PROGETTO

January 1, 2022

[1]:
```
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import numpy as np #LINEAR ALGEBRA
from matplotlib import pyplot as plt #DATA VISUALIZATION
import matplotlib.pyplot as plt #plotting
import seaborn as sns #plotting
import warnings
warnings.filterwarnings('ignore') #don't disturb
```

[2]:
```
df=pd.read_csv('globalterrorismdb_0718dist.csv',encoding='ISO-8859-1')#ho␣
↪risolto così il problema della codifica utf-8 che dava errori con il codec␣
↪di default usato da pandas
```

[3]:
```
df.head(10)
```

[3]:
```
        eventid  iyear  imonth  iday approxdate  extended resolution  country  \
0  197000000001   1970       7     2        NaN         0        NaN       58
1  197000000002   1970       0     0        NaN         0        NaN      130
2  197001000001   1970       1     0        NaN         0        NaN      160
3  197001000002   1970       1     0        NaN         0        NaN       78
4  197001000003   1970       1     0        NaN         0        NaN      101
5  197001010002   1970       1     1        NaN         0        NaN      217
6  197001020001   1970       1     2        NaN         0        NaN      218
7  197001020002   1970       1     2        NaN         0        NaN      217
8  197001020003   1970       1     2        NaN         0        NaN      217
9  197001030001   1970       1     3        NaN         0        NaN      217

           country_txt  region  …  \
0   Dominican Republic       2  …
1               Mexico       1  …
2          Philippines       5  …
3               Greece       8  …
4                Japan       4  …
5        United States       1  …
6              Uruguay       3  …
7        United States       1  …
8        United States       1  …
9        United States       1  …
```

```
181684  START Primary Collection       0          0         0        0       NaN
181685  START Primary Collection      -9         -9         0       -9       NaN
181686  START Primary Collection       0          0         0        0       NaN
181687  START Primary Collection      -9         -9         1        1       NaN
181688  START Primary Collection       0          0         0        0       NaN
181689  START Primary Collection      -9         -9         0       -9       NaN
181690  START Primary Collection      -9         -9         0       -9       NaN

[10 rows x 135 columns]
```

[5]:
```
df.dtypes
```

[5]:
```
eventid        int64
iyear          int64
imonth         int64
iday           int64
approxdate    object
               …
INT_LOG        int64
INT_IDEO       int64
INT_MISC       int64
INT_ANY        int64
related       object
Length: 135, dtype: object
```

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 181691 entries, 0 to 181690
Columns: 135 entries, eventid to related
dtypes: float64(55), int64(22), object(58)
memory usage: 187.1+ MB
```

[7]:
```
df.describe()
```

[7]:
```
            eventid          iyear         imonth           iday  \
count  1.816910e+05  181691.000000  181691.000000  181691.000000
mean   2.002705e+11    2002.638997       6.467277      15.505644
std    1.325957e+09      13.259430       3.388303       8.814045
min    1.970000e+11    1970.000000       0.000000       0.000000
25%    1.991021e+11    1991.000000       4.000000       8.000000
50%    2.009022e+11    2009.000000       6.000000      15.000000
75%    2.014081e+11    2014.000000       9.000000      23.000000
max    2.017123e+11    2017.000000      12.000000      31.000000

            extended        country         region       latitude  \
count  181691.000000  181691.000000  181691.000000  177135.000000
```

```
approxdate    181691
                …
INT_LOG       181691
INT_IDEO      181691
INT_MISC      181691
INT_ANY       181691
related       181691
Length: 135, dtype: int64
```

[15]:
```python
df.isna().sum()/len(df)*100
```

[15]:
```
eventid        0.000000
iyear          0.000000
imonth         0.000000
iday           0.000000
approxdate    94.914993
                 …
INT_LOG        0.000000
INT_IDEO       0.000000
INT_MISC       0.000000
INT_ANY        0.000000
related       86.219461
Length: 135, dtype: float64
```

[16]:
```python
df.isnull().sum()/len(df)*100
```

[16]:
```
eventid        0.000000
iyear          0.000000
imonth         0.000000
iday           0.000000
approxdate    94.914993
                 …
INT_LOG        0.000000
INT_IDEO       0.000000
INT_MISC       0.000000
INT_ANY        0.000000
related       86.219461
Length: 135, dtype: float64
```

[17]:
```python
name_cols=df.columns
```

[18]:
```python
name_cols
```

[18]:
```
Index(['eventid', 'iyear', 'imonth', 'iday', 'approxdate', 'extended',
       'resolution', 'country', 'country_txt', 'region',
       …
       'addnotes', 'scite1', 'scite2', 'scite3', 'dbsource', 'INT_LOG',
```

```
       'INT_IDEO', 'INT_MISC', 'INT_ANY', 'related'],
      dtype='object', length=135)
```

[19]:
```
thresh=len(df)*.5# using thresh to drop >50% nulls
df.dropna(thresh=thresh,axis=1,inplace=True)
```

[20]:
```
df.isnull().sum()/len(df)*100
```

[20]:
```
eventid              0.000000
iyear                0.000000
imonth               0.000000
iday                 0.000000
extended             0.000000
country              0.000000
country_txt          0.000000
region               0.000000
region_txt           0.000000
provstate            0.231712
city                 0.238867
latitude             2.507554
longitude            2.508104
specificity          0.003302
vicinity             0.000000
summary             36.396409
crit1                0.000000
crit2                0.000000
crit3                0.000000
doubtterr            0.000550
multiple             0.000550
success              0.000000
suicide              0.000000
attacktype1          0.000000
attacktype1_txt      0.000000
targtype1            0.000000
targtype1_txt        0.000000
targsubtype1         5.709144
targsubtype1_txt     5.709144
corp1               23.418882
target1              0.350045
natlty1              0.858050
natlty1_txt          0.858050
gname                0.000000
guncertain1          0.209146
individual           0.000000
nperps              39.140629
nperpcap            38.245703
claimed             36.391456
```

```
weaptype1            0.000000
weaptype1_txt        0.000000
weapsubtype1        11.430396
weapsubtype1_txt    11.430396
weapdetail          37.244553
nkill                5.676120
nkillus             35.470111
nkillter            36.852678
nwound               8.977330
nwoundus            35.611010
nwoundte            38.055270
property             0.000000
ishostkid            0.097969
scite1              36.430533
dbsource             0.000000
INT_LOG              0.000000
INT_IDEO             0.000000
INT_MISC             0.000000
INT_ANY              0.000000
dtype: float64
```

[21]:
```
df.shape # as you can see it's happened that decreased data a lot
```

[21]:
```
(181691, 58)
```

[22]:
```
df.drop_duplicates()#now I want to drop duplicates
```

[22]:
```
             eventid  iyear  imonth  iday  extended  country  \
0       197000000001   1970       7     2         0       58
1       197000000002   1970       0     0         0      130
2       197001000001   1970       1     0         0      160
3       197001000002   1970       1     0         0       78
4       197001000003   1970       1     0         0      101
…                …      …      …     …         …        …
181686  201712310022   2017      12    31         0      182
181687  201712310029   2017      12    31         0      200
181688  201712310030   2017      12    31         0      160
181689  201712310031   2017      12    31         0       92
181690  201712310032   2017      12    31         0      160

                 country_txt  region                    region_txt  \
0         Dominican Republic       2  Central America & Caribbean
1                     Mexico       1                North America
2                Philippines       5               Southeast Asia
3                     Greece       8               Western Europe
4                      Japan       4                    East Asia
…                        …      …                          …
```

```
nkillus             float64
nkillter            float64
nwound              float64
nwoundus            float64
nwoundte            float64
property              int64
ishostkid           float64
scite1               object
dbsource             object
INT_LOG               int64
INT_IDEO              int64
INT_MISC              int64
INT_ANY               int64
dtype: object
```

[24]:

```
int_cols=df.select_dtypes(include=np.number).columns.tolist()#INTEGER cols
#as said df is a pandas DataFrame. I would like to find all columns of numeric␣
↪type.
#Simple one-line answer to create a new dataframe with only numeric columns:
#df.select_dtypes(include=np.number)
#If you want the names of numeric columns:
#df.select_dtypes(include=np.number).columns.tolist()
# Filling data with respective medians and modes
for i in int_cols:
  df[i] = df[i].fillna(df[i].median())
df.isnull().sum()#ovviamente i null-values rimangono nei tipi stringa ossia␣
↪"object"
```

[24]:

```
eventid              0
iyear                0
imonth               0
iday                 0
extended             0
country              0
country_txt          0
region               0
region_txt           0
provstate          421
city               434
latitude             0
longitude            0
specificity          0
vicinity             0
summary          66129
crit1                0
crit2                0
crit3                0
```

```
 'city',
 'summary',
 'attacktype1_txt',
 'targtype1_txt',
 'targsubtype1_txt',
 'corp1',
 'target1',
 'natlty1_txt',
 'gname',
 'weaptype1_txt',
 'weapsubtype1_txt',
 'weapdetail',
 'scite1',
 'dbsource']
```

[26]:
```python
for i in string_cols:
    print(df[i].describe())    # checking whether I can input mode to this data
    print('--------------------')
```

```
count     181691
unique       205
top         Iraq
freq       24636
Name: country_txt, dtype: object
--------------------
count                         181691
unique                            12
top       Middle East & North Africa
freq                           50474
Name: region_txt, dtype: object
--------------------
count      181270
unique       2855
top       Baghdad
freq         7645
Name: provstate, dtype: object
--------------------
count      181257
unique      36674
top       Unknown
freq         9775
Name: city, dtype: object
--------------------
count                                                115562
unique                                               112492
top       09/00/2016: Sometime between September 18, 201…
freq                                                    100
```

```
Name: weaptype1_txt, dtype: object
--------------------
count                    160923
unique                       30
top       Unknown Explosive Type
freq                      44980
Name: weapsubtype1_txt, dtype: object
--------------------
count        114021
unique        19148
top       Explosive
freq          20925
Name: weapdetail, dtype: object
--------------------
count                                                115500
unique                                                83988
top       Committee on Government Operations United Stat…
freq                                                    205
Name: scite1, dtype: object
--------------------
count                      181691
unique                         26
top       START Primary Collection
freq                        78002
Name: dbsource, dtype: object
--------------------
```

[27]:

```
#Square brackets can be used to access the content of a Serie and a DataFrame
for i in ['provstate','city','target1','natlty1_txt']:
  df[i]=df[i].fillna(df[i].mode()[0])  # cause its string
```

[28]:

```
df.columns
```

[28]:

```
Index(['eventid', 'iyear', 'imonth', 'iday', 'extended', 'country',
       'country_txt', 'region', 'region_txt', 'provstate', 'city', 'latitude',
       'longitude', 'specificity', 'vicinity', 'summary', 'crit1', 'crit2',
       'crit3', 'doubtterr', 'multiple', 'success', 'suicide', 'attacktype1',
       'attacktype1_txt', 'targtype1', 'targtype1_txt', 'targsubtype1',
       'targsubtype1_txt', 'corp1', 'target1', 'natlty1', 'natlty1_txt',
       'gname', 'guncertain1', 'individual', 'nperps', 'nperpcap', 'claimed',
       'weaptype1', 'weaptype1_txt', 'weapsubtype1', 'weapsubtype1_txt',
       'weapdetail', 'nkill', 'nkillus', 'nkillter', 'nwound', 'nwoundus',
       'nwoundte', 'property', 'ishostkid', 'scite1', 'dbsource', 'INT_LOG',
       'INT_IDEO', 'INT_MISC', 'INT_ANY'],
      dtype='object')
```

[29]:

```
df.head()
```

```
[29]:          eventid  iyear  imonth  iday  extended  country          country_txt  \
      0  197000000001   1970       7     2         0       58  Dominican Republic
      1  197000000002   1970       0     0         0      130              Mexico
      2  197001000001   1970       1     0         0      160         Philippines
      3  197001000002   1970       1     0         0       78              Greece
      4  197001000003   1970       1     0         0      101               Japan

         region                   region_txt provstate  … nwoundus  nwoundte  \
      0       2  Central America & Caribbean   Baghdad  …      0.0       0.0
      1       1                North America   Federal  …      0.0       0.0
      2       5               Southeast Asia    Tarlac  …      0.0       0.0
      3       8               Western Europe    Attica  …      0.0       0.0
      4       4                    East Asia   Fukouka  …      0.0       0.0

         property  ishostkid  scite1 dbsource  INT_LOG  INT_IDEO  INT_MISC  INT_ANY
      0         0        0.0     NaN     PGIS        0         0         0        0
      1         0        1.0     NaN     PGIS        0         1         1        1
      2         0        0.0     NaN     PGIS       -9        -9         1        1
      3         1        0.0     NaN     PGIS       -9        -9         1        1
      4         1        0.0     NaN     PGIS       -9        -9         1        1

      [5 rows x 58 columns]
```

```
[30]: #exploratory data analysis phase

      #•          Year, Month, day: time information of terrorist incidents
      #•          county: numeric code with country name following
      #•          region: numeric code with region name following
      #•          specificity: the geospatial resolution of the latitude and longitude␣
       ↪fields. The most specific resolution uniformly available throughout the␣
       ↪dataset is the center of the city, village, or town in which the attack␣
       ↪occurred
      #•          vicinity: The incident occurred in the immediate vicinity of the city␣
       ↪in
      #question or not
      #•          summary: description of the incident
      #-Eventid A 12-digit Event ID system. First 8 numbers - date recorded␣
       ↪"yyyymmdd". Last 4 numbers - sequential
      #-Iyear This field contains the year in which the incident occurred.
      #-Imonth This field contains the number of the month in which the incident␣
       ↪occurred.
      #-Iday This field contains the numeric day of the month on which the incident␣
       ↪occurred.
      #-Country This field identifies the country code
      #-country_txt This field identifies the country or location where the incident␣
       ↪occurred.
      #-Region This field identifies the region code based on 12 regions
```

```
#-Approxdate -> null al 95% (inutile)
#-Suicide 1 = "Yes" The incident was a suicide attack. 0 = "No" There is no␣
↪indication that the incident was a suicide
#-attacktype1_txt The general method of attack and broad class of tactics used.
#-target1 The specific person, building, installation that was targeted and/or␣
↪victimized
#-natlty1_txt The nationality of the target that was attacked
#-gname The name of the group that carried out the attack
#-weaptype1_txt General type of weapon used in the incident


df.rename(columns={'iyear':'Year','imonth':'Month','iday':'Day','country_txt':
↪'Country','region_txt':'Region','attacktype1_txt':'AttackType','target1':
↪'Target','nkill':'Killed','nwound':'Wounded','summary':'Summary','gname':
↪'Group','targtype1_txt':'Target_type','weaptype1_txt':'Weapon_type','motive':
↪'Motive'},inplace=True)
```

[31]:
```
df.describe()
```

[31]:
```
            eventid           Year          Month            Day  \
count  1.816910e+05  181691.000000  181691.000000  181691.000000
mean   2.002705e+11    2002.638997       6.467277      15.505644
std    1.325957e+09      13.259430       3.388303       8.814045
min    1.970000e+11    1970.000000       0.000000       0.000000
25%    1.991021e+11    1991.000000       4.000000       8.000000
50%    2.009022e+11    2009.000000       6.000000      15.000000
75%    2.014081e+11    2014.000000       9.000000      23.000000
max    2.017123e+11    2017.000000      12.000000      31.000000

            extended        country         region       latitude  \
count  181691.000000  181691.000000  181691.000000  181691.000000
mean        0.045346     131.968501       7.160938      23.498173
std         0.208063     112.414535       2.933408      18.377236
min         0.000000       4.000000       1.000000     -53.154613
25%         0.000000      78.000000       5.000000      11.849620
50%         0.000000      98.000000       6.000000      31.467463
75%         0.000000     160.000000      10.000000      34.538561
max         1.000000    1004.000000      12.000000      74.633553

          longitude    specificity  …      nkillter        Wounded  \
count  1.816910e+05  181691.000000  …  181691.000000  181691.000000
mean  -4.461064e+02       1.451437  …       0.320825       2.883296
std    2.021946e+05       0.995416  …       3.346474      34.309747
min   -8.618590e+07       1.000000  …       0.000000       0.000000
25%    6.655000e+00       1.000000  …       0.000000       0.000000
50%    4.324651e+01       1.000000  …       0.000000       0.000000
75%    6.835734e+01       1.000000  …       0.000000       2.000000
```

```
max    1.793667e+02       5.000000  …      500.000000    8191.000000

            nwoundus       nwoundte       property      ishostkid  \
count  181691.000000  181691.000000  181691.000000  181691.000000
mean        0.025076       0.066382      -0.544556       0.058996
std         2.453378       1.172976       3.122889       0.461022
min         0.000000       0.000000      -9.000000      -9.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.000000       0.000000       1.000000       0.000000
75%         0.000000       0.000000       1.000000       0.000000
max       751.000000     200.000000       1.000000       1.000000

             INT_LOG       INT_IDEO       INT_MISC        INT_ANY
count  181691.000000  181691.000000  181691.000000  181691.000000
mean       -4.543731      -4.464398       0.090010      -3.945952
std         4.543547       4.637152       0.568457       4.691325
min        -9.000000      -9.000000      -9.000000      -9.000000
25%        -9.000000      -9.000000       0.000000      -9.000000
50%        -9.000000      -9.000000       0.000000       0.000000
75%         0.000000       0.000000       0.000000       0.000000
max         1.000000       1.000000       1.000000       1.000000

[8 rows x 41 columns]
```

[32]:

```
pd.set_option('display.max_columns', 58)
df.describe()
```

[32]:

```
            eventid           Year          Month            Day  \
count  1.816910e+05  181691.000000  181691.000000  181691.000000
mean   2.002705e+11    2002.638997       6.467277      15.505644
std    1.325957e+09      13.259430       3.388303       8.814045
min    1.970000e+11    1970.000000       0.000000       0.000000
25%    1.991021e+11    1991.000000       4.000000       8.000000
50%    2.009022e+11    2009.000000       6.000000      15.000000
75%    2.014081e+11    2014.000000       9.000000      23.000000
max    2.017123e+11    2017.000000      12.000000      31.000000

            extended        country         region       latitude  \
count  181691.000000  181691.000000  181691.000000  181691.000000
mean        0.045346     131.968501       7.160938      23.698173
std         0.208063     112.414535       2.933408      18.377236
min         0.000000       4.000000       1.000000     -53.154613
25%         0.000000      78.000000       5.000000      11.849620
50%         0.000000      98.000000       6.000000      31.467463
75%         0.000000     160.000000      10.000000      34.538561
max         1.000000    1004.000000      12.000000      74.633553
```

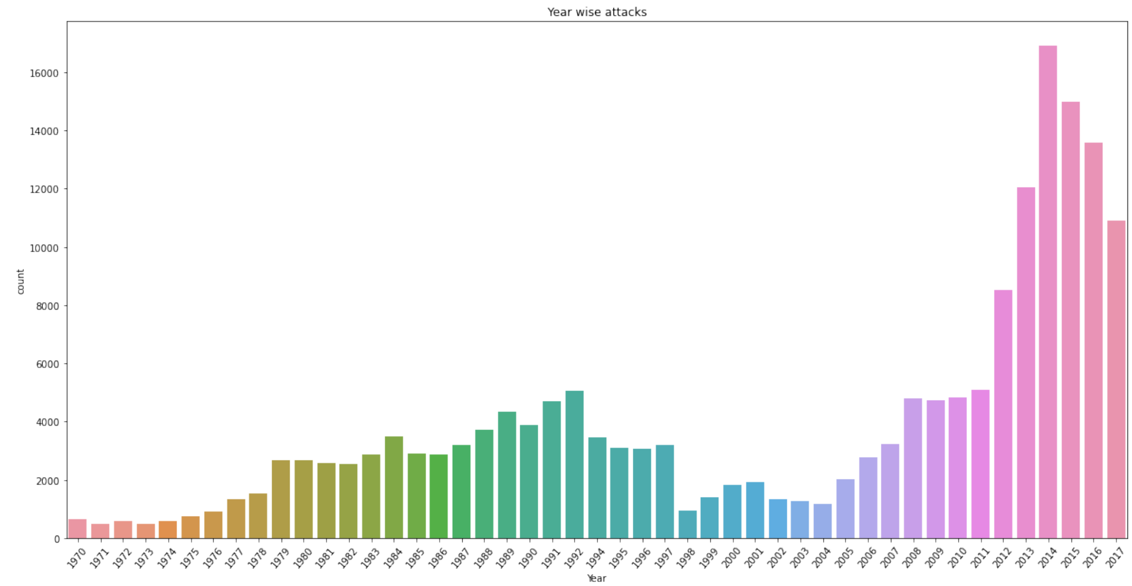
```
 45  nkillus            181691 non-null  float64
 46  nkillter           181691 non-null  float64
 47  Wounded            181691 non-null  float64
 48  nwoundus           181691 non-null  float64
 49  nwoundte           181691 non-null  float64
 50  property           181691 non-null  int64
 51  ishostkid          181691 non-null  float64
 52  scite1             115500 non-null  object
 53  dbsource           181691 non-null  object
 54  INT_LOG            181691 non-null  int64
 55  INT_IDEO           181691 non-null  int64
 56  INT_MISC           181691 non-null  int64
 57  INT_ANY            181691 non-null  int64
dtypes: float64(19), int64(22), object(17)
memory usage: 80.4+ MB
```

[34]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')#it was been imported from starting phase
plt.figure(figsize=(20,10))
sns.countplot(df['Year']).set_title('Year wise attacks')
plt.xticks(rotation=50)

#alternative:
#Barplot
#import seaborn as sns

#x_year = terror_df['Year'].unique()
#y_year = terror_df['Year'].value_counts(dropna=False).sort_index()
#plt.figure(figsize=(15,10))
#plt.title("Attack in Years")
#plt.xlabel("Attack Years")
#plt.ylabel("Number of attacks each year")
#plt.xticks(rotation=45)
#sns.barplot(x=x_year, y=y_year, palette= 'rocket')
#plt.show()
```

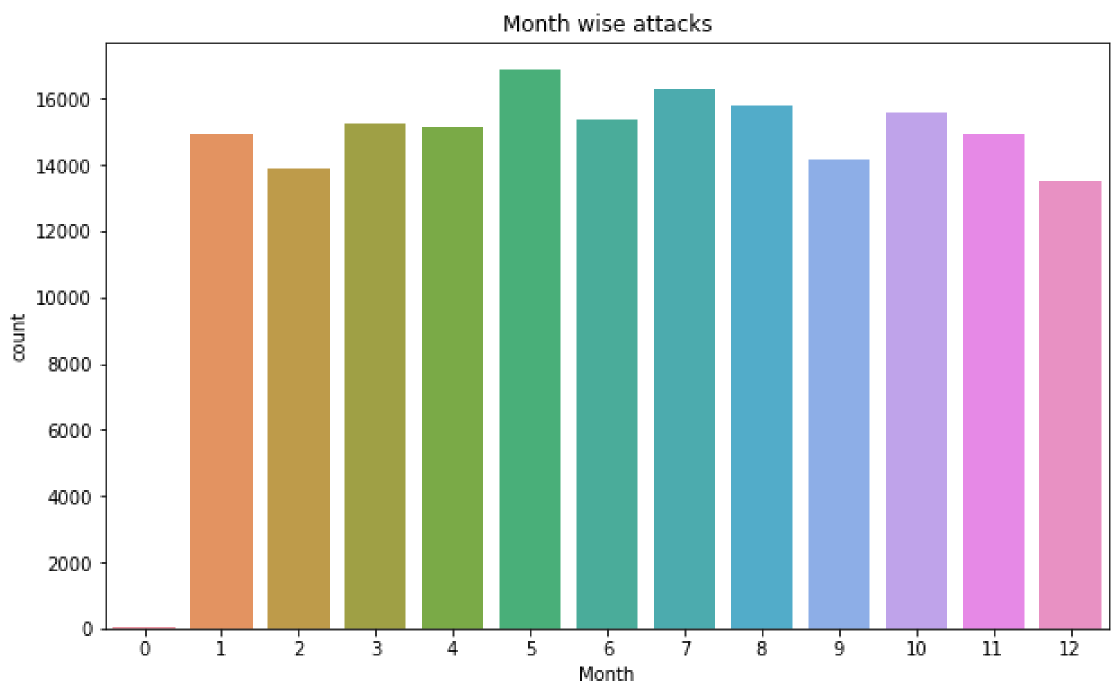
[34]:

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46]),
 [Text(0, 0, '1970'),
  Text(1, 0, '1971'),
  Text(2, 0, '1972'),
  Text(3, 0, '1973'),
  Text(4, 0, '1974'),
  Text(5, 0, '1975'),
```



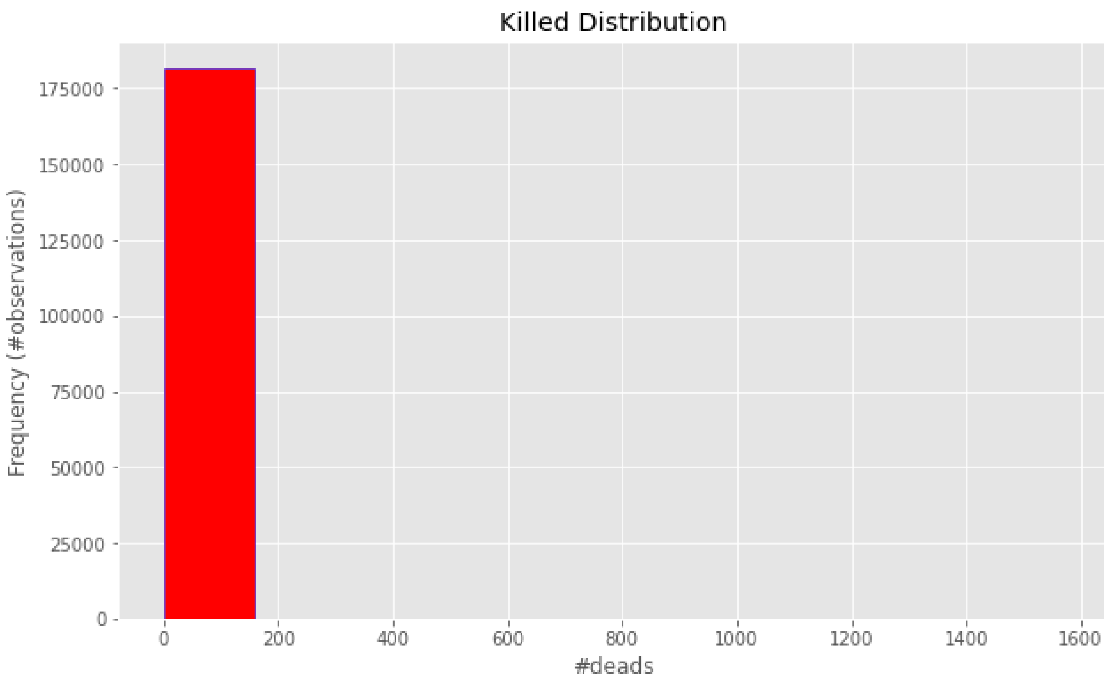

```
[35]: #concludes that the maximum number of attacks per year coincides with the year␣
      ↪2014, in general and without distinction of country

      plt.figure(figsize=(10,6))
      sns.countplot(df['Month']).set_title('Month wise attacks')
```

```
[35]: Text(0.5, 1.0, 'Month wise attacks')
```



[36]:
```
df_Y_M = df[['Month','Year','Killed']]#select only 3 cols
df_2014 = df[df.Year == '2014']#filtering only 2014
plt.figure(figsize=(10,6))
plt.style.use('ggplot')
df.Killed.plot(kind='hist', color='red', edgecolor='blue')
plt.title('Killed Distribution')
plt.xlabel('#deads')
plt.ylabel('Frequency (#observations)')

#sns.countplot(df_2014['Killed']).set_title('Month wise attacks')
```

[36]:
```
Text(0, 0.5, 'Frequency (#observations)')
```



[37]:
```
#df.Related lost at cleaning phase as higher 50% missing values
#Square brackets can be used to access the content of a Serie and a DataFrame
print('Maximum people killed in an attack are',df['Killed'].max(),'that took␣
 ↪place in',df.loc[df['Killed'].idxmax()].Country)
print('Country with most attacks: ',df['Country'].value_counts().idxmax())
print('City with most attacks: ',df['city'].value_counts().index[1])
print("Region with the most attacks:",df['Region'].value_counts().idxmax())
print("Year with the most attacks:",df['Year'].value_counts().idxmax())
print("Month with the most attacks:",df['Month'].value_counts().idxmax())
```

```python
k=df['Month'].value_counts().idxmax()
if k==1:
    print('Month with the most attacks:January')
elif k==2:
    print('Month with the most attacks:February')
elif k==3:
    print('Month with the most attacks:March')
elif k==4:
    print('Month with the most attacks:April')
elif k==5:
    print('Month with the most attacks:May')
elif k==6:
    print('Month with the most attacks:June')
elif k==7:
    print('Month with the most attacks:July')
elif k==8:
    print('Month with the most attacks:August')
elif k==9:
    print('Month with the most attacks:September')
elif k==10:
    print('Month with the most attacks:October')
elif k==11:
    print('Month with the most attacks:November')
elif k==12:
    print('Month with the most attacks:December')
print("Group with the most attacks:",df['Group'].value_counts().index[1])
print("Most Attack Types:",df['AttackType'].value_counts().idxmax())
```

```
Maximum people killed in an attack are 1570.0 that took place in Iraq
Country with most attacks:  Iraq
City with most attacks:  Baghdad
Region with the most attacks: Middle East & North Africa
Year with the most attacks: 2014
Month with the most attacks: 5
Month with the most attacks:May
Group with the most attacks: Taliban
Most Attack Types: Bombing/Explosion
```

[38]:

```python
df['casualities']=df['Killed']+df['Wounded']
#DURING INITIAL PHASE IN ANOTHER FILE.PY :
#df.Wounded.count()
#165380
#df.Killed.count()
#171378
#residue=df.Killed.count() - df.Wounded.count()
#residue
```

```
#5998--->3.4998 % di Killati ---> come fanno ad essere morte più persone di␣
↪quelle ferite ? A quel punto considero il numero delle vittime direttamente␣
↪come la somma e sono sicuro'
```

[39]:
```
df.Country.count()
```

[39]:
```
181691
```

[40]:
```
df.groupby('Country')['Country'].nunique()
```

[40]:
```
Country
Afghanistan    1
Albania        1
Algeria        1
Andorra        1
Angola         1
              ..
Yemen          1
Yugoslavia     1
Zaire          1
Zambia         1
Zimbabwe       1
Name: Country, Length: 205, dtype: int64
```

[41]:
```
df['Year'].value_counts(dropna=False).sort_index() #there is the confirm for␣
↪2014
```

[41]:
```
1970     651
1971     471
1972     568
1973     473
1974     581
1975     740
1976     923
1977    1319
1978    1526
1979    2662
1980    2662
1981    2586
1982    2544
1983    2870
1984    3495
1985    2915
1986    2860
1987    3183
1988    3721
1989    4324
```

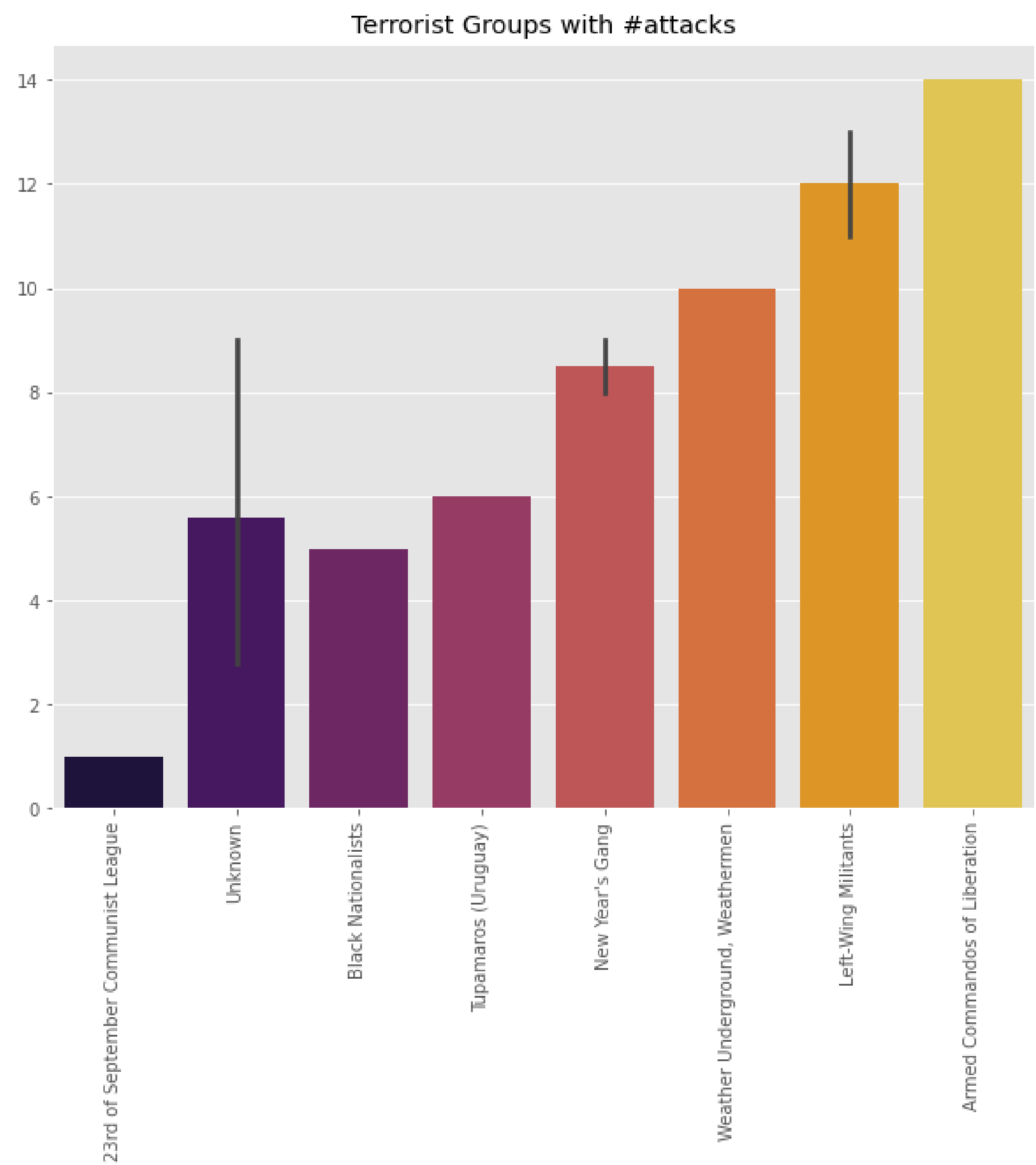
```
1990     3887
1991     4683
1992     5071
1994     3456
1995     3081
1996     3058
1997     3197
1998      934
1999     1395
2000     1814
2001     1906
2002     1333
2003     1278
2004     1166
2005     2017
2006     2758
2007     3242
2008     4805
2009     4721
2010     4826
2011     5076
2012     8522
2013    12036
2014    16903
2015    14965
2016    13587
2017    10900
Name: Year, dtype: int64
```

[42]:
```python
df.Group
```

[42]:
```
0                                         MANO-D
1                 23rd of September Communist League
2                                        Unknown
3                                        Unknown
4                                        Unknown
                           …
181686                                Al-Shabaab
181687                        Muslim extremists
181688    Bangsamoro Islamic Freedom Movement (BIFM)
181689                                   Unknown
181690                                   Unknown
Name: Group, Length: 181691, dtype: object
```

[43]:
```python
#plotting of the different groups of terrorists
sns.barplot(df['Group'][1:15].values,df['Group'][1:15].
 ↪index,palette=('inferno'))
```

```
plt.xticks(rotation=90)
fig=plt.gcf()
fig.set_size_inches(10,8)#1inch=2.54cm
plt.title('Terrorist Groups with #attacks')
plt.savefig('C:\\esercitazione\\notebooks\\Materiale_Progetto_CNDA\\attacks.
↪png')
plt.show()
```



[44]:
```
df.index
```

[44]: RangeIndex(start=0, stop=181691, step=1)

[45]:
```
df.country
```

[45]:
```
0          58
1         130
2         160
3          78
4         101
         …
181686    182
181687    200
181688    160
181689     92
181690    160
Name: country, Length: 181691, dtype: int64
```

[46]:
```
df.Country#previous "country_txt"
```

[46]:
```
0         Dominican Republic
1                     Mexico
2                Philippines
3                     Greece
4                      Japan
                 …
181686               Somalia
181687                 Syria
181688           Philippines
181689                 India
181690           Philippines
Name: Country, Length: 181691, dtype: object
```

[47]:
```
country_wise=df['Country'].value_counts().reset_index()
country_wise.rename(columns={"index":'Country Name'},inplace=True)
country_wise
```

[47]:
```
            Country Name  Country
0                   Iraq    24636
1               Pakistan    14368
2            Afghanistan    12731
3                  India    11960
4               Colombia     8306
..                   …        …
200        International        1
201    Wallis and Futuna        1
202        South Vietnam        1
203              Andorra        1
```

```
204   Antigua and Barbuda        1

[205 rows x 2 columns]
```

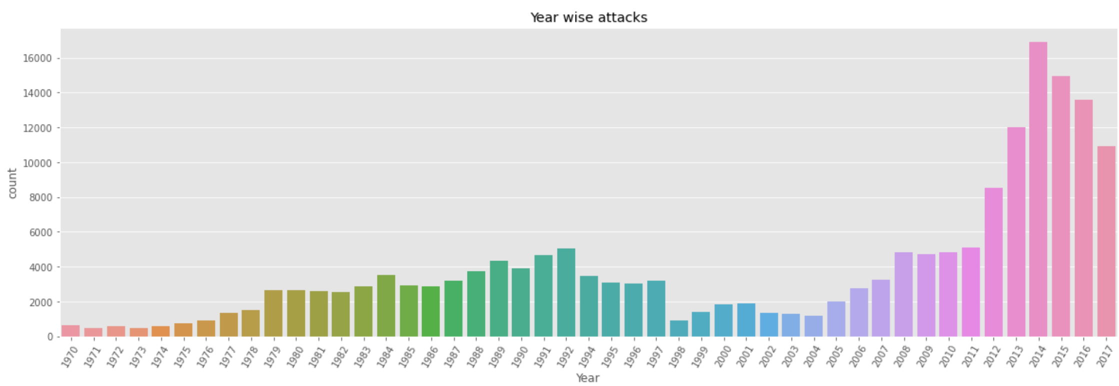
[48]:
```
df.Country.count()
```

[48]:
```
181691
```

[49]:
```
plt.figure(figsize=(20,6))
sns.countplot(df['Year']).set_title('Year wise attacks')
plt.xticks(rotation=60)
```

[49]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46]),
 [Text(0, 0, '1970'),
  Text(1, 0, '1971'),
  Text(2, 0, '1972'),
  Text(3, 0, '1973'),
  Text(4, 0, '1974'),
  Text(5, 0, '1975'),
  Text(6, 0, '1976'),
  Text(7, 0, '1977'),
  Text(8, 0, '1978'),
  Text(9, 0, '1979'),
  Text(10, 0, '1980'),
  Text(11, 0, '1981'),
  Text(12, 0, '1982'),
  Text(13, 0, '1983'),
  Text(14, 0, '1984'),
  Text(15, 0, '1985'),
  Text(16, 0, '1986'),
  Text(17, 0, '1987'),
  Text(18, 0, '1988'),
  Text(19, 0, '1989'),
  Text(20, 0, '1990'),
  Text(21, 0, '1991'),
  Text(22, 0, '1992'),
  Text(23, 0, '1994'),
  Text(24, 0, '1995'),
  Text(25, 0, '1996'),
  Text(26, 0, '1997'),
  Text(27, 0, '1998'),
  Text(28, 0, '1999'),
  Text(29, 0, '2000'),
  Text(30, 0, '2001'),
  Text(31, 0, '2002'),
```

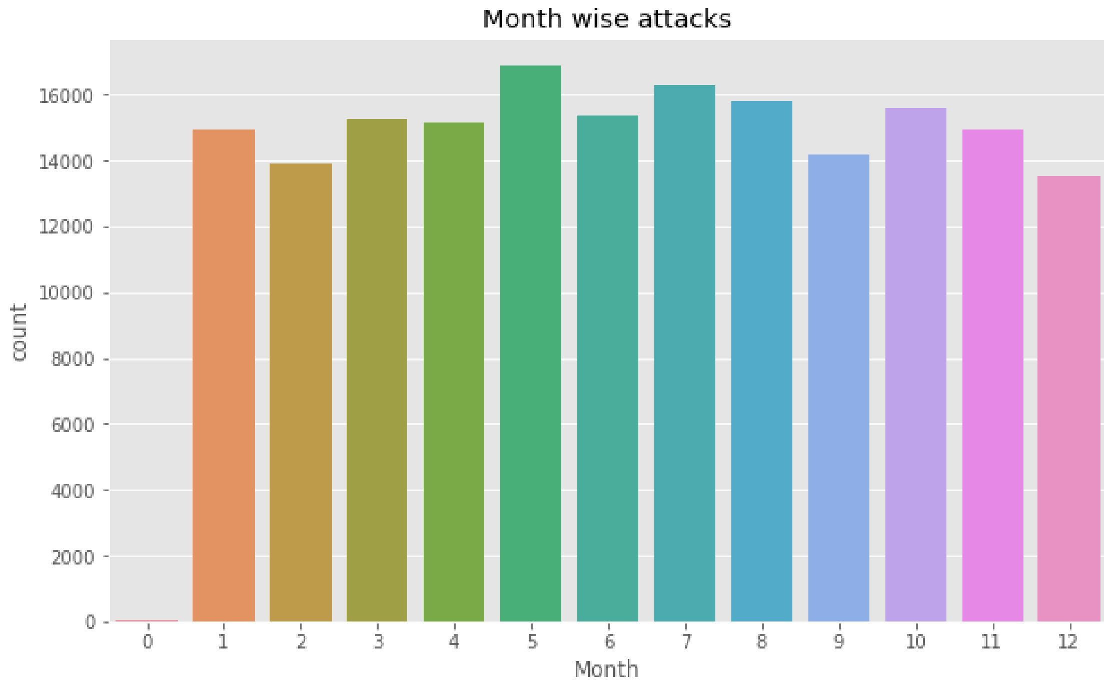
```
Text(32, 0, '2003'),
Text(33, 0, '2004'),
Text(34, 0, '2005'),
Text(35, 0, '2006'),
Text(36, 0, '2007'),
Text(37, 0, '2008'),
Text(38, 0, '2009'),
Text(39, 0, '2010'),
Text(40, 0, '2011'),
Text(41, 0, '2012'),
Text(42, 0, '2013'),
Text(43, 0, '2014'),
Text(44, 0, '2015'),
Text(45, 0, '2016'),
Text(46, 0, '2017')])
```



OFFICIAL VERSION

[50]:
```
plt.figure(figsize=(10,6))
sns.countplot(df['Month']).set_title('Month wise attacks')
```

[50]:
```
Text(0.5, 1.0, 'Month wise attacks')
```



[51]:
```
#checked : found an inconsistent fact: #months=13? No..
#Month can't be zeros, dropping those zeros
df[df['Month']==0]#I want to show what I want to detect and delete
```

[51]:

	eventid	Year	Month	Day	extended	country	Country	region	\
1	197000000002	1970	0	0	0	130	Mexico	1	
1123	197200000002	1972	0	0	0	160	Philippines	5	
1690	197300000001	1973	0	0	1	45	Colombia	3	
2164	197400000002	1974	0	0	0	69	France	8	
2165	197400000003	1974	0	0	0	98	Italy	8	
2744	197500000001	1975	0	0	0	153	Pakistan	6	
3484	197600000001	1976	0	0	0	209	Turkey	10	
3485	197600000002	1976	0	0	0	209	Turkey	10	
4407	197700000001	1977	0	0	0	101	Japan	4	
4408	197700000002	1977	0	0	0	101	Japan	4	
4409	197700000003	1977	0	0	0	101	Japan	4	
4410	197700000004	1977	0	0	0	69	France	8	
4411	197700000005	1977	0	0	0	69	France	8	
5726	197800000001	1978	0	0	0	30	Brazil	3	
5727	197800000002	1978	0	0	0	61	El Salvador	2	
7252	197900000001	1979	0	0	0	101	Japan	4	
7253	197900000002	1979	0	0	0	45	Colombia	3	
7254	197900000003	1979	0	0	0	160	Philippines	5	
15163	198200000001	1982	0	0	0	38	Canada	1	
26987	198600000001	1986	0	0	0	186	Sri Lanka	6	

```
                            Region             provstate           city  \
1                    North America               Federal    Mexico city
1123                Southeast Asia                 Capiz          Roxas
1690                 South America               Unknown        unknown
2164                Western Europe                 Paris          Paris
2165                Western Europe                 Lazio           Rome
2744                    South Asia                Punjab     Rawalpindi
3484    Middle East & North Africa              Istanbul       Istanbul
3485    Middle East & North Africa                Ankara         Ankara
4407                     East Asia                 Tokyo          Tokyo
4408                     East Asia                 Tokyo          Tokyo
4409                     East Asia                 Tokyo          Tokyo
4410                Western Europe  Pyrenees-Atlantiques        Bayonne
4411                Western Europe  Pyrenees-Atlantiques        Bayonne
5726                 South America     Rio Grande do Sul   Porto Alegre
5727   Central America & Caribbean          San Salvador   San Salvador
7252                     East Asia               Unknown        Unknown
7253                 South America                Bogota         Bogota
7254                Southeast Asia               Unknown        Unknown
15163                North America               Ontario        Toronto
26987                   South Asia               Unknown        Unknown

        latitude   longitude  specificity  vicinity Summary  crit1  crit2  \
1      19.371887  -99.086624          1.0         0     NaN      1      1
1123   11.586558  122.753716          1.0         0     NaN      1      1
1690   31.467463   43.246506          5.0         0     NaN      1      1
2164   48.856644    2.342330          1.0         0     NaN      1      1
2165   41.890961   12.490069          1.0         0     NaN      1      1
2744   33.594013   73.069077          1.0         0     NaN      1      1
3484   41.106178   28.689863          1.0         0     NaN      1      1
3485   39.930771   32.767540          1.0         0     NaN      1      1
4407   35.689125  139.747742          1.0         0     NaN      1      1
4408   35.689125  139.747742          1.0         0     NaN      1      1
4409   35.689125  139.747742          1.0         0     NaN      1      1
4410   43.492949   -1.474841          1.0         0     NaN      1      1
4411   43.492949   -1.474841          1.0         0     NaN      1      1
5726  -30.034108  -51.217839          1.0         0     NaN      1      1
5727   13.692880  -89.199161          1.0         0     NaN      1      1
7252   31.467463   43.246506          5.0         0     NaN      1      1
7253    4.667128  -74.106056          1.0         0     NaN      1      1
7254   31.467463   43.246506          5.0         0     NaN      1      1
15163  43.666667  -79.416667          1.0         0     NaN      1      1
26987  31.467463   43.246506          5.0         0     NaN      1      1

       crit3  doubtterr  multiple  success  suicide  attacktype1  \
1          1        0.0       0.0        1        0            6
```

1123	1	0.0	0.0	1	0	3
1690	1	0.0	0.0	1	0	6
2164	1	-9.0	0.0	0	0	3
2165	1	0.0	0.0	1	0	3
2744	1	0.0	0.0	1	0	3
3484	0	1.0	0.0	1	0	9
3485	0	1.0	0.0	1	0	9
4407	1	0.0	0.0	1	0	3
4408	1	0.0	0.0	1	0	3
4409	1	0.0	0.0	1	0	3
4410	1	0.0	0.0	1	0	3
4411	1	0.0	0.0	1	0	3
5726	1	-9.0	0.0	1	0	6
5727	1	-9.0	0.0	1	0	6
7252	1	-9.0	0.0	1	0	9
7253	1	0.0	0.0	1	0	9
7254	1	0.0	0.0	1	0	9
15163	1	0.0	0.0	1	0	3
26987	1	-9.0	0.0	1	0	6

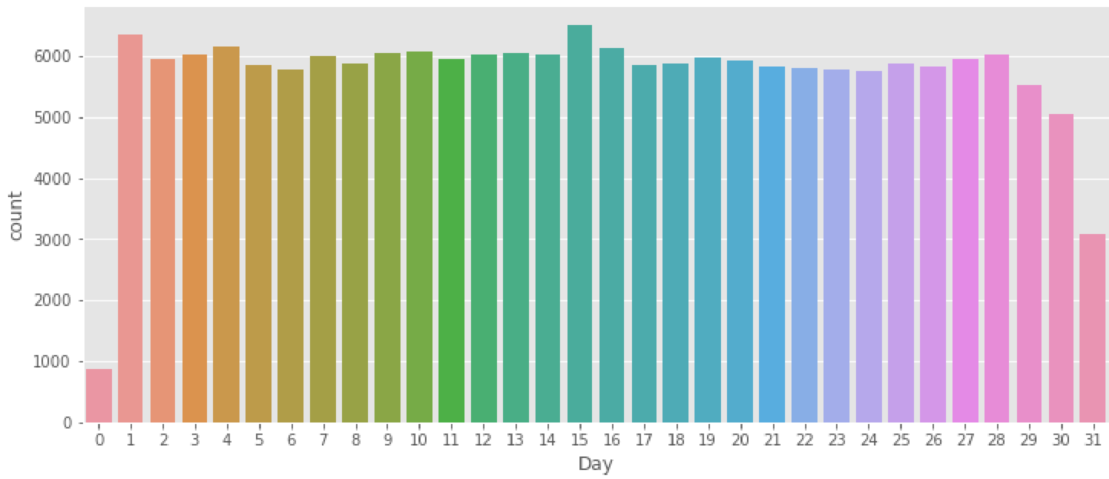
	AttackType	targtype1	Target_type \
1	Hostage Taking (Kidnapping)	7	Government (Diplomatic)
1123	Bombing/Explosion	6	Airports & Aircraft
1690	Hostage Taking (Kidnapping)	1	Business
2164	Bombing/Explosion	1	Business
2165	Bombing/Explosion	6	Airports & Aircraft
2744	Bombing/Explosion	6	Airports & Aircraft
3484	Unknown	4	Military
3485	Unknown	4	Military
4407	Bombing/Explosion	8	Educational Institution
4408	Bombing/Explosion	1	Business
4409	Bombing/Explosion	8	Educational Institution
4410	Bombing/Explosion	10	Journalists & Media
4411	Bombing/Explosion	1	Business
5726	Hostage Taking (Kidnapping)	14	Private Citizens & Property
5727	Hostage Taking (Kidnapping)	1	Business
7252	Unknown	14	Private Citizens & Property
7253	Unknown	10	Journalists & Media
7254	Unknown	14	Private Citizens & Property
15163	Bombing/Explosion	7	Government (Diplomatic)
26987	Hostage Taking (Kidnapping)	14	Private Citizens & Property

	targsubtype1	targsubtype1_txt	… \
1	45.0	Diplomatic Personnel (outside of embassy, cons…	…
1123	42.0	Aircraft (not at an airport)	…
1690	9.0	Farm/Ranch	…
2164	3.0	Bank/Commerce	…

```
[0 rows x 59 columns]
```

