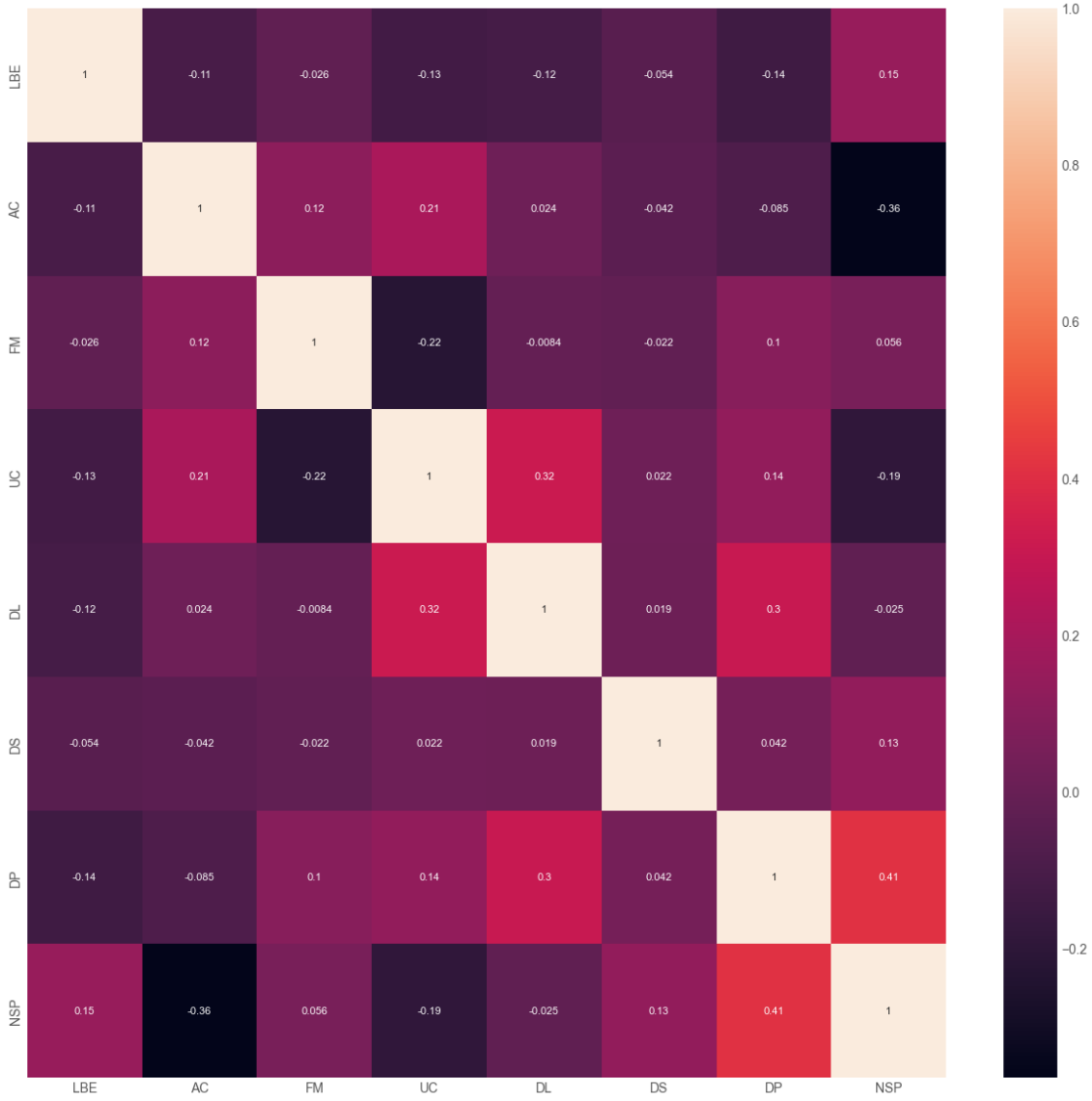
	count	mean	std	min	25%	50%	75%	max
LBE	2126.0	133.303857	9.840844	106.0	126.0	133.0	140.0	160.000000
AC	2126.0	2.294434	2.743893	0.0	0.0	1.0	4.0	10.000000
FM	2126.0	1.573755	2.583095	0.0	0.0	0.0	2.0	7.241298
UC	2126.0	3.557158	2.647940	0.0	1.0	3.0	5.0	11.000000
DL	2126.0	1.269133	1.916901	0.0	0.0	0.0	2.0	7.000000
DS	2126.0	0.000011	0.000189	0.0	0.0	0.0	0.0	0.003293
DP	2126.0	0.010554	0.034923	0.0	0.0	0.0	0.0	0.126058
NSP	2126.0	1.304327	0.614377	1.0	1.0	1.0	1.0	3.000000

