

```
from google.colab import drive
drive.mount('/content/drive')
```

↗ Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

```
# 2 - CARGA DEL FICHERO DE DATOS
file = '/content/drive/MyDrive/ColabNotebooks3/sprint1_AS/ASI_casoPractico.csv'
data = pd.read_csv(file, sep = ';')
data.head()
```

↗

	ID	b	e	LBE	AC	FM	UC	ASTV	MSTV	ALTV	...	Min	Max	Nmax	Nzeros	Mode
0	1	240	357	120	0	0	0	73	0.5	43	...	62	126	2	0	120
1	2	5	632	132	4	0	4	17	2.1	0	...	68	198	6	1	141
2	3	177	779	133	2	0	5	16	2.1	0	...	68	198	5	1	141
3	4	411	1192	134	2	0	6	16	2.4	0	...	53	170	11	0	137
4	5	533	1147	132	4	0	5	16	2.4	0	...	53	170	9	0	137

```
# CLASE 1
```

```
# IMPORTACION DE LIBRERIAS
import numpy as np
import pandas as pd
```

```
# LIBRERIAS PARA HACER GRAFICOS
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as xp
```

```
# 2 - INFORMACION DEL CONJUNTO DE DATOS
data.info()
```

↗

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2126 entries, 0 to 2125
Data columns (total 26 columns):
#   Column      Non-Null Count  Dtype
---  -
0   ID           2126 non-null   int64
1   b            2126 non-null   int64
2   e            2126 non-null   int64
3   LBE          2126 non-null   int64
4   AC           2126 non-null   int64
5   FM           2126 non-null   int64
6   UC           2126 non-null   int64
7   ASTV         2126 non-null   int64
8   MSTV         2126 non-null   float64
9   ALTV         2126 non-null   int64
10  MLTV         2126 non-null   float64
11  DL           2126 non-null   int64
12  DS           2126 non-null   int64
13  DP           2126 non-null   int64
14  DR           2126 non-null   int64
15  Width        2126 non-null   int64
16  Min          2126 non-null   int64
17  Max          2126 non-null   int64
18  Nmax         2126 non-null   int64
19  Nzeros       2126 non-null   int64
20  Mode         2126 non-null   int64
21  Mean         2126 non-null   int64
22  Median       2126 non-null   int64
23  Variance     2126 non-null   int64
24  Tendency     2126 non-null   int64
25  Target       2126 non-null   int64
dtypes: float64(2), int64(24)
memory usage: 432.0 KB
```

```
# 2 - NUMERO DE VALORES UNICOS PARA CADA VARIABLE
data.agg(['nunique']).T
```



nunique

ID	2126
b	979
e	1064
LBE	48
AC	22
FM	96
UC	19
ASTV	75
MSTV	57
ALTV	87
MLTV	249
DL	15
DS	2
DP	5
DR	1
Width	154
Min	109
Max	86
Nmax	18
Nzeros	9
Mode	88
Mean	103
Median	95
Variance	133
Tendency	3
Target	2

2 -ELIMINAR COLUMNS NO NECESARIAS

```
data = data.drop(["ID","b","e","DR"], axis = 1)
data.head()
```



	LBE	AC	FM	UC	ASTV	MSTV	ALTV	MLTV	DL	DS	DP	Width	Min	Max	Nmax	Nzeros
0	120	0	0	0	73	0.5	43	2.4	0	0	0	64	62	126	2	0
1	132	4	0	4	17	2.1	0	10.4	2	0	0	130	68	198	6	1
2	133	2	0	5	16	2.1	0	13.4	2	0	0	130	68	198	5	1
3	134	2	0	6	16	2.4	0	23.0	2	0	0	117	53	170	11	0

3 - ANALISIS DESCRIPTIVO - NULOS

```
data.isnull().sum()
```



ID	0
b	0
e	0
LBE	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
Target      0
dtype: int64

```

```

# 3 - MEDIDAS DE CENTRALIZACION, LOCALIZACION Y DISPERION
data.describe().T

