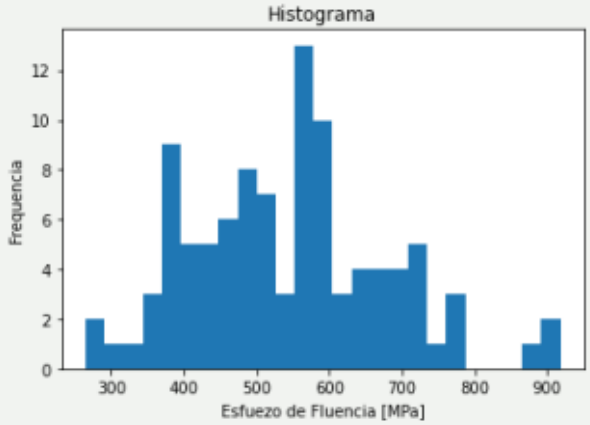
- JULIAN AUGUSTO CORTES GOMEZ - COD: 1803147
- UMNG - INGENIERIA FORENSE
- PROF. ALEJANDRO CASTELLANOS

```
from google.colab import drive
drive.mount('/content/drive')
from random import random
from scipy import stats
!pip install matplotlib
import matplotlib.pyplot as plt
import os
import pandas as pd
!pip install fitter
!pip install xlrd
from fitter import Fitter, get_distributions, get_common_distributions
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (fr
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Requirement already satisfied: fitter in /usr/local/lib/python3.7/dist-packages (1.4
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: easydev in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: scipy>=0.18 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: colorama in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: colorlog in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: pexpect in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.7/dist-packa
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Requirement already satisfied: xlrd in /usr/local/lib/python3.7/dist-packages (1.1.0)
```

Caso 1: Propiedades de un material

Usted realiza 100 ensayos de tensión para determinar el esfuerzo de fluencia de una acero 4140. La medida de esta propiedad tiene le comportamiento que se muestra en la figura. Para un diseño de una pieza se establece el esfuerzo de fluencia mínimo en 400 MPa, determine la probabilidad de que el material de la pieza no cumpla con la resistencia deseada.

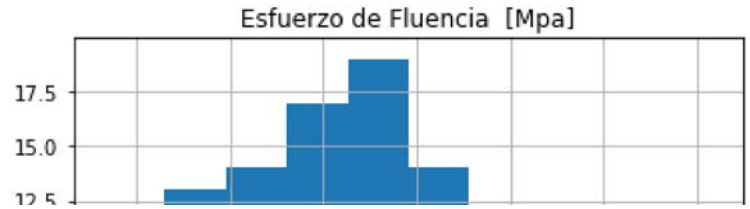



```
archivo = '/content/drive/MyDrive/Datos.xlsx'
df = pd.read_excel(archivo, sheet_name='Caso1')
df.describe()
```

	Esfuerzo de Fluencia [Mpa]
count	100.000000
mean	544.642391
std	135.052183
min	265.005088
25%	443.460679
50%	551.909467
75%	622.837458
max	917.492766

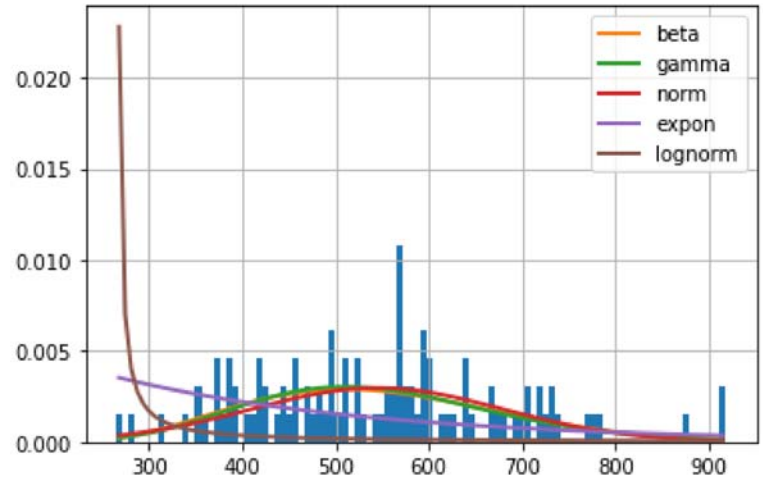
```
df.hist()
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fafbb37e8d0>]],
      dtype=object)
```




```
f = Fitter(df, distributions=['gamma','lognorm','beta','expon','norm'])
f.fit()
f.summary()
```

	sumsquare_error	aic	bic	kl_div
beta	0.000250	1368.595460	-1271.469685	inf
gamma	0.000251	1371.396014	-1275.784370	inf
norm	0.000251	1376.407981	-1280.119568	inf
expon	0.000396	1364.032245	-1234.801837	inf
lognorm	0.001040	1736.097382	-1133.567899	inf




```
f.get_best(method='sumsquare_error')
```

```
{'beta': {'a': 4.903736596010129,
  'b': 11.61718725546137,
  'loc': 178.75133994774788,
  'scale': 1232.666495391395}}
```

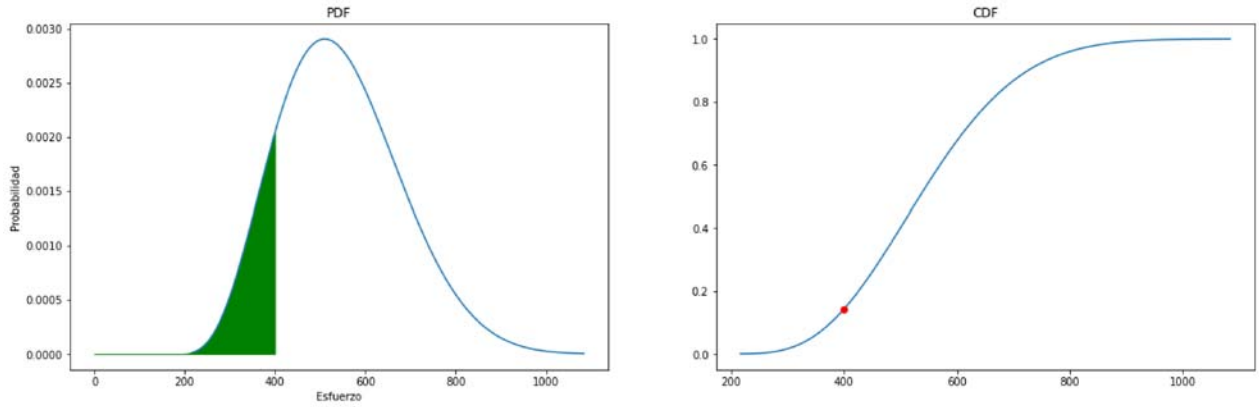
```
f.fitted_param["beta"]
```

```
(4.903736596010129, 11.61718725546137, 178.75133994774788, 1232.666495391395)
```

```
a = f.fitted_param["beta"][0]
b = f.fitted_param["beta"][1]
loc = f.fitted_param["beta"][2]
scale = f.fitted_param["beta"][3]
import numpy as np
x = np.linspace(stats.beta(a,b,loc,scale).ppf(0.0001),
```

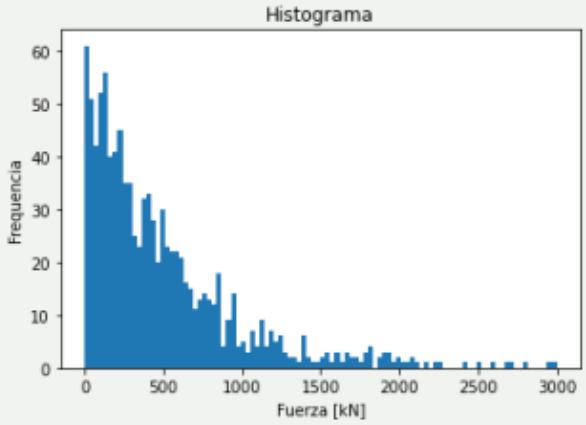
```
                stats.beta(a,b,loc,scale).ppf(0.9999),
                1000)
dist = stats.beta(a,b,loc,scale)
valor = 400
fig,(ax1,ax2) = plt.subplots(1,2,figsize=(20,6))
ax1.plot(x, dist.pdf(x))
px = np.arange(0, valor, 0.001)
ax1.fill_between(px, dist.pdf(px), color = 'g')
ax1.set_title("PDF")
ax1.set_xlabel('Esfuerzo')
ax1.set_ylabel('Densidad')

fda_beta = dist.cdf(x)

ax2.plot(x, fda_beta)
ax2.plot(valor,dist.cdf(valor),"ro")
ax2.set_title("CDF")
ax1.set_xlabel('Esfuerzo')
ax1.set_ylabel('Probabilidad')

prob1 = dist.cdf(valor)
print(f'La probabilidad de sacar menos de {valor}', f'es: {prob1}')
```

```
La probabilidad de sacar menos de 400 es: 0.14252938006727864
```