[44]:
```
###Con el siguiente código se graficará la columna 'NSP' que corresponde a la␣
 ↪clasificación por Estado del Feto
plt.figure()
data_chart = outliers_df.copy()
data_chart['NSP'] = data['NSP'].replace({1: 'Normal', 2:'Suspect', 3:
 ↪'Pathological'})
ax = sns.countplot(data=data_chart, x="NSP", order=["Normal", "Suspect",␣
 ↪"Pathological"])
ax.set_title('Distribucion de Categorias NSP')
ax.set_ylabel('Casos')
ax.set_ylabel('Categoria NSP')
plt.show()
```



[45]:
```
fig, ax = plt.subplots(figsize=(14,14))
sns.heatmap(outliers_df.corr(), annot=True, annot_kws={"size": 8})
```

[45]: <Axes: >



[46]:
```
#Como correlaciona con NSP

outliers_df.corr()['NSP'].sort_values(ascending=False)
```

[46]:
```
NSP    1.000000
DP     0.411484
LBE    0.148151
DS     0.131934
FM     0.056283
DL    -0.025371
UC    -0.193780
AC    -0.364431
```

```
Name: NSP, dtype: float64
```

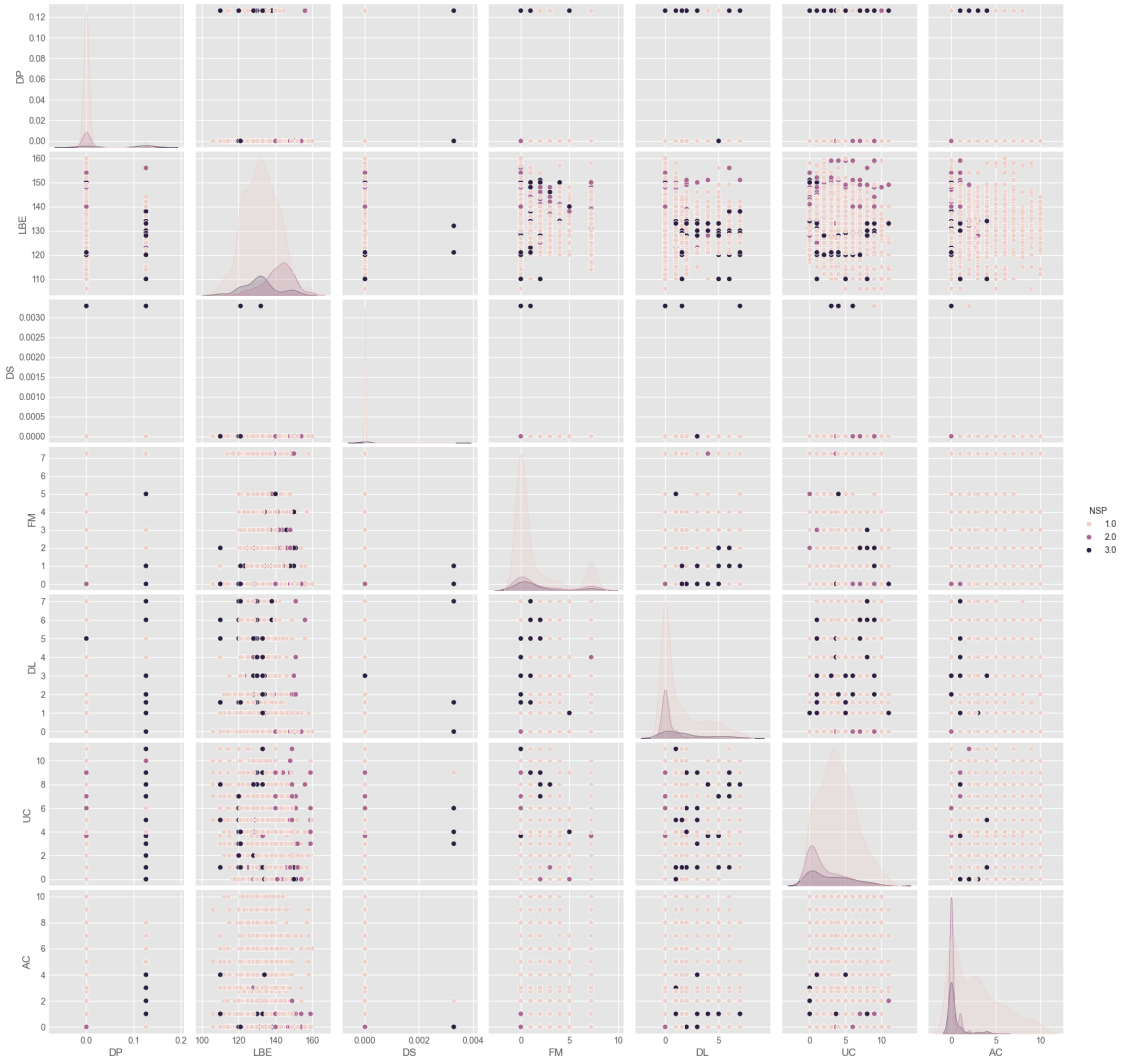
0.6 Análisis Bivariado

La matriz de correlaciones claramente muestra que hay un par de variables que estan altamente correlacionadas con nuestra variable objetivo NSP:

Hagamos un analysis bivariado para entender el comportamiento

[47]:
```
data_chart_vibar = outliers_df[['DP', 'LBE', 'DS', 'FM', 'DL', 'UC', 'AC',␣
  ↪'NSP']]
sns.pairplot(data_chart_vibar, hue='NSP')
```

