1-intro

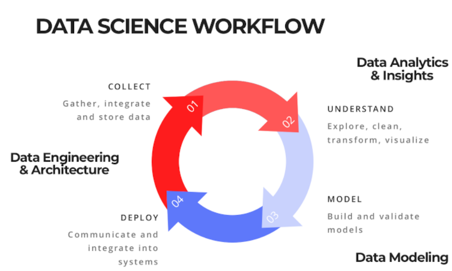
April 27, 2023

Estadística y computación para metagenómica

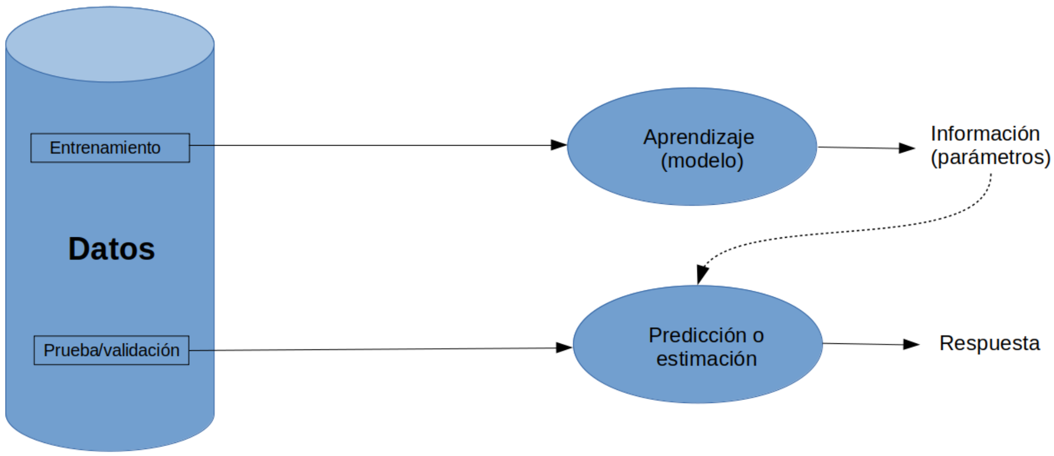
Víctor Muñiz Sánchez

Junio 2023

1 Introducción



Un esquema de aprendizaje máquina.

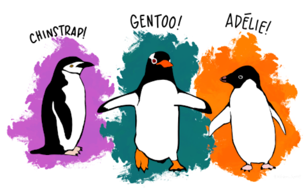


Sin embargo, hay un elemento muy importante que falta en ésa ilustración…

Veámoslo con algunos ejemplos.

1.1 Representación de datos.

1.1.1 Ejemplo 1. Palmer penguins



Artwork by @allison_horst

¿Cómo se obtuvieron los datos?

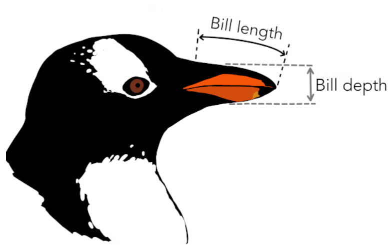
Github

+Info

[45]:
```
import pandas as pd
import seaborn as sns
import numpy as np
%matplotlib inline

from palmerpenguins import load_penguins
penguins = load_penguins()
penguins.dropna(inplace=True)
penguins.drop(['year'],axis=1,inplace=True)
penguins
```

[45]:
```
       species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
0       Adelie  Torgersen            39.1           18.7              181.0
1       Adelie  Torgersen            39.5           17.4              186.0
2       Adelie  Torgersen            40.3           18.0              195.0
4       Adelie  Torgersen            36.7           19.3              193.0
5       Adelie  Torgersen            39.3           20.6              190.0
..         …          …             …              …                 …
339  Chinstrap      Dream            55.8           19.8              207.0
340  Chinstrap      Dream            43.5           18.1              202.0
341  Chinstrap      Dream            49.6           18.2              193.0
342  Chinstrap      Dream            50.8           19.0              210.0
343  Chinstrap      Dream            50.2           18.7              198.0

     body_mass_g     sex
0         3750.0    male
1         3800.0  female
2         3250.0  female
4         3450.0  female
5         3650.0    male
..           …       …
339       4000.0    male
340       3400.0  female
341       3775.0    male
342       4100.0    male
343       3775.0  female

[333 rows x 7 columns]
```

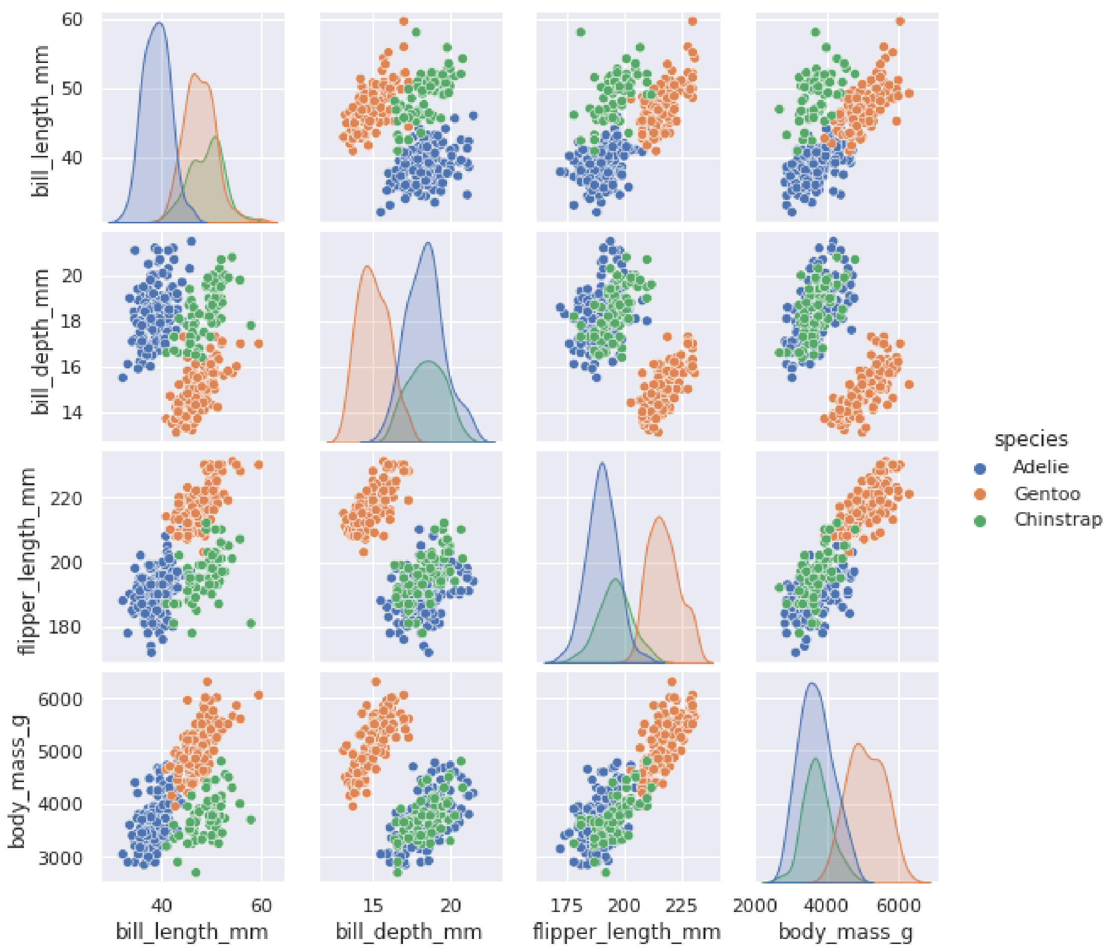
[ ]:
```
## penguins.to_csv('palmer_penguins.csv',index=False)
```



Demos una mirada a los datos. ¿Qué te gustaría 'ver'?