Caso 2: Fuerza de una Prensa

Una prensa mecánica presenta fallas de forma constante, por lo que se realiza un estudio con respecto a la fuerza que aplica en cada proceso. Se realizan 1000 pruebas registrando la fuerza que genera la prensa. De acuerdo con el fabricante, la prensa soporta fuerzas hasta de 1300 kN. Con base en los datos, determine la probabilidad de que la prensa exceda la fuerza permitida.




```
archivo = '/content/drive/MyDrive/Datos.xlsx'
df = pd.read_excel(archivo, sheet_name='Caso 2')
df.describe()
```

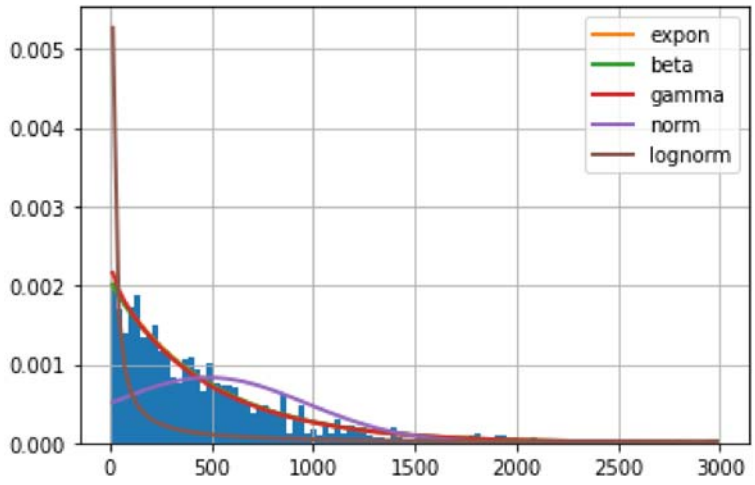
	Fueza [kN]
count	1000.000000
mean	485.928757
std	481.054123
min	0.490356
25%	144.741650
50%	347.637513
75%	652.556676
max	2997.212340

```
df.hist()
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fafbb274cd0>]],
      dtype=object)
```


```
f = Fitter(df, distributions=['gamma','lognorm','beta','expon','norm'])
f.fit()
f.summary()
```

	sumsquare_error	aic	bic	kl_div
expon	9.297130e-07	1858.333291	-20782.329664	inf
beta	9.322977e-07	1856.695012	-20765.737886	inf
gamma	1.015085e-06	1852.661219	-20687.570441	inf
norm	1.067366e-05	2190.374970	-18341.670866	inf
lognorm	2.701613e-05	2031.982822	-17406.108576	inf




```
f.get_best(method='sumsquare_error')
```

```
{'expon': {'loc': 0.4903556588589409, 'scale': 485.43840138541145}}
```