[47]:
```
<seaborn.axisgrid.PairGrid at 0x7f9128aed870>
```



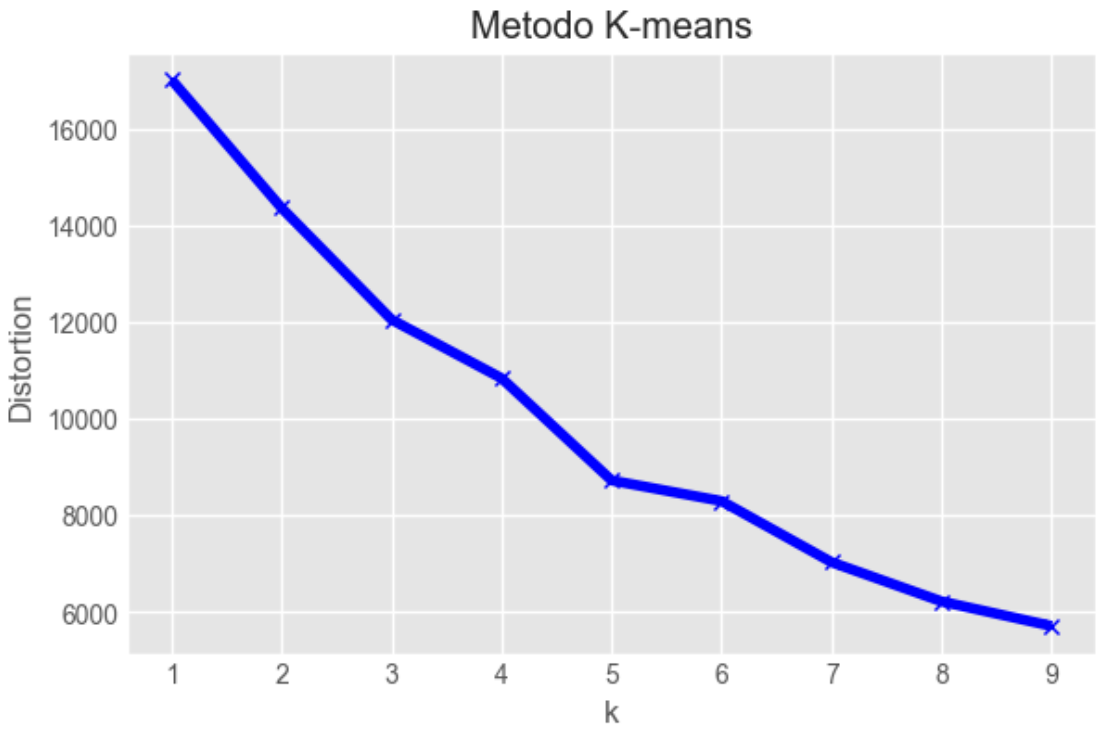
0.7 Agrupación y modelado

Para poder hacer la agrupacion de los datos utilizaremos el modelo K-Means, para lo que primero aplicaremos la stransformacion de estandarizar los valores de nuestro data set. Acto seguido, definiremos cual es el numero optimo de K's para utilizar en el modelo y, por ultimo, analizaremos los resultados de la aplicacion de dicho modelo.

[48]:
```
#Cambiar acá si se desea repetir el análisis con otra base de datos.
data_non = outliers_df
```

[49]:
```
# Primero estandarizamos los valores del modelo
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data_non)
```

[57]:
```
# Utilizaremos primero K-Means para tratar de generar las agrupaciones con el␣
 ↪mismo numero de clases que la variable NSP

distortions = []
K = range(1,10)



for k in K:
    modelo_kmeans = KMeans(n_clusters=k)
    modelo_kmeans.fit(data_scaled)
    distortions.append(modelo_kmeans.inertia_)

plt.figure(figsize=(6, 4))
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('Metodo K-means')
plt.show()
```



Para la selección del número óptimo de clústeres utilizaremos el método del codo para determinar nuestro valor óptimo. Este método funciona al correr el modelo de K-medias en nuestro conjunto de datos K veces, calculando simultáneamente el valor de la suma de cuadrados de las diferencias. Al visualizar estos resultados (errores vs. K), podemos observar que a medida que aumenta el valor de K, el valor de los errores disminuye, ya que los puntos de datos están más cerca de los centroides. El "codo" en la gráfica nos ayuda a identificar cuándo la tasa de cambio disminuye significativamente.

En nuestra gráfica, no observamos una reducción repentina en los valores; la disminución parece ser más gradual. Sin embargo, podemos apreciar que entre los valores de K de 3 y 4 hay un punto de inflexión. Por lo tanto, elegiremos 3 como nuestro número óptimo de clústeres.

[58]:
```
optimal_k = 3  # replace with the number you found optimal
kmeans = KMeans(n_clusters=optimal_k)
kmeans_labels = kmeans.fit_predict(data_scaled)
data_non['KMedias_Labels'] = kmeans_labels
```

Ya que tenemos nuestra agrupacion, podemos analizar informacion estadistica sobre nuestros clusters. Haremos lo siguiente: 1. Analizar los valores promedio de nuestros clusters 2. Observar el tamaño de nuestros clusters 3. Obtener datos estadisticos generales 5. Obtener el "silhoute score" y representacion visual de los clusters