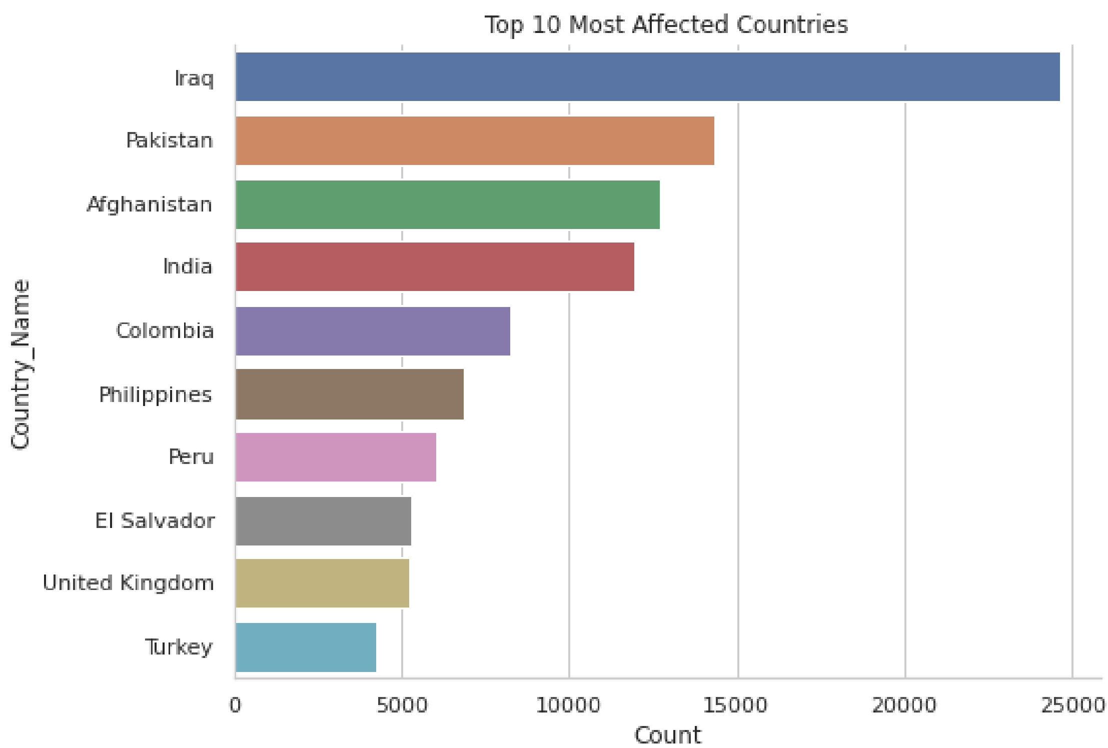
[54]:
```python
plt.figure(figsize=(12,5))
sns.countplot(df['Day'])
```

[54]:
```
<AxesSubplot:xlabel='Day', ylabel='count'>
```



[55]:
```python
#same problem of the months: the nominal number, without exceptions, of days is
↪"30" or "31" but there is not any month with 32 days
df[df['Day']==0]
```

[55]:
```
             eventid  Year  Month  Day  extended  country  \
2       197001000001  1970      1    0         0      160
3       197001000002  1970      1    0         0       78
4       197001000003  1970      1    0         0      101
96      197003000001  1970      3    0         0      160
165     197004000001  1970      4    0         1       65
…                …     …      …    …         …        …
104603  201112170006  2011     12    0         0      155
104611  201112170021  2011     12    0         0      153
104612  201112170022  2011     12    0         0      153
104613  201112170024  2011     12    0         0      153
104684  201112220039  2011     12    0         0      153

             Country  region              Region  \
2        Philippines       5      Southeast Asia
3             Greece       8      Western Europe
4              Japan       4           East Asia
96       Philippines       5      Southeast Asia
165         Ethiopia      11  Sub-Saharan Africa
```



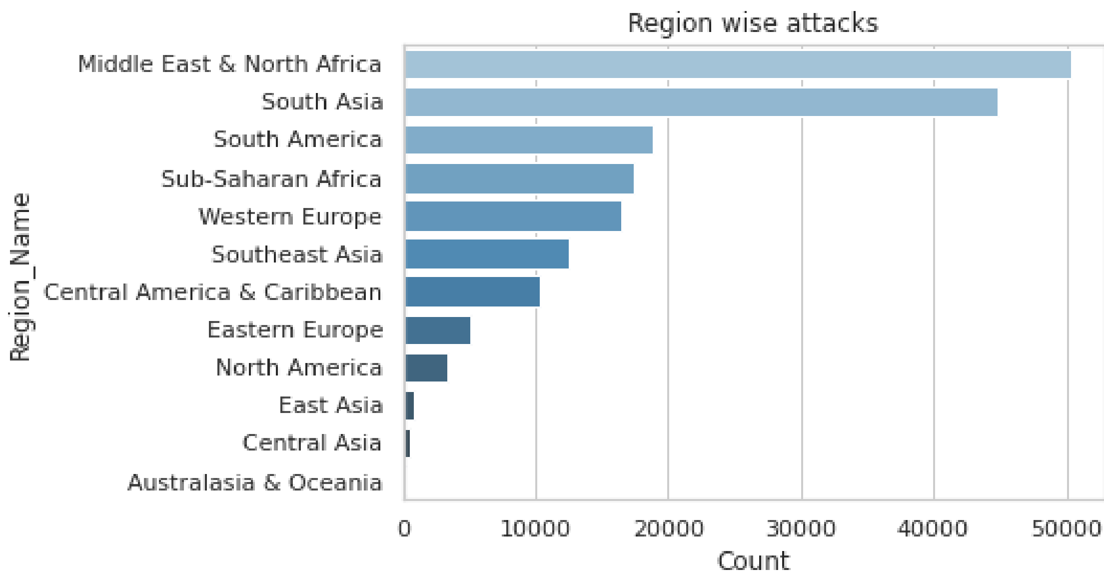
[66]:
```
#nel grafico tutto blu stavo riadattando del codice usato durante la Basic␣
↪Statistical Description fatta a lezione
#comunque non mi serve perché non c'è corrispondenza con i luoghi quindi␣
↪comunica poco..
```

[67]:
```
region_wise=df['Region'].value_counts().reset_index()
region_wise.rename(columns={"index":'Region_Name','Region':
↪'Count'},inplace=True)
region_wise
```

[67]:

	Region_Name	Count
0	Middle East & North Africa	50317
1	South Asia	44866
2	South America	18838
3	Sub-Saharan Africa	17450
4	Western Europe	16450
5	Southeast Asia	12438
6	Central America & Caribbean	10260
7	Eastern Europe	5136
8	North America	3416
9	East Asia	790
10	Central Asia	562
11	Australasia & Oceania	277

[68]:
```
ax = sns.barplot(x="Count", y="Region_Name", data=region_wise,
                 palette="Blues_d").set_title('Region wise attacks')
```



[69]:
```
#Middle East & North Africe are most Affected regions
```

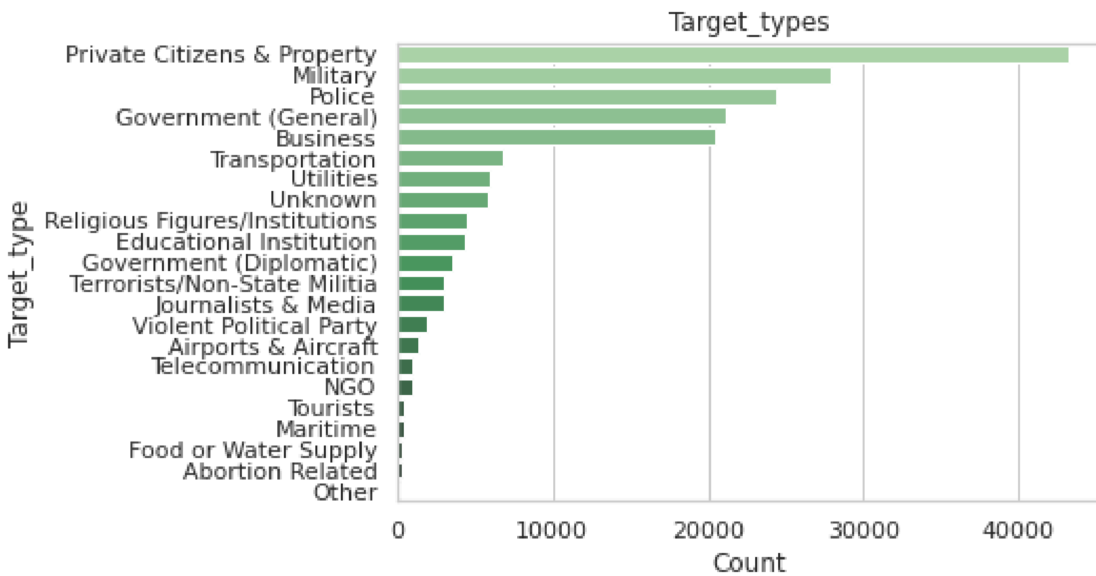
[70]:
```
city_wise=df['city'].value_counts().reset_index()
city_wise.rename(columns={"index":'City_Name','city':'Count'},inplace=True)
city_wise
```

[70]:

	City_Name	Count
0	Unknown	10062
1	Baghdad	7582
2	Karachi	2647
3	Lima	2356
4	Mosul	2263
…	…	…
36545	H'doura	1
36546	Tamarasheni	1
36547	Kororamae	1
36548	Kitgum Matidi	1
36549	Kubentog	1

[36550 rows x 2 columns]

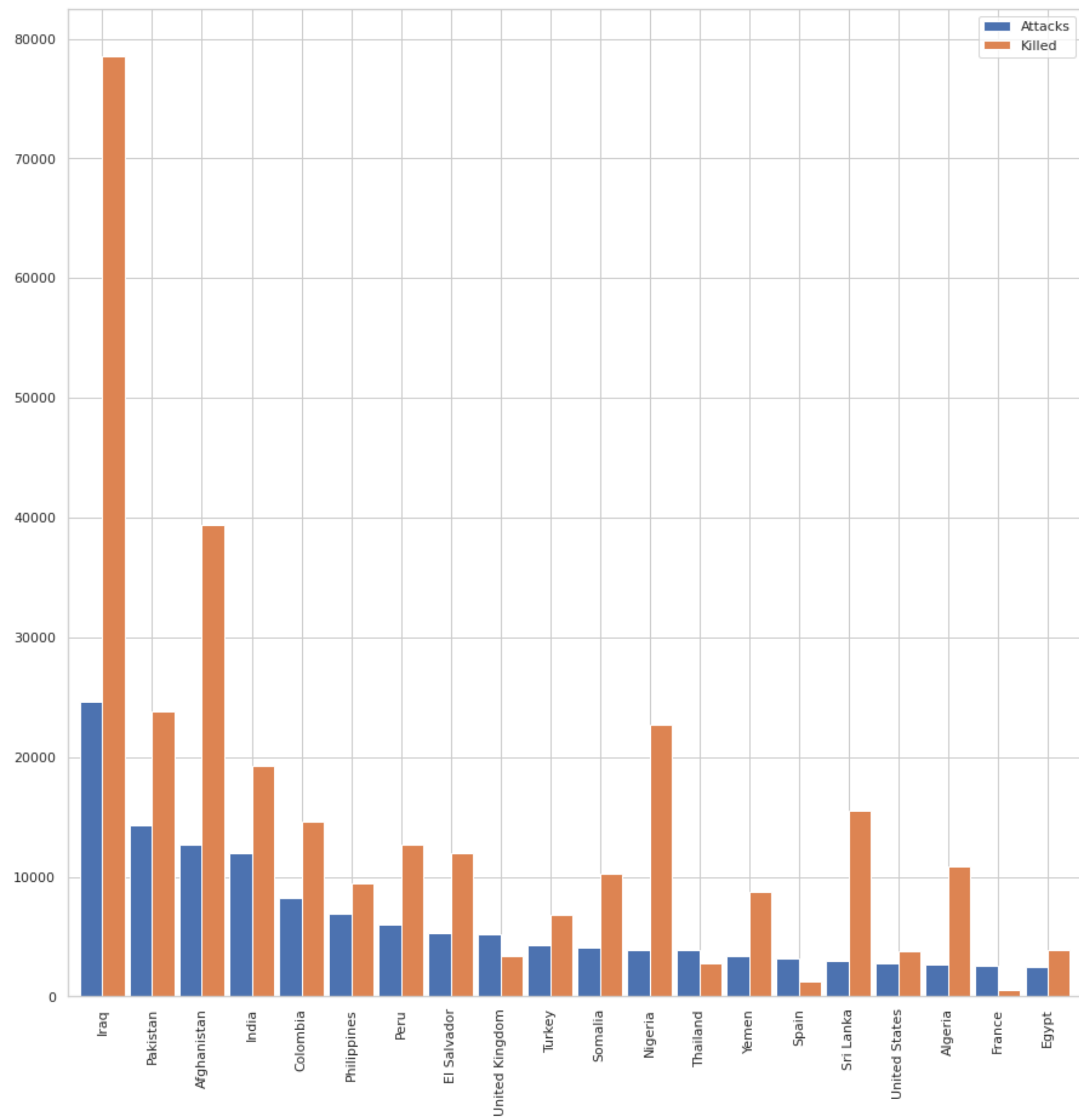
[71]:
```
ax = sns.barplot(x="Count", y="City_Name", data=city_wise[:
↪10],palette="Reds_d").set_title('Top 10 Attacked cities')
```



[80]:
```
nationality_type=df['natlty1_txt'].value_counts().reset_index()
nationality_type.rename(columns={"index":'Target_Nationality','natlty1_txt':
↪'Count'},inplace=True)
nationality_type
```

[80]:
```
                      Target_Nationality  Count
0                                   Iraq  25638
1                               Pakistan  13861
2                                  India  12069
3                            Afghanistan  10918
4                               Colombia   7860
..                                   ...    ...
210                            St. Lucia      1
211                            Greenland      1
212                  Antigua and Barbuda      1
213  Commonwealth of Independent States      1
214                     Marshall Islands      1

[215 rows x 2 columns]
```

[81]:
```
ax = sns.barplot(x="Count", y="Target_Nationality", data=nationality_type[:
↪10],palette="afmhot").set_title('Top 10 nationals targeted in these attacks')
```



[89]:
```
country_terrorism_percentages = df['Country'].value_counts(normalize=True)[:20].
    ↪to_frame()
```

[90]:
```
country_terrorism_percentages
```

[90]:

	Country
Iraq	0.136150
Pakistan	0.079264
Afghanistan	0.070343
India	0.065990
Colombia	0.045531
Philippines	0.038020

[93]:
```
                Killed
Country
Afghanistan   39374.0
Albania          42.0
Algeria       10846.0
Andorra           0.0
Angola         3003.0
…                 …
Yemen          8775.0
Yugoslavia      119.0
Zaire           316.0
Zambia           70.0
Zimbabwe        153.0

[205 rows x 1 columns]
```

[94]:
```
#nell'analisi sono stati trattati:
#top 10 most Affected Countries --> region, prov, city (ideas for other␣
 ↪analysis)
#in realtà province e basta non sono state trattate
#top attack type
#top target type based on Entities (ex: civilians, soldiers)
#top target type based on Context (ex: Airports)
#top 10 nationals targetered
#top group of terrorists
#top weapon
#attack to killed ratio for country
#top Month
#top Years
#top Days

#ora mi concentro sugli aspetti statistici e cerco di ingegnarmi per␣
 ↪correlazioni

df["Killed"].unique().mean() #questa media dovrebbe essere di 159 persone morte␣
 ↪al giorno a città (?)
```

[94]:
```
158.4780487804878
```

[95]:
```
##################################sto a fare qualche prova di layout alternativi
df.guncertain1
```

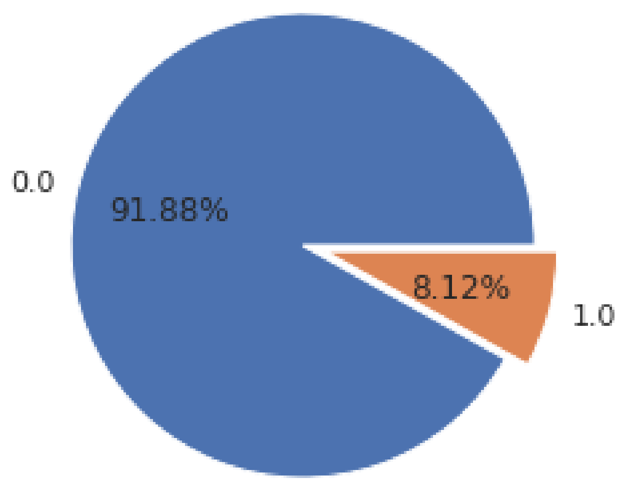
[95]:
```
0          0.0
5          0.0
6          0.0
7          0.0
8          0.0
```

```
          …
181686    0.0
181687    0.0
181688    0.0
181689    0.0
181690    0.0
Name: guncertain1, Length: 180800, dtype: float64
```

[96]:
```
df['guncertain1'].isna().sum()
```

[96]:
```
0
```

[97]:
```
def pie(feature) :
    global df
    plt.pie(df[feature].value_counts(),labels=list(df[feature].value_counts().
 ↪index),
        autopct ='%1.2f%%' , labeldistance = 1.1,explode = [0.05 for i in␣
 ↪range(len(df[feature].value_counts()))] )
    plt.show()


pie('guncertain1')
```



[98]:
```
#This variable indicates whether or not the information reported by sources␣
 ↪about the
#Perpetrator Group Name(s) is based on speculation or dubious claims of␣
 ↪responsibility.
```

```
181686              Somalia     1.0         1
181687                Syria     2.0         1
181688          Philippines     0.0         1

[160817 rows x 3 columns]
```

[129]:
```
Group_success
```

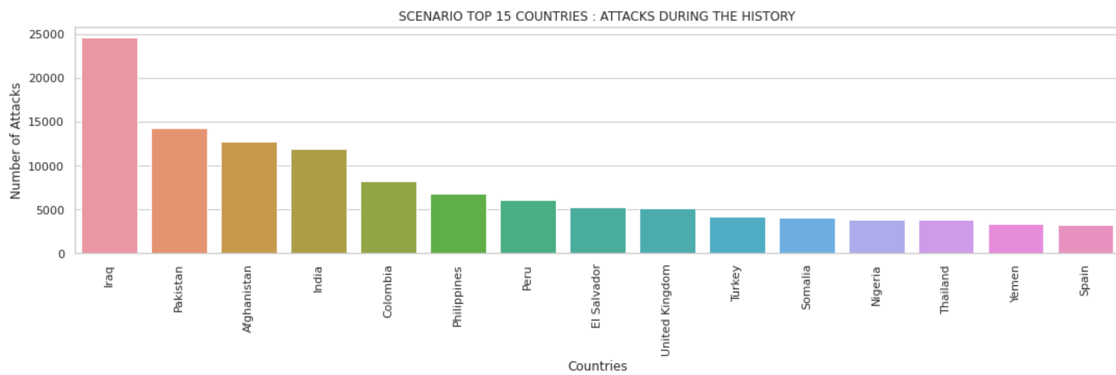
[129]:
```
              Killed  success
Country
Afghanistan  36542.0    11128
Albania         42.0       64
Algeria      10788.0     2531
Andorra          0.0        1
Angola        2965.0      469
…                …       …
Yemen         8399.0     2836
Yugoslavia     114.0      179
Zaire          316.0       44
Zambia          70.0       58
Zimbabwe       151.0       94

[202 rows x 2 columns]
```

