

**Documento para utilización  
Modelo de clasificación Enero 2022**

César Hooper  
Marzo 2023  
Concepción

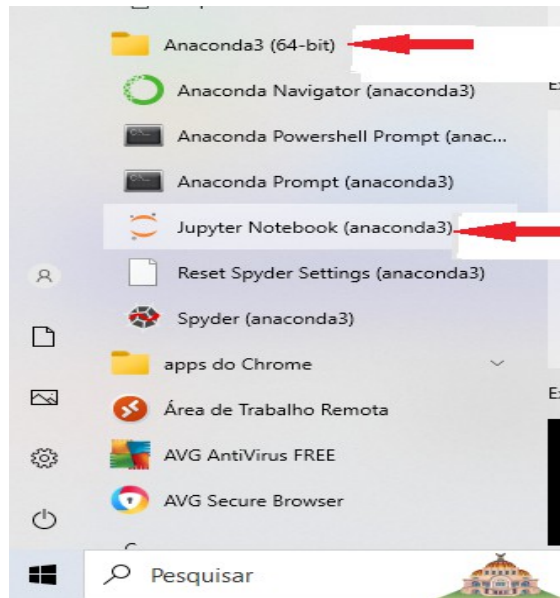
# Procedimiento para cargar Jupyter Notebook

## Paso 0

Instalar Anaconda. Éste software se puede descargar desde [www.anaconda.com](http://www.anaconda.com) para la versión de windows deseada. Es importante instalar la versión adecuada para la versión de windows (32 bit ó 64 bit)

## Paso 1

Una vez instalado anaconda dirigirse a INICIO y buscar carpeta ANACONDA. Desplegarla y presionar JUPYTER NOTEBOOK (flecha roja)



## Paso 2

Se abrirá el prompt de carga del navegador de Jupyter Notebook

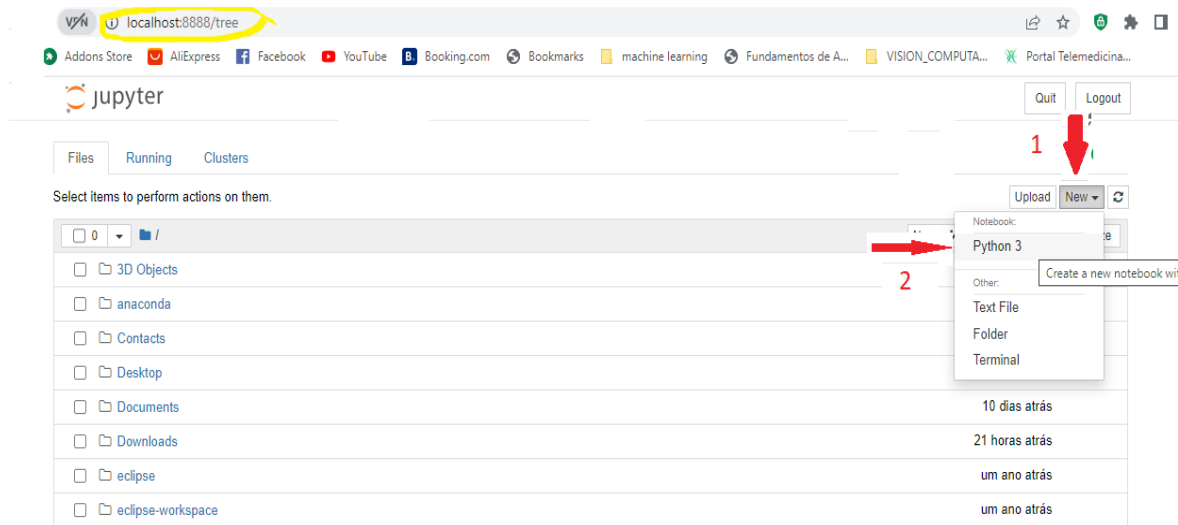
```
Jupyter Notebook (anaconda3)
[I 2023-02-24 19:17:00.696 LabApp] JupyterLab extension loaded from C:\Users\Cesar Hooper\anaconda\anaconda3\lib\site-packages\jupyterlab
[I 2023-02-24 19:17:00.696 LabApp] JupyterLab application directory is C:\Users\Cesar Hooper\anaconda\anaconda3\share\jupyterlab
[I 19:17:00.712 NotebookApp] Serving notebooks from local directory: C:\Users\Cesar Hooper
[I 19:17:00.712 NotebookApp] Jupyter Notebook 6.3.0 is running at:
[I 19:17:00.712 NotebookApp] http://localhost:8888/?token=aa79e089a43ace370a98b9a78537e99be8932248432db5f8
[I 19:17:00.712 NotebookApp] or http://127.0.0.1:8888/?token=aa79e089a43ace370a98b9a78537e99be8932248432db5f8
[I 19:17:00.712 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 19:17:01.009 NotebookApp]

To access the notebook, open this file in a browser:
file:///C:/Users/Cesar%20Hooper/AppData/Roaming/jupyter/runtime/nbserver-3800-open.html
Or copy and paste one of these URLs:
http://localhost:8888/?token=aa79e089a43ace370a98b9a78537e99be8932248432db5f8
or http://127.0.0.1:8888/?token=aa79e089a43ace370a98b9a78537e99be8932248432db5f8
```