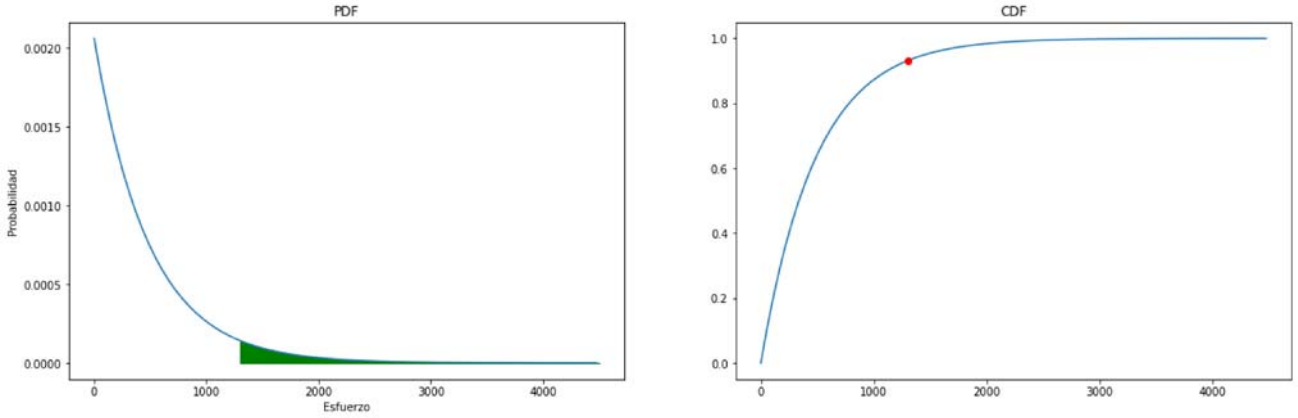
```
f.fitted_param["expon"]
```

```
(0.4903556588589409, 485.43840138541145)
```

```
loc = f.fitted_param["expon"][0]
scale = f.fitted_param["expon"][1]

import numpy as np
x2 = np.linspace(stats.expon(loc, scale).ppf(0.0001),
                 stats.expon(loc, scale).ppf(0.9999),
                 1000)

dist2 = stats.expon(loc, scale)
```

```
valor = 1300
fig,(ax1,ax2) = plt.subplots(1,2,figsize=(20,6))
ax1.plot(x2, dist2.pdf(x2))
px2 = np.arange(valor,4500, 0.001)
ax1.fill_between(px2, dist2.pdf(px2), color = 'g')
ax1.set_title("PDF")
ax1.set_xlabel('Esfuerzo')
ax1.set_ylabel('Densidad')

fda_expon = dist2.cdf(x2)

ax2.plot(x2, fda_expon)
ax2.plot(valor,dist2.cdf(valor),"ro")
ax2.set_title("CDF")
ax1.set_xlabel('Esfuerzo')
ax1.set_ylabel('Probabilidad')

prob1 = dist2.cdf(valor)
print(f'La probabilidad de sacar mas de {valor}kN', f'es: {1-prob1}')
```

```
La probabilidad de sacar mas de 1300kN es: 0.06877042123805688
```




```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf("ForenseMiNIp.ipynb")
```