[130]:
```
Group_totals
```

[130]:
```
              Killed
Country
Afghanistan  39374.0
Albania         42.0
Algeria      10846.0
Andorra          0.0
Angola        3003.0
…                …
Yemen         8775.0
Yugoslavia     119.0
Zaire          316.0
Zambia          70.0
Zimbabwe       153.0

[205 rows x 1 columns]
```

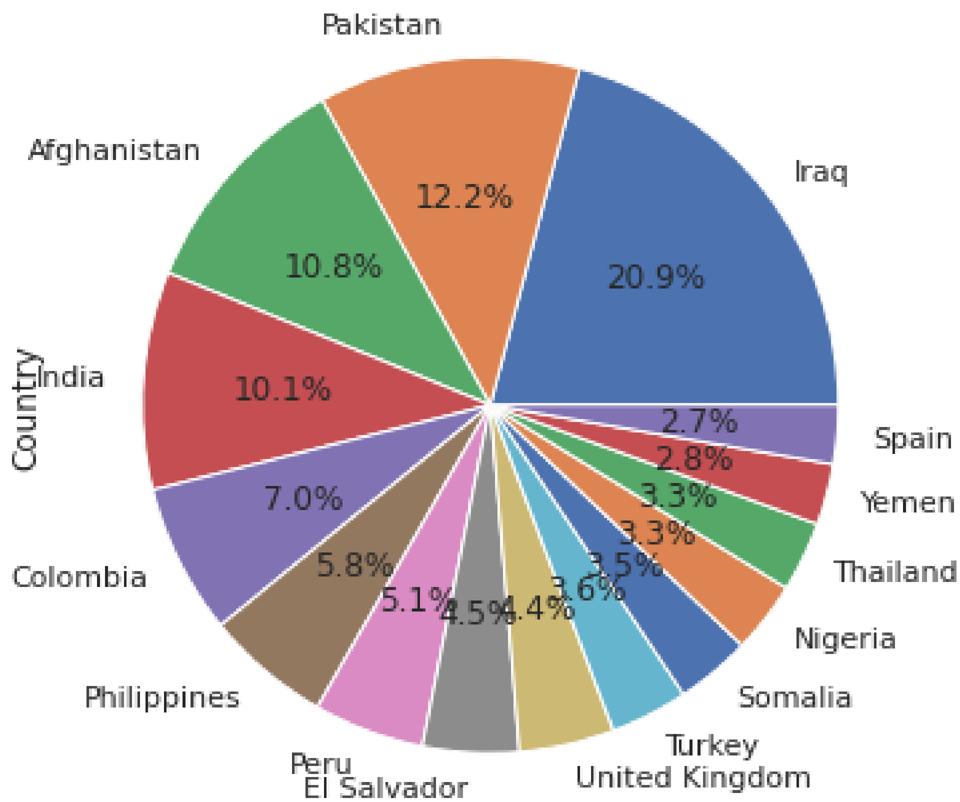
[131]:
```
# Series.value_counts(normalize=False, sort=True, ascending=False, bins=None,␣
↪dropna=True)[source]
#Return a Series containing counts of unique values.
#The resulting object will be in descending order so that the first element is␣
↪the most frequently-occurring element. Excludes NA values by default.
```

```
#Return a Series containing counts of unique values.
#The resulting object will be in descending order so that the first element is␣
↪the most frequently-occurring element. Excludes NA values by default.
plt.subplots(figsize=(18,4))
sns.barplot(df['Country'].value_counts()[:15].index,df['Country'].
↪value_counts()[:15].values)
plt.title('SCENARIO TOP 15 COUNTRIES : ATTACKS DURING THE HISTORY')
plt.xlabel('Countries')
plt.ylabel('Number of Attacks')
plt.xticks(rotation = 90)
plt.show()
```



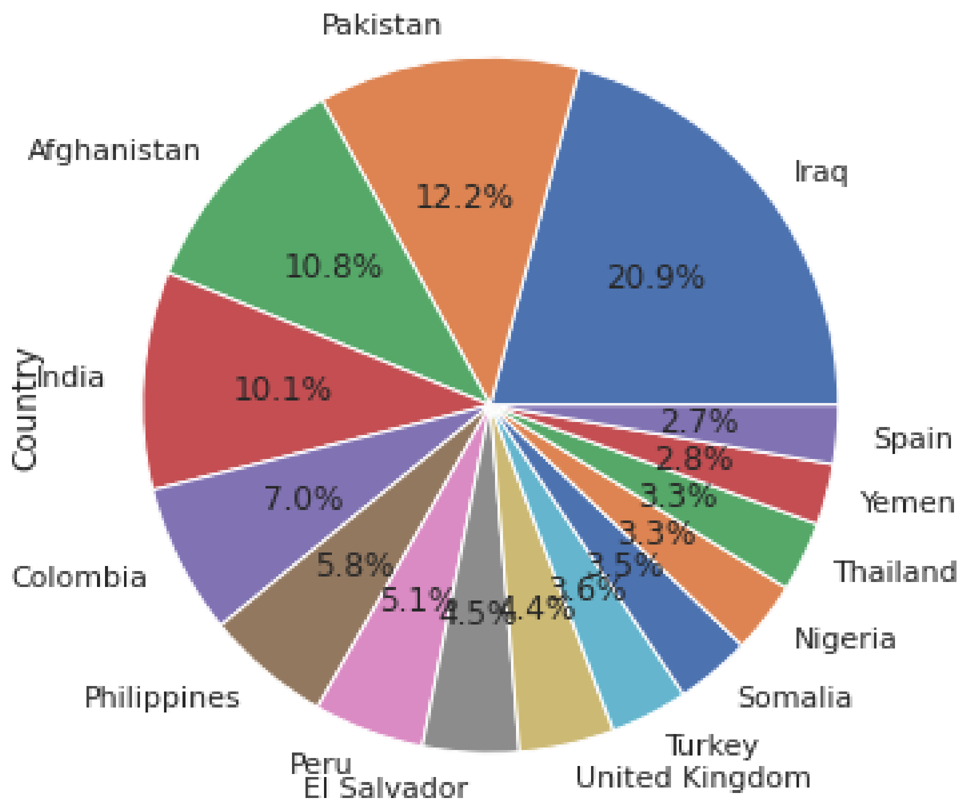
[132]:
```
top15_perc=df['Country'].value_counts()[:15]
top15_perc.dropna()#Remove missing values.
top15_perc.plot(kind='pie',autopct="%1.1f%%",figsize=(6,6))
```

[132]: <AxesSubplot:ylabel='Country'>



[133]:
```
top15_perc=df['Country'].value_counts()[:15]
#top15_perc.dropna()#Remove missing values.
top15_perc.plot(kind='pie',autopct="%1.1f%%",figsize=(6,6))
```

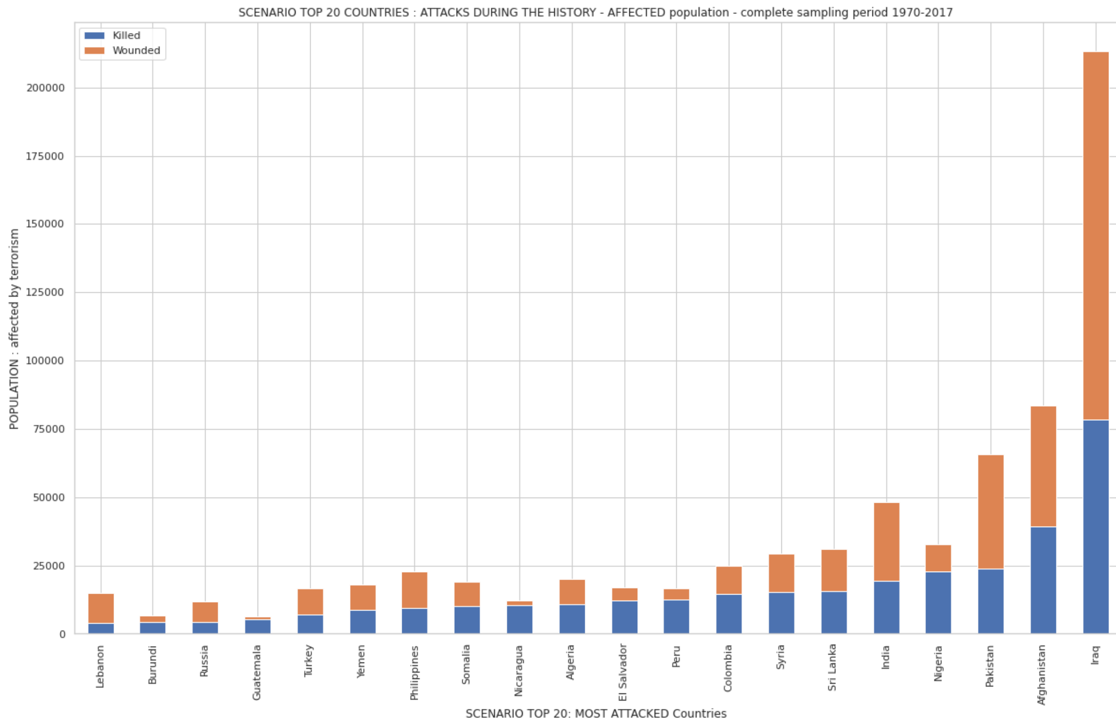
[133]: <AxesSubplot:ylabel='Country'>



[134]:
```
#obv it's equal because Data Preprocessing (Data Cleaning) has happened !
```

[135]:
```
subdataset=df[['Country','Killed','Wounded']]

sub_TOTALS=subdataset.groupby(['Country'])[['Killed','Wounded']].sum()
sub_TOTALS=sub_TOTALS.sort_values('Killed')
top_sub_TOP_TOTALS=sub_TOTALS.tail(20)
top_sub_TOP_TOTALS.plot(kind='bar',stacked=True,figsize=(20,12))
plt.xlabel('SCENARIO TOP 20: MOST ATTACKED Countries ')
plt.ylabel('POPULATION : affected by terrorism')
plt.title('SCENARIO TOP 20 COUNTRIES : ATTACKS DURING THE HISTORY - AFFECTED␣
↪population - complete sampling period 1970-2017 ')
```

[135]:
```
Text(0.5, 1.0, 'SCENARIO TOP 20 COUNTRIES : ATTACKS DURING THE HISTORY -
AFFECTED population - complete sampling period 1970-2017 ')
```

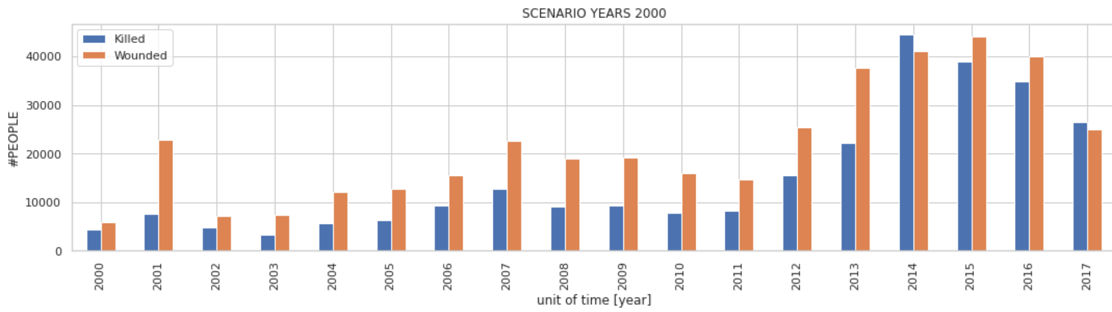


[136]:
```
df.Year.unique()
```

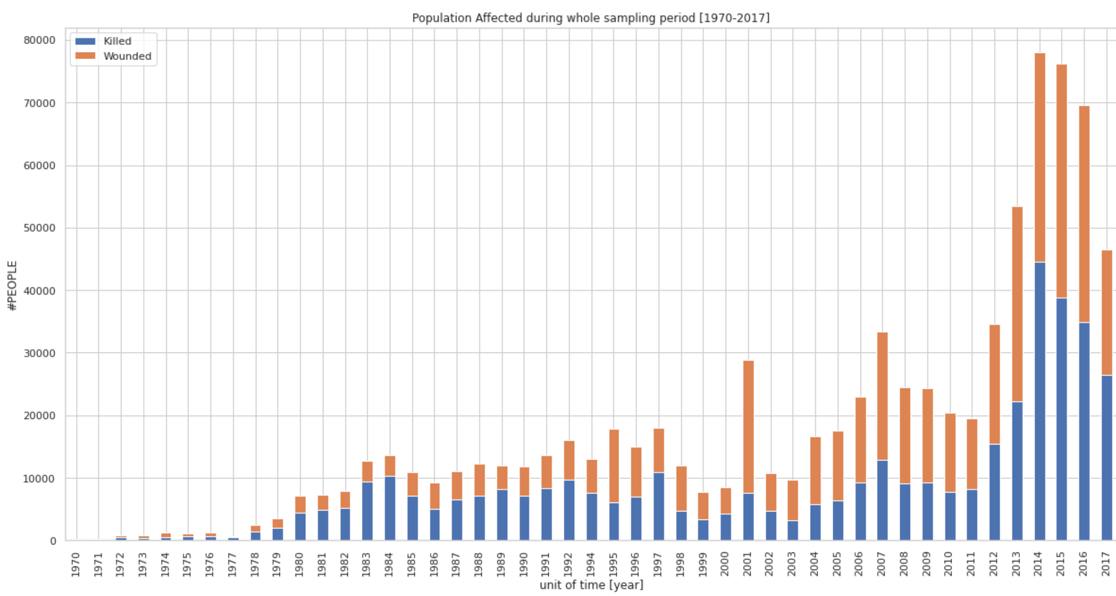
[136]:
```
array([1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980,
       1981, 1986, 1982, 1983, 1984, 1985, 1987, 1988, 1989, 1990, 1991,
       1992, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003,
       2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014,
       2015, 2016, 2017])
```

[137]:
```
Period_2000=df[['Year','Killed','Wounded']]
Period_2000=Period_2000.groupby('Year')[['Killed','Wounded']].sum()
Period_2000=Period_2000.tail(18)
Period_2000.reset_index(inplace=True)
Period_2000.plot(x='Year',y=['Killed','Wounded'],kind='bar',figsize=(18,4))
plt.title('SCENARIO YEARS 2000')
plt.ylabel('#PEOPLE')
plt.xlabel('unit of time [year]')
```

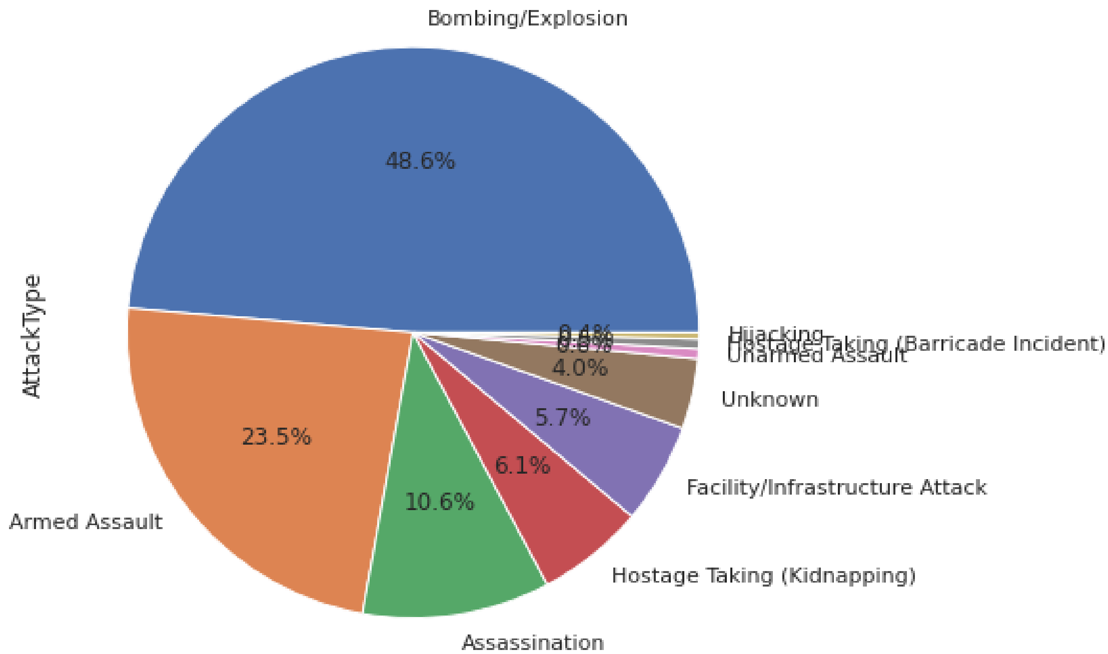
[137]:
```
Text(0.5, 0, 'unit of time [year]')
```



[138]:
```
#Total people Affected from year 1970 to 2017
y1970=df[df['Killed']>0][['Year','Killed','Wounded']]
y1970.dropna()
y1970=y1970.groupby(['Year'])[['Killed','Wounded']].sum()
y1970.plot(kind='bar',stacked=True,figsize=(20,10))
plt.title('Population Affected during whole sampling period [1970-2017]')
plt.ylabel('#PEOPLE')
plt.xlabel('unit of time [year]')
plt.show()
```



[139]:
```
plt.figure(figsize=(8,8))
df['AttackType'].value_counts().plot.pie(autopct="%1.1f%%")
plt.tight_layout()
```



[140]:
```
#df.AttackType.unique()     #-->array(['Assassination', 'Armed Assault',␣
 ↪'Bombing/Explosion',
                            #'Facility/Infrastructure Attack', 'Hijacking',␣
 ↪'Unknown',
                            #'Hostage Taking (Kidnapping)', 'Unarmed Assault',
                            #'Hostage Taking (Barricade Incident)'],␣
 ↪dtype=object)
df_copy=df.copy()
#troiata : NUM_ATTACKS = len(df_copy.AttackType.unique()) #not EQUIVALENT␣
 ↪NUM_ATTACKS=df['Attacktype'].value_counts()
NUM_ATTACKS=df_copy['AttackType'].value_counts()
NUM_ATTACKS=list(NUM_ATTACKS)#casting -------> containing COUNTERS
array=['Assassination', 'Armed Assault', 'Bombing/Explosion','Facility/
 ↪Infrastructure Attack', 'Hijacking', 'Unknown','Hostage Taking␣
 ↪(Kidnapping)', 'Unarmed Assault','Hostage Taking (Barricade Incident)']
dict_new = {'AttackType':array,'Count':NUM_ATTACKS}
#array ----> names , #NUM_ATTACKS ---> counter[j]
#single_dict_new = dict_new[j]={'AttackType'[j]:array[j],'Count'[j]:
 ↪NUM_ATTACKS[j]}
df_copy = pd.DataFrame(dict_new)
fig = px.pie(df_copy, values='Count', names='AttackType', title='100% shared on␣
 ↪different attack')
#fig.update_traces(textposition='inside', textinfo='percent+label')
```

```
fig.show()

########################
```

[141]:
```
df.shape
```

[141]: (180800, 59)

[142]:
```
df_copy.shape
```

[142]: (9, 2)

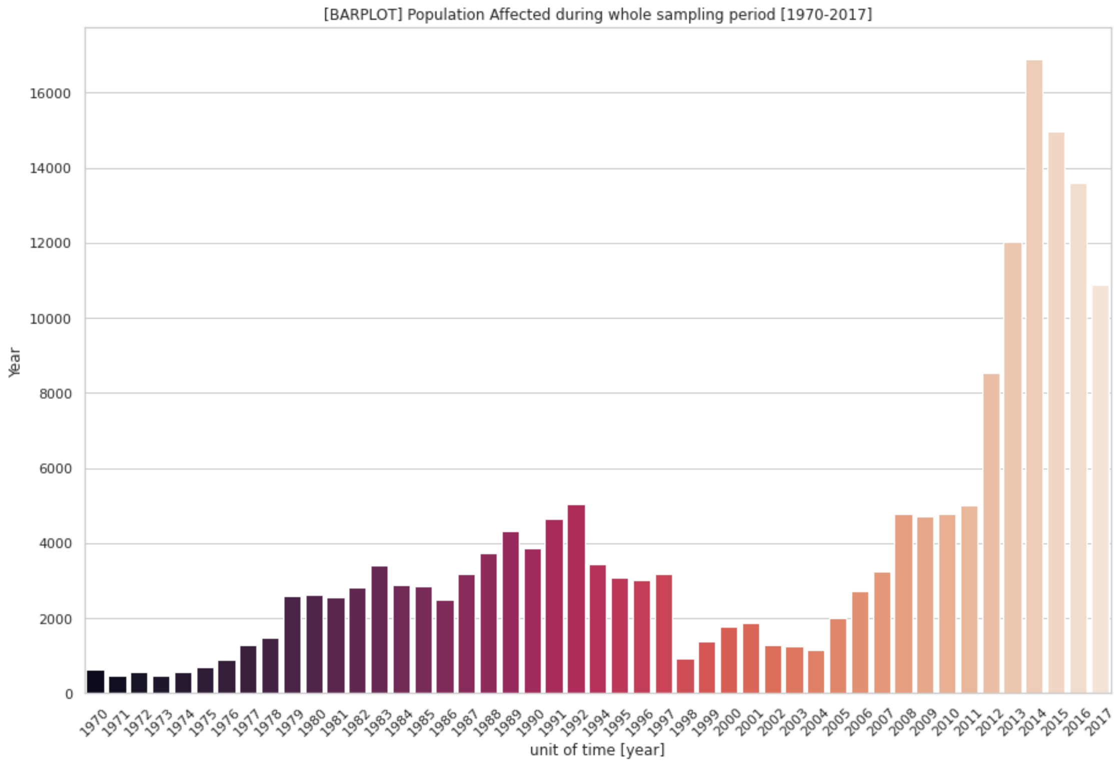
[143]:
```
#rows for df_copy =#rows for dict_new = ['Assassination', 'Armed Assault',␣
↪'Bombing/Explosion','Facility/Infrastructure Attack', 'Hijacking',␣
↪'Unknown','Hostage Taking (Kidnapping)', 'Unarmed Assault','Hostage Taking␣
↪(Barricade Incident)']
```

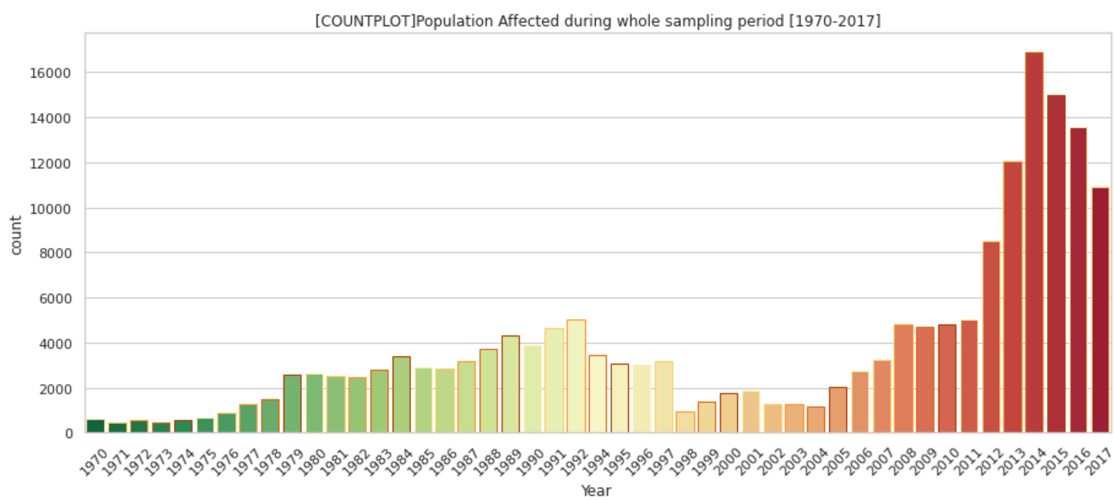
[144]:
```
#Total people Affected from year 1970 to 2017
#y1970=df[df['Killed']>0][['Year','Killed','Wounded']]
#y1970.dropna()
#y1970=y1970.groupby(['Year'])[['Killed','Wounded']].sum()
#CLASSICAL PLOT
#y1970.plot(kind='bar',stacked=True,figsize=(20,10))
#plt.title('Population Affected during whole sampling period [1970-2017]')
#plt.ylabel('#PEOPLE')
#plt.xlabel('unit of time [year]')
#plt.show()

#PLOT USING BAR PLOT
#x1970 = df['Year'].unique()

#y_1970 = df[df['Killed']>0][['Year','Killed','Wounded']]
#y_1970 = df['Year'].value_counts(dropna=True).sort_index()
#y_1970=y_1970.groupby(['Year'])[['Killed','Wounded']]
#y_1970 = y_1970['Year'].value_counts().sort_index()
#plt.figure(figsize=(20,10))
#plt.title("Population Affected during whole sampling period [1970-2017]")
#plt.xlabel("unit of time [year]")
#plt.ylabel("#PEOPLE")
#plt.xticks(rotation=45)
#sns.barplot(x=x1970, y=y_1970, palette= 'rocket')
#plt.show()
```

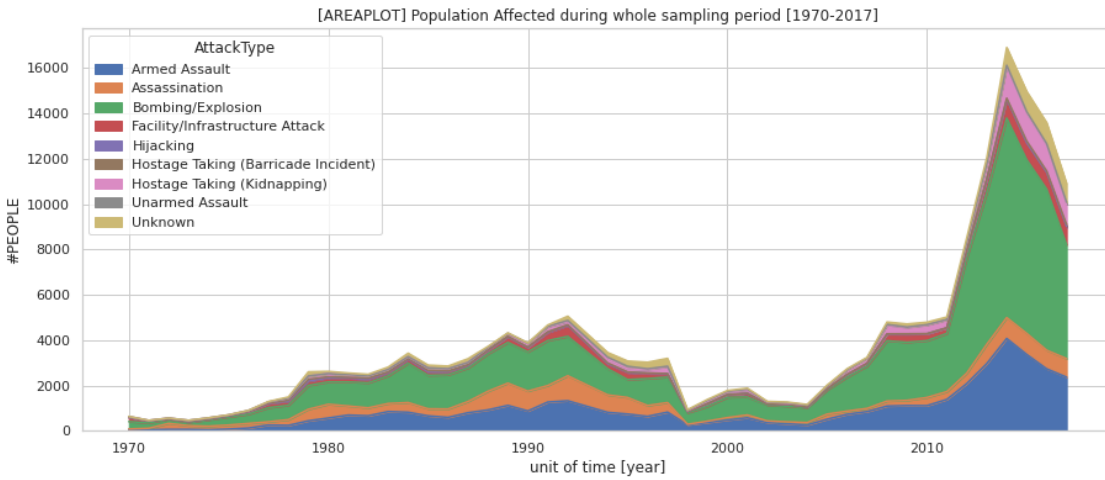
[145]:
```
#Barplot
import seaborn as sns

x_1970_2017 = df['Year'].unique()
```

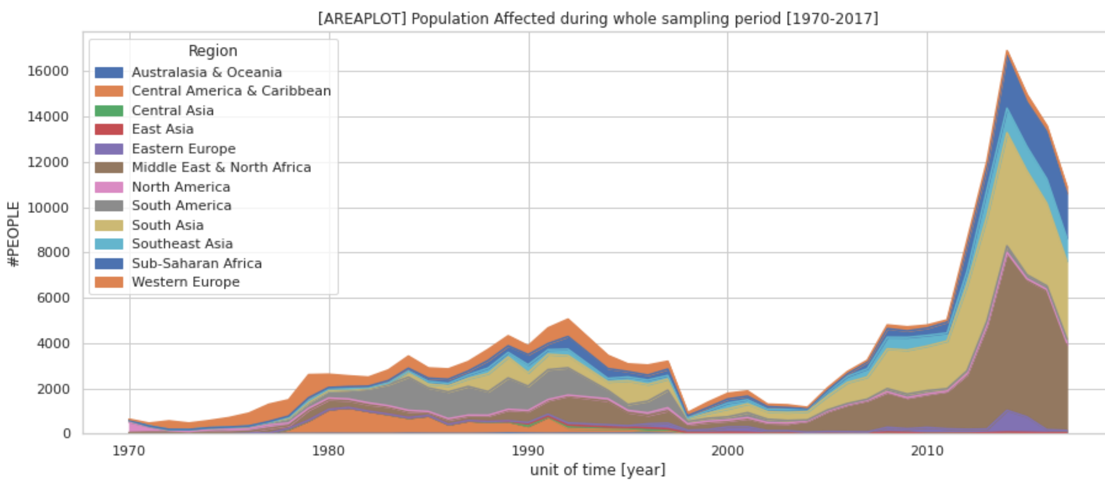
```
y_1970_2017 = df['Year'].value_counts(dropna=False).sort_index()
plt.figure(figsize=(15,10))
plt.title("[BARPLOT] Population Affected during whole sampling period␣
↪[1970-2017]")
plt.xlabel("unit of time [year]")
plt.ylabel("#PEOPLE")
plt.xticks(rotation=45)
sns.barplot(x=x_1970_2017, y=y_1970_2017, palette= 'rocket')
plt.show()

#Countplot
plt.subplots(figsize=(15,6))
#sns.countplot('Year', data=df, palette='RdYlGn_r',edgecolor=sns.
↪color_palette("YlOrBr", 5))
sns.countplot('Year', data=df, palette='RdYlGn_r',edgecolor=sns.
↪color_palette("YlOrBr", 5))
plt.xticks(rotation=45)
plt.title('[COUNTPLOT]Population Affected during whole sampling period␣
↪[1970-2017]')
plt.show()

#Area plot
#pd.crosstab(df.Year,df.Region).plot(kind='area',figsize=(15,6))
pd.crosstab(df.Year,df.AttackType).plot(kind='area',figsize=(15,6))
plt.title('[AREAPLOT] Population Affected during whole sampling period␣
↪[1970-2017]')
plt.xlabel("unit of time [year]")
plt.ylabel("#PEOPLE")
plt.show()

#Area plot
#pd.crosstab(df.Year,df.Region).plot(kind='area',figsize=(15,6))
pd.crosstab(df.Year,df.Region).plot(kind='area',figsize=(15,6))
plt.title('[AREAPLOT] Population Affected during whole sampling period␣
↪[1970-2017]')
plt.xlabel("unit of time [year]")
plt.ylabel("#PEOPLE")
plt.show()

#Area plot
#pd.crosstab(df.Year,df.Region).plot(kind='area',figsize=(15,6))
pd.crosstab(df.Year,df.Weapon_type).plot(kind='area',figsize=(15,6))
plt.title('[AREAPLOT] Population Affected during whole sampling period␣
↪[1970-2017]')
plt.xlabel("unit of time [year]")
plt.ylabel("#PEOPLE")
```

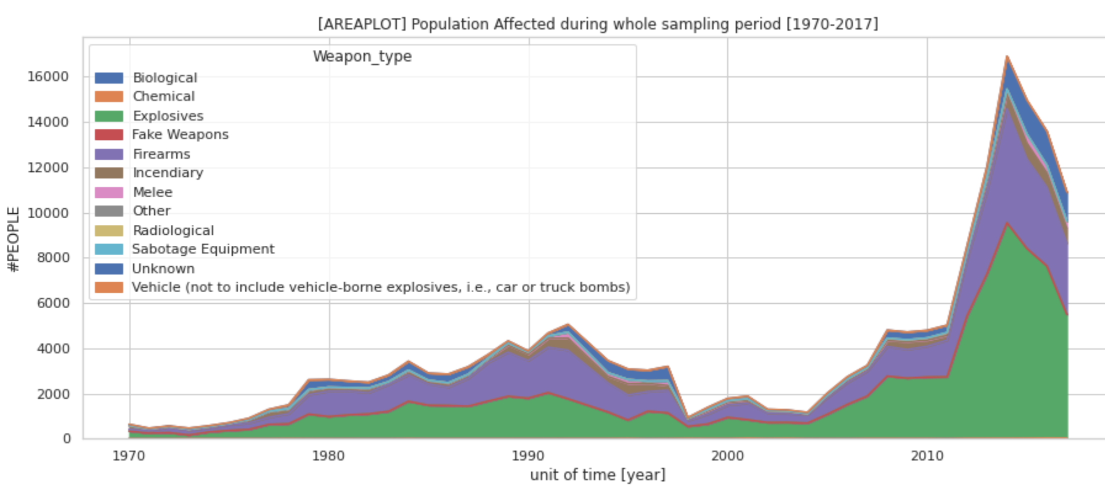
```python
plt.show()
```






[AREAPLOT] Population Affected during whole sampling period [1970-2017]
AttackType
Armed Assault
Assassination
Bombing/Explosion
Facility/Infrastructure Attack
Hijacking
Hostage Taking (Barricade Incident)
Hostage Taking (Kidnapping)
Unarmed Assault
Unknown
#PEOPLE
unit of time [year]

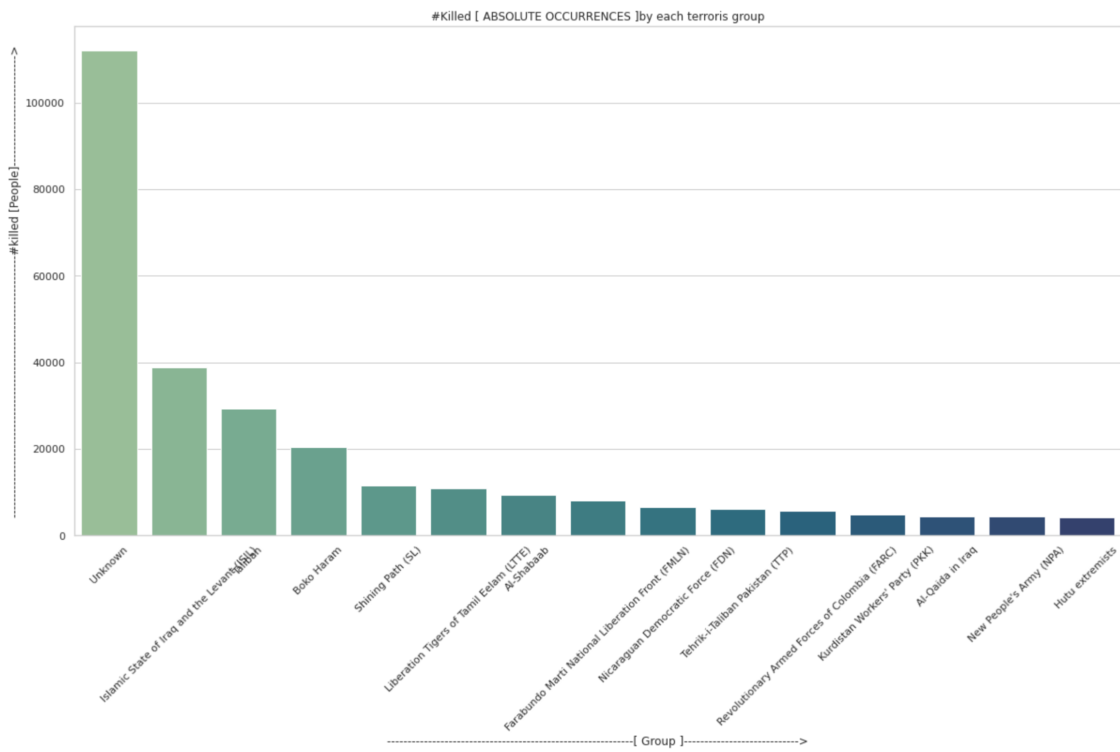

[AREAPLOT] Population Affected during whole sampling period [1970-2017]
Region
Australasia & Oceania
Central America & Caribbean
Central Asia
East Asia
Eastern Europe
Middle East & North Africa
North America
South America
South Asia
Southeast Asia
Sub-Saharan Africa
Western Europe
#PEOPLE
unit of time [year]


[AREAPLOT] Population Affected during whole sampling period [1970-2017]
Weapon_type
Biological
Chemical
Explosives
Fake Weapons
Firearms
Incendiary
Melee
Other
Radiological
Sabotage Equipment
Unknown
Vehicle (not to include vehicle-borne explosives, i.e., car or truck bombs)
#PEOPLE
unit of time [year]

```
Revolutionary Armed Forces of Colombia (FARC)       5620.0
Kurdistan Workers' Party (PKK)                      4921.0
Al-Qaida in Iraq                                    4380.0
New People's Army (NPA)                             4346.0
Hutu extremists                                     4095.0
```

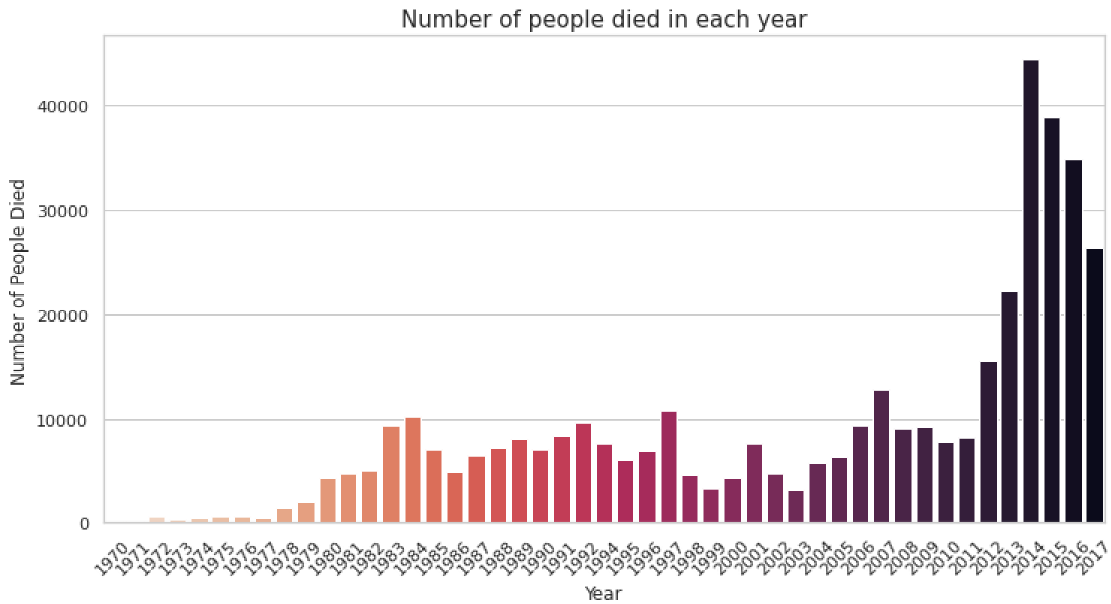
[149]:
```
print('People Killed by each group in terrorist activity')
plt.subplots(figsize=(20,10))
sns.barplot(group_killed_people.index, group_killed_people.Killed.
 ↪values,palette="crest")
plt.title('#Killed [ ABSOLUTE OCCURRENCES ]by each terroris group')
plt.xlabel('------------------------------------------------------------[ Group␣
 ↪]----------------------------->')
plt.
 ↪ylabel('---------------------------------------------------------------#killed␣
 ↪[People]--------------------------->')
plt.xticks(rotation= 45)
plt.show()
```

```
People Killed by each group in terrorist activity
```



[150]:
```
died_people = df[['Year','Killed']].groupby(['Year']).sum()
plt.subplots(figsize=(12,6))
```

```
sns.barplot(died_people.index, died_people.Killed.values,palette="rocket_r")
plt.title("Number of people died in each year",fontsize=15)
plt.ylabel("Number of People Died")
plt.xlabel('Year')
plt.xticks(rotation = 45)
plt.show()
```



[151]:
```
print('TOP 15 CITIES ATTACKED BY TERRORISM')
attack_cities = df.city.value_counts()[:15]
attack_cities
```

```
TOP 15 CITIES ATTACKED BY TERRORISM
```

[151]:
```
Unknown         10062
Baghdad          7582
Karachi          2647
Lima             2356
Mosul            2263
Belfast          2169
Santiago         1615
Mogadishu        1573
San Salvador     1546
Istanbul         1033
Athens           1016
Bogota            974
Kirkuk            924
Beirut            903
```

```
Medellin          846
Name: city, dtype: int64
```

[152]:
```
print('TOP 15 COUNTRIES ATTACKED BY TERRORISM')
attack_countries = df.Country.value_counts()[:15]
attack_countries
```

```
TOP 15 COUNTRIES ATTACKED BY TERRORISM
```

[152]:
```
Iraq              24616
Pakistan          14331
Afghanistan       12718
India             11931
Colombia           8232
Philippines        6874
Peru               6059
El Salvador        5277
United Kingdom     5208
Turkey             4267
Somalia            4124
Nigeria            3907
Thailand           3840
Yemen              3345
Spain              3212
Name: Country, dtype: int64
```

[153]:
```
print('Which Region Has Suffered Most Attacks (1970-2017) ? TOP 15 REGIONS␣
 ↪ATTACKED BY TERRORISM')
attack_regions = df.Region.value_counts()[:15]
attack_regions
```

```
Which Region Has Suffered Most Attacks (1970-2017) ? TOP 15 REGIONS ATTACKED BY
TERRORISM
```

[153]:
```
Middle East & North Africa     50317
South Asia                     44866
South America                  18838
Sub-Saharan Africa             17450
Western Europe                 16450
Southeast Asia                 12438
Central America & Caribbean    10260
Eastern Europe                  5136
North America                   3416
East Asia                        790
Central Asia                     562
Australasia & Oceania            277
Name: Region, dtype: int64
```

[154]:
```python
df.city
```

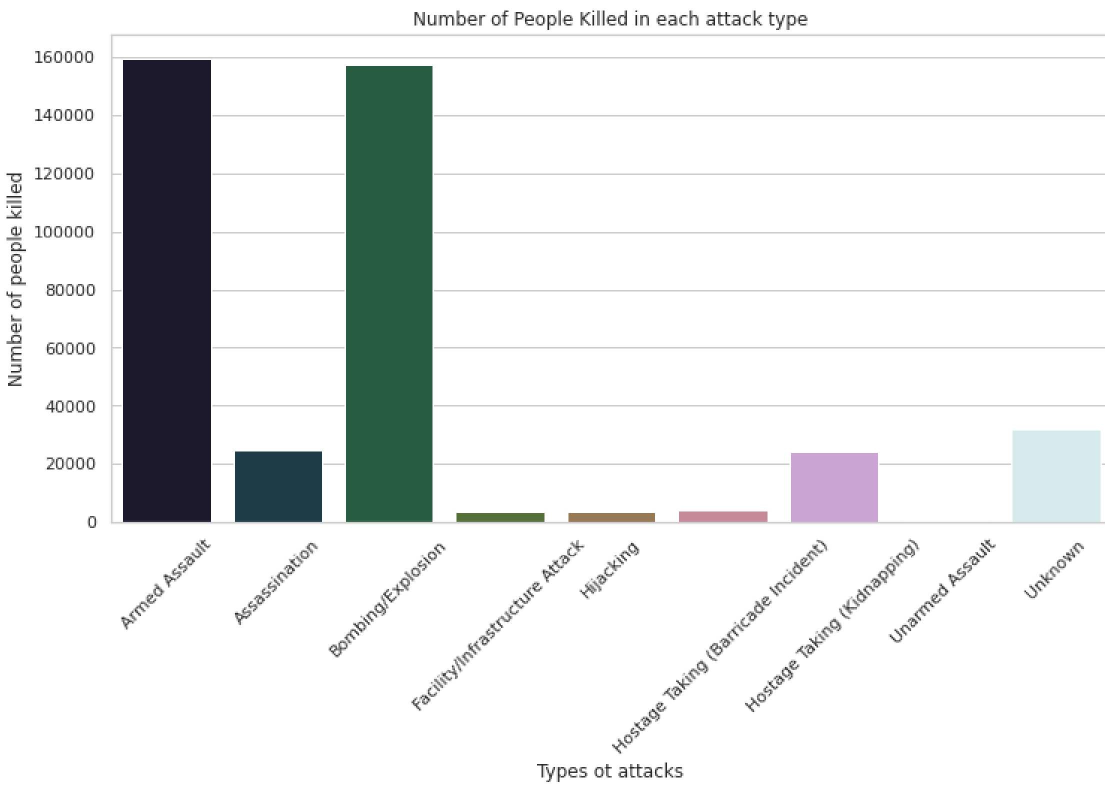
[154]:
```
0          Santo Domingo
5                  Cairo
6             Montevideo
7                Oakland
8                Madison
               …
181686     Ceelka Geelow
181687            Jableh
181688          Kubentog
181689            Imphal
181690     Cotabato City
Name: city, Length: 180800, dtype: object
```

[155]:
```python
attack_killed = df[['AttackType','Killed']].groupby(["AttackType"],axis=0).sum()
attack_killed
```

[155]:

	Killed
AttackType	
Armed Assault	159640.0
Assassination	24776.0
Bombing/Explosion	157235.0
Facility/Infrastructure Attack	3640.0
Hijacking	3715.0
Hostage Taking (Barricade Incident)	4478.0
Hostage Taking (Kidnapping)	24129.0
Unarmed Assault	879.0
Unknown	32165.0

[156]:
```python
## People Killed in each attack type
plt.subplots(figsize=(12,6))
sns.barplot(attack_killed.index, attack_killed.Killed.
 ↪values,palette="cubehelix")
plt.title('Number of People Killed in each attack type')
plt.xlabel('Types ot attacks')
plt.ylabel('Number of people killed')
plt.xticks(rotation= 45)
plt.show()
```

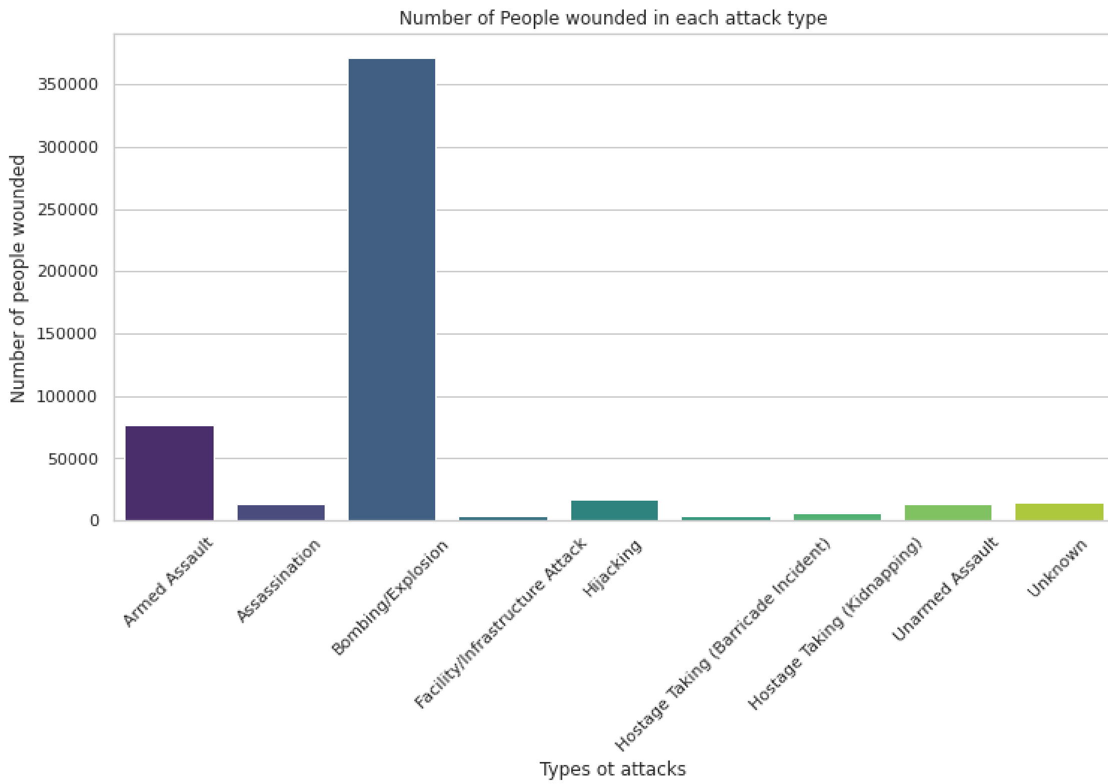


[157]:
```
attack_wounded = df[['AttackType','Wounded']].groupby(["AttackType"],axis=0).
 ↪sum()
attack_wounded
```

[157]:
```
                                     Wounded
AttackType
Armed Assault                        77169.0
Assassination                        13849.0
Bombing/Explosion                   372226.0
Facility/Infrastructure Attack        3764.0
Hijacking                            17001.0
Hostage Taking (Barricade Incident)   3966.0
Hostage Taking (Kidnapping)           6438.0
Unarmed Assault                      13999.0
Unknown                              14683.0
```

[158]:
```
## People Wounded in each attack type
plt.subplots(figsize=(12,6))
sns.barplot(attack_wounded.index, attack_wounded.Wounded.
 ↪values,palette="viridis")
plt.title('Number of People wounded in each attack type')
plt.xlabel('Types ot attacks')
```

```
plt.ylabel('Number of people wounded')
plt.xticks(rotation= 45)
plt.show()
```



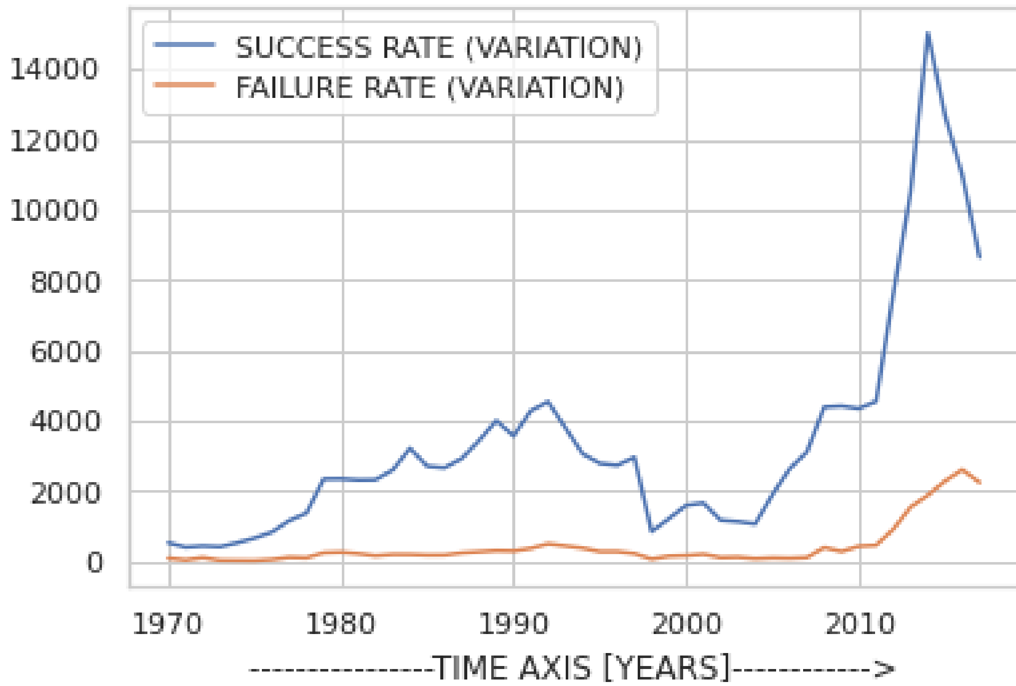
[ ]:

```
[159]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 180800 entries, 0 to 181690
Data columns (total 59 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   eventid           180800 non-null  int64
 1   Year              180800 non-null  int64
 2   Month             180800 non-null  int64
 3   Day               180800 non-null  int64
 4   extended          180800 non-null  int64
 5   country           180800 non-null  int64
 6   Country           180800 non-null  object
 7   region            180800 non-null  int64
 8   Region            180800 non-null  object
```

```
#Name: numbers, dtype: int64

success_counts=df[df.success==1]['Year'].value_counts().sort_index()
fail_counts=df[df.success==0]['Year'].value_counts().sort_index()
plt.plot(success_counts,label="SUCCESS RATE (VARIATION)")
plt.plot(fail_counts,label="FAILURE RATE (VARIATION)")
plt.legend(loc=2)
plt.xlabel("----------------TIME AXIS [YEARS]------------>")
plt.show
```

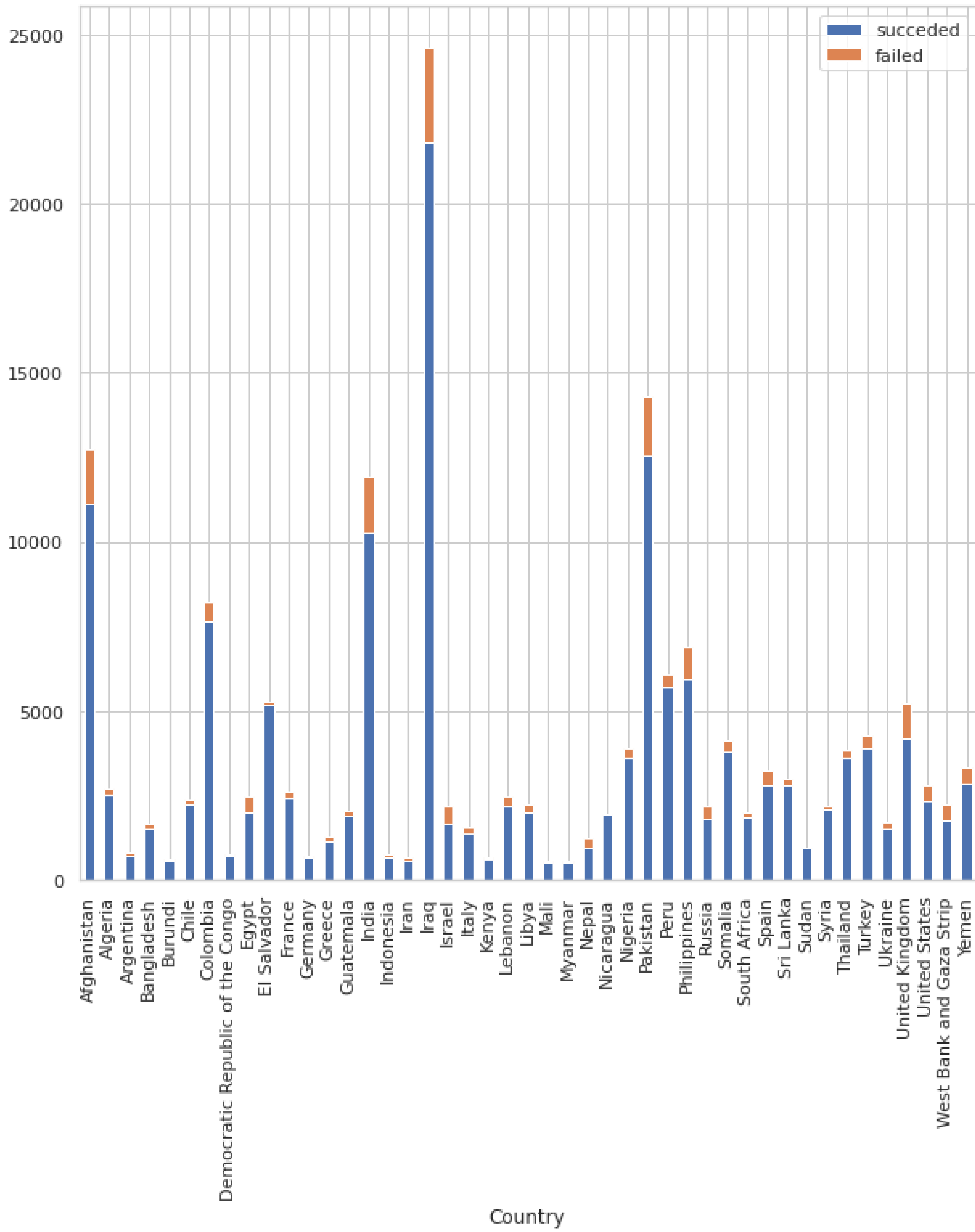
[167]: <function matplotlib.pyplot.show(close=None, block=None)>



[168]:
```
# Resto sorpreso perché sembra ci siano stati molti più atti di successo che␣
↪atti falliti.
#Può significare sia:
#-----i dati non sono completi e gli atti terroristici falliti vengono␣
↪segnalati al 100%.
#-----l'alto tasso di successo è reale, almeno secondo la metrica con cui sono␣
↪stati segnalati.
#c'è da ricordare che alcuni rientrano dentro tutte le categorie, sono certi di␣
↪essere avvenuti però.....

#Sospetto che la realtà sia un mix.
```

```
#Un atto terroristico fallito potrebbe non colpire i titoli dei giornali e␣
↪quindi sarà assente dai dati.
#Diamo un'occhiata a questo tasso di successo per paese per vedere se posso␣
↪avere maggiori informazioni.
#Manterrò solo i paesi in cui sono stati segnalati almeno 500 atti per motivi␣
↪di leggibilità.

#idea : df_test['date'] = pd.to_datetime(df[['day', 'month', 'year']])
df_test = df.copy()
df_test['Date'] = pd.to_datetime(df_test[['Day', 'Month', 'Year']])#anzichè␣
↪usare eventId che è scomodo
count_by_country = df_test[df_test.success== 1].groupby('Country').
↪count()['Date']

#serve un casting:
df_test_1=pd.DataFrame(index=count_by_country[count_by_country>500].index.
↪unique())
#df_test_1 è un sottoinsieme di count_by_country che però non è un dataframe e␣
↪invece df_test_1 ora lo è diventato


df_test_1["succeded"]= df_test[df_test.success== 1].groupby('Country')['Date'].
↪count().fillna(0)
df_test_1["failed"]= df_test[df_test.success!= 1].groupby('Country')['Date'].
↪count().fillna(0)
df_test_1[df_test_1.failed>0].plot(kind='bar', stacked=True,figsize=(10,10))
```

[168]: <AxesSubplot:xlabel='Country'>



[169]:
```
df.head(2)
```

[169]:
```
        eventid  Year  Month  Day  extended  country            Country  \
0  197000000001  1970      7    2         0       58  Dominican Republic
5  197001010002  1970      1    1         0      217       United States
```

```
  region                       Region provstate           city   latitude  \
0      2  Central America & Caribbean   Baghdad  Santo Domingo  18.456792
5      1                North America  Illinois          Cairo  37.005105

  longitude  specificity  vicinity  \
0 -69.951164          1.0         0
5 -89.176269          1.0         0

                                              Summary  crit1  crit2  crit3  \
0                                                 NaN      1      1      1
5  1/1/1970: Unknown African American assailants …      1      1      1

  doubtterr  multiple  success  suicide  attacktype1     AttackType  \
0       0.0       0.0        1        0            1  Assassination
5       0.0       0.0        1        0            2  Armed Assault

  targtype1                  Target_type  targsubtype1  \
0        14  Private Citizens & Property          68.0
5         3                       Police          22.0

                             targsubtype1_txt  …  \
0                              Named Civilian  …
5  Police Building (headquarters, station, school)  …

                     Target natlty1         natlty1_txt               Group  \
0              Julio Guzman    58.0  Dominican Republic              MANO-D
5  Cairo Police Headquarters   217.0       United States  Black Nationalists

 guncertain1  individual  nperps  nperpcap  claimed  weaptype1  Weapon_type  \
0         0.0           0   -99.0       0.0      0.0         13      Unknown
5         0.0           0   -99.0     -99.0      0.0          5     Firearms

 weapsubtype1  weapsubtype1_txt                   weapdetail Killed  \
0        12.0               NaN                          NaN    1.0
5         5.0  Unknown Gun Type  Several gunshots were fired.    0.0

  nkillus  nkillter  Wounded  nwoundus  nwoundte  property  ishostkid  \
0     0.0       0.0      0.0       0.0       0.0         0        0.0
5     0.0       0.0      0.0       0.0       0.0         1        0.0

                                             scite1        dbsource INT_LOG  \
0                                               NaN            PGIS       0
5  "Police Chief Quits," Washington Post, January…  Hewitt Project      -9

  INT_IDEO  INT_MISC  INT_ANY  casualities
0        0         0        0          1.0
```

```
5        -9          0       -9           0.0

[2 rows x 59 columns]
```

[170]:
```
df_test.head(2)
```

[170]:
```
        eventid  Year  Month  Day  extended  country             Country  \
0  197000000001  1970      7    2         0       58  Dominican Republic
5  197001010002  1970      1    1         0      217       United States

   region                       Region provstate          city   latitude  \
0       2  Central America & Caribbean   Baghdad  Santo Domingo  18.456792
5       1                North America  Illinois          Cairo  37.005105

   longitude  specificity  vicinity  \
0 -69.951164          1.0         0
5 -89.176269          1.0         0

                                         Summary  crit1  crit2  crit3  \
0                                            NaN      1      1      1
5  1/1/1970: Unknown African American assailants …      1      1      1

   doubtterr  multiple  success  suicide  attacktype1     AttackType  \
0        0.0       0.0        1        0            1  Assassination
5        0.0       0.0        1        0            2  Armed Assault

   targtype1                  Target_type  targsubtype1  \
0         14  Private Citizens & Property          68.0
5          3                       Police          22.0

                             targsubtype1_txt  … natlty1  \
0                              Named Civilian  …    58.0
5  Police Building (headquarters, station, school)  …   217.0

          natlty1_txt                Group guncertain1 individual  nperps  \
0  Dominican Republic               MANO-D         0.0          0   -99.0
5       United States  Black Nationalists          0.0          0   -99.0

   nperpcap  claimed  weaptype1  Weapon_type  weapsubtype1  weapsubtype1_txt  \
0       0.0      0.0         13      Unknown          12.0               NaN
5     -99.0      0.0          5     Firearms           5.0  Unknown Gun Type

                  weapdetail Killed nkillus  nkillter  Wounded  nwoundus  \
0                        NaN    1.0     0.0       0.0      0.0       0.0
5  Several gunshots were fired.    0.0     0.0       0.0      0.0       0.0

   nwoundte  property  ishostkid  \
```

```
0       0.0          0        0.0
5       0.0          1        0.0

                                                  scite1        dbsource INT_LOG  \
0                                                    NaN            PGIS       0
5  "Police Chief Quits," Washington Post, January…  Hewitt Project      -9

  INT_IDEO  INT_MISC  INT_ANY  casualities       Date
0        0         0        0          1.0 1970-07-02
5       -9         0       -9          0.0 1970-01-01

[2 rows x 60 columns]
```

[171]:
```python
df_test.Date
```

[171]:
```
0        1970-07-02
5        1970-01-01
6        1970-01-02
7        1970-01-02
8        1970-01-02
            …
181686   2017-12-31
181687   2017-12-31
181688   2017-12-31
181689   2017-12-31
181690   2017-12-31
Name: Date, Length: 180800, dtype: datetime64[ns]
```

[172]:
```python
count_by_country
```

[172]:
```
Country
Afghanistan    11128
Albania           64
Algeria         2531
Andorra            1
Angola           469
               …
Yemen           2836
Yugoslavia       179
Zaire             44
Zambia            58
Zimbabwe          94
Name: Date, Length: 202, dtype: int64
```

[173]:
```python
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
```

[196]:
```
#Dummy benchmark : predict 100% success
print("Benchmark: " )
# ALTERNATIVE print(accuracy_score(target_test['success'],np.
 ↪ones(len(target_test['success']))))
print(metrics.accuracy_score(y_test,y_test))
```

```
Benchmark:
1.0
```

[197]:
```
ids = features_test.index#save all indexes
```

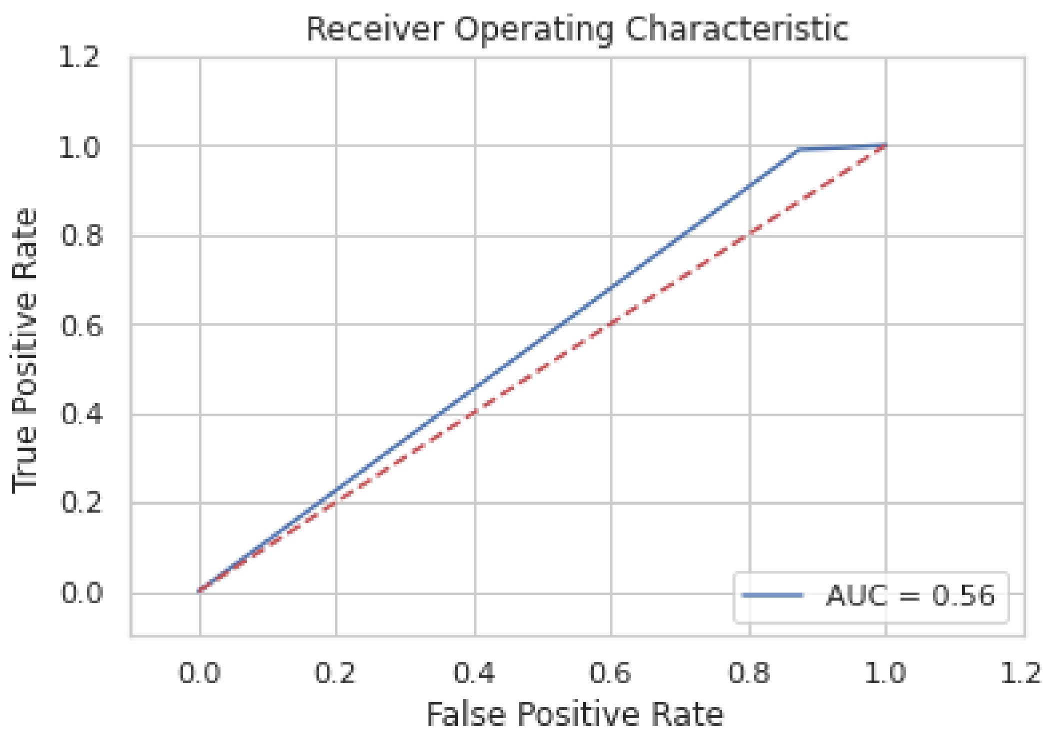
[198]:
```
#Random Forest
forest=RandomForestClassifier(n_estimators=10)
forest = forest.fit( features_train, target_train )
output = forest.predict(features_test).astype(int)
results = pd.DataFrame(data=output,index=ids,columns=['prediction'])
results = target_test.join(results)#.
 ↪drop(['attack','target','weapon','country'],axis=1)
```

[199]:
```
print("Random Forest accuracy score: " )
print(metrics.accuracy_score(results['success'],results['prediction']))
```

```
Random Forest accuracy score:
0.8692256056630255
```

[200]:
```
from sklearn.metrics import roc_curve, auc
false_positive_rate, true_positive_rate, thresholds = roc_curve(target_test,␣
 ↪output)
roc_auc = auc(false_positive_rate, true_positive_rate)
print('AUC = %0.4f'% roc_auc)
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate, 'b',label='AUC = %0.2f'%␣
 ↪roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.1,1.2])
plt.ylim([-0.1,1.2])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

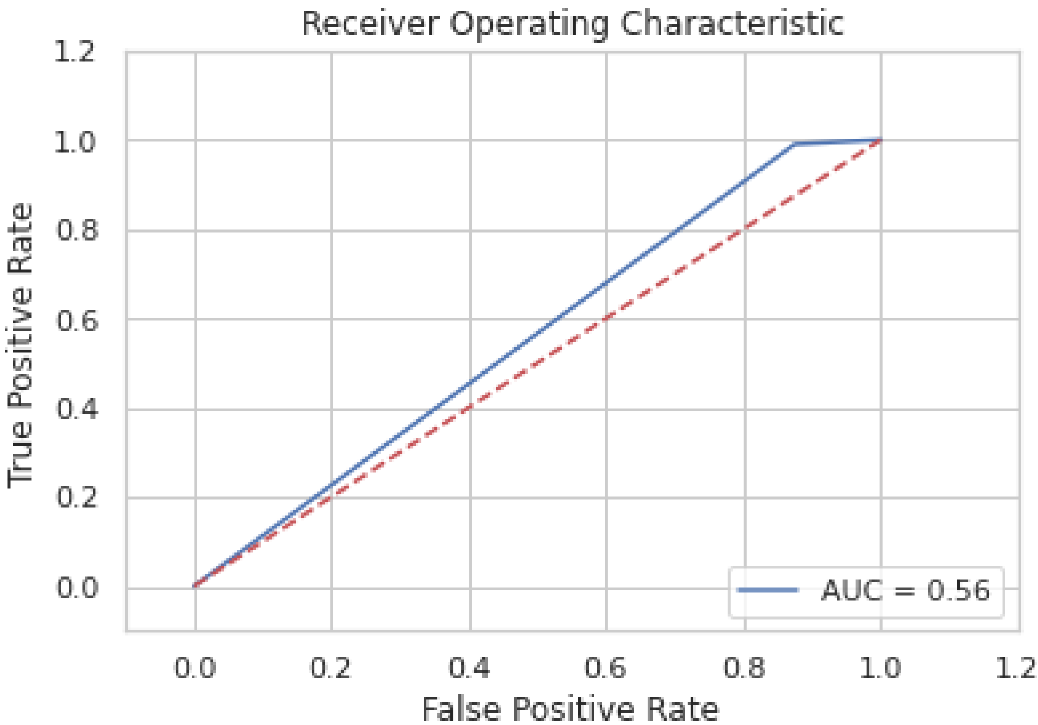
```
AUC = 0.5581
```



[201]:
```
############################ reitero l'analisi con più stimatori
#Random Forest
forest=RandomForestClassifier(n_estimators=15)
forest = forest.fit( features_train, target_train )
output = forest.predict(features_test).astype(int)
results = pd.DataFrame(data=output,