[59]:
```
cluster_medias = kmeans.cluster_centers_
features = data_non.columns[:-1]
centers_df = pd.DataFrame(cluster_medias, columns=features)
```

```
print(centers_df)
```

```
        LBE        AC        FM        UC        DL        DS        DP  \
0 -0.326492  0.589890 -0.057627  0.508934  0.321158 -0.057476 -0.302284
1  0.416149 -0.533448 -0.002518 -0.590410 -0.508026 -0.057476 -0.302284
2 -0.477995 -0.296568  0.319145  0.435537  0.994299  0.610246  3.209499

        NSP
0 -0.488753
1  0.224767
2  1.399482
```

[62]:
```
# Observamos el tamaño de los clusters
cluster_sizes = data_non['KMedias_Labels'].value_counts()
print(cluster_sizes)
```

```
KMedias_Labels
1    973
0    970
2    183
Name: count, dtype: int64
```

[64]:
```
# Group the data by cluster label and compute summary statistics
clustered_data = data_non.groupby('KMedias_Labels')
cluster_summary = clustered_data.describe().T
print(cluster_summary)
```

```
KMedias_Labels           0           1           2
LBE count       970.000000  973.000000  183.000000
    mean        130.076289  137.405961  128.601093
    std           8.803341    9.668404    7.151377
    min         106.000000  112.000000  110.000000
    25%         124.000000  130.000000  125.000000
…                     …           …           …
NSP min           1.000000    1.000000    1.000000
    25%           1.000000    1.000000    1.000000
    50%           1.000000    1.000000    3.000000
    75%           1.000000    2.000000    3.000000
    max           3.000000    3.000000    3.000000

[64 rows x 3 columns]
```

Dado que los datos que utilizamos para crear las agrupaciones estan estandarizados, los resultados de cluster_medias nos da que tantas desviaciones estandar esta cada grupo. Podemos observar que:

0.7.1 Cluster 0:

- Los centroides de este agrupamiento tiene un valor promedio bajo de “LBE”

- Tiene un valor significativamente alto para “AC”, lo que significa que este feature es importante para definir este cluster
- “UC” y “DL” estan por encima del promedio
- “FM”, “DS” y “DP” tienen un valor promedio.

0.7.2 Cluster 1:

- LBE tambien esta por encima del promedio
- AC y UC estan significativamente por debajo del promedio, lo que indica que las caracteristicas de este cluster son lo opuesto del cluster 0
- DL esta por debajo del proemdio
- Como el cluster 0, “FM”, “DS”, “DP” estan dentro del promedio

0.7.3 Cluster 2:

- LBE es el mas bajo de todos los clusters
- “FM” y “UC” estan por encima del promedio, con “FM” teniendo un valor significativamente por en encima del promedio
- “DL” y “DS” son muy altos, “DL” siendo el mas alto de todos los clusters
- “DP” es muy alto, indicando que este puede ser un feature particularmente importante

0.7.4 Clasificación por Estado del Feto de 3 Clases, resumida en la columna NSP

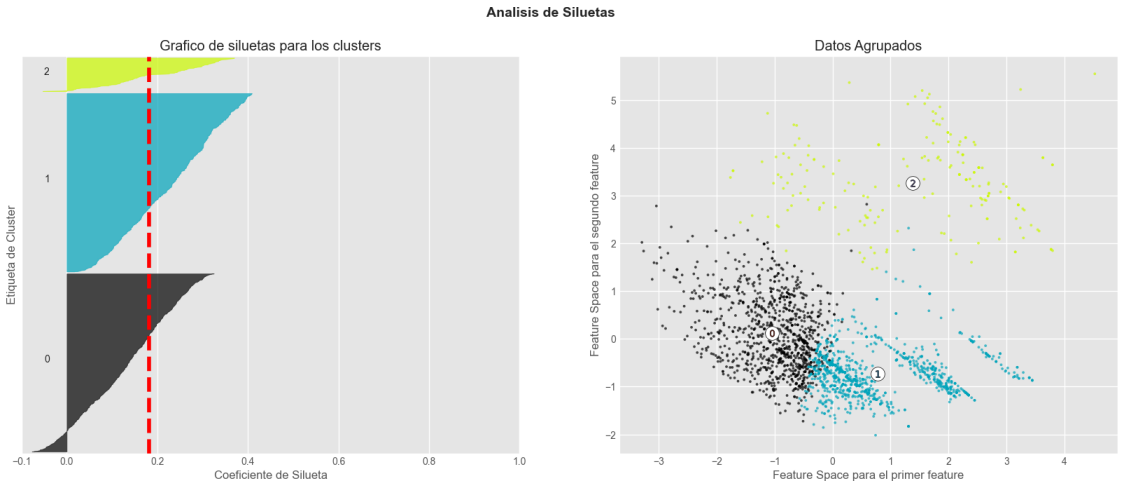
- NSP: Normal=1; Sospechoso=2; Patológico=3

Con esto podemos intuir lo siguiente: - Cluster 0 puede representar a un grupo con actividad alta de “AC - Acelaraciones” Y “UC - Contracciones Uterinas” pareciera estar relacionado a un “NSP” normal - Cluster 1 puede estar representado por niveles bajos de “AC - Aceleraciones” y “UC - Contracciones Uterinas”, donde el NSP pareciera tender a ligeramente sospechoso - Cluster 2 esta capturando instancias de LBE bajo, pero alto movimiento fetal “FM”, contracciones uterinas “UC”, desaceleraciones “DL”, deceleraciones severas “DS” y deceleraciones prolongadas “DP”. Lo que puede indicar casos significaticamente distintos a los demas. Tendiendo a un NSP patologico problematico.

[73]:
```
# Computamos el "silhouette_score" de cada muestra
silhouette_avg = silhouette_score(data_scaled, kmeans_labels)
sample_silhouette_values = silhouette_samples(data_scaled, kmeans_labels)

# Hacemos una reduccion de dimensionalidad de los datos
pca = PCA(n_components=2)
reduced_features = pca.fit_transform(data_scaled)

# Comenzamos a crear las graficas
fig, (ax1, ax2) = plt.subplots(1, 2)
fig.set_size_inches(18, 7)

# Los coeficientes de silueta van de -.1 a 1
ax1.set_xlim([-0.1, 1])
# Creamos una separacion visual entre los grupos
```

```
ax1.set_ylim([0, len(data_scaled) + (len(np.unique(kmeans_labels)) + 1) * 10])

y_lower = 10
for i in range(len(np.unique(kmeans_labels))):
    # Agrupamos los valores de las siluetas para cada cluster
    ith_cluster_silhouette_values = sample_silhouette_values[kmeans_labels == i]

    ith_cluster_silhouette_values.sort()

    size_cluster_i = ith_cluster_silhouette_values.shape[0]
    y_upper = y_lower + size_cluster_i

    color = cm.nipy_spectral(float(i) / len(np.unique(kmeans_labels)))
    ax1.fill_betweenx(np.arange(y_lower, y_upper), 0,␣
 ↪ith_cluster_silhouette_values,
                      facecolor=color, edgecolor=color, alpha=0.7)
    ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

    # Computamos nuestro valor y_lower para ajustar los rangos
    y_lower = y_upper + 10

ax1.set_title("Grafico de siluetas para los clusters")
ax1.set_xlabel("Coeficiente de Silueta")
ax1.set_ylabel("Etiqueta de Cluster")

# Valor promedio de la silueta
ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

ax1.set_yticks([])  # Cambiamos los cortes del eje x
ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

# Segunda grafica mostrando los clusters
colors = cm.nipy_spectral(kmeans_labels.astype(float) / len(np.
 ↪unique(kmeans_labels)))
ax2.scatter(reduced_features[:, 0], reduced_features[:, 1], marker='.', s=30,␣
 ↪lw=0, alpha=0.7,
            c=colors, edgecolor='k')

# Etiquetado de los clusters
centers = pca.transform(kmeans.cluster_centers_)
# Etiquetado con circulos
ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
            c="white", alpha=1, s=200, edgecolor='k')

for i, c in enumerate(centers):
    ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1, s=50, edgecolor='k')
```

```
ax2.set_title("Datos Agrupados")
ax2.set_xlabel("Feature Space para el primer feature")
ax2.set_ylabel("Feature Space para el segundo feature")

plt.suptitle(("Analisis de Siluetas"), fontsize=14, fontweight='bold')

plt.show()
```