```



	count	mean	std	min	25%	50%	75%	max
ID	2126.0	1063.500000	613.867657	1.0	532.25	1063.5	1594.75	2126.0
b	2126.0	878.439793	894.084748	0.0	55.00	538.0	1521.00	3296.0
e	2126.0	1702.877234	930.919143	287.0	1009.00	1241.0	2434.75	3599.0
LBE	2126.0	133.303857	9.840844	106.0	126.00	133.0	140.00	160.0
AC	2126.0	2.722484	3.560850	0.0	0.00	1.0	4.00	26.0
FM	2126.0	7.241298	37.125309	0.0	0.00	0.0	2.00	564.0
UC	2126.0	3.659925	2.847094	0.0	1.00	3.0	5.00	23.0
ASTV	2126.0	46.990122	17.192814	12.0	32.00	49.0	61.00	87.0
MSTV	2126.0	1.332785	0.883241	0.2	0.70	1.2	1.70	7.0
ALTV	2126.0	9.846660	18.396880	0.0	0.00	0.0	11.00	91.0
MLTV	2126.0	8.187629	5.628247	0.0	4.60	7.4	10.80	50.7
DL	2126.0	1.570085	2.499229	0.0	0.00	0.0	3.00	16.0
DS	2126.0	0.003293	0.057300	0.0	0.00	0.0	0.00	1.0
DP	2126.0	0.126058	0.464361	0.0	0.00	0.0	0.00	4.0
DR	2126.0	0.000000	0.000000	0.0	0.00	0.0	0.00	0.0
Width	2126.0	70.445908	38.955693	3.0	37.00	67.5	100.00	180.0
Min	2126.0	93.579492	29.560212	50.0	67.00	93.0	120.00	159.0
Max	2126.0	164.025400	17.944183	122.0	152.00	162.0	174.00	238.0
Nmax	2126.0	4.068203	2.949386	0.0	2.00	3.0	6.00	18.0
Nzeros	2126.0	0.323612	0.706059	0.0	0.00	0.0	0.00	10.0
Mode	2126.0	137.452023	16.381289	60.0	129.00	139.0	148.00	187.0
Mean	2126.0	134.610536	15.593596	73.0	125.00	136.0	145.00	182.0
Median	2126.0	138.090310	14.466589	77.0	129.00	139.0	148.00	186.0
Variance	2126.0	18.808090	28.977636	0.0	2.00	7.0	24.00	269.0
Tendency	2126.0	0.320320	0.610829	-1.0	0.00	0.0	1.00	1.0
Target	2126.0	0.221543	0.415383	0.0	0.00	0.0	0.00	1.0

```

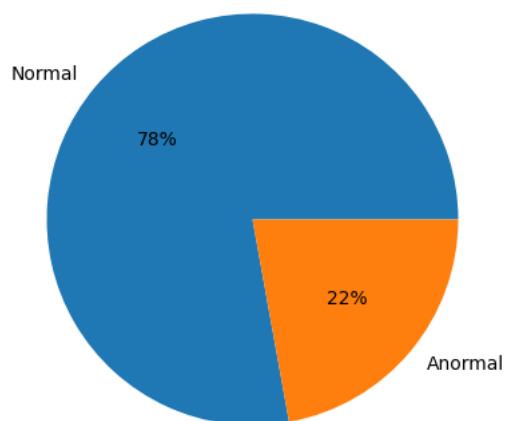
# 3 - DISTRIBUCION DE FRECUENCIAS DEL TARGET
a, b = data['Target'].value_counts()
plt.figure(figsize = (13,5))
plt.subplot(121)
plt.pie([a, b], labels=["Normal", "Anormal"], autopct="%1.0f%%")
print(data["Target"].value_counts())

```

```

Target
0    1655
1     471
Name: count, dtype: int64

```

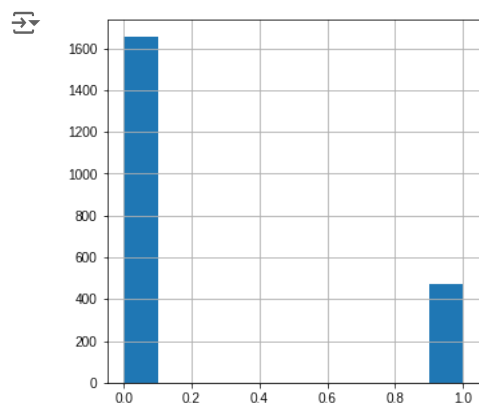


```
# 3 - HISTOGRAMA
```

```

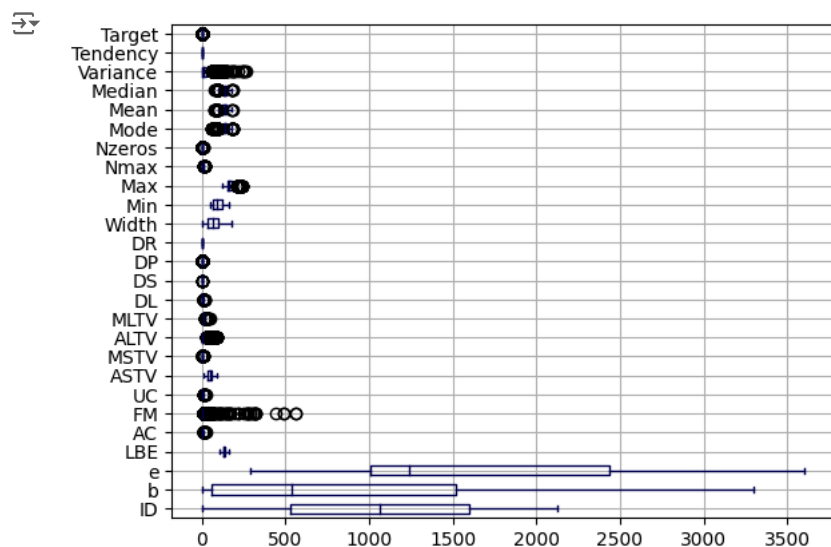
fig, ax = plt.subplots(1,1,figsize=(5,5))
plt.style.use("seaborn")
data['Target'].hist(ax=ax);

```



```
# BOX PLOT
```

```
data_box_plot = data.boxplot(vert=False, color = "#000054")
```



