

Importación de librerías

```
In [ ]: import pandas as pd
import numpy as np
import re
```

Comienzo de la depuración

```
In [ ]: df_ventas = pd.read_csv("items_ordered_2years.csv", sep=",")
```

En este dataframe los duplicados son valores que no tienen valor, puesto que no puede haber dos productos con el mismo item_id. Por tanto, son valores que hay que eliminar

```
In [ ]: #Eliminación de duplicados
df_ventas = df_ventas.drop_duplicates()

df_ventas.head()
```

```
Out[ ]:
```

	num_order	item_id	created_at	product_i
0	ce30c2f02458457e3c7b563a636ae2a1	0916c05c5c3f65f59d813a78ac35c8d2	2018-11-06 16:52:13	8643
1	ce30c2f02458457e3c7b563a636ae2a1	ff323b39ae36843396d2e53ce549fb10	2018-11-06 16:52:13	8765
2	ce30c2f02458457e3c7b563a636ae2a1	199916dff95259f4d2daab6664ca9c0	2018-11-06 16:52:13	278
3	83e75d608f11c8163599806420903ab9	8ca334ec2493501139327ce0165a1a84	2018-12-17 12:26:54	1300
4	9b687270d8e7eed9717022af5961a190	ce0a1683c6b1a0248b330344ec592ddf	2017-01-12 14:19:03	4194

```
In [ ]: df_ventas.nunique()
```

```
Out[ ]:
```

num_order	278045
item_id	906317
created_at	277896
product_id	26396
qty_ordered	57
base_cost	52202
price	5028
discount_percent	45
customer_id	118799
city	21012
zipcode	11132
dtype:	int64

Sigue habiendo mas registros que valores únicos en item_id, lo cual indica que hay que revisar qué puede estar generando esa duplicidad de id.

Revisión de ITEM_ID

```
In [ ]: df_ventas["item_id"].nunique()
```

```
Out[ ]: 906317
```

```
In [ ]: item_nu = df_ventas.groupby(
        "item_id", as_index=False
    ).size()

mascara = item_nu.iloc[:,1] > 1

item_repetidos = item_nu[mascara]

item_repetidos.head()
```

```
Out[ ]:
```

	item_id	size
111	000802dde903f8282df756d2aab8d024	2
135	000a8c9981c4dc672abd207bdc1b85f7	2
236	001183d1253fa3d736eda6c92c826360	2
375	001b62e0e6c6fcf7fd771422bc91cb12	2
385	001bd90aa12b86fc11c4a416b3495cd5	2

```
In [ ]: mascara2 = df_ventas["item_id"].isin(
        item_repetidos["item_id"].tolist()
    )

df_ventas_repetido = df_ventas[mascara2].sort_values("item_id")

df_ventas_repetido.head()
```

```
Out[ ]:
```

	num_order	item_id	created_at	prod
649241	8051708bf244ba8518b5e5b61a74486b	000802dde903f8282df756d2aab8d024	2018-12-17 23:32:08	
649259	8051708bf244ba8518b5e5b61a74486b	000802dde903f8282df756d2aab8d024	2018-12-17 23:32:08	
447200	6d19dedf8710788ea7d774c3fc977a9b	000a8c9981c4dc672abd207bdc1b85f7	2018-09-02 16:35:36	
447182	6d19dedf8710788ea7d774c3fc977a9b	000a8c9981c4dc672abd207bdc1b85f7	2018-09-02 16:35:36	
766220	27df73640a5bbe494d1db3e3bf16b906	001183d1253fa3d736eda6c92c826360	2018-08-07 09:29:02	

Las duplicidades del item_id aparecen debido a presumiblemente errores en el zipcode/city.
Se tratará despues de corregir las columnas de city y zipcode

Revisión de valores nulos

```
In [ ]: df_ventas.isna().sum()
```

```
Out[ ]: num_order      0
        item_id       0
        created_at    0
        product_id    0
        qty_ordered   0
        base_cost     2358
        price         0
        discount_percent 0
        customer_id   0
        city          2910
        zipcode       2910
        dtype: int64
```

Valores nulos en zipcode y city similares. Pueden ser coincidentes.

```
In [ ]: mascara_na = df_ventas["zipcode"].notna()
        df_ventas[mascara_na ].isna().sum()
```

```
Out[ ]: num_order      0
        item_id       0
        created_at    0
        product_id    0
        qty_ordered   0
        base_cost     2355
        price         0
        discount_percent 0
        customer_id   0
        city          0
        zipcode       0
        dtype: int64
```

Efectivamente, al filtrar por zipcodes que no tuvieran na en la columna zipcode desaparecían los valores perdidos en la de city. Esos registros dificilmente pueden ser imputados por algoritmos como randomforest o knn, sin arriesgarse a estar añadiendo información de forma casi aleatoria. Por ello, se prescindirá de esos registros.

```
In [ ]: df_ventas = df_ventas[mascara_na]
```

Revisión de zipcodes

```
In [ ]: df_ventas["longitud_zip"] = df_ventas["zipcode"].dropna().apply(len)
        df_ventas["longitud_zip"].unique().tolist()[0:5]
```

```
Out[ ]: [5, 2, 8, 4, 6]
```