Las siluetas son utilizadas para comparar la distancia entre distintos clústeres, donde cada barra representa un punto de datos. El largo de cada barra representa el score de silueta de cada punto de datos, indicando qué tan similar es cada punto de datos a su clúster.

En la gráfica de silueta de la izquierda podemos observar que: - Los clústeres 0 y 1 tienen barras que se extienden más a la derecha, lo que quiere decir que son más similares a los miembros de su grupo y, en promedio, están mejor representados con su propio clúster. - El clúster 2 tiene un score de silueta más bajo que los otros dos grupos, representado por barras más pequeñas, lo que sugiere que los puntos de datos de este clúster pueden estar no tan densamente agrupados o peor separados de otros clústeres. - El clúster 0 tiene valores negativos, lo que puede ser preocupante, ya que significa que algunos miembros del clúster están más cercanos a otros clústeres que a su propio clúster.

La gráfica de dispersión de la derecha muestra la distribución de los puntos de datos representados en un espacio bidimensional, gracias a la utilización de una técnica PCA de reducción de dimensionalidad.

En la gráfica de dispersión de la derecha podemos observar que: - Los clústeres 0 y 1 están mejor definidos, como lo indica la concentración alrededor de sus centros. - El clúster 2 está mucho más disperso y es menos denso, lo que puede identificar oportunidades de mejora en el agrupamiento. - El centro de los clústeres está posicionado en los centroides de sus clústeres, denotados por las etiquetas del clúster. Idealmente, deberían estar en las ubicaciones donde la densidad de los datos es más alta, lo que parece ser el caso para los clústeres 0 y 1. Sin embargo, el centro del Clúster 2 no parece estar en un punto altamente denso.

En conclusión, el Clúster 0 y 1 parecen estar bien definidos y tener buena estructura, con miembros del clúster cerca de sus centros y lejos de miembros de otros clústeres, mientras que el Clúster 2 es menos cohesivo, con sus miembros estando más dispersos, indicando que puede estar capturando información más amplia o diversa que los otros. Esto puede dar evidencia de que existe oportunidad de mejora en el agrupamiento, especialmente en el Clúster 2.

0.8 Referencias

- [1] XGBoost Documentation. (s.f.). XGBoost Documentation. Recuperado de https://xgboost.readthedocs.io/en/stable/
- [2] CUN. (s.f.). Diccionario Médico. Recuperado de https://www.cun.es/diccionario-medico/terminos/cardiotocografia
- [3] SISPORTO. (s.f.). A Brief History of the SISPORTO Project. Recuperado de https://sisporto.med.up.pt/a-brief-history-of-the-sisporto-project/
- [4] ScikitLearn (s.f.) Selecting the number of clusters with silhouette analysis on KMeans clustering https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

[74]:
```
data.columns
```

[74]:
```
Index(['LBE', 'LB', 'AC', 'FM', 'UC', 'ASTV', 'MSTV', 'ALTV', 'MLTV', 'DL',
       'DS', 'DP', 'DR', 'Width', 'Min', 'Max', 'Nmax', 'Nzeros', 'Mode',
       'Mean', 'Median', 'Variance', 'Tendency', 'A', 'B', 'C', 'D', 'E', 'AD',
       'DE', 'LD', 'FS', 'SUSP', 'CLASS', 'NSP'],
      dtype='object')
```