Haz doble clic (o pulsa Intro) para editar

Pregunta 1

```
# Descripción de las variables y su tipo según el PDF
variables = {
    'ID': 'Cualitativa',
    'b': 'Cualitativa',
    'e': 'Cualitativa',
    'LBE': 'Cuantitativa continua',
    'AC': 'Cuantitativa discreta',
    'FM': 'Cuantitativa discreta',
    'UC': 'Cuantitativa discreta',
    'ASTV': 'Cuantitativa continua',
    'MSTV': 'Cuantitativa continua',
    'ALTV': 'Cuantitativa continua',
    'MLTV': 'Cuantitativa continua',
    'DL': 'Cuantitativa discreta',
    'DS': 'Cuantitativa discreta',
    'DP': 'Cuantitativa discreta',
    'DR': 'Cuantitativa discreta',
    'Width': 'Cuantitativa continua',
    'Min': 'Cuantitativa continua',
    'Max': 'Cuantitativa continua',
    'Nmax': 'Cuantitativa discreta',
    'Nzeros': 'Cuantitativa discreta',
    'Mode': 'Cuantitativa continua',
    'Mean': 'Cuantitativa continua',
    'Median': 'Cuantitativa continua',
    'Variance': 'Cuantitativa continua',
    'Tendency': 'Cuantitativa continua',
    'Target': 'Cualitativa'
}

# Crear un DataFrame para mostrar la información
info = []

for column in data.columns:
    dtype = data[column].dtype
    tipo_variable = variables.get(column, 'Desconocido')
    info.append({
        'Variable': column,
        'Tipo': tipo_variable,
        'DType': dtype
    })

info_df = pd.DataFrame(info)

# Mostrar el DataFrame con la información
print(info_df)
```

	Variable	Tipo	DType
0	ID	Cualitativa	int64
1	b	Cualitativa	int64
2	e	Cualitativa	int64
3	LBE	Cuantitativa continua	int64
4	AC	Cuantitativa discreta	int64
5	FM	Cuantitativa discreta	int64
6	UC	Cuantitativa discreta	int64
7	ASTV	Cuantitativa continua	int64
8	MSTV	Cuantitativa continua	float64
9	ALTV	Cuantitativa continua	int64
10	MLTV	Cuantitativa continua	float64
11	DL	Cuantitativa discreta	int64
12	DS	Cuantitativa discreta	int64
13	DP	Cuantitativa discreta	int64
14	DR	Cuantitativa discreta	int64
15	Width	Cuantitativa continua	int64
16	Min	Cuantitativa continua	int64
17	Max	Cuantitativa continua	int64
18	Nmax	Cuantitativa discreta	int64
19	Nzeros	Cuantitativa discreta	int64
20	Mode	Cuantitativa continua	int64
21	Mean	Cuantitativa continua	int64
22	Median	Cuantitativa continua	int64
23	Variance	Cuantitativa continua	int64
24	Tendency	Cuantitativa continua	int64
25	Target	Cualitativa	int64

Pregunta 2

```
# Calcular la proporción de estados fetales normales y anormales
estado_fetal_counts = data['Target'].value_counts()
total_estados = estado_fetal_counts.sum()

# Asumimos que el valor 1 representa "normal" y otros valores representan "anormal"
proporcion_normales = estado_fetal_counts.get(1, 0) / total_estados
proporcion_anormales = 1 - proporcion_normales

# Mostrar los resultados
print(f'Proporción de estados fetales normales: {proporcion_normales:.2f}')
print(f'Proporción de estados fetales anormales: {proporcion_anormales:.2f}')
```

↗

Proporción de estados fetales normales: 0.22
Proporción de estados fetales anormales: 0.78

Pregunta 3

```
# Seleccionar las variables de interés
variables = ['FM', 'ALTV', 'Median']

# Crear un diccionario para almacenar las estadísticas
stats = {}

for var in variables:
    data1 = data[var]

    # Medidas de centralización
    mean = data1.mean()
    median = data1.median()

    # Medidas de localización
    q1 = data1.quantile(0.25)
    q2 = data1.median() # Es la mediana
    q3 = data1.quantile(0.75)

    # Medidas de dispersión
    data_range = data1.max() - data1.min()
    variance = data1.var()
    std_dev = data1.std()

    # Almacenar las estadísticas en el diccionario
    stats[var] = {
        'Mean': mean,
        'Median': median,
        'Q1': q1,
        'Q2 (Median)': q2,
        'Q3': q3,
        'Range': data_range,
        'Variance': variance,
        'Standard Deviation': std_dev
    }

# Crear un DataFrame con las estadísticas
stats_df = pd.DataFrame(stats)

# Mostrar el DataFrame con las estadísticas
print(stats_df)
```

↗

	FM \
Mean	7.241298
Median	0.0
Q1	0.0
Q2 (Median)	0.0
Q3	2.0
Range	ID 563.0
b	564.0
e	...
Variance	1378.288571
Standard Deviation	37.125309

