In [1]:
```
from scipy.io import arff
```

In [2]:
```
data, meta = arff.loadarff("riesgo.arff")
```

In [24]:
```
import pandas as pd
import numpy as np
```

In [4]:
```
df = pd.DataFrame(data,columns=meta.names())
```

In [5]:
```
df[df.columns.drop("Estado")] = df.drop("Estado",axis=1).astype(int)
```

In [6]:
```
df["Estado"] = df["Estado"].astype(str)
```

In [7]:
```
def to_str_w_bytes(data):
    data = data.replace("b'", "")
    data = data.replace(" ", "")
    data = data.replace("'", "")
    return data
```

In [8]:
```
df["Estado"] = df["Estado"].apply(to_str_w_bytes)
```

In [9]:
```
df.head()
```

Out[9]:

	Estado	Acccidente	A-Volcanica	Alud	Aluvion	Avenida-torrencial	Linea-costeera	Colapso-estructural	Contaminacion	Deslizamiento	...	Panico
0	TAMAULIPAS	0	0	0	0	0	0	0	0	0	...	0
1	DISTRITO-FEDERAL	0	0	0	0	0	0	0	0	0	...	0
2	GUANAJUATO	0	0	0	0	0	0	0	0	0	...	0
3	MICHOACAN	0	0	0	0	0	0	0	0	0	...	0
4	VERACRUZ	0	0	0	0	0	0	0	0	0	...	0

5 rows × 40 columns

In [13]:
```
df_sum = df.groupby(by="Estado").sum()
```

In [12]:
```
from mlxtend.frequent_patterns import apriori, association_rules
```

In [14]:
```
# Building the model # Building the model
frq_items = apriori(df_sumsum, min_support = 0.05, use_colnames = True)

# Collecting the inferred rules in a dataframe
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False, False])
print(rules.head())
frq_items = apriori(basket_France, min_support = 0.05, use_colnames = True)

# Collecting the inferred rules in a dataframe
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False, False])
print(rules.head())
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-14-a8e9032a7ecc> in <module>
      1 # Building the model # Building the model
----> 2 frq_items = apriori(df_sum, min_support = 0.05, use_colnames = True)
      3
      4 # Collecting the inferred rules in a dataframe
      5 rules = association_rules(frq_items, metric ="lift", min_threshold = 1)

c:\users\joalc\documents\proyectos\desastresnaturales\venv\lib\site-packages\mlxtend\frequent_patterns\apr
iori.py in apriori(df, min_support, use_colnames, max_len, verbose, low_memory)
    236                             'Got %s.' % min_support)
    237
--> 238     fpc.valid_input_check(df)
    239
    240     if hasattr(df, "sparse"):

c:\users\joalc\documents\proyectos\desastresnaturales\venv\lib\site-packages\mlxtend\frequent_patterns\fpc
ommon.py in valid_input_check(df)
    114             s = ('The allowed values for a DataFrame'
    115                  ' are True, False, 0, 1. Found value %s' % (val))
--> 116             raise ValueError(s)
    117
    118

ValueError: The allowed values for a DataFrame are True, False, 0, 1. Found value 4
```

In [ ]:

In [35]:
```
(unique, counts) = np.unique(df.drop(["Estado"], axis=1).values.sum(axis=1), return_counts=True)
frequencies = np.asarray((unique, counts)).T
frequencies
```

Out[35]:
```
array([[    0,  1373],
       [    1, 41059]], dtype=int64)
```

In [ ]: