1-intro

January 21, 2021

#

Ciencia de Datos

Víctor Muñiz Sánchez

Maestría en Cómputo Estadístico

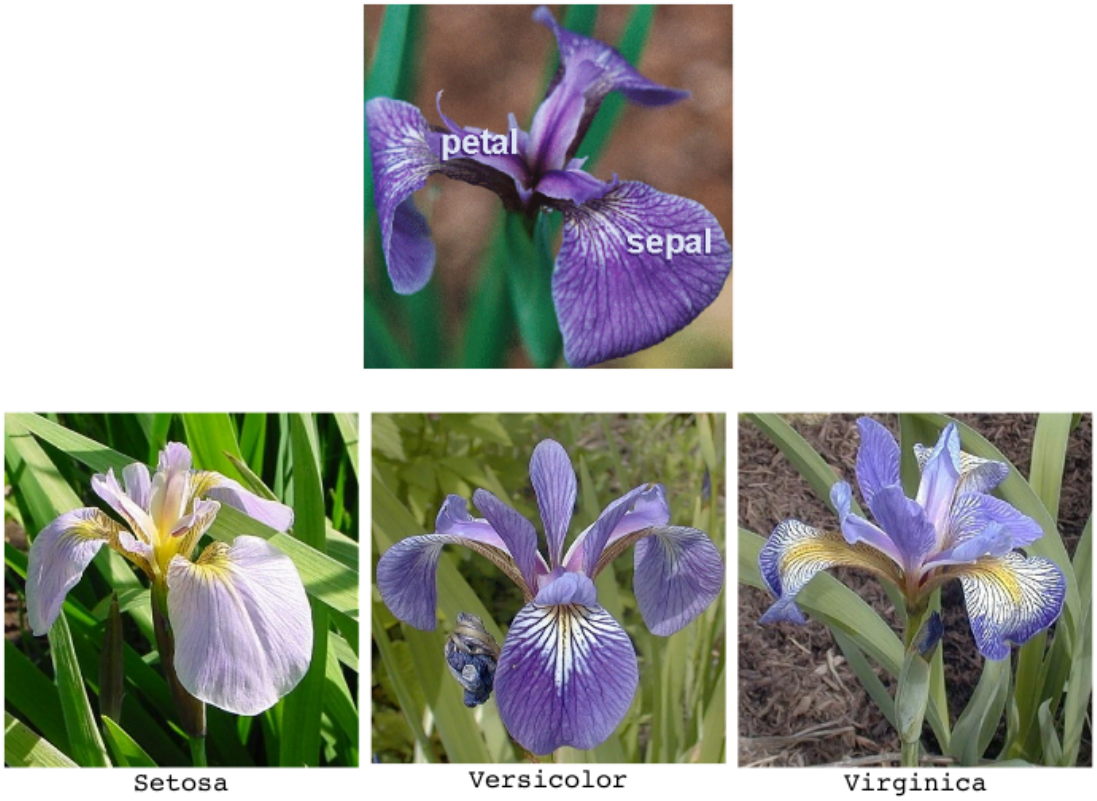
Enero a junio 2021

1 Representación de datos.

1.1 Ejemplo 1. Iris dataset.

[82]:
```
import IPython.display as ipd
ipd.Image("figs/iris-all.png",width=500)
```

[82]:



[98]:
```
import seaborn as sns
%matplotlib inline

iris = sns.load_dataset('iris')
iris.head(10)
```

[98]:

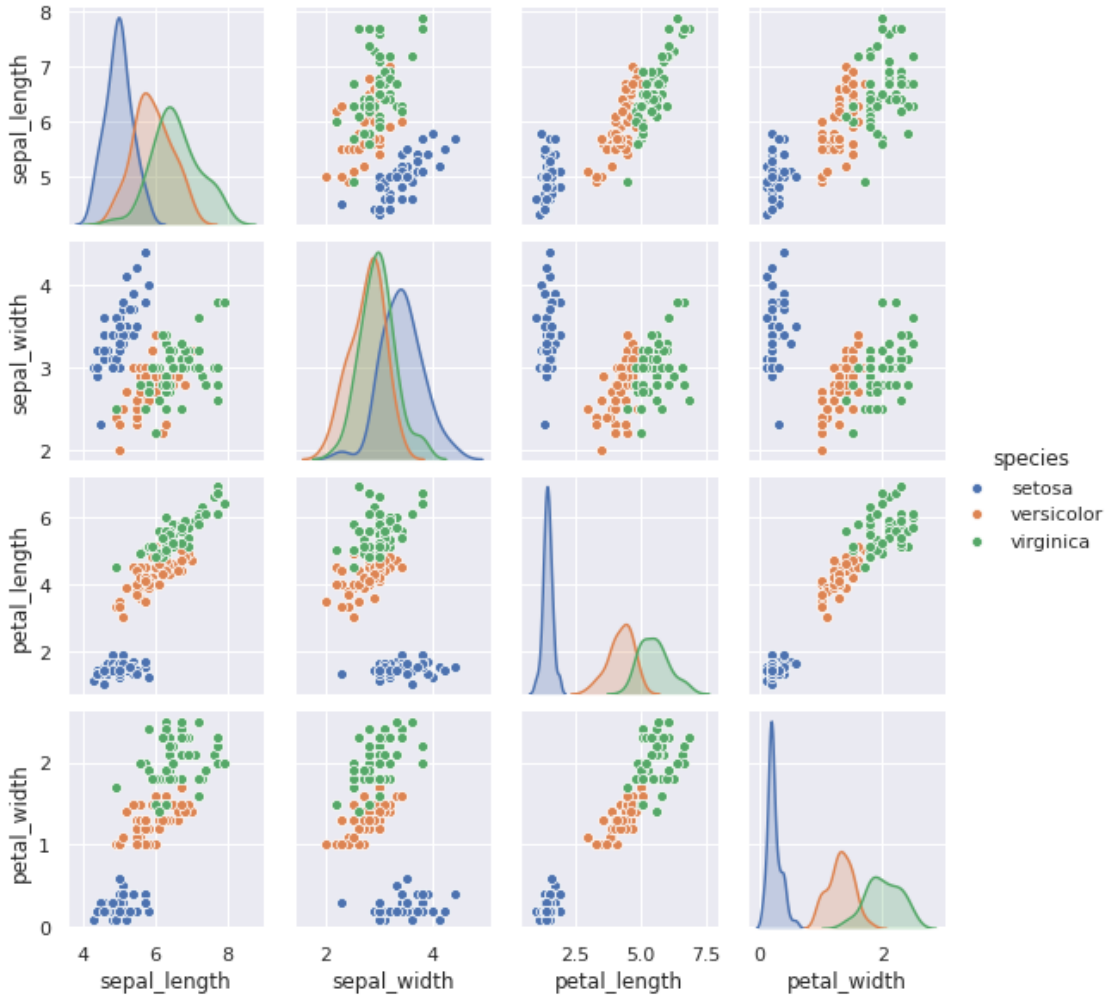
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa

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

Aprendizaje No Supervisado: detectar patrones en los datos.

[84]:
```
sns.set()
sns.pairplot(iris, hue='species', height=2)
```

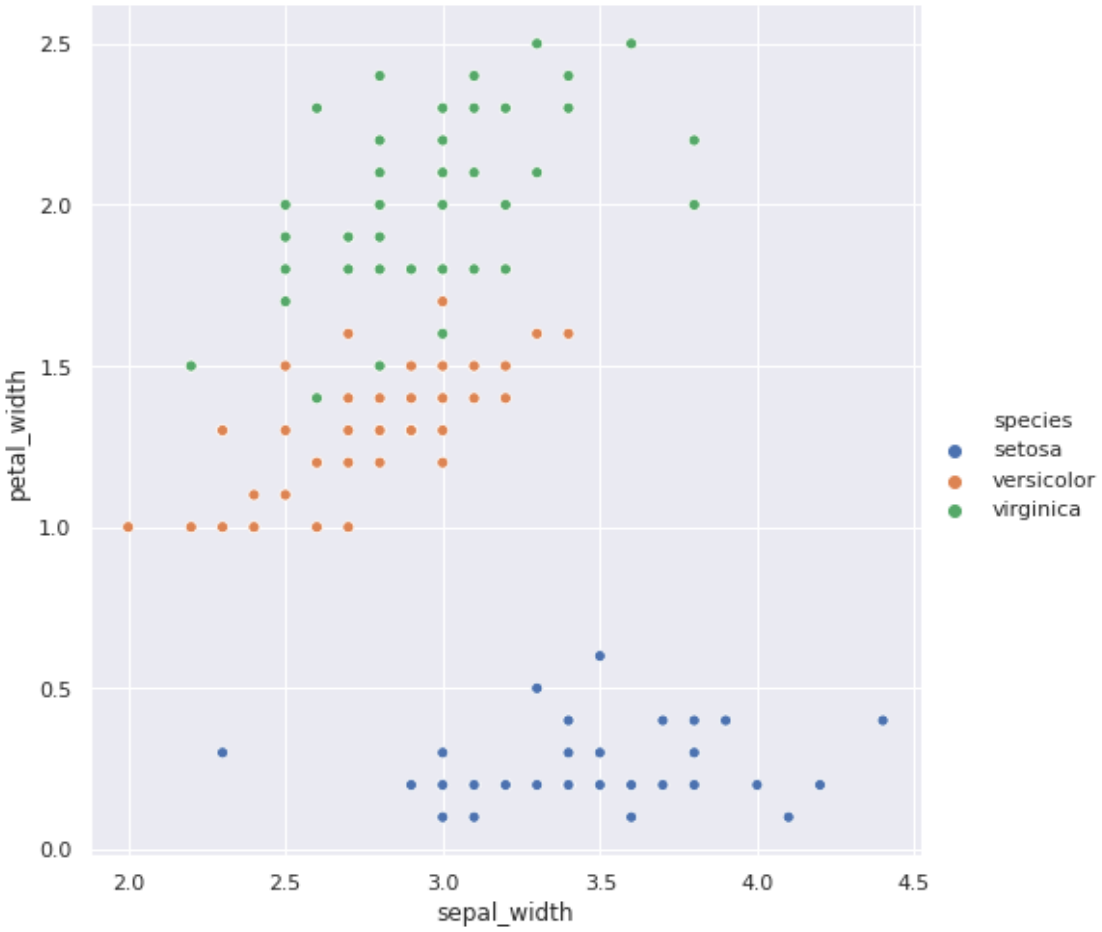
[84]: <seaborn.axisgrid.PairGrid at 0x7fd9e819d8e0>



Veamos solo un par de variables…

[85]:
```
sns.relplot(x='sepal_width', y='petal_width', hue='species', data=iris,␣
↪height=7)
```

[85]: <seaborn.axisgrid.FacetGrid at 0x7fd9dbf20670>



Simplifiquemos el problema.

Unimos las categorías *`versicolor'* y *`virginica'* en la clase *`vers_virg'*

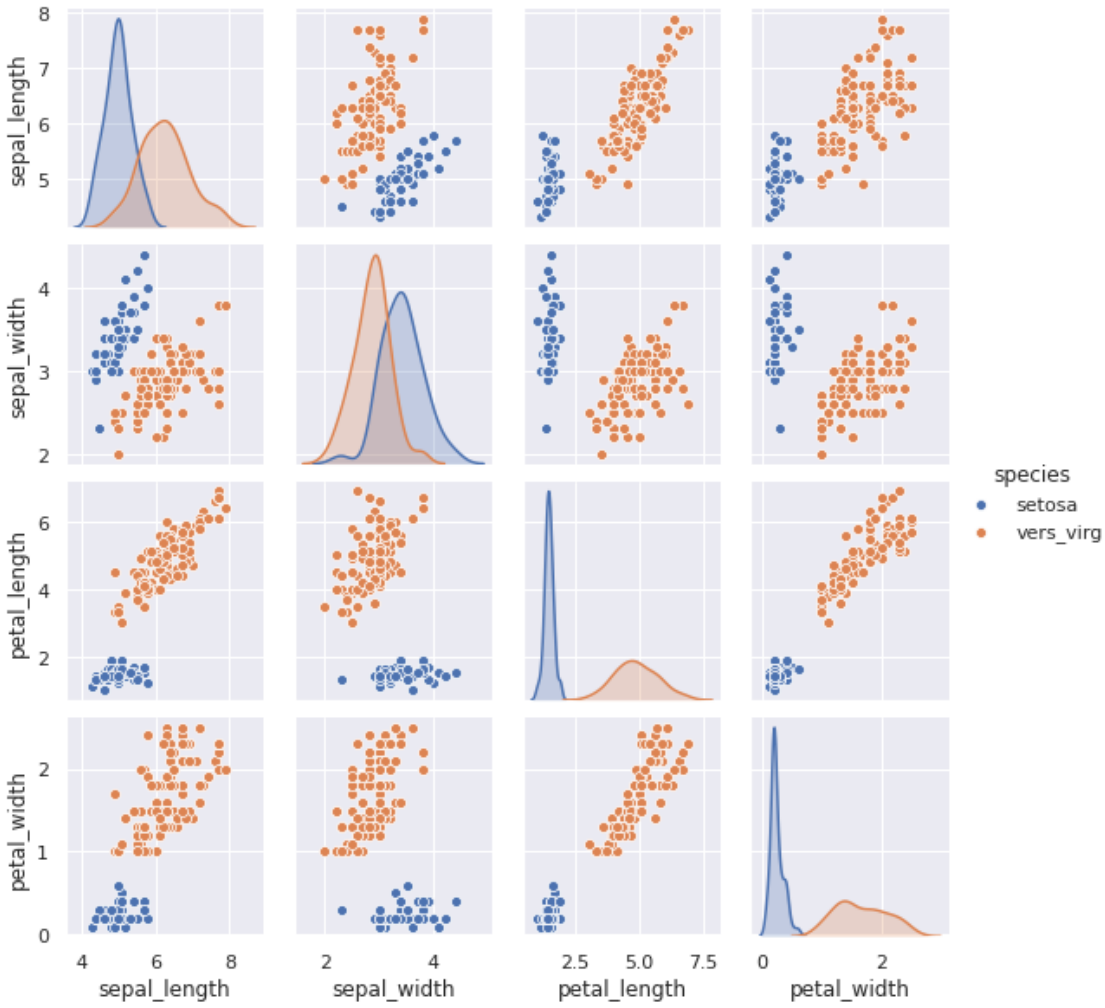
[86]:
```
iris2 = iris
spec2 = iris2['species'].copy()
spec2[spec2.eq('versicolor') | spec2.eq('virginica')] = 'vers_virg'
iris2['species'] = spec2
iris2.iloc[[0,10,50,100],:]
```

[86]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
10	5.4	3.7	1.5	0.2	setosa
50	7.0	3.2	4.7	1.4	vers_virg
100	6.3	3.3	6.0	2.5	vers_virg

[87]:
```
sns.pairplot(iris2, hue='species', height=2)
```

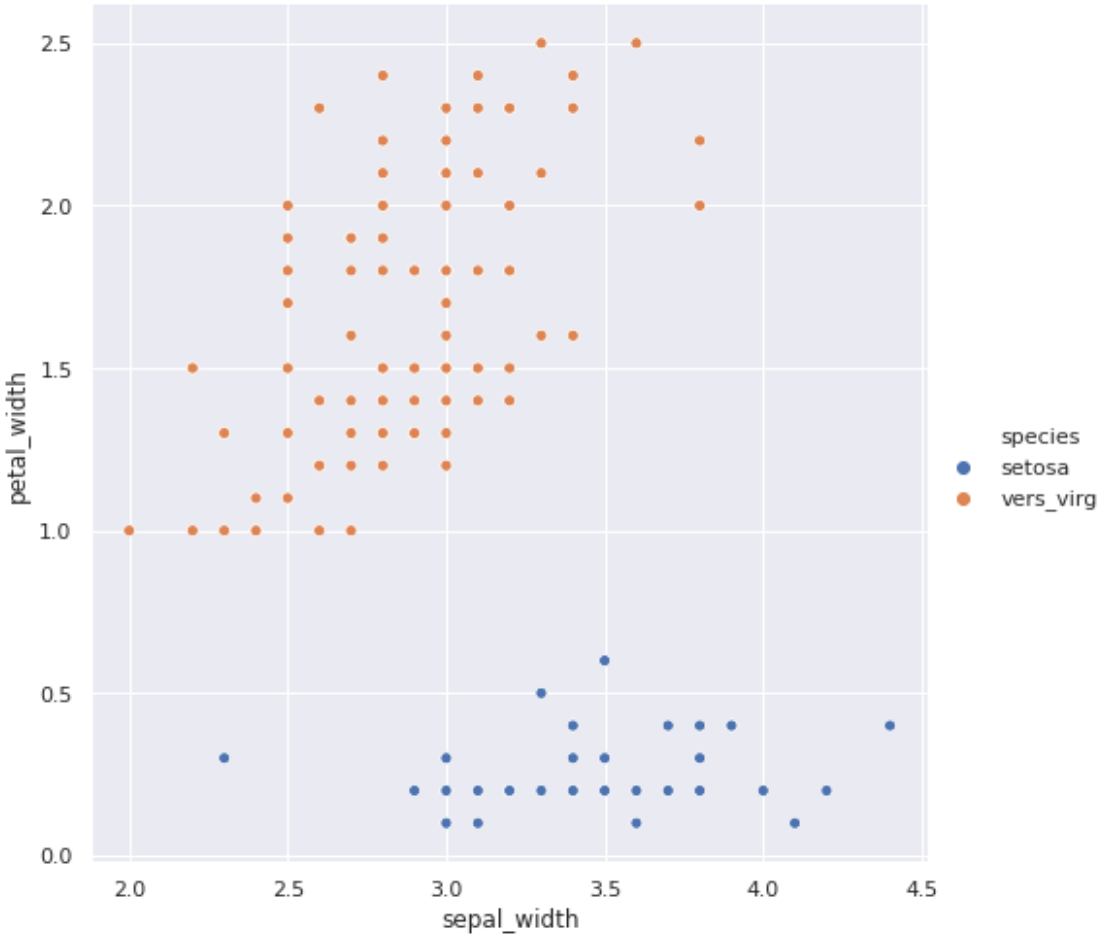
[87]: <seaborn.axisgrid.PairGrid at 0x7fd9db886760>



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

[90]:
```
sns.relplot(x='sepal_width', y='petal_width', hue='species', data=iris2,␣
↪height=7)
```

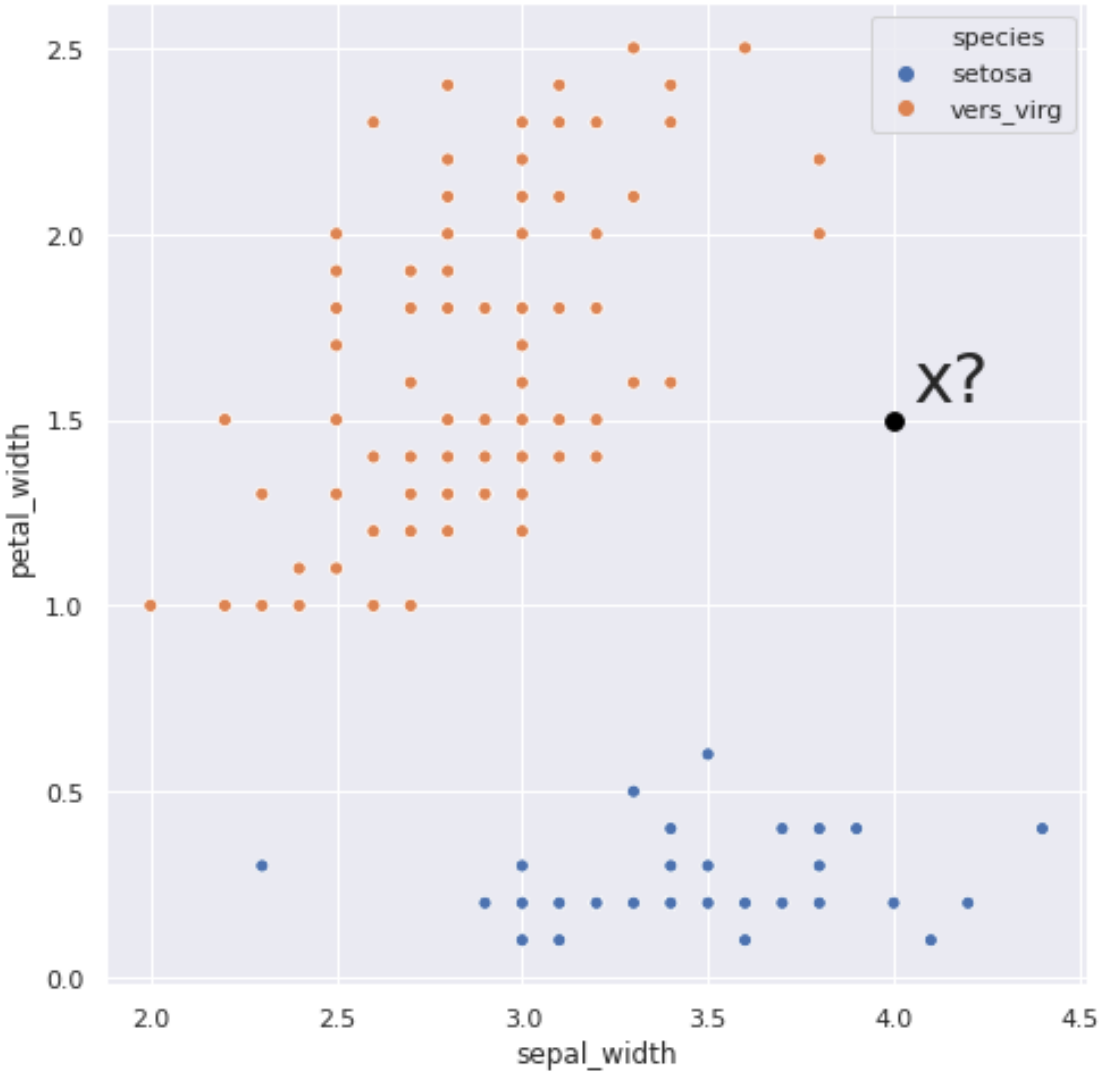
[90]: <seaborn.axisgrid.FacetGrid at 0x7fd9e9a50f40>



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

[91]:
```
import matplotlib.pyplot as plt

new_x = [4, 1.5]
fig = plt.figure(figsize = (6,6))
ax=fig.add_axes([0,0,1,1])
sns.scatterplot(x='sepal_width', y='petal_width', hue='species', data=iris2)
ax.scatter(x=new_x[0], y=new_x[1], color='black', s=60)
ax.text(new_x[0]+.05, new_x[1]+.05, 'x?', fontsize = 30)
```

[91]: Text(4.05, 1.55, 'x?')



¿Qué clase le asignarías? ¿Porqué?

Tenemos entonces un conjunto de entrenamiento:

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

Queremos una función

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

que nos responda la pregunta: ¿$\mathbf{x}_{new}$ es similar a Iris Setosa (-1) o a Iris Versicolor-Virginica ($+1$)?

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:

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

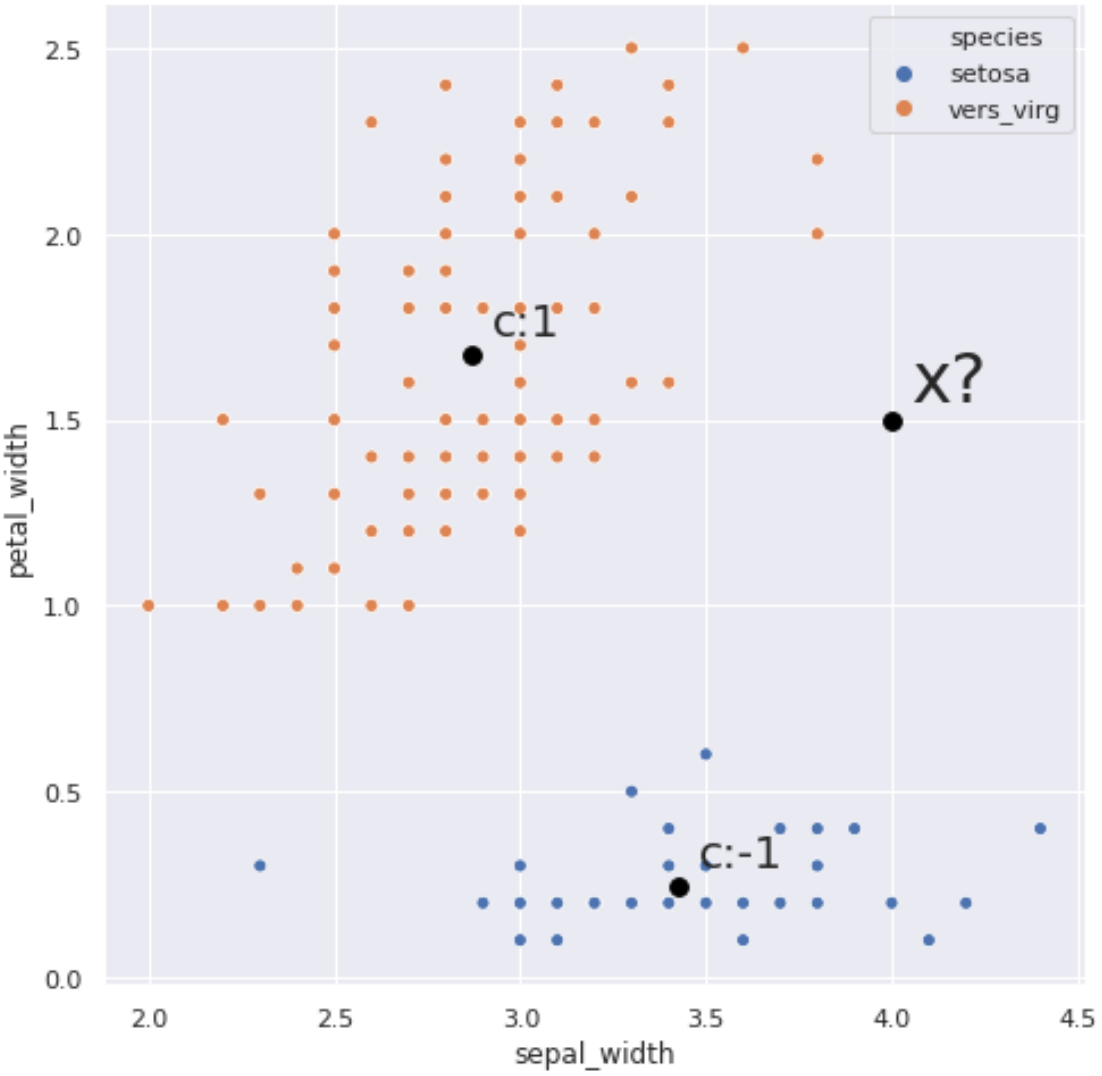
y

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

[92]:
```
import numpy as np

xtemp = iris2.loc[iris2['species']=='setosa',['sepal_width', 'petal_width']]
x_set = np.array(xtemp)
xtemp = iris2.loc[iris2['species']!='setosa',['sepal_width', 'petal_width']]
x_ver_vir = np.array(xtemp)
x_means = np.vstack((x_set.mean(axis=0),x_ver_vir.mean(axis=0)))
all_mean = x_means.mean(axis=0)
```

[93]:
```
ax.scatter(x_means[:,0], x_means[:,1], color='black', s=60)
for i in range(0,x_means.shape[0]):
    ax.text(x_means[i,0]+.05, x_means[i,1]+.05, 'c:'+str(round(2*i-1)),␣
 ↪fontsize = 20)
fig
```

[93]:



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

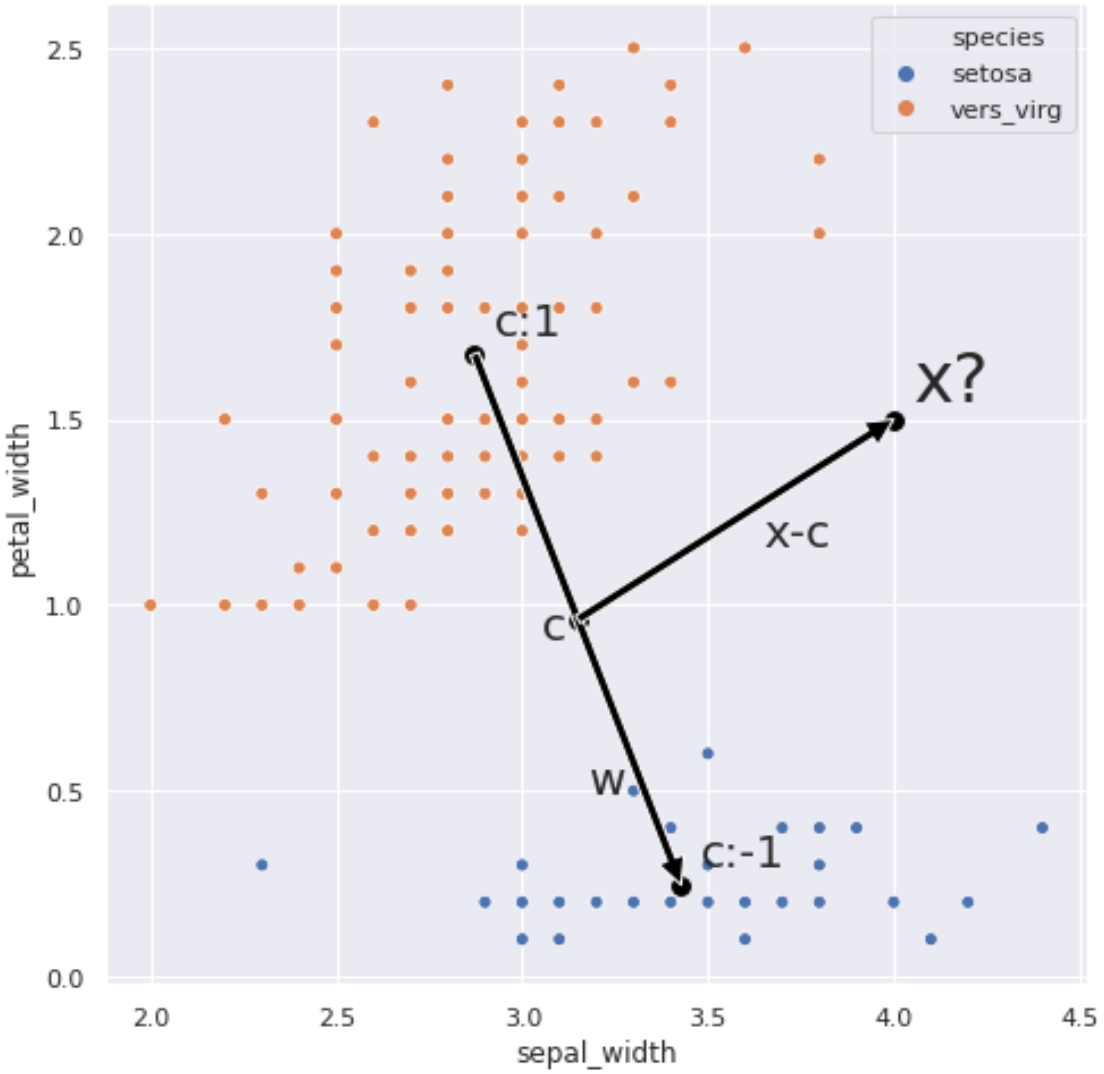
$\mathbf{c} = (\mathbf{c}_+ - \mathbf{c}_-)/2$,

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

[94]:
```
ax.scatter(all_mean[0], all_mean[1], color='black', s=60)
ax.text(all_mean[0]-.1, all_mean[1]-.05, 'c', fontsize = 20)
```

```
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]-.25, x_means[0,1]+.25, '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]+.5, all_mean[1]+.2, 'x-c', fontsize = 20)
fig
```

[94]:



Lo anterior, induce un modelo lineal de clasificación:

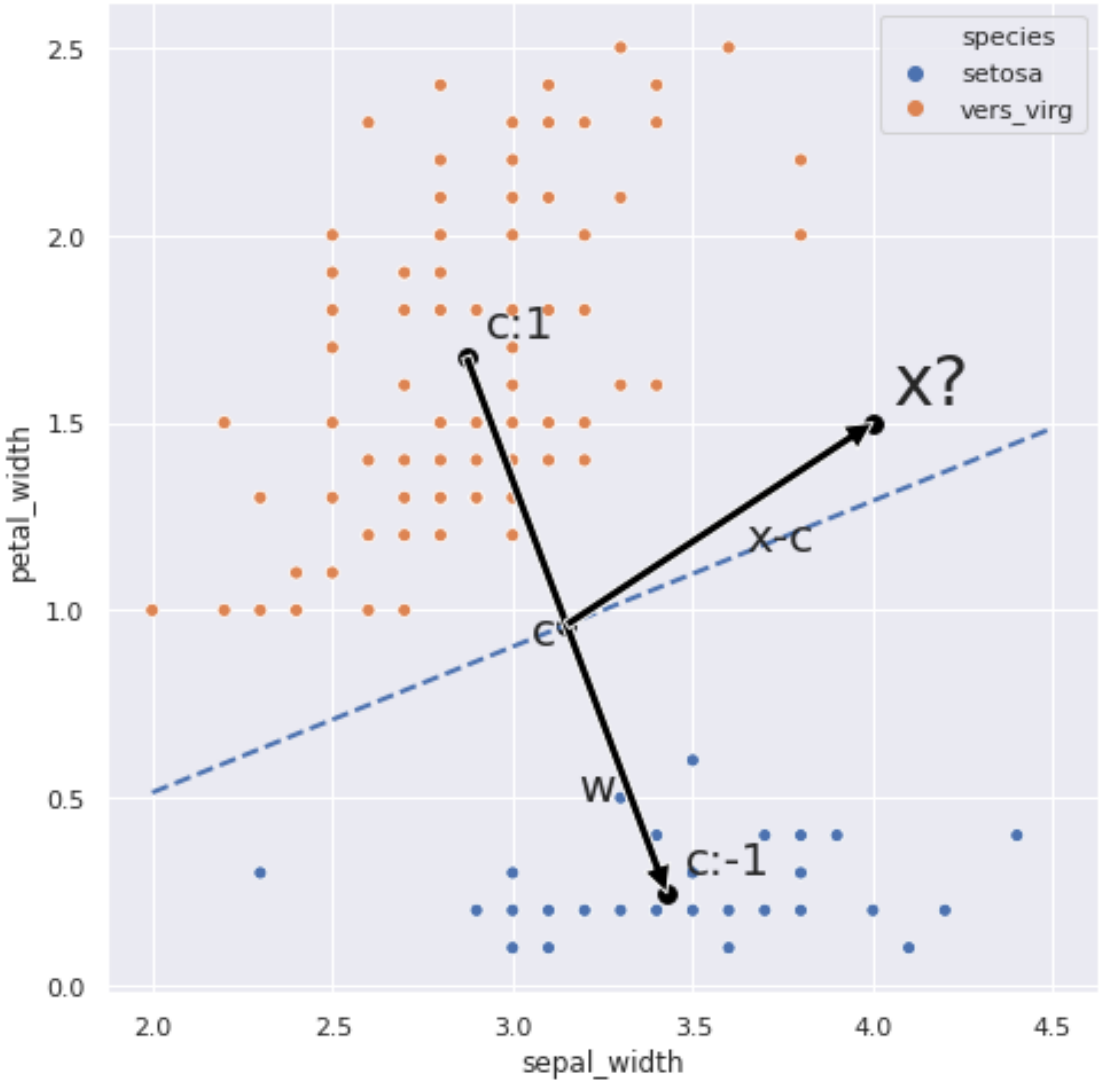
$w_0 + \mathbf{w}'\mathbf{x}$.

Entonces, podemos clasificar mediante

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

[95]:
```
xx1 = np.linspace(2,4.5,100)
xx2 = all_mean[1]-(((xx1-all_mean[0])*w[0])/w[1])
ax.plot([xx1[0],xx1[99]],[xx2[0],xx2[99]],linestyle='--', linewidth=2)
fig
```

[95]:



Obviamente, ésta solución no es la mejor opción.

Por ejemplo, ¿qué pasa si tenemos outliers?

1.2 Ejemplo 2. Digitos MNIST

La base de datos está incluida en varios módulos (por ejemplo, sklearn), pero para usar la versión original de 28 x 28 pixeles, usaremos la que está contenida

en keras.datasets.

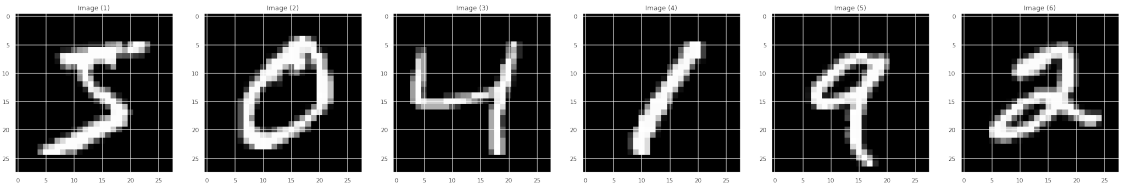
[101]:
```
import keras
from keras.datasets import mnist
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
%matplotlib inline

# Cargamos los datos
nmax = 10000
(x_train, y_train), (x_test, y_test) = mnist.load_data()
img_width = img_height = x_train[0].shape[0]
x_t = x_train.reshape(x_train.shape[0], img_width*img_height)
X = x_t[0:nmax,]
X = X/255
y_train = y_train[0:nmax]
# por el momento, no usaremos estos...
#X_te = x_test.reshape(x_test.shape[0], img_width*img_height)
#X_te = X_te/255
#y_train = keras.utils.to_categorical(y_train, num_classes)

num_classes = 10
```

Visualización de algunas imágenes

[105]:
```
def show_images(images, cols = 1, titles = None):
    """Display a list of images in a single figure with matplotlib.
    """
    assert((titles is None)or (len(images) == len(titles)))
    n_images = len(images)
    if titles is None: titles = ['Image (%d)' % i for i in range(1,n_images +␣
 ↪1)]
    fig = plt.figure()
    for n, (image, title) in enumerate(zip(images, titles)):
        a = fig.add_subplot(cols, np.ceil(n_images/float(cols)), n + 1)
        if image.ndim == 2:
            plt.gray()
        plt.imshow(image)
        a.set_title(title)
    fig.set_size_inches(np.array(fig.get_size_inches()) * n_images)
    plt.show()

imgs = []
for i in range(6):
    imgs.append(x_train[i].reshape((28,28)))
```

```
show_images(imgs,1)
```

```
<ipython-input-105-51190c471de8>:9: MatplotlibDeprecationWarning:

Passing non-integers as three-element position specification is deprecated since
3.3 and will be removed two minor releases later.
```



Considera ésta representación de los datos

[102]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
scaler = StandardScaler()
pca = PCA(n_components=2)

# Ajustamos en datos de entrenamiento
scaler.fit(X)
train_img = scaler.transform(X)

# Obtenemos las representaciones
x_pca = pca.fit_transform(train_img)
```

[4]:
```
import plotly.express as px
pca_dataset = pd.DataFrame({'pc1': x_pca[:, 0], 'pc2': x_pca[:, 1], 'digit':␣
 ↪y_train})
fig = px.scatter(pca_dataset, x='pc1', y='pc2', color= 'digit',␣
 ↪hover_data=['digit'])
fig.update_layout(autosize=False,width=800,height=800)
fig.show()
```

¿Qué tan útil es ésta representación?

Veámoslo con una aplicación interactiva…

[99]:
```
from tkinter import *
from digitsTab import *
```

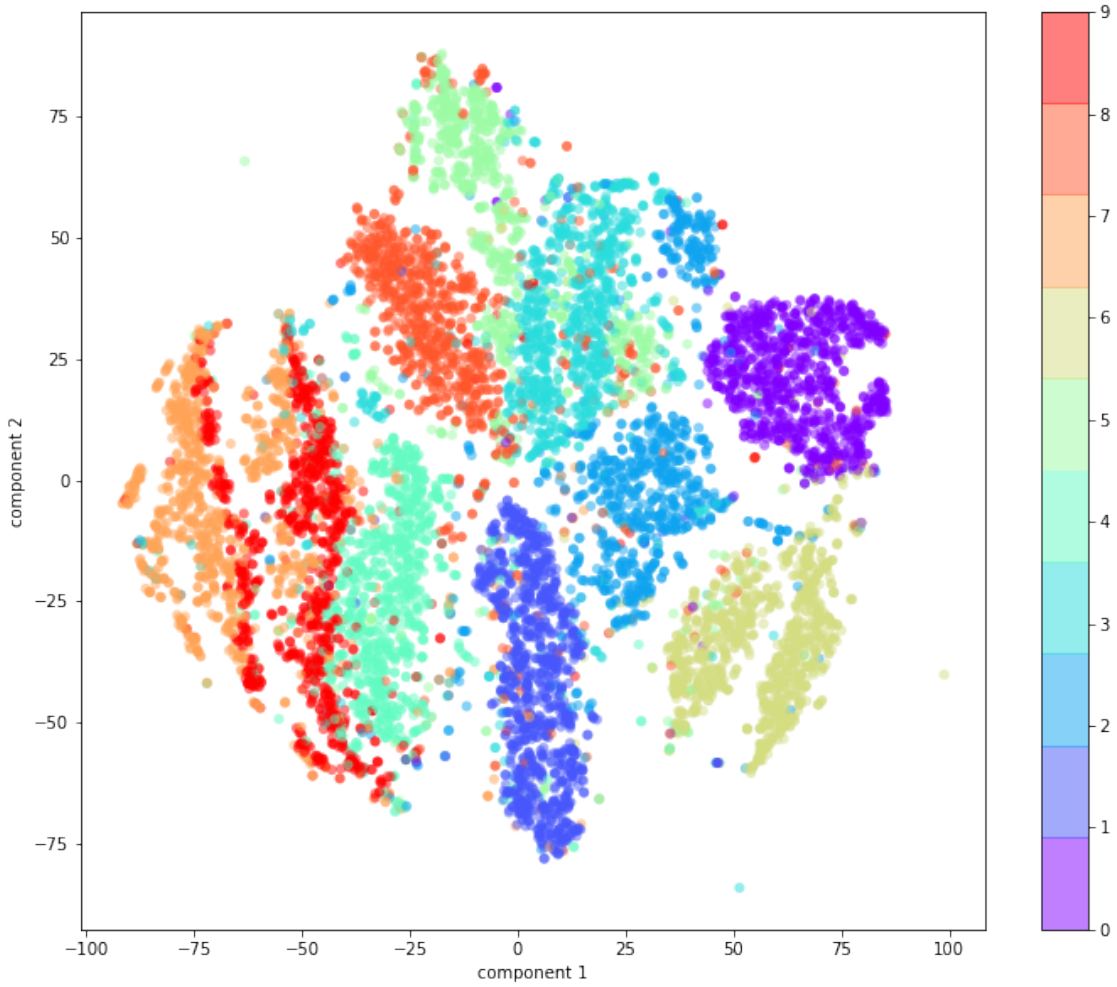
[108]:
```
# ejecutamos el applet
root = Tk()
root.geometry("400x400")
```

```
root.resizable(0, 0)
app = digits_representation(root, img_width, img_height, scaler, pca, x_pca,␣
 ↪y_train)
root.mainloop()
```

Otra representación… compara con la anterior (tarda en ejecutarse…)

[7]:
```
from sklearn.manifold import TSNE

import numpy as np
np.random.seed(0)
tsne = TSNE()
X_tsne = tsne.fit_transform(train_img)

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

[11]:
```
plt.figure(figsize = (12,10))

plt.scatter(tsne_dataset['pc1'], tsne_dataset['pc2'],c=y_train,␣
 ↪edgecolor='none', alpha=0.5,
            cmap=plt.cm.get_cmap('rainbow', 10))
plt.xlabel('component 1')
plt.ylabel('component 2')
plt.colorbar()
plt.show()
```



Gráfica interactiva

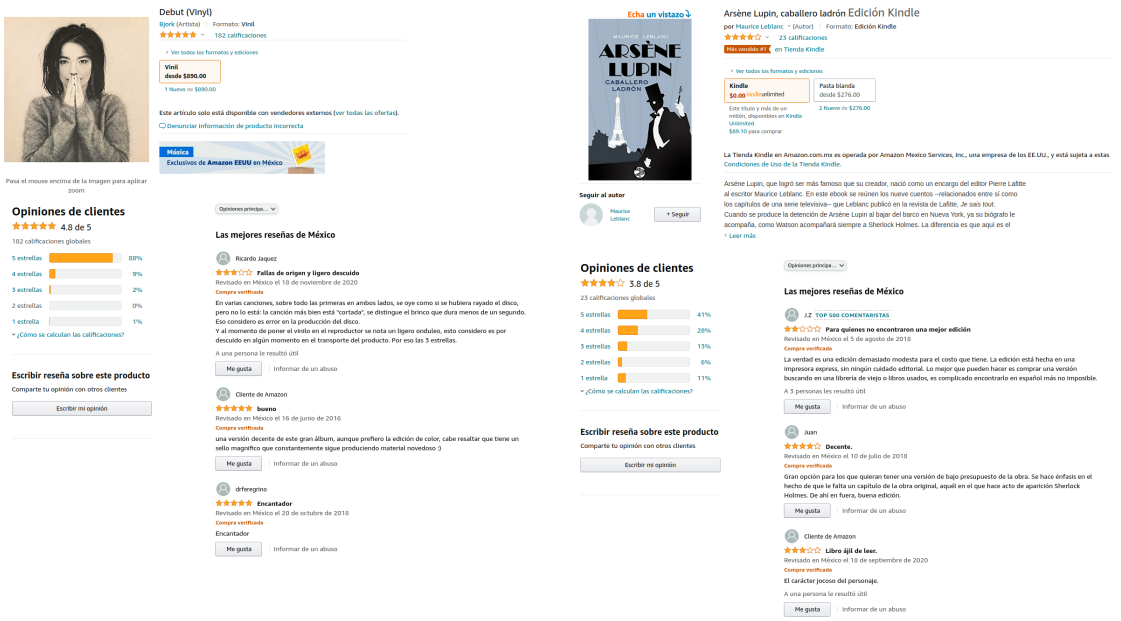
[10]:
```
fig = px.scatter(tsne_dataset, x='pc1', y='pc2', color= 'digit',␣
 ↪hover_data=['digit'])
fig.update_layout(
    autosize=False,
    width=800,
```

```
    height=800,
)
fig.show()
```

1.3 Ejemplo 3. Textos de opinión

[2]:
```
ipd.Image("figs/text_opiniones.png")
```

[2]:



Aquí, usaremos textos en español que corresponden a opiniones de usuarios en los siguientes productos: automóviles, hoteles, lavadoras, libros, teléfonos celulares, música, computadoras y películas (Julian Brooke and Maite Taboada. https://www.sfu.ca/~mtaboada/SFU_Review_Corpus.html).

No todo el código lo vas a poder ejecutar, ya que faltan varias cosas que en éste momento no es importante saber (algunas las veremos posteriormente en el curso). Fijate solo en la idea del ejemplo…

[96]:
```
from sklearn.feature_extraction.text import CountVectorizer
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
import my_functions
from my_functions import *

t_data = pd.read_csv('../data/spanish_reviews/reviews_text_caract.csv',␣
 ↪header=0)
```

```
t_data
```

[96]:
```
                        file  categoria sentimiento
0         coches_no_1_11.txt     coches          no
1         coches_no_1_13.txt     coches          no
2         coches_no_1_15.txt     coches          no
3         coches_no_1_18.txt     coches          no
4         coches_no_1_19.txt     coches          no
..                       …        …         …
395  peliculas_yes_5_23.txt  peliculas         yes
396   peliculas_yes_5_4.txt  peliculas         yes
397   peliculas_yes_5_5.txt  peliculas         yes
398   peliculas_yes_5_7.txt  peliculas         yes
399   peliculas_yes_5_9.txt  peliculas         yes

[400 rows x 3 columns]
```

[9]:
```
dir_data = '../data/spanish_reviews/all_files/'
# leer y preprocesar textos
preprocesador = preprocesaTexto(idioma='es', _tokeniza=False,␣
 ↪_muestraCambios=False, _quitarAcentos=True,
                                 _remueveStop=True,␣
 ↪_stemming=False,_lematiza=False)

files_txt = dir_data+t_data['file']
files_txt = files_txt.tolist()

corpus = []
raw_txt = []
for f in files_txt:
    file = open(f, 'r', encoding='latin-1')
    txt = file.read()
    raw_txt.append(txt)
    txt_prep = preprocesador.preprocesa(txt)
    corpus.append(txt_prep)

y1 = t_data['categoria'].astype('category').cat.codes
y2 = t_data['sentimiento'].astype('category').cat.codes
```

Veamos algunos ejemplos del corpus, tanto el original como el preprocesado.

[5]:
```
raw_txt[10]
```

[5]:
```
'Hola a todos. Tengo un Scenic RX4 con dos años y medio y 60.000 Km. La mayoría
de defectos que citan los "desilusionados" los he tenido yo también: me
entregaron el coche con un fallo del turbo (no me creo que le hicieran revisión
pre-entrega, o si la hicieron les dió igual darme el coche con la avería. sic),
```

rotura del turbo con 55.000 Km (pese a estar fuera de garantía por kilometraje, Renault se hizo cargo del 80%. "Solo" pagué 300 euros), infinitos fallos con el sensor de presión de los neumáticos (reconozco que es una chorrada de avería, pero me obligó a llevar el coche al concesionario al menos 6 veces), el sistema de anclaje de la tapa de la gasolina y el motor del cierre de esta, están hechos para que se rompan al poco tiempo (solo hay que verlo) y por último el famoso sistema de escape. Me han cambiado 2 veces el catalizador, dos veces el tramo intermedio y una el silencioso final. Acoples y cambios en el diseño del escape no han servido de nada y todas las semanas reviso los bajos del coche (como si tuviera miedo a que me pusieran una bomba, vaya). Solo conozco un caso de propietario de este vehículo que no se le haya roto (la verdad es que aun tiene pocos Km). Conozco personalmente casos peores con "fundidas de embrague", roturas de suspensiones, etc, que espero no afecten a mi unidad. En fin, el coche todo lo que tiene de bonito lo tiene de malo y si Renault ha dejado de fabricarlo … por algo será. \n\nEn honor a la verdad he de decir que salvo la tapa de la gasolina y el 20% de la reparación del turbo, del resto de reparaciones se ha hecho cargo Renault, incluso las que se han producido fuera del periodo de garantía, sintoma inequivoco que reconocen la chapuza de coche que comercializan. Por cierto, la ampliación de la garantía a 3 años o 50.000 Km que costaba 300 euros, me la regaló Renault la tercera vez que llevé el coche al taller. En definitiva, que pedí que se quedará la casa el coche y me diera el valor actual del mercado, pero solo lo hacían si compraba otro Renault, lo cual, lógicamente, hasta que no me pase el disgusto que me ha producido una birria de casi 4 millones de pesetas, no pienso volver a hacer. \n\nSaludos y perdón por el rollo, pero es que me apetecía desahogarme. '

[10]:
```
corpus[10]
```

[10]: 'hola scenic rx dos años medio km mayoria defectos citan desilusionados entregaron coche fallo turbo creo hicieran revision preentrega si hicieron dio igual darme coche averia sic rotura turbo km pese garantia kilometraje renault hizo cargo solo pague euros infinitos fallos sensor presion neumaticos reconozco chorrada averia obligo llevar coche concesionario menos veces sistema anclaje tapa gasolina motor cierre hechos rompan tiempo solo verlo ultimo famoso sistema escape cambiado veces catalizador dos veces tramo intermedio silencioso final acoples cambios diseño escape servido todas semanas reviso bajos coche si miedo pusieran bomba vaya solo conozco caso propietario vehiculo roto verdad aun pocos km conozco personalmente casos peores fundidas embrague roturas suspensiones etc espero afecten unidad fin coche bonito malo si renault dejado fabricarlo honor verdad decir salvo tapa gasolina reparacion turbo resto reparaciones hecho cargo renault incluso producido periodo garantia sintoma inequivoco reconocen chapuza coche comercializan cierto ampliacion garantia años km costaba euros regalo renault tercera vez lleve coche taller definitiva pedi quedara casa coche diera valor actual mercado solo hacian si compraba renault logicamente pase disgusto producido birria casi millones pesetas pienso volver hacer saludos perdon rollo apetecia desahogarme '

Veamos una forma de representación. (NO EJECUTAR…)

[12]:
```
from gensim.models.keyedvectors import KeyedVectors
wordvectors_file_vec = '../../banxico/nlp_2020/data/fasttext-sbwc.vec'
size_w = 100000
wordvectors = KeyedVectors.load_word2vec_format(wordvectors_file_vec)
```

[13]:
```
def frase_a_vec(frase, modelo_wv):
    N = 0
    acc = np.zeros(modelo_wv.vector_size)
    for palabra in frase:
        if palabra in modelo_wv.vocab:
            acc += modelo_wv[palabra]
            N += 1
    return acc if N==0 else acc/N

def sentencias_a_vec(frases, modelo_wv):
    vectores = np.zeros((len(frases), modelo_wv.vector_size))
    for (i, frase) in enumerate(frases):
        vectores[i,:] = frase_a_vec(frase, modelo_wv)
    return vectores
```

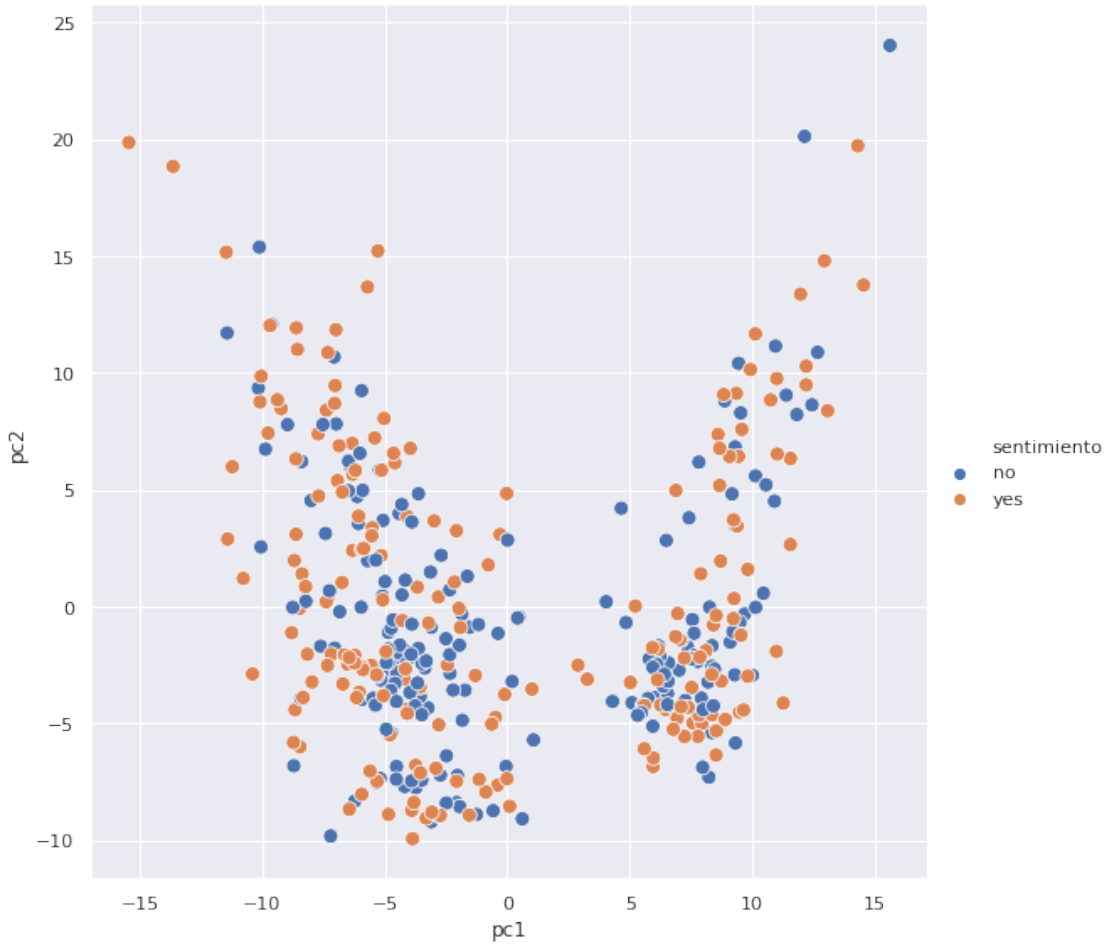
[14]:
```
docs_vec = np.zeros((len(corpus), wordvectors.vector_size))

for (i, txt) in enumerate(corpus):
    tokens = word_tokenize(txt)
    docs_vec[i,:] = frase_a_vec(tokens, wordvectors)
```

[ ]:
```
from sklearn.decomposition import PCA, KernelPCA
from sklearn.preprocessing import StandardScaler
import plotly.express as px

X = StandardScaler().fit_transform(docs_vec)
pca = PCA(n_components=5)
docs_pca = pca.fit_transform(X)
proj = pd.DataFrame(docs_pca,columns = ['pc1','pc2','pc3','pc4','pc5'])

proj_docs = pd.DataFrame({'pc1': proj['pc1'], 'pc2': proj['pc2'], 'pc3':␣
↪proj['pc3'], 'topico': t_data['categoria'],
                          'sentimiento':t_data['sentimiento']})
```

Visualmente, obtenemos lo siguiente. Observa cómo se ve respecto al tópico y al sentimiento.

[117]:
```
sns.relplot(x='pc1', y='pc2', hue='topico', data=proj_docs, height=8, s=80)
plt.show()
```

[118]:
```
sns.relplot(x='pc1', y='pc2', hue='sentimiento', data=proj_docs, height=8, s=80)
plt.show()
```



Gráfica interactiva

[17]:
```
fig = px.scatter(proj_docs, x='pc1', y='pc2', hover_data=['sentimiento'], color␣
↪= 'topico')
fig.update_layout(
    autosize=False,
    width=600,
    height=600,
)
fig.show()
```