VlyxbaMCaEEEIICSvQKCZ+Z9GixfLnn9vkzp27cuL4Cfn++xmMKSaEEEJImIJGMfE7adOmlRw5c0iOH
NklZqyYcvbsGXn8iKtPEEIIISTsQKOY+J1s2d7RE+0KFyksdfUaxbHk0eNHxreEEEIIIa8eGsXE7/z+
+++yZPES/flh3nx58OCBRIoYyfiWEEIIIeTVQ6OY+J2nT5/pJdiePXXsmCRMllBYtmku8+PGMbwkhhBB
CXj00ionfiRkjhsSNG1fixIkjUSJHlsOHD8vqVavlz61/GlcQQggghhLxaaBQTv7N161aZMmWq/P33P7
J8+QpZvGixHDx4SM6cPWtc2Zrbt2/LuLHjpGGDRjKg/wC5ePGi8Q0hhBBCiG+gUUz8zsWLl2TsuG+kZ
68e8tXXYyRV6lRSr35dqV69mnGFcxBysXTJkORkRI4ZMnTZFevXuJfHiMfSCEEIIIb6FRjHx09i97pef
f5Ftf26TdWvXyc0bNyVChAjGt665f5f/++nDp1Sv+7dOky2b9/f4jf3rlzR3799Vf5Yd4PcuHCeMs8ZZ
pW/6TiRuPuf38fvSy8QtCCCHkzYBGMfE7derW1h5h5fhFEcP35cqlWvpmOMrXD//gO5fPmyXL16ViSHii
QrV6yUQ4cOGd8qAQ4fXmLHjiPxEySQiJEiGmeJt4z85bAMWXnA7WfNfoawEEIIebOgUUz8TooUKSRz5
sySIUMGqVmzpqR+O7U2Zq0QNWoU7WNoU7WkuULCAlC5TWpIkSarjkU2iRYsm7733PylVqqQkiJ/AOEsIIYQQ
4hk0ionfmT7Te1m4YKH8/PPPPEEiNmDBk/7lvLk+Vg9GbPkUNOnzote/7ZI+fPn5OUysgmhBBCCPElNIq
J3zmwf7+0btNaIkeOIlGiRNGhFFi32ColS5aQcOHCyaZNmyV37tzy7v/eNBM4o/cX8SrmsG4BQQ4zyr0fNLw
Os37BBLl64IAsXLtKhE9GiRjO+dQ/WN67foL507tJJypQtIxEjMnaYEEIIIb6FRjHxO06VKlAenNLw
oQJtWFct14diRvP2kQ7QgghhJCXAY1i4nemTZ0mhQp9IF+O+O2+5RTzWwnSxs8e4sXEEIIefXQKCZ+49GjR/LgwPGjBsxXE
VIYQQQsirh0Yx8Rv//LNHft3wq5Qtxbcv6v5MMsXwGlnhyrbyE2L9yjNmu8oPvMeEEEIIIWEFGsXEb9y4cUxbkks
JFkl07d0nUqFGDP1hijRBCCCEkrECjmPiNx48deyd69+3QM8ZmzZ/W/W/W/W/5ocxxYQQQQggJS4S7cev6M+Nv
nzKwQMHZfnyfXFlCclEsIYQ264v3LxFM73+sC9p1aK1zzzh6vwhSeNENY7CFiev3Eod3zg
i3lK7dm2ZNGWicUQICVRoFJM3hkA0it8EaBS/BQDJ8ghB
BCCCEBD41iQgghhBAS8NAoJoQQQQgghAQ+NYkIIIYQQEvDQKCaEEEIIIQEPjWJCCCGEEBLw0CgmhBBCC
CEBD41iQgghhBAS8NAoJoQQQQgghAQ+NYvJacOPGDfmwZi0OWKUcYYQQgghxHfQKCZhnkePsmC+Qvl
nXeyypMnT4yzhBBBCCCG+I9yNefX8efGX8TEib5c+ufsmvXLkmRIoUcOHBQunXqNJmBBQunXvanwjcvPmTfnt19/k5Mm
TsnLFKtm0aZPxzevzHxz4yjN5sI4cPJ0SHljSPn/LLvvLSaudM4Cj1be5aQpHGiGkfO2X3qmv
z09znj6OVw9vo9WbX3vHHknN4VskjLQmmNI+IttWvXlklTJhpHhJBAhUYxCdNcvX5Ve4/hXRJBAhUYxCdNcvXJV5s+fL/ny55NLF
y/JX3/tDmEUP3z4UE6dPCU3xHE8/IvhsmLFCuOb1/yqozi+TtOSbeF/xhHYQsaxb6BRjEhDB8
goRprly5Inv27JVBAwbJkCkMFDZPWVfLbb78Z34xj78Z34pEjxqVAqKaxVPJ++FuSuWcWzCZYQQghh
MGWTc+LGyeOli6d6tV6dWnt5JL9GeYEkOBwK7xeOELi6dWnt5QtV06KFClMHPnp0iPgTKKFChMqlMrjUHPRB
lQsIBxRAghhBDiO2gUE0IIIIIQEgAl4ICHRjEhhBBCCCEBD41iQgghhBBCCCEBD41iQgghhBAS8NAoJo
0YxIYQQQggJeGgUE0IQAhjhBBCCCEBD41iQgghhBAS8NAoJoQQQggghAQ+NYkIIIYQQEvDQKCaEEEIIIQEPjWWeM
ozebcOpTO0/KoAMXnL52TzYfuWwchZ4qOZNLtMgRjCPnHLt8R7b9d9U4Clv0rpBFWhZKaxw5Z+Qvh+T
SrQfGkXOyvRVHGuRLbRwFDrVr15ZJUyYaR4SQQIVGMXljeF2M4hrjf5edJ68ZR4R4j1WjuPjIX+XYpT
vGkXPKZUsq3zbIZRwFDjSKCSGA4ROEEEIISTgoVFMCCGEEEIICHhrFhBBCCCEk4KFRTAghhBBCAh
YQQQgghJOChUUwIIYQQQgghJOChUUwIIYQQQgghJOChUUwIIYQQQgghJOChUUwIIYQQQgghJOChUUwIIYQQQgghJOChUUwIIYQQ
EEII8R00ikmYBgx0iVLpX+//tK3Tz+ZO2OPc2sfN1y9dlxZVe5m49jrBgX9ycs66kadDZyykBQBaEF+nMT2k