Funciones para limpieza de city y zipcode

```
In [ ]: #Limpieza de zipcodes con RegEx
def num_guion(string):
    """ Get a string with the numbers and hyphens of another string

    Args:
        df: string used to extract the string with numbers abd hyphens

    Returns:
        df: the string with numbers and hyphens
    """
    aux = re.match("([\d-]+)", str(string))
    try:
        return str(aux.group())
    except:
        return string

# Limpieza de nombres de ciudad

# def AcentosLimpiador(text):
#     acentos = {'ñ': 'n', 'á': 'a', 'é': 'e', 'í': 'i', 'ó': 'o', 'ú': 'u',
#     for ele in acentos:
#         if ele in text:
#             text = text.replace(ele, acentos[ele])
#     return text

# def CityCleaner(text):
#     stopWordSpanish = set(stopwords.words('spanish'))
#     wordTokens = word_tokenize(AcentosLimpiador(text.lower()).rstrip())
#     filteredSentence = [element for element in wordTokens if not element in stopWordSpanish]
#     return filteredSentence
```

Limpieza de ambas columnas para facilitar el merge con la tabla de zipcodes

```
In [ ]: # Limpieza zipcode
df_ventas["zipcode"] = df_ventas["zipcode"].apply(lambda x: num_guion(x))
# df_ventas["city"] = df_ventas["city"].apply(lambda x: CityCleaner(x))
# df_ventas["city"] = df_ventas["city"].apply(lambda x: "".join(x))
```

Se carga el dataframe con la información de provincias, países, zipcodes y ciudades.

```
In [ ]: zipcodes_csv = pd.read_csv("DataScrapped.csv")
zipcodes_csv.drop("Unnamed: 0", axis=1, inplace=True)
zipcodes_csv.drop_duplicates(inplace=True)
```

```
In [ ]: zipcodes_csv.head()
```

Out []:

	Country	Region	City	Zipcode
0	Spain	Castilla - La Mancha	Tarazona De La Mancha	02100
4	Spain	Comunidad Valenciana	Alboraya	46120
74	Spain	Comunidad Valenciana	Benidorm	03503
76	Spain	Castilla - La Mancha	Yunquera De Henares	19210
88	Spain	Cataluna	Cornella De Llobregat	08940

Se genera una copia del dataframe con las columnas city y zipcode para un primer join/merge

```
In [ ]: # Nombres de las columnas a minuscula

column_list = zipcodes_csv.columns.tolist()
for i in range(len(column_list)):
    column_list[i] = column_list[i].lower()
zipcodes_csv.columns = column_list

# Obtener dataframe con solo city y zipcode, y pasar city a minusculas para
# zipcodes_to_join = zipcodes_csv.loc[:, ["zipcode", "city"]]
# zipcodes_to_join["city"] = zipcodes_to_join["city"].apply(lambda x: CityCl
# zipcodes_to_join["city"] = zipcodes_to_join["city"].apply(lambda x: "...joi
```

Para obtener aquellos zipcodes que estén mal escritos pero que estén bien la ciudad, se realiza un primer merge por columna city

```
In [ ]: df_ventas.drop("city", inplace=True, axis=1)
```

```
In [ ]: df_ventas_2 = df_ventas.merge(zipcodes_csv, how="left", on="zipcode")
```

```
In [ ]: df_ventas_2.head()
```

Out[]:

		num_order	item_id	created_at	product_i
0	ce30c2f02458457e3c7b563a636ae2a1	0916c05c5c3f65f59d813a78ac35c8d2	2018-11-06 16:52:13	8643	
1	ce30c2f02458457e3c7b563a636ae2a1	ff323b39ae36843396d2e53ce549fb10	2018-11-06 16:52:13	8765	
2	ce30c2f02458457e3c7b563a636ae2a1	199916dff95259f4d2daab6664ca9c0	2018-11-06 16:52:13	278	
3	83e75d608f11c8163599806420903ab9	8ca334ec2493501139327ce0165a1a84	2018-12-17 12:26:54	1300	
4	9b687270d8e7eed9717022af5961a190	ce0a1683c6b1a0248b330344ec592ddf	2017-01-12 14:19:03	4194	

```
In [ ]: mascara_null_2 = df_ventas_2["city"].notna()
df_ventas_2 = df_ventas_2[mascara_null_2]
```

```
In [ ]: df_ventas_2.drop_duplicates(inplace=True)
```

Solucionar el problema de id duplicadas

Se parte de la hipótesis de que son registros que se han duplicado al hacerse una corrección sobre la ubicación donde se mandaba el producto pero que no se eliminó el registro anterior. No se puede saber cual es el correcto, por lo que se decide tomar uno de los dos valores de forma aleatoria, que solo afectará a la ciudad/zipcode, mateniendose la comunidad y país de igual manera.

```
In [ ]: #Diccionario de la función de agregación por campo

list_columns = df_ventas_2.drop("item_id", axis = 1).columns.to_list()
list_agg = ["first"] * len(list_columns)
dict_agg = dict(zip(list_columns, list_agg))
```

```
In [ ]: # Nos deshacemos de los duplicados

df_ventas_3 = df_ventas_2.groupby(
    "item_id", as_index=False
).agg(dict_agg)

df_ventas_3.head()
```

```
Out[ ]:
```

		item_id	num_order	created_at	product_i
0	000010d95384a6ba3d57dd870e7b337c	65717498f0771a49497d80f11160093c	2017-09-22 15:46:37	564	
1	00001a8fb0bd42b1e16ba731e30cc490	09b538e85ce396ecbb70695f91007830	2018-09-12 21:27:08	2874	
2	0000302bc9b9a670dfcb14381555ff45	bc150db52b5a565d31b1c70969638ca9	2018-11-19 16:36:10	996	
3	000039147df4aacf0aa8b3a552e8ecdb	434cf1eaf255b367ce2d3343bb96b1fe	2017-09-06 12:08:44	6896	
4	000091029a220c2fdf12700f07f70b1d	f268c24275ad1d887925fca2909e2c2d	2018-09-29 09:45:14	966	

```
In [ ]: #Exportación de ventas_1

df_ventas_3.to_csv('df_ventas_1.csv')
```