	ALTV \
Mean	9.84666
Median	0.0
Q1	0.0
Q2 (Median)	0.0
Q3	11.0
Range	ID 90.0
b	91.0
e	...
Variance	338.445182
Standard Deviation	18.39688

Median

Mean	138.09031
Median	139.0
Q1	129.0
Q2 (Median)	139.0
Q3	148.0
Range	ID 185.0
b	186.0
e	...
Variance	209.282193
Standard Deviation	14.466589

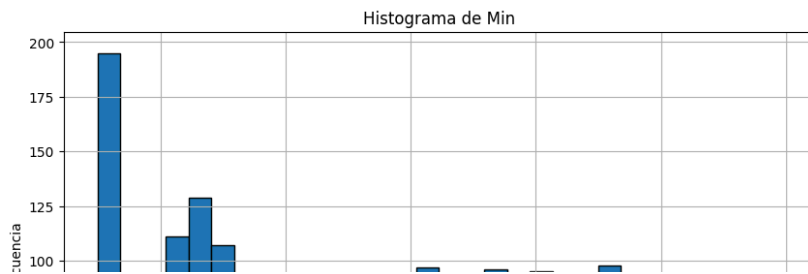
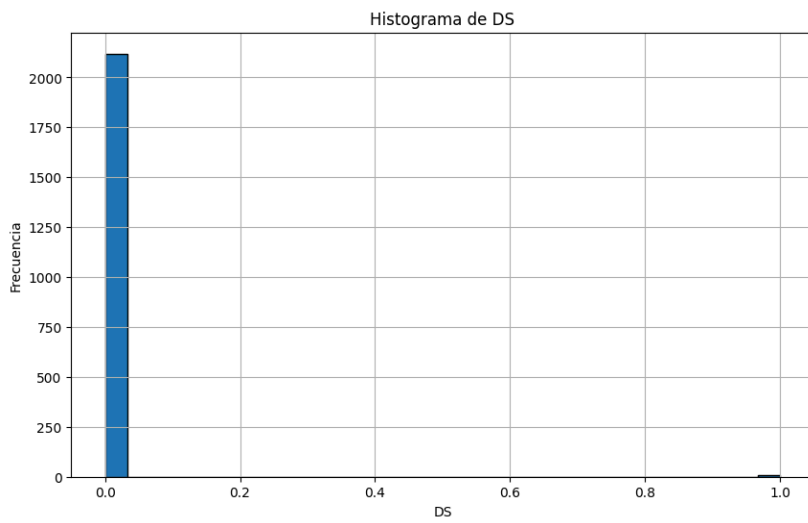
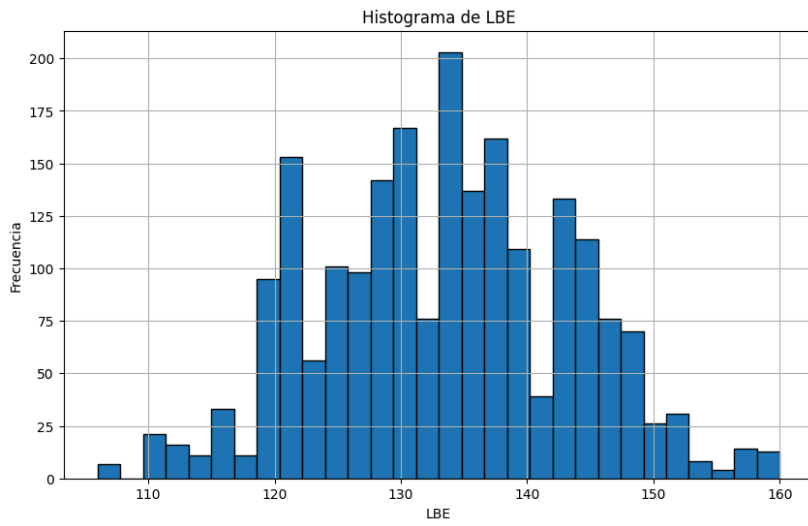
Pregunta 4

```
import matplotlib.pyplot as plt
```

```
# Seleccionar las variables de interés
variables = ['LBE', 'DS', 'Min']
```

```
# Crear histogramas para cada variable
for var in variables:
```

```
    plt.figure(figsize=(10, 6))
    plt.hist(data[var], bins=30, edgecolor='black')
    plt.title(f'Histograma de {var}')
    plt.xlabel(var)
    plt.ylabel('Frecuencia')
    plt.grid(True)
    plt.show()
```



Pregunta 5

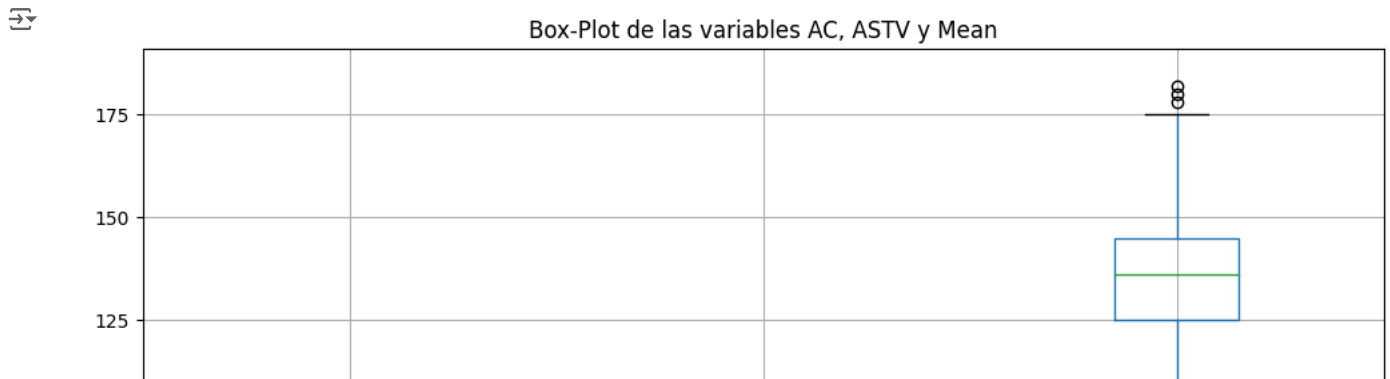
```
# Seleccionar las variables de interés
variables = ['AC', 'ASTV', 'Mean']

# Crear un gráfico box-plot para cada variable
plt.figure(figsize=(12, 8))

# Generar el box-plot para las variables seleccionadas
data[variables].boxplot()

# Configurar el título y etiquetas de los ejes
plt.title('Box-Plot de las variables AC, ASTV y Mean')
plt.xlabel('Variables')
plt.ylabel('Valores')
plt.grid(True)

# Mostrar el gráfico
plt.show()
```



```
# CLASE 1
```

```
# IMPORTACION DE LIBRERIAS
```

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

```
# LIBRERIAS PARA HACER GRAFICOS
```

```
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as xp
```

```
# 2 - CARGA DEL FICHERO DE DATOS
```

```
file = '/content/drive/MyDrive/ColabNotebooks3/Aprendisaje_supervisado_1,S2/ASI_casoPractico (1).csv'
data = pd.read_csv(file, sep = ';')
data.head()
```

```

ID      b      e  LBE  AC  FM  UC  ASTV  MSTV  ALTV  ...  Min  Max  Nmax  Nzeros  Mode
0    1  240   357  120   0   0   0    73   0.5   43  ...  62  126    2      0   120
1    2    5   632  132   4   0   4    17   2.1    0  ...  68  198    6      1   141
2    3  177   779  133   2   0   5    16   2.1    0  ...  68  198    5      1   141
3    4  411  1192  134   2   0   6    16   2.4    0  ...  53  170   11      0   137
4    5  533  1147  132   4   0   5    16   2.4    0  ...  53  170    9      0   137

```

5 rows × 26 columns

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount
```

```
# 2 - INFORMACION DEL CONJUNTO DE DATOS
```

```
data.info()
```


```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2126 entries, 0 to 2125
Data columns (total 26 columns):
#   Column      Non-Null Count  Dtype
---  -
0   ID          2126 non-null   int64
1   b           2126 non-null   int64
2   e           2126 non-null   int64
3   LBE         2126 non-null   int64
4   AC          2126 non-null   int64
5   FM          2126 non-null   int64
6   UC          2126 non-null   int64
7   ASTV        2126 non-null   int64
8   MSTV        2126 non-null   float64
9   ALTV        2126 non-null   int64
10  MLTV        2126 non-null   float64
11  DL          2126 non-null   int64
12  DS          2126 non-null   int64
13  DP          2126 non-null   int64
14  DR          2126 non-null   int64
15  Width       2126 non-null   int64
16  Min         2126 non-null   int64
17  Max         2126 non-null   int64
18  Nmax        2126 non-null   int64
19  Nzeros      2126 non-null   int64
20  Mode        2126 non-null   int64
21  Mean        2126 non-null   int64
22  Median      2126 non-null   int64
23  Variance    2126 non-null   int64
24  Tendency    2126 non-null   int64
25  Target      2126 non-null   int64
dtypes: float64(2), int64(24)
memory usage: 432.0 KB


```

```
# 2 - NUMERO DE VALORES UNICOS PARA CADA VARIABLE
```


```
data.agg(['nunique']).T
```



	nunique
ID	2126
b	979
e	1064
LBE	48
AC	22
FM	96
UC	19
ASTV	75
MSTV	57
ALTV	87
MLTV	249
DL	15
DS	2
DP	5
DR	1
Width	154
Min	109
Max	86
Nmax	18
Nzeros	9
Mode	88
Mean	103
Median	95
Variance	133
Tendency	3
Target	2



```
# 2 -ELIMINAR COLUMNS NO NECESARIAS
data = data.drop(["ID","b","e","DR"], axis = 1)
data.head()
```




	LBE	AC	FM	UC	ASTV	MSTV	ALTV	MLTV	DL	DS	...	Min	Max	Nmax	Nzeros	Mode
0	120	0	0	0	73	0.5	43	2.4	0	0	...	62	126	2	0	120
1	132	4	0	4	17	2.1	0	10.4	2	0	...	68	198	6	1	141
2	133	2	0	5	16	2.1	0	13.4	2	0	...	68	198	5	1	141
3	134	2	0	6	16	2.4	0	23.0	2	0	...	53	170	11	0	137
4	132	4	0	5	16	2.4	0	19.9	0	0	...	53	170	9	0	137

5 rows × 22 columns



```
# 3 - ANALISIS DESCRIPTIVO - NULOS
data.isnull().sum()
```



[Mostrar salida oculta](#)


```
# 3 - MEDIDAS DE CENTRALIZACION, LOCALIZACION Y DISPERSION
data.describe().T
```




[Mostrar salida oculta](#)

```
# 3 - DISTRIBUCION DE FRECUENCIAS DEL TARGET
```


```
a, b = data['Target'].value_counts()
plt.figure(figsize = (13,5))
plt.subplot(121)
plt.pie([a, b], labels=["Normal", "Anormal"], autopct="%1.0f%%")
print(data["Target"].value_counts())
```

 [Mostrar salida oculta](#)

```
# 3 - HISTOGRAMA
fig, ax = plt.subplots(1,1,figsize=(5,5))
plt.style.use("seaborn")
data['Target'].hist(ax=ax);
```


 [Mostrar salida oculta](#)

```
# BOX PLOT
data_box_plot = data.boxplot(vert=False, color = "#000054")
```


 [Mostrar salida oculta](#)

```
# CLASE 2
```

```
# 3 - MATRIZ DE CORRELACIONES
plt.figure(figsize=(20,20))
corr = data.corr()
cmap = sns.color_palette("light:b", as_cmap=True)
sns.heatmap(corr, annot=True, cmap=cmap)
```

 [Mostrar salida oculta](#)

```
# 3 - RELACION UNIVARIANTE DEL TARGET CON LAS VARIABLES EXPLICATIVAS
plt.figure(figsize = (9,6))
sns.boxplot(x = 'Target', y = 'ASTV', data = data, palette="Blues")
plt.title('Baseline Fetal Heart Rate (FHR) for each status')
```

 [Mostrar salida oculta](#)

```
# MUESTREO
# ENTRENAMIENTO Y TEST
from sklearn.model_selection import train_test_split
X = data.loc[:, data.columns != "Target"]
y = data.loc[:, data.columns == "Target"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.50, random_state = 0)
```

```
# MODELIZACION
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
# PREDICCIONES
modelNB = gnb.fit(X_train, y_train.values.ravel())
y_pred_train = modelNB.predict_proba(X_train)
y_pred_test = modelNB.predict_proba(X_test)
```

```

# # VALIDACION
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve, auc

# CURVA ROC Y AUC PARA TRAINING
fpr_train, tpr_train, _ = roc_curve(y_train, y_pred_train[:, 1])
roc_auc_train = auc(fpr_train, tpr_train)

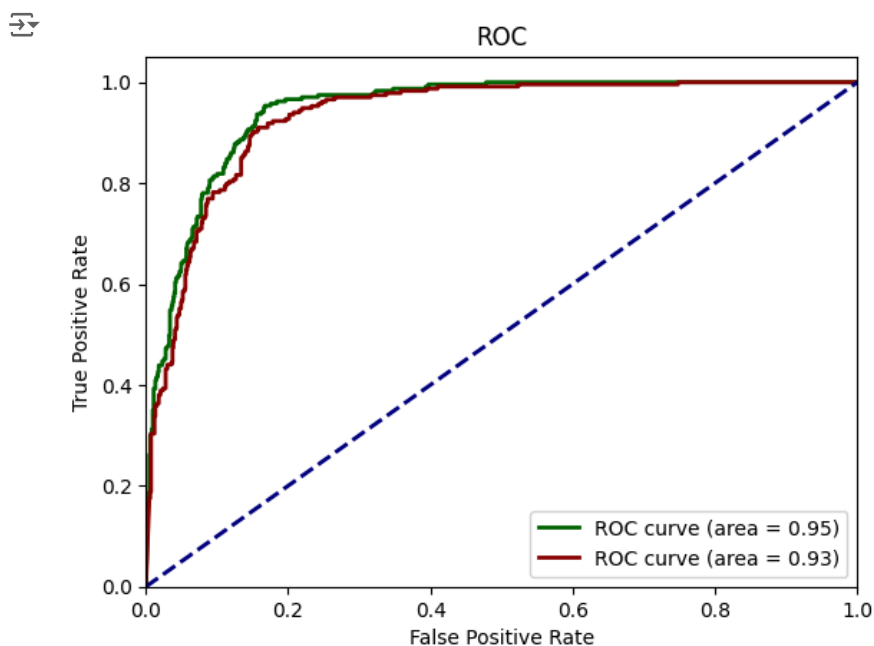
# CURVA ROC Y AUC PARA TEST
fpr_test, tpr_test, _ = roc_curve(y_test, y_pred_test[:, 1])
roc_auc_test = auc(fpr_test, tpr_test)