Este prompt debe permanecer abierto mientras estés trabajando en tus proyectos Jupyter Notebook

### Paso 3

Una vez cargado, se abrirá automáticamente en alguno de los navegadores el NAVEGADOR JUPYTER NOTEBOOK (ésto puede tardar un poco). Aparecerá lo siguiente, el **homepage**.



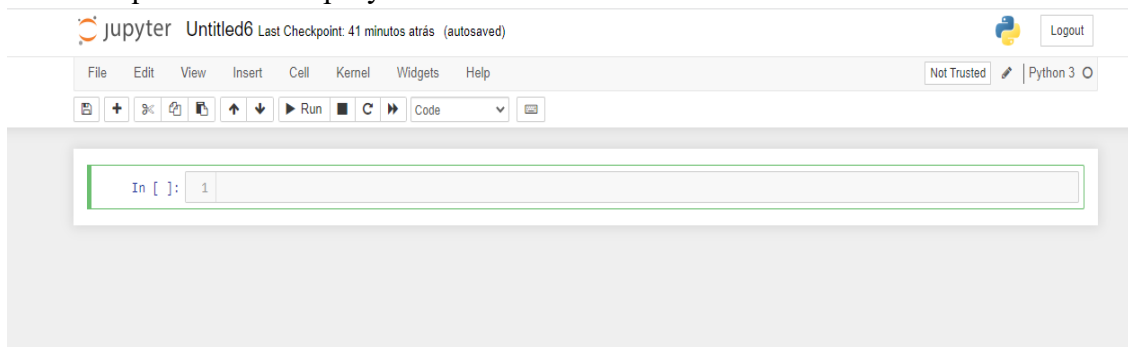
**Obs 1:** Si quieres cambiar de navegador sólo copias la dirección (en la barra de direcciones marcado con círculo amarillo) y lo pegas en el navegador que desees. **Sólo si deseas cambiar de navegador.**

**OBS 2:** En caso que no se despliegue el navegador jupyter notebook en el nuevo navegador, y solicite un token, debes copiar el token que está en el prompt y lo pegas donde se solicita. (donde dice `localhost:8888/?token=` "el código que está aquí es el que debes copiar y pegar")

Una vez desplegado el navegador Jupyter Notebook presionas donde marcan las flechas 1 y 2. Se desplegará la siguiente ventana

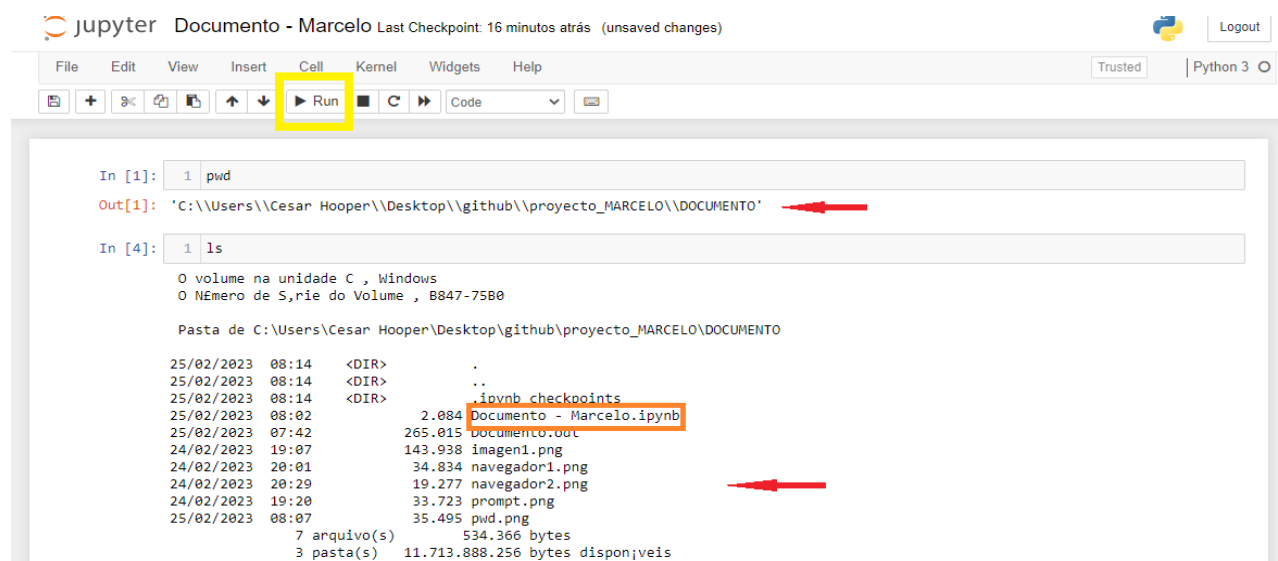
### Paso 4

Esta es la ventana con la que se trabaja y se programa en JN. Donde dice Untitled6 puedes marcar y poner el nombre que desees a tu proyecto.



- Los cambios que realices se guardan automáticamente cada cierto tiempo; sin embargo, te recomiendo guardar manualmente, antes de cerrar la sesión.
- No olvides que el **prompt debe permanecer abierto** mientras trabajes en tu JN, y cerrarlo cuando quieras finalizar la sesión.
- Puedes crear una carpeta de proyectos JN, entrar en ella a través del HomePage y luego abrir el navegador para programar y trabajar.

Para saber en qué carpeta estás trabajando puedes escribir el comando **pwd** en la barra de trabajo y presionar **Run** (recuadro amarillo)



```
jupyter Documento - Marcelo Last Checkpoint: 16 minutos atrás (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
+ -> Run - C Code
In [1]: 1 pwd
Out[1]: 'C:\\Users\\Cesar Hooper\\Desktop\\github\\proyecto_MARCELO\\DOCUMENTO'
In [4]: 1 ls
O volume na unidade C , Windows
O Número de S,rie do Volume , B847-75B0
Pasta de C:\\Users\\Cesar Hooper\\Desktop\\github\\proyecto_MARCELO\\DOCUMENTO
25/02/2023 08:14 <DIR> ..
25/02/2023 08:14 <DIR> ..
25/02/2023 08:14 <DIR> .ipynb_checkpoints
25/02/2023 08:02 2.084 Documento - Marcelo.ipynb
25/02/2023 07:42 265.015 Documento.out
24/02/2023 19:07 143.938 imagen1.png
24/02/2023 20:01 34.834 navegador1.png
24/02/2023 20:29 19.277 navegador2.png
24/02/2023 19:20 33.723 prompt.png
25/02/2023 08:07 35.495 pwd.png
7 arquivo(s) 534.366 bytes
3 pasta(s) 11.713.888.256 bytes disponíveis
```

Se despliega la dirección del directorio actual de trabajo. En el ejemplo, estoy en un directorio llamado DOCUMENTO, que a su vez está dentro de otro directorio llamado proyecto\_MARCELO, etcétera. Es importante saber en qué directorio estás trabajando para cargar datos que estén en otros directorios y poder trabajar con ellos.

Con el comando **ls** puedes ver qué documentos y directorios hay dentro de tu directorio de trabajo. No olvides presionar **run**.

Si observas la figura, en un recuadro anaranjado, está el nombre del Jupyter Notebook que creé para ejemplificar este proceso. Se llama **Documento – Marcelo.ipynb**. Los JN se guardan con la extensión ipynb.

Es importante que tu proyecto tenga nombre para que luego lo puedas identificar.

Cuando inicies sesión y quieras recuperar tu trabajo o proyecto, debes realizar los pasos 1 al 3, luego buscas tu directorio de trabajo en el homepage, vas hasta tu proyecto y lo abres.

Si por casualidad cierras el prompt, se produce algún error o sencillamente se bloquea, deberás iniciar sesión nuevamente con los pasos 1 al 3.

## Procedimiento para cargar el modelo y los datos en el jupyter Notebook

**Paso 0:** Descarga el modelo en la carpeta donde tienes guardado tu proyecto JN. El modelo pesa aproximadamente 22Mb y está guardado en formato pkl. El modelo se llama **modelo\_enero2022.pkl**. Este es un formato escrito en lenguaje codificado de máquina, no es un lenguaje de programación ni menos lenguaje humano. Dejo como ejemplo una pequeña sección del modelo codificado. Python puede usar este modelo codificado y realizar cálculos.

[illegible]

Vamos al proceso de carga.

**Paso 1:** importar (cargar) la librería joblib. Para ello, en la línea de trabajo del Jupyter Notebook escribimos el siguiente comando **import joblib**. Esta librería nos permite cargar el modelo en el jupyter notebook.

**Paso 2:** con el comando **joblib.load** cargar el modelo. Escribe en la siguiente línea de trabajo el este comando. **Incluyendo comillas**

```
modelo = joblib.load('modelo_enero2022.pkl')
```

No olvides presionar **run**.

En la siguiente imagen se ejemplifica el procedimiento señalados en los pasos 1 y 2. (Agrega los otros imports **pandas**, **numpy** y **matplotlib.pyplot** tal como aparecen ahí)

## Cargando el Modelo

```
In [9]: 1 # importando joblib.  
2 import joblib  
3 import pandas as pd  
4 import numpy as np  
5 import matplotlib.pyplot as plt
```

```
In [4]: 1 # Cargando el modelo  
2 modelo = joblib.load('modelo_enero2022.pkl')
```

Y éso sería todo. El modelo está listo para ponerse a trabajar.

Sin embargo, el modelo necesita que los datos estén formateados de una manera en particular. Este modelo fue entrenado con una matriz de datos de 14 columnas, con valores que tenían ciertos rangos.

- a. El modelo no va a funcionar si la matriz de datos con los que se trabaje **no tiene 14 columnas** respetando el orden de cada variable.
- b. Este programador no se responsabiliza de los resultados si los rangos de valores de los datos son diferentes de aquellos con los que fue entrenado el modelo.

**Paso 3:**

Diríjete a tu administrador de archivos e ingresa a la carpeta donde están guardados tus datos. Copia la dirección que aparece en la barra de dirección. Si usas windows es probable que la dirección aparezca así:

```
C:\Users\Cesar Hooper\...archivo1\proyecto_MARCELO
```

pega esa dirección en una barra de trabajo del JN, de este modo, incluyendo las comillas

```
path = 'C:\Users\Cesar Hooper\...archivo1\proyecto_MARCELO'
```

**Antes de presionar run** debes cambiar los backslash por slash.

```
path = 'C:/Users/Cesar Hooper/...archivo1/proyecto_MARCELO/'
```

Presiona run. Ese es el camino donde están tus archivos con los datos.

**Paso 4:** Para cargar cualquier archivo **xlsx** que se encuentre en esa dirección debes escribir el siguiente comando:

```
datos = pd.load_excel(path + 'nombre_archivo.xlsx')
```

No olvidar las comillas y presionar run.

**Oberservación:** limpieza de los datos.

Como se mencionó al finalizar el paso 2, los datos deben estar formateados respetando el formato con el que se entrenó el modelo. En este caso, 14 columnas con valores en ciertos rangos. Las columnas deben estar ordenadas respetando el orden de los datos de entrenamiento del modelo. Los datos no pueden contener valores nulos (estar vacíos) ni valores que no sean numéricos (letras, símbolos, palabras). Los datos deben ser sólo numéricos.

Para evitar errores, vamos a revisar brevemente un procedimiento de limpieza y formatación de los datos. Este procedimiento deberá realizarse previamente y cada vez que se desee aplicar el modelo a un nuevo conjunto de datos.

**Paso 5:** una vez cargado lo datos, ejecutamos el comando

```
datos.head(10)
```

este comando nos muestra las 10 primeras filas de datos.

En la siguiente figura se muestra la aplicación del comando y el resultado.

Desde la fila 0 hasta la fila 5, es decir, las 6 primeras filas, parecen valores que no son numéricos. (NaN significa not a number).

Por otra parte, en las primeras 2 columnas están la fecha y la hora. Debemos limpiar los valores NaN, además de sacar fecha y hora.

```
In [20]: 1 datos.head(10)
```

```
Out[20]:
```

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10	Unnamed: 11	Unnamed: 12	Unnamed: 13
0	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaT	hora	LSOI 5min	LSOI S 30	LSOI 4	LSOI D	5 min	30 min	4 hr	Day	CCI 5	CCI 30	PRECIO	CC
6	2022-01-03	10:20	5	9	11	16	19	21	25	28	31	33	4768.75	-
7	2022-01-03	10:21	3	9	11	16	19	21	25	28	31	33	4766	-
8	2022-01-03	10:22	3	9	11	16	19	21	25	28	31	33	4766.75	-
9	2022-01-03	10:23	3	9	11	16	19	21	25	28	31	33	4768.75	-

**Paso 6:** los siguientes comandos eliminan los NaN, y retiran fecha y hora.

**Obs:** los datos que estoy usando de ejemplo venían con la clasificación (target) que eliminé. Tú no tendrías que incluir esa línea en los comandos, pues tus datos no deberían estar clasificados.

```
In [62]: 1 # comandos para quitar fecha, hora y eliminar NaN
2 dataset = datos.dropna().copy()
3 fecha = dataset['Unnamed: 0']
4 hora = dataset['Unnamed: 1']
5 target = dataset['Unnamed: 16'] # cargué datos con clasificación
6 dataset = dataset.drop(['Unnamed: 0', 'Unnamed: 1', 'Unnamed: 16'], axis=1)
7 |
```

```
Out[62]:
```

	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10	Unnamed: 11	Unnamed: 12	Unnamed: 13	Unnamed: 14
7	3	9	11	16	19	21	25	28	31	33	4766	-43	-164
8	3	9	11	16	19	21	25	28	31	33	4766.75	-44	-161
9	3	9	11	16	19	21	25	28	31	33	4768.75	-38	-155
10	3	9	11	16	18	21	25	27	31	33	4771.5	-30	-145
11	2	9	11	16	18	21	25	27	30	33	4775.25	14	-126

```
In [67]: 1 # con el comando shape comprobamos que los datos tienen el formato de 14 columnas y la misma cantidad de filas
2 dataset.shape, target.shape, fecha.shape, hora.shape
```

```
Out[67]: ((9931, 14), (9931,), (9931,), (9931,))
```

Ahora que los datos están formateados vamos a realizar el procedimiento de normalización. Este procedimiento se realiza debido a que la varianza de los lados es grande, si los comparamos entre columnas, es grande. Tenemos datos que varían entre 1 y 10 por un lado, datos que varían en el orden de los 4700, y valores negativos entre -10 y -160. El modelo fue entrenado con valores normalizados. En qué consiste esta normalización? Sencillamente: se toman los datos de cada columna, se promedian y calcula la desviación estándar y luego, a cada dato de la columna se resta el promedio y divide por la desviación estándar. Este procedimiento se realiza con cada columna de datos.

Para ésto usamos una librería de python.

Sigue las instrucciones que se muestran en la figura a continuación:

```

In [37]: 1 # randomizamos los datos
          2 dataset_randomized = dataset.sample(frac=1)
          3 dataset_randomized.head()

Out[37]:
```

	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10	Unnamed: 11	Unnamed: 12	Unnamed: 13	Unnamed: 14
185	5	8	13	16	18	21	25	28	30	33	4768.5	136	-75
3183	5	7	10	17	19	22	25	28	31	33	4712.5	-145	-33
3923	2	9	13	15	19	22	26	28	30	33	4633.25	73	-36
7224	5	9	11	17	19	22	25	28	30	33	4336.75	36	-88
128	2	6	13	16	18	21	25	28	30	33	4767.25	57	-114

```

In [41]: 1 # importando librería para normalización de datos
          2 from sklearn.preprocessing import StandardScaler
          3 scaler = StandardScaler()
          4
          5 # (fiteamos) los datos randomizados
          6 scaler.fit(dataset_randomized)

Out[41]: StandardScaler()

In [42]: 1 # normalización de los datos no randomizados
          2 dataset_normalized = scaler.transform(dataset)
          3 dataset_normalized

```

## paso 7:

- primero barajamos un poco los datos para el pre-procesamiento de normalización
- luego importamos la librería StandardScaler
- fiteamos los datos randomizados y creamos la función de normalización **scaler**
- aplicamos la función de normalización sobre los datos originales (sin randomizar) dataset.

Ahora los datos están listos para que el modelo pueda clasificarlos.

```

3 scaler = StandardScaler()
4
5 # (fiteamos) los datos randomizados
6 scaler.fit(dataset_randomized)

Out[41]: StandardScaler()

In [42]: 1 # normalización de los datos no randomizados
          2 dataset_normalized = scaler.transform(dataset)
          3 dataset_normalized

Out[42]: array([[ -0.28470634,  1.29439514, -0.18305857, ..., -0.31207686,
                  -1.00349502,  1.40903243],
                [ -0.28470634,  1.29439514, -0.18305857, ..., -0.32088118,
                  -0.98054784,  1.41370375],
                [ -0.28470634,  1.29439514, -0.18305857, ..., -0.26805526,
                  -0.93465347,  1.42649062],
                ...,
                [ -0.28470634, -0.34910523, -1.0711654 , ...,  0.00487868,
                  0.92406833, -0.34154848],
                [ -0.28470634, -0.34910523, -1.0711654 , ..., -0.00392564,
                  0.91641927, -0.34313559],
                [ -0.28470634, -0.34910523, -1.0711654 , ...,  0.00487868,
                  0.92406833, -0.33996137]])

```

In [ ]: 1

Sólo falta el procedimiento para aplicar el modelo sobre los datos. En ésto nos enfocaremos en los siguientes pasos.