Aprendizaje No Supervisado: detectar patrones en los datos.

[46]:
```
sns.set()
sns.pairplot(penguins, hue='species', height=2);
```



Para fines didácticos, simplificaremos el problema.

- Unimos las categorías *'Adelie'* y *'Chinstrap'* en la clase *'adel_chins'*
- Además, usaremos datos estandarizados (más adelante, retomaremos éste concepto)

[47]:
```
spec2 = penguins['species'].copy()
spec2[spec2.eq('Adelie') | spec2.eq('Chinstrap')] = 'adel_chins'
penguins['species'] = spec2
penguins
```

[47]:
```
        species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
0    adel_chins  Torgersen            39.1           18.7              181.0
1    adel_chins  Torgersen            39.5           17.4              186.0
2    adel_chins  Torgersen            40.3           18.0              195.0
4    adel_chins  Torgersen            36.7           19.3              193.0
5    adel_chins  Torgersen            39.3           20.6              190.0
..          …          …             …              …                  …
339  adel_chins      Dream            55.8           19.8              207.0
```

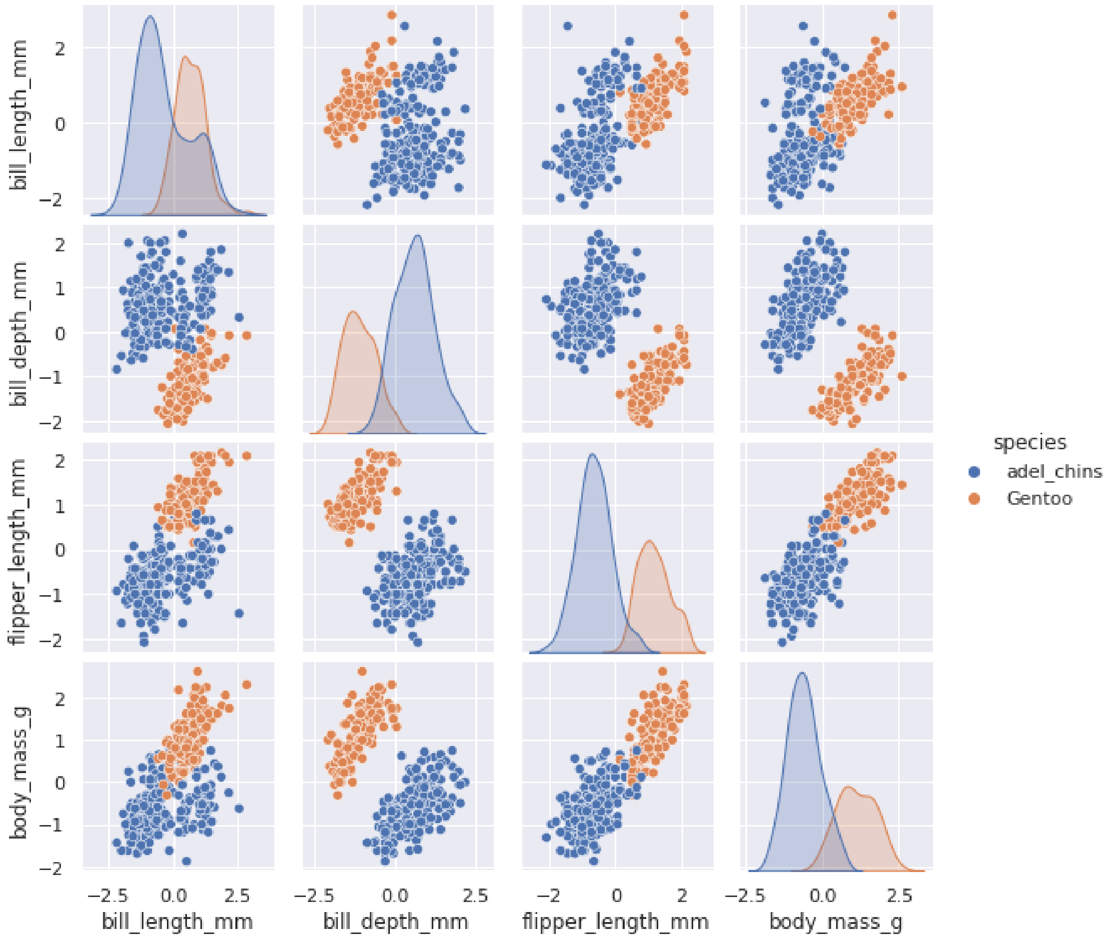
```
340  adel_chins      Dream            43.5          18.1               202.0
341  adel_chins      Dream            49.6          18.2               193.0
342  adel_chins      Dream            50.8          19.0               210.0
343  adel_chins      Dream            50.2          18.7               198.0

     body_mass_g     sex
0         3750.0    male
1         3800.0  female
2         3250.0  female
4         3450.0  female
5         3650.0    male
..           …      …
339       4000.0    male
340       3400.0  female
341       3775.0    male
342       4100.0    male
343       3775.0  female

[333 rows x 7 columns]
```

[48]:

```
from sklearn.preprocessing import StandardScaler

col_vars = ['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm',␣
 ↪'body_mass_g']
x_peng = penguins[col_vars]

ss = StandardScaler()
scaled_x = ss.fit_transform(x_peng)
scaled_x = pd.DataFrame(ss.fit_transform(x_peng), columns=col_vars, index =␣
 ↪penguins.index)
penguins_sc = pd.concat([scaled_x, penguins['species']], axis=1)
```

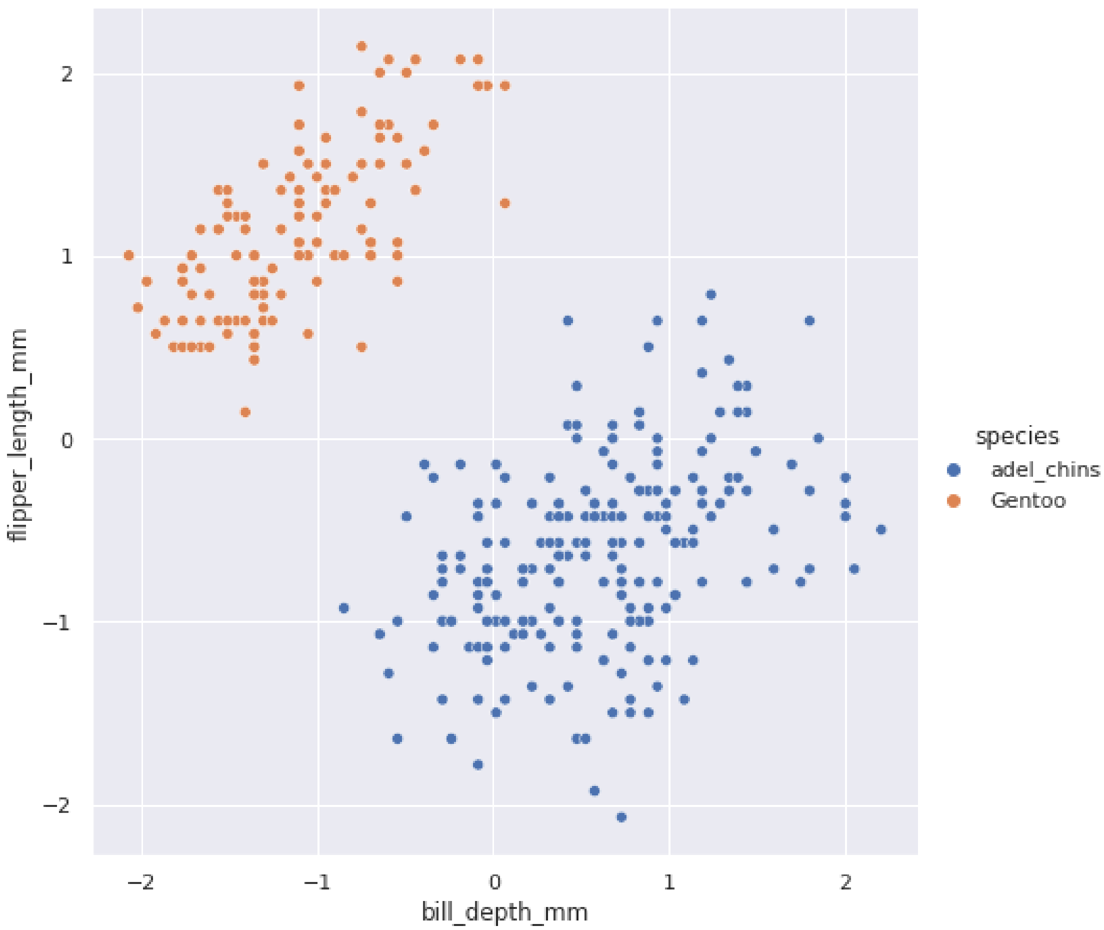
[49]:

```
sns.pairplot(penguins_sc, hue='species', height=2);
```



Nuevamente, tenemos un problema de aprendizaje no supervisado. Podemos identificar claramente al menos 2 grupos

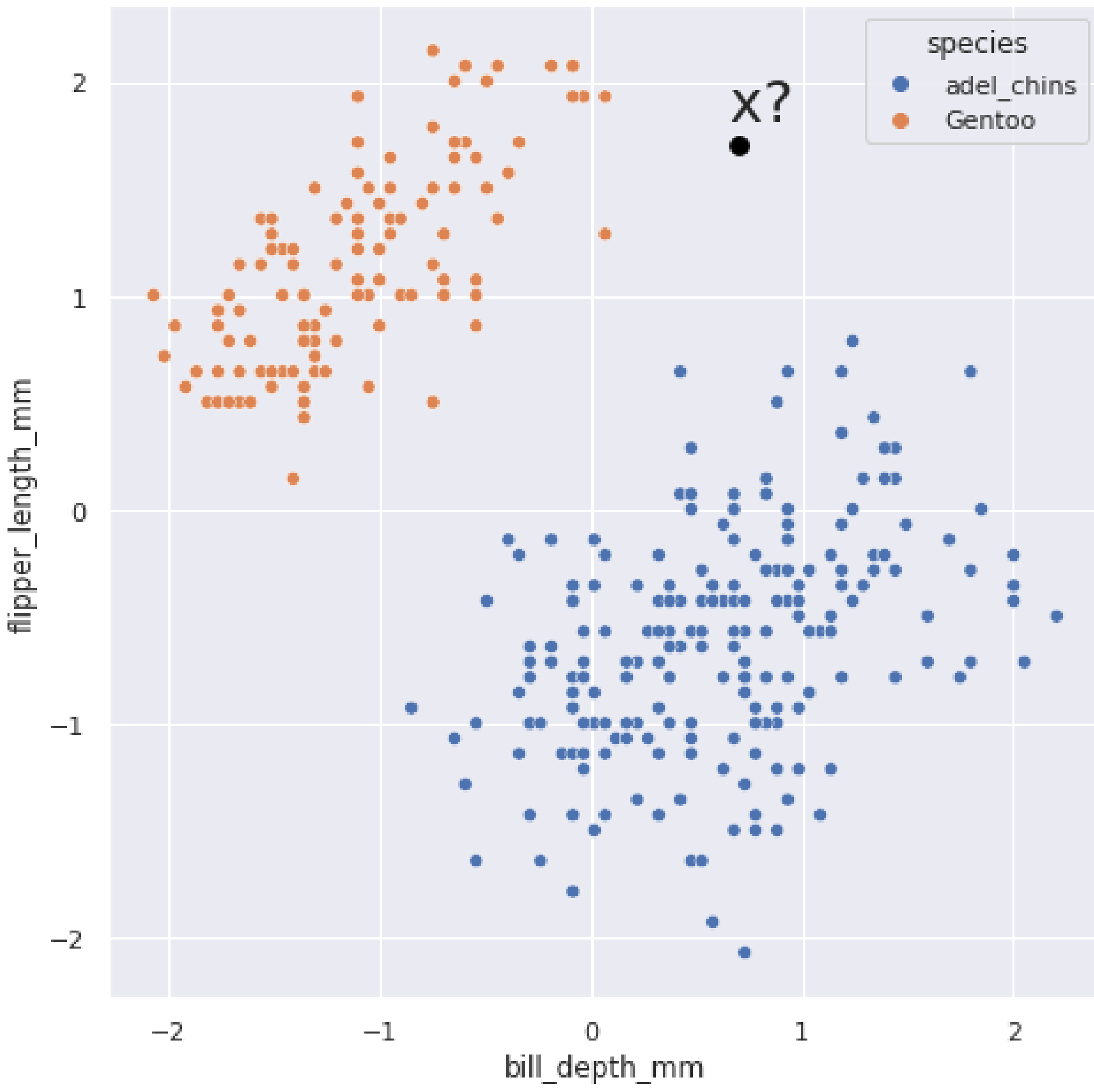
[50]:
```
sns.relplot(x='bill_depth_mm', y='flipper_length_mm', hue='species',␣
    ↪data=penguins_sc, height=7);
```



Aprendizaje Supervisado: aparece un dato nuevo y queremos saber a qué categoría pertenece

[51]:

```
import matplotlib.pyplot as plt

new_x = [.7, 1.7]
fig = plt.figure(figsize = (6,6))
ax=fig.add_axes([0,0,1,1])
sns.scatterplot(x='bill_depth_mm', y='flipper_length_mm', hue='species',␣
 ↪data=penguins_sc)
ax.scatter(x=new_x[0], y=new_x[1], color='black', s=60)
ax.text(new_x[0]-.13, new_x[1]+.12, ' x?', fontsize = 25);
```



¿Qué clase le asignarías? ¿Porqué?

Con lo que sabes o has aprendido hasta ahora, ¿se te ocurre algún método para estimar la clase del pinguino?

Tenemos entonces un conjunto de entrenamiento:

$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n); \quad \mathbf{x} \in \mathbb{R}^2, y \in \{-1, 1\}$$

Queremos una función

$$f : \mathbb{R}^2 \mapsto \{-1, 1\}$$

que nos responda la pregunta: ¿$\mathbf{x}_{new}$ es similar a Gentoo (-1) o a Adelie-Chinstrap (+1)?

Una solución basado en la distancia como medida de similaridad.

Usaremos un criterio basado en el vecino más cercano, pero en este caso, vamos a "*resumir*" los puntos de ambas categorías en solo 2 vecinos:

$\mathbf{c}_+ = \frac{1}{n_+} \sum_{i|y_i=1} \mathbf{x}_i$

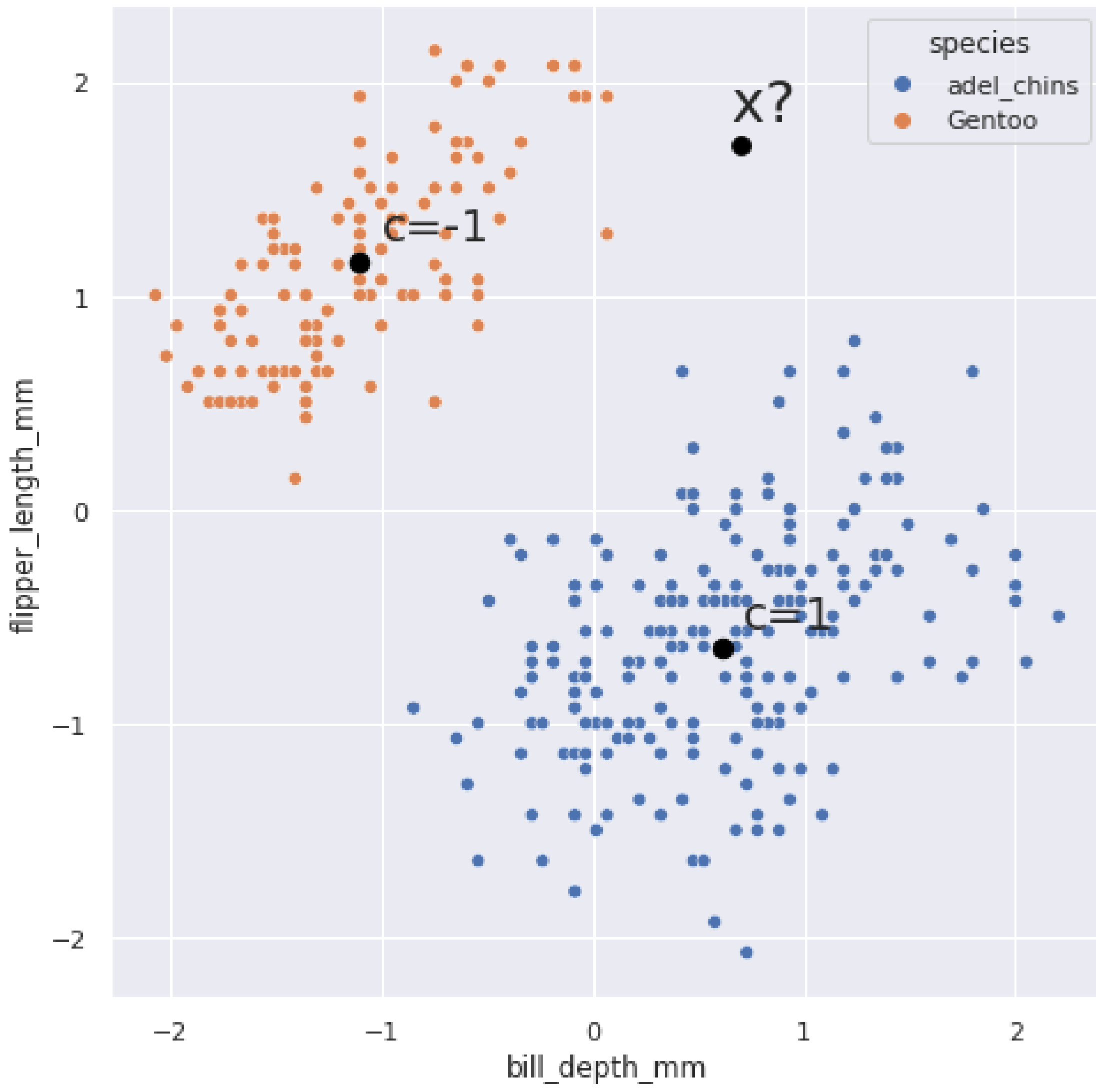
y

$\mathbf{c}_- = \frac{1}{n_-} \sum_{i|y_i=-1} \mathbf{x}_i$

[52]:
```
x_gent = np.array(penguins_sc.
 ↪loc[penguins_sc['species']=='Gentoo',['bill_depth_mm', 'flipper_length_mm']])
x_adel_chins = np.array(penguins_sc.loc[penguins_sc['species']!
 ↪='Gentoo',['bill_depth_mm', 'flipper_length_mm']])
x_means = np.vstack((x_gent.mean(axis=0),x_adel_chins.mean(axis=0)))
all_mean = x_means.mean(axis=0)
```

[53]:
```
ax.scatter(x_means[:,0], x_means[:,1], color='black', s=70)
for i in range(0,x_means.shape[0]):
    ax.text(x_means[i,0]+.1, x_means[i,1]+.1, 'c='+str(round(2*i-1)), fontsize␣
 ↪= 20)
fig
```

[53]:



Entonces, nos interesa saber si

¿$\|\mathbf{c}_{-} - \mathbf{x}\|^2 > \|\mathbf{c}_{+} - \mathbf{x}\|^2$?

equivalente a

¿$\|\mathbf{c}_{-}\|^2 - \|\mathbf{c}_{+}\|^2 + 2\langle \mathbf{x}, \mathbf{c}_{+}\rangle - 2\langle \mathbf{x}, \mathbf{c}_{-}\rangle > 0$?

Construimos un clasificador muy sencillo, para esto, define

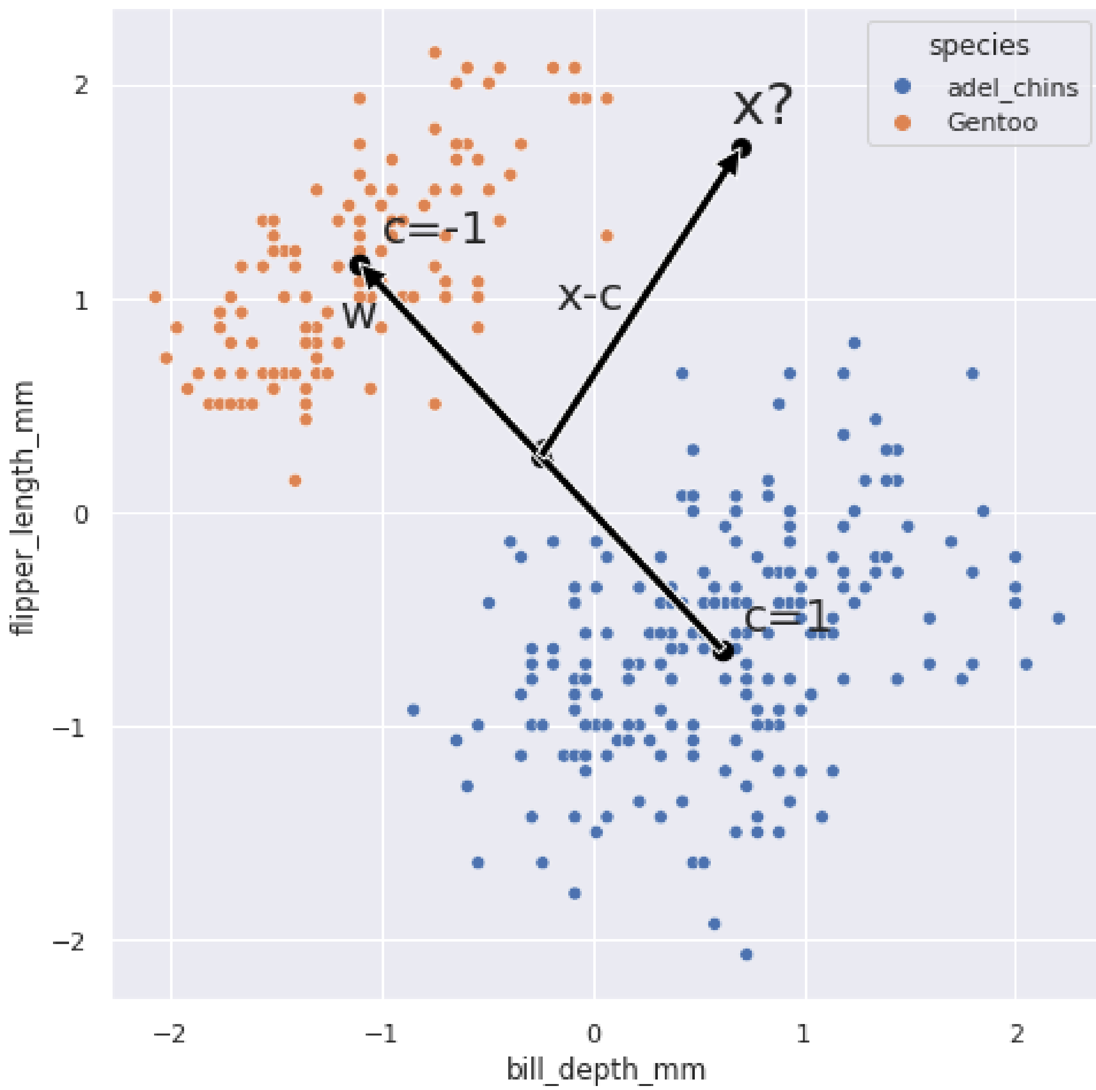
$\mathbf{c} = (\mathbf{c}_{+} - \mathbf{c}_{-})/2,$

$\mathbf{w} = \mathbf{c}_{+} - \mathbf{c}_{-}$

[54]:
```
w = x_adel_chins.mean(axis=0)-x_gent.mean(axis=0)
ax.scatter(all_mean[0], all_mean[1], color='black', s=60)
ax.text(all_mean[0]-.05, all_mean[1]-.02, 'c', fontsize = 20)
```

```python
ax.annotate('',xy = (x_means[0,0], x_means[0,1]), xytext = (x_means[1,0],␣
↪x_means[1,1]),
               arrowprops=dict(facecolor='black'))
ax.text(x_means[0,0]-.1, x_means[0,1]-.3, 'w', fontsize = 20)
ax.annotate('',xy = (new_x[0], new_x[1]), xytext = (all_mean[0], all_mean[1]),
               arrowprops=dict(facecolor='black'))
ax.text(all_mean[0]+.07, all_mean[1]+.7, 'x-c', fontsize = 20)
fig
```

[54]:



Lo anterior, induce un modelo lineal de clasificación:

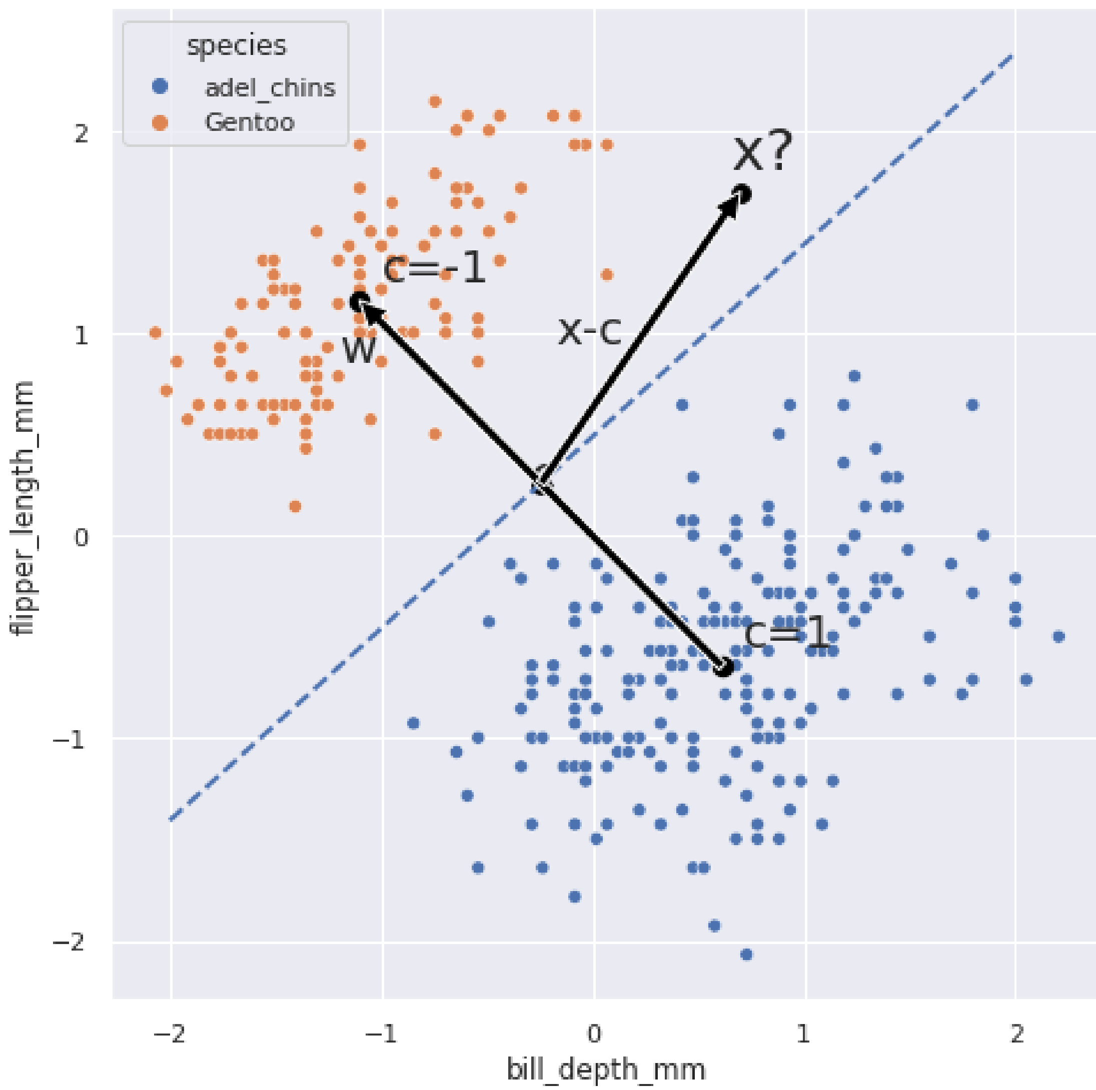
$w_0 + \mathbf{w}'\mathbf{x}$.

Entonces, podemos clasificar mediante

$y = \text{sign}\langle(\mathbf{x} - \mathbf{c}), \mathbf{w}\rangle$

[55]:
```
cg = x_gent.mean(axis=0)
cac = x_adel_chins.mean(axis=0)
w = cac-cg
w0 = (np.dot(cg,cg)-np.dot(cac,cac))*.5
xx1 = np.linspace(-2,2,100)
xx2 = -(w[0]*xx1+w0)/w[1]
ax.plot([xx1[0],xx1[99]],[xx2[0],xx2[99]],linestyle='--', linewidth=2)
fig
```

[55]:



Obviamente, ésta solución no es la mejor opción.

Por ejemplo, ¿qué pasa si tenemos outliers?

1.1.2 Ejemplo 2: OTUS

[56]:
```python
import contextily as ctx
import geopandas as gpd
from mpl_toolkits.axes_grid1 import make_axes_locatable
import pandas as pd
import seaborn as sns

otus = pd.read_csv("../data/OTUS_conservados.csv", index_col="X")
otus.head()
```

[56]:
```
                      New_Labels  Latitude  Longitude  X4457032  X4471583  \
X
1001.skm3  WarmT-SumDry-HotSum0    33.194   -117.241  0.003306       0.0
1001.skd3  WarmT-SumDry-HotSum0    33.194   -117.241  0.002699       0.0
1001.skm1  WarmT-SumDry-HotSum0    33.194   -117.241  0.001304       0.0
1001.skb3  WarmT-SumDry-HotSum0    33.194   -117.241  0.009130       0.0
1001.skm2  WarmT-SumDry-HotSum0    33.194   -117.241  0.001022       0.0

           X9560  X4468101  X198079   X101868  X4360511  …  X210657  \
X                                                        …
1001.skm3    0.0       0.0      0.0  0.005785   0.00000  …      0.0
1001.skd3    0.0       0.0      0.0  0.001080   0.00018  …      0.0
1001.skm1    0.0       0.0      0.0  0.002934   0.00000  …      0.0
1001.skb3    0.0       0.0      0.0  0.004966   0.00000  …      0.0
1001.skm2    0.0       0.0      0.0  0.000341   0.00000  …      0.0

           X218246    X48487    X81081  X1787355  X6159  X154268   X855996  \
X
1001.skm3  0.00000  0.004752  0.007231  0.000000    0.0      0.0  0.002273
1001.skd3  0.00000  0.003419  0.029872  0.000180    0.0      0.0  0.000720
1001.skm1  0.00000  0.004563  0.004563  0.000326    0.0      0.0  0.001304
1001.skb3  0.00016  0.004645  0.008169  0.000961    0.0      0.0  0.002243
1001.skm2  0.00000  0.005112  0.010907  0.000682    0.0      0.0  0.002045

            X99400  X716037
X
1001.skm3  0.00000  0.00000
1001.skd3  0.00000  0.00000
1001.skm1  0.00000  0.00000
1001.skb3  0.00016  0.00032
1001.skm2  0.00000  0.00000

[5 rows x 606 columns]
```

[57]:
```python
otus.shape
```

[57]: (3043, 606)

[58]:
```
otus_gdf = gpd.GeoDataFrame(otus.copy(), geometry=gpd.
 ↪points_from_xy(otus['Longitude'], otus['Latitude']), crs='EPSG:4326')

fig, ax = plt.subplots(1, figsize=(15, 15))
otus_con_gdf.plot(marker='.', c='black', ax=ax, markersize=70, zorder=1)
plt.title("Ubicación de las muestras",fontsize=30)
plt.axis("off")

ctx.add_basemap(ax, source=ctx.providers.CartoDB.Positron, crs='EPSG:4326')
```

[59]:
```
cont_names = list(otus.drop(["New_Labels","Latitude", "Longitude"], axis=1).
 ↪columns)
y_names = ["New_Labels"]
train_df = otus[cont_names+y_names]
train_df
```

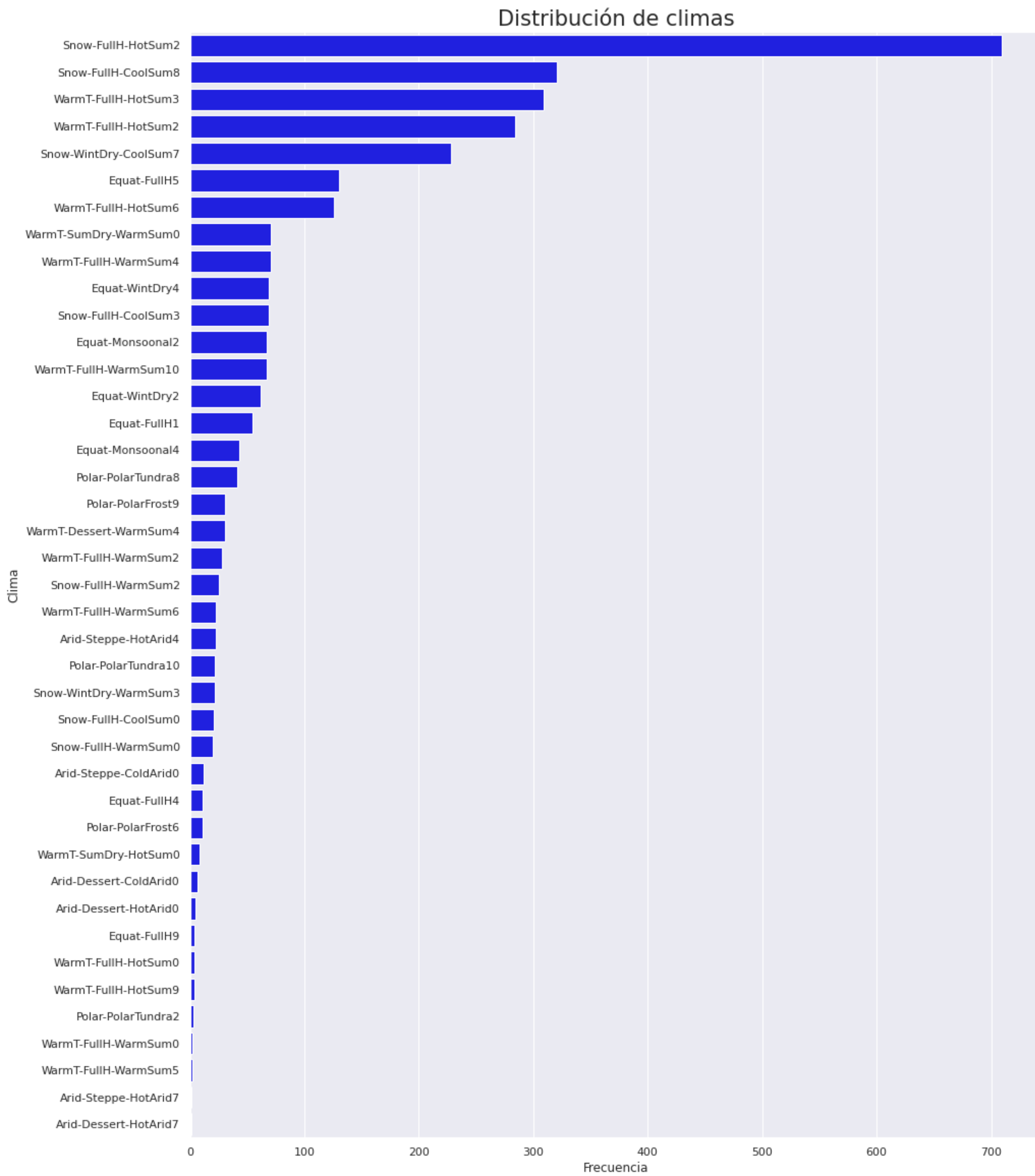
[59]:
```
                                                X4457032  X4471583  X9560  \
X
1001.skm3                                       0.003306  0.000000    0.0
1001.skd3                                       0.002699  0.000000    0.0
1001.skm1                                       0.001304  0.000000    0.0
1001.skb3                                       0.009130  0.000000    0.0
1001.skm2                                       0.001022  0.000000    0.0
…                                                    …         …      …
1883.2011.282.crump.artic.ltreb.main.lane4.noindex  0.000000  0.000000    0.0
1883.2011.329.crump.artic.ltreb.main.lane4.noindex  0.000110  0.000000    0.0
```

```
1883.2011.348.crump.artic.ltreb.main.lane4.noindex  0.000528  0.000024    0.0
1883.2011.3.crump.artic.ltreb.main.lane4.noindex    0.000000  0.000000    0.0
1883.2011.37.crump.artic.ltreb.main.lane4.noindex   0.000000  0.000000    0.0

                                                    X4468101  X198079  \
X
1001.skm3                                           0.000000      0.0
1001.skd3                                           0.000000      0.0
1001.skm1                                           0.000000      0.0
1001.skb3                                           0.000000      0.0
1001.skm2                                           0.000000      0.0
…                                                        …        …
1883.2011.282.crump.artic.ltreb.main.lane4.noindex  0.000000      0.0
1883.2011.329.crump.artic.ltreb.main.lane4.noindex  0.000028      0.0
1883.2011.348.crump.artic.ltreb.main.lane4.noindex  0.000000      0.0
1883.2011.3.crump.artic.ltreb.main.lane4.noindex    0.000000      0.0
1883.2011.37.crump.artic.ltreb.main.lane4.noindex   0.000000      0.0

                                                     X101868  X4360511  X9961  \
X
1001.skm3                                           0.005785  0.000000    0.0
1001.skd3                                           0.001080  0.000180    0.0
1001.skm1                                           0.002934  0.000000    0.0
1001.skb3                                           0.004966  0.000000    0.0
1001.skm2                                           0.000341  0.000000    0.0
…                                                        …         …    …
1883.2011.282.crump.artic.ltreb.main.lane4.noindex  0.000000  0.000028    0.0
1883.2011.329.crump.artic.ltreb.main.lane4.noindex  0.000000  0.000000    0.0
1883.2011.348.crump.artic.ltreb.main.lane4.noindex  0.000000  0.000000    0.0
1883.2011.3.crump.artic.ltreb.main.lane4.noindex    0.000023  0.000000    0.0
1883.2011.37.crump.artic.ltreb.main.lane4.noindex   0.000000  0.000261    0.0

                                                    X3944484  X1105039  …  \
X                                                                       …
1001.skm3                                           0.000000  0.000000  …
1001.skd3                                           0.000000  0.000000  …
1001.skm1                                           0.000000  0.000000  …
1001.skb3                                           0.000000  0.000000  …
1001.skm2                                           0.000000  0.000000  …
…                                                        …         …  …
1883.2011.282.crump.artic.ltreb.main.lane4.noindex  0.000028  0.000000  …
1883.2011.329.crump.artic.ltreb.main.lane4.noindex  0.000000  0.000000  …
1883.2011.348.crump.artic.ltreb.main.lane4.noindex  0.000000  0.000312  …
1883.2011.3.crump.artic.ltreb.main.lane4.noindex    0.000000  0.000162  …
1883.2011.37.crump.artic.ltreb.main.lane4.noindex   0.000725  0.000000  …

                                                    X218246    X48487  \
```

```
X
1001.skm3                                        0.00000  0.004752
1001.skd3                                        0.00000  0.003419
1001.skm1                                        0.00000  0.004563
1001.skb3                                        0.00016  0.004645
1001.skm2                                        0.00000  0.005112
…                                                  …        …
1883.2011.282.crump.artic.ltreb.main.lane4.noindex  0.00000  0.000000
1883.2011.329.crump.artic.ltreb.main.lane4.noindex  0.00000  0.000000
1883.2011.348.crump.artic.ltreb.main.lane4.noindex  0.00000  0.000000
1883.2011.3.crump.artic.ltreb.main.lane4.noindex    0.00000  0.000000
1883.2011.37.crump.artic.ltreb.main.lane4.noindex   0.00000  0.000000

                                                    X81081  X1787355  \
X
1001.skm3                                         0.007231  0.000000
1001.skd3                                         0.029872  0.000180
1001.skm1                                         0.004563  0.000326
1001.skb3                                         0.008169  0.000961
1001.skm2                                         0.010907  0.000682
…                                                     …        …
1883.2011.282.crump.artic.ltreb.main.lane4.noindex  0.000000  0.000000
1883.2011.329.crump.artic.ltreb.main.lane4.noindex  0.000000  0.000000
1883.2011.348.crump.artic.ltreb.main.lane4.noindex  0.000000  0.000000
1883.2011.3.crump.artic.ltreb.main.lane4.noindex    0.000000  0.000000
1883.2011.37.crump.artic.ltreb.main.lane4.noindex   0.000000  0.000000

                                                     X6159  X154268  \
X
1001.skm3                                         0.000000      0.0
1001.skd3                                         0.000000      0.0
1001.skm1                                         0.000000      0.0
1001.skb3                                         0.000000      0.0
1001.skm2                                         0.000000      0.0
…                                                     …      …
1883.2011.282.crump.artic.ltreb.main.lane4.noindex  0.006404      0.0
1883.2011.329.crump.artic.ltreb.main.lane4.noindex  0.000000      0.0
1883.2011.348.crump.artic.ltreb.main.lane4.noindex  0.000024      0.0
1883.2011.3.crump.artic.ltreb.main.lane4.noindex    0.000835      0.0
1883.2011.37.crump.artic.ltreb.main.lane4.noindex   0.000000      0.0

                                                   X855996    X99400  \
X
1001.skm3                                         0.002273  0.000000
1001.skd3                                         0.000720  0.000000
1001.skm1                                         0.001304  0.000000
1001.skb3                                         0.002243  0.000160
```

```
1001.skm2                                        0.002045  0.000000
…                                                     …         …
1883.2011.282.crump.artic.ltreb.main.lane4.noindex  0.000000  0.001601
1883.2011.329.crump.artic.ltreb.main.lane4.noindex  0.000000  0.000248
1883.2011.348.crump.artic.ltreb.main.lane4.noindex  0.000000  0.000744
1883.2011.3.crump.artic.ltreb.main.lane4.noindex    0.000000  0.000000
1883.2011.37.crump.artic.ltreb.main.lane4.noindex   0.000000  0.000000

                                                    X716037  \
X
1001.skm3                                           0.00000
1001.skd3                                           0.00000
1001.skm1                                           0.00000
1001.skb3                                           0.00032
1001.skm2                                           0.00000
…                                                     …
1883.2011.282.crump.artic.ltreb.main.lane4.noindex  0.00000
1883.2011.329.crump.artic.ltreb.main.lane4.noindex  0.00000
1883.2011.348.crump.artic.ltreb.main.lane4.noindex  0.00000
1883.2011.3.crump.artic.ltreb.main.lane4.noindex    0.00000
1883.2011.37.crump.artic.ltreb.main.lane4.noindex   0.00000

                                                              New_Labels
X
1001.skm3                                           WarmT-SumDry-HotSum0
1001.skd3                                           WarmT-SumDry-HotSum0
1001.skm1                                           WarmT-SumDry-HotSum0
1001.skb3                                           WarmT-SumDry-HotSum0
1001.skm2                                           WarmT-SumDry-HotSum0
…                                                                …
1883.2011.282.crump.artic.ltreb.main.lane4.noindex   Snow-FullH-CoolSum8
1883.2011.329.crump.artic.ltreb.main.lane4.noindex   Snow-FullH-CoolSum8
1883.2011.348.crump.artic.ltreb.main.lane4.noindex   Snow-FullH-CoolSum8
1883.2011.3.crump.artic.ltreb.main.lane4.noindex     Snow-FullH-CoolSum8
1883.2011.37.crump.artic.ltreb.main.lane4.noindex    Snow-FullH-CoolSum8

[3043 rows x 604 columns]
```

[60]:
```python
fig, ax = plt.subplots(1,1 ,figsize=(15, 20))
sns.countplot(data=train_df, y="New_Labels", order=train_df.New_Labels.
 ↪value_counts().index, color="blue")
plt.xlabel("Frecuencia")
plt.ylabel("Clima")
plt.title("Distribución de climas", fontsize=21)
plt.show()
```



Veamos otras representaciones de los datos, particularmente, reduciendo su dimensión de manera lineal y no lineal

[61]:
```
y = np.array(train_df[y_names]).ravel()
X = np.array(train_df.drop(y_names,axis=1)).astype('float')
```

Representación con PCA

[72]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import plotly.express as px

X_std = StandardScaler().fit_transform(X)
ncomp=3
otus_pca=PCA(ncomp)
otus_pca.fit_transform(X_std)
proj = pd.DataFrame(otus_pca.transform(X_std),columns = ['pc1','pc2','pc3'])
pca_proj = pd.DataFrame({'pc1': proj['pc1'], 'pc2': proj['pc2'], 'clima': y})

# Grafica interactiva
fig = px.scatter(pca_proj, x='pc1', y='pc2', hover_data=['clima'], color =␣
↪'clima')
fig.update_layout(
    autosize=False,
    width=800,
    height=800,
)
fig.show()
```

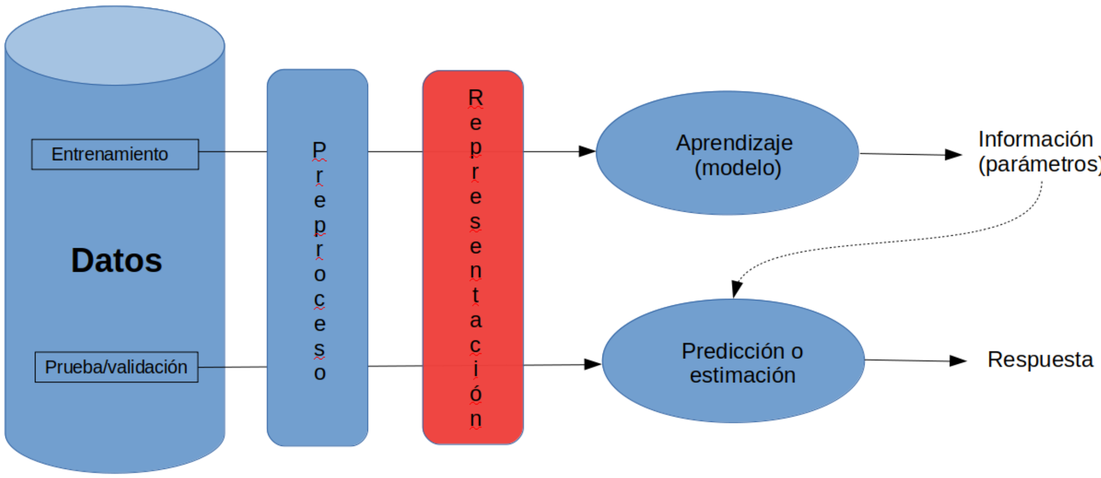
[70]:

```
from sklearn.manifold import TSNE

tsne = TSNE(n_components=2, perplexity=500)
#X_tsne = tsne.fit_transform(train_img)
#tsne_dataset = pd.DataFrame({'pc1': X_tsne[:, 0], 'pc2': X_tsne[:, 1], 'digit':
↪ y_train})
X_tsne = tsne.fit_transform(X_std)

tsne_dataset = pd.DataFrame({'pc1': X_tsne[:, 0], 'pc2': X_tsne[:, 1], 'clima':␣
↪y})
```

[73]:

```
# Grafica interactiva
fig = px.scatter(tsne_dataset, x='pc1', y='pc2', hover_data=['clima'], color =␣
↪'clima')
fig.update_layout(
    autosize=False,
    width=800,
    height=800,
)
fig.show()
```

Entonces, nuestro esquema de aprendizaje máquina revisitado es:


Entrenamiento
Datos
Prueba/validación
Preproceso
Representación
Aprendizaje
(modelo)
Información
(parámetros)
Predicción o
estimación
Respuesta