# GRAFICA
plt.figure()
lw = 2

plt.plot(fpr_train, tpr_train, color="darkgreen", lw=lw, label="ROC curve (area = %0.2f)" % roc_auc_train,)
plt.plot(fpr_test, tpr_test, color="darkred", lw=lw, label="ROC curve (area = %0.2f)" % roc_auc_test,)

plt.plot([0, 1], [0, 1], color="navy", lw=lw, linestyle="--")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC")
plt.legend(loc="lower right")
plt.show()

```



```
# VALIDACION. CURVA ROC, AUC Y MATRIZ DE CONFUSION
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve, auc
"""

fpr_train_svm, tpr_train_svm, _ = roc_curve(y_train, y_pred_train_svm[:, 1])
roc_auc_train_svm = auc(fpr_train_svm, tpr_train_svm)

# CURVA ROC Y AUC PARA TEST
fpr_test_svm, tpr_test_svm, _ = roc_curve(y_test, y_pred_test_svm[:, 1])
roc_auc_test_svm = auc(fpr_test_svm, tpr_test_svm)

"""

# CURVA ROC Y AUC PARA TRAINING
fpr_train, tpr_train, _ = roc_curve(y_train, y_pred_train[:, 1])
roc_auc_train = auc(fpr_train, tpr_train)

# CURVA ROC Y AUC PARA TEST
fpr_test, tpr_test, _ = roc_curve(y_test, y_pred_test[:, 1])
roc_auc_test = auc(fpr_test, tpr_test)
```

✓ Solucion Sprint 2

1_Las tres variables explicativas que presentan mayor correlación con el target son:

ASTV con una correlación de 0.493391 ALTV con una correlación de 0.489400 DP con una correlación de 0.340647

```
# Calcular la matriz de correlación
correlation_matrix = data.corr()

# Obtener la correlación con el target
target_correlation = correlation_matrix["Target"].sort_values(ascending=False)

# Obtener las 3 variables explicativas con mayor correlación con el target
top_3_correlated_variables = target_correlation.index[1:4]

top_3_correlated_variables, target_correlation[top_3_correlated_variables]
```

```
(Index(['ASTV', 'ALTV', 'DP'], dtype='object'),
 ASTV    0.493391
 ALTV    0.489400
 DP      0.340647
 Name: Target, dtype: float64)
```

2_Cree un conjunto de entrenamiento y otro de test a partir del conjunto inicial de datos donde el conjunto de entrenamiento contenga el 60% de las observaciones.

```
from sklearn.model_selection import train_test_split

# Definir las características (X) y el target (y)
X = data.loc[:, data.columns != "Target"]
y = data.loc[:, data.columns == "Target"]

# Crear conjunto de entrenamiento y test con el 60% para entrenamiento
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.40, random_state=0)

# Verificar los tamaños de los conjuntos
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((1275, 21), (851, 21), (1275, 1), (851, 1))
```

Conjunto de entrenamiento (60%): 1275 observaciones. Conjunto de prueba (40%): 851 observaciones.

3_A partir del conjunto de entrenamiento ajuste un modelo a partir del algoritmo de Naive Bayes.

```
from sklearn.naive_bayes import GaussianNB
```

```
# Crear y ajustar el modelo de Naive Bayes con el conjunto de entrenamiento
gnb = GaussianNB()
modelNB = gnb.fit(X_train, y_train.values.ravel())
```

```
# Verificar el ajuste del modelo
modelNB
```



```
▼ GaussianNB
GaussianNB()
```

Evaluar el Modelo

```
from sklearn.metrics import accuracy_score

# Hacer predicciones en el conjunto de prueba
y_pred_test = modelNB.predict(X_test)

# Calcular la precisión del modelo
accuracy = accuracy_score(y_test, y_pred_test)
accuracy
```

4_Obtenga la curva ROC y el área bajo la curva para los conjuntos de entrenamiento y test

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

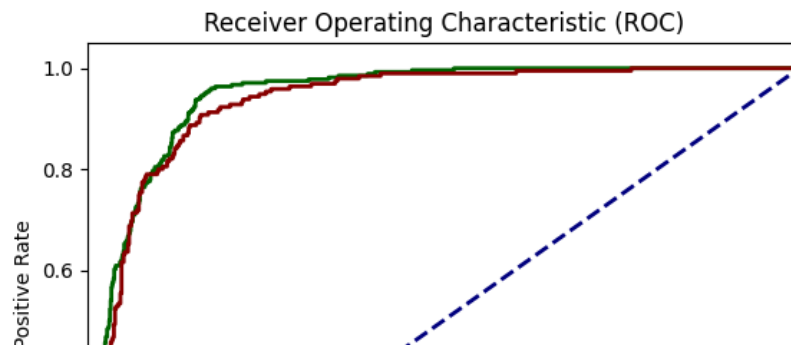
# Obtener las probabilidades de predicción para el conjunto de entrenamiento y prueba
y_pred_train_proba = modelNB.predict_proba(X_train)[: , 1]
y_pred_test_proba = modelNB.predict_proba(X_test)[: , 1]

# Calcular la curva ROC y el área bajo la curva (AUC) para el conjunto de entrenamiento
fpr_train, tpr_train, _ = roc_curve(y_train, y_pred_train_proba)
roc_auc_train = auc(fpr_train, tpr_train)

# Calcular la curva ROC y el área bajo la curva (AUC) para el conjunto de prueba
fpr_test, tpr_test, _ = roc_curve(y_test, y_pred_test_proba)
roc_auc_test = auc(fpr_test, tpr_test)

# Graficar la curva ROC
plt.figure()
lw = 2
plt.plot(fpr_train, tpr_train, color="darkgreen", lw=lw, label="ROC curve train (area = %0.2f)" % roc_auc_train)
plt.plot(fpr_test, tpr_test, color="darkred", lw=lw, label="ROC curve test (area = %0.2f)" % roc_auc_test)
plt.plot([0, 1], [0, 1], color="navy", lw=lw, linestyle="--")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operating Characteristic (ROC)")
plt.legend(loc="lower right")
plt.show()

roc_auc_train, roc_auc_test
```




```
#####
# SVM LINEAL #
#####

# Tratamiento de datos
# =====
import pandas as pd
import numpy as np

# Preprocesado, modelado y validacion
# =====
from sklearn import svm
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve, auc

# Gráficos
# =====
import matplotlib.pyplot as plt

# Configuración warnings
# =====
import warnings
warnings.filterwarnings('ignore')
from google.colab import drive
drive.mount('/gdrive')
file = '/gdrive/MyDrive/Sprint3_Supervizado/ASI_casoPractico (2).csv'
#file = '/content/data/ASI_casoPractico.csv'
data = pd.read_csv(file, sep = ';')
data = data.drop(["ID", "b", "e", "DR"], axis = 1)

# Solo se cogen dos variables a modo de ejemplo: Mean y Variance

X = data.loc[:, data.columns != "Target"]
y = data.loc[:, data.columns == "Target"]
print(X)
print(y)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.50, random_state = 0)
```

Drive already mounted at /gdrive; to attempt to forcibly remount, call drive.mount("/gdrive", force_remount=True).

	LBE	AC	FM	UC	ASTV	MSTV	ALTV	MLTV	DL	DS	...	Width	Min	Max	\
0	120	0	0	0	73	0.5	43	2.4	0	0	...	64	62	126	
1	132	4	0	4	17	2.1	0	10.4	2	0	...	130	68	198	
2	133	2	0	5	16	2.1	0	13.4	2	0	...	130	68	198	
3	134	2	0	6	16	2.4	0	23.0	2	0	...	117	53	170	
4	132	4	0	5	16	2.4	0	19.9	0	0	...	117	53	170	
...
2121	140	0	0	6	79	0.2	25	7.2	0	0	...	40	137	177	
2122	140	1	0	9	78	0.4	22	7.1	0	0	...	66	103	169	
2123	140	1	0	7	79	0.4	20	6.1	0	0	...	67	103	170	
2124	140	1	0	9	78	0.4	27	7.0	0	0	...	66	103	169	
2125	142	1	1	5	74	0.4	36	5.0	0	0	...	42	117	159	

	Nmax	Nzeros	Mode	Mean	Median	Variance	Tendency
0	2	0	120	137	121	73	1
1	6	1	141	136	140	12	0
2	5	1	141	135	138	13	0
3	11	0	137	134	137	13	1
4	9	0	137	136	138	11	1
...
2121	4	0	153	150	152	2	0
2122	6	0	152	148	151	3	1
2123	5	0	153	148	152	4	1
2124	6	0	152	147	151	4	1
2125	2	1	145	143	145	1	0

[2126 rows x 21 columns]

	Target
0	1
1	0
2	0
3	0
4	0
...	...
2121	1
2122	1
2123	1

```
2124      1
2125      0
```

```
[2126 rows x 1 columns]
```

```
# SVM Lineal
svmLineal = svm.SVC(C = 1
                    ,kernel='linear'
                    ,degree = 1
                    ,probability = True)

# Ajuste del modelo
svmLineal.fit(X_train, y_train)

# VALIDACION CON EL CONJUNTO DE TEST
y_proba_train_svm = svmLineal.predict_proba(X_train)
y_proba_test_svm = svmLineal.predict_proba(X_test)

# Entrenamiento
fpr_train_svm, tpr_train_svm, _ = roc_curve(y_train, y_proba_train_svm[:, 1])
roc_auc_train_svm = auc(fpr_train_svm, tpr_train_svm)
#
# Test
fpr_test_svm, tpr_test_svm, _ = roc_curve(y_test, y_proba_test_svm[:, 1])
roc_auc_test_svm = auc(fpr_test_svm, tpr_test_svm)

# Curva ROC y AUC
plt.figure()
lw = 2

plt.plot(fpr_train_svm, tpr_train_svm, color="darkgreen", lw=lw, label="ROC curve (area = %0.2f)" % roc_auc_train_svm,)
plt.plot(fpr_test_svm, tpr_test_svm, color="darkred", lw=lw, label="ROC curve (area = %0.2f)" % roc_auc_test_svm,)

plt.plot([0, 1], [0, 1], color="navy", lw=lw, linestyle="--")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC")
plt.legend(loc="lower right")
plt.show()
```