## Paso 8: Aplicación de Modelo sobre los datos.

Para ello simplemente debes escribir los comandos que aparecen a continuación

```
In [71]: 1 # aplicación modelo sobre los datos limpios y normalizados.
2 # recordar que el modelo se cargó con el nombre de modelo
3
4 # este comando entrega el resultado de la clasificación -1, 0 ó 1
5 y_pred = modelo.predict(dataset_normalized)
6
7 # este comando entrega la probabilidad de cada resultado
8 y_proba = modelo.predict_proba(dataset_normalized)
```

```
In [75]: 1 # verificamos dimensiones de los resultados
2 # lo importante es que tengan misma cantidad de filas.
3 y_pred.shape, y_proba.shape
```

```
Out[75]: ((9931,), (9931, 3))
```

```
In [76]: 1 # estos son los resultados de la clasificación
2 y_pred
```

```
Out[76]: array([0, 0, 0, ..., 1, 1, 1], dtype=int64)
```

```
In [48]: 1 # resultado de las probabilidades
2 y_proba
```

```
Out[48]: array([[0.1 , 0.89 , 0.01 ],
 [0.16 , 0.83 , 0.01 ],
 [0.3 , 0.65 , 0.05 ],
 ...,
 [0.025, 0.005, 0.97 ],
 [0.07 , 0.01 , 0.92 ],
 [0.025, 0.005, 0.97 ]])
```

- `y_pred = modelo.predict(datos_normalizados)` : este comando entrega una columna de 9931 datos, con la clasificación (-1, 0 ó 1).
- `y_proba = modelo.predict_proba(datos_normalizados)` : este comando entrega un arreglo con las probabilidades de cada resultado.

**Paso 9:** para que puedas analizar los resultados con mayor facilidad, vamos a construir una tabla con los resultados, incluyendo fecha y hora.

```
In [77]: 1 # resultados
2 resultados = pd.DataFrame()
3 resultados['fecha'] = fecha
4 resultados['hora'] = hora
5 resultados['Prediccion'] = y_pred
6 resultados['%pred: -1'] = y_proba[:,0]
7 resultados['%pred: 0'] = y_proba[:,1]
8 resultados['%pred: 1'] = y_proba[:,2]
9
10 resultados.head(10)
```

```
Out[77]:
```

	fecha	hora	Prediccion	%pred: -1	%pred: 0	%pred: 1
7	2022-01-03	10:21	0	0.100	0.890	0.010
8	2022-01-03	10:22	0	0.160	0.830	0.010
9	2022-01-03	10:23	0	0.300	0.650	0.050
10	2022-01-03	10:24	0	0.185	0.795	0.020
11	2022-01-03	10:25	0	0.065	0.855	0.080
12	2022-01-03	10:26	0	0.115	0.860	0.025
13	2022-01-03	10:27	0	0.060	0.885	0.055
14	2022-01-03	10:28	0	0.060	0.880	0.060
15	2022-01-03	10:29	0	0.065	0.810	0.125
16	2022-01-03	10:30	0	0.035	0.915	0.050

**Paso 10:** finalmente, para que puedas trabajar con los resultados en formato de excel, escribes este comando para guardar la tabla de resultados en formato `xlsx`.

14	2022-01-03	10:28	0	0.060	0.880	0.060
15	2022-01-03	10:29	0	0.065	0.810	0.125
16	2022-01-03	10:30	0	0.035	0.915	0.050

```
In [79]: 1 resultados.to_excel('RESULTADOS_ENERO2022.xlsx', index=False)
```

**fin**

Esta tabla de resultados se guarda en la dirección actual en que estás trabajando. Si quieres guardarla en otro archivo debes agregar un path con la dirección del otro archivo.

Ponle un nombre que puedas reconocer fácilmente a la tabla con los resultados.

Así, puedes abrir la tabla de resultados en excel y analizarlos como gustes.

### Comentarios finales:

Creo que me alargué demasiado, a pesar de que traté de ser sucinto en las instrucciones. Disculpa si algunos pasajes parecen *bobamente* pedagógicos, pero ésto se debe a dos motivos. Primero que, trabajar con datos es un tema delicado; escribir mal una instrucción puede causar errores que, si no sabes interpretar el origen del error, causa estancamiento y frustración. Créeme, mucha frustración. En segundo lugar, fui profesor por muchos años, así que debe ser un defecto profesional enseñar cosas con detallado procedimiento.

De cualquier modo, estoy seguro de que lamentablemente se me escaparon detalles. Algunos de ellos tendrás que resolverlos con intuición y otros puedes consultar libremente.

Espero que este documento sea de utilidad y te permita trabajar de manera simple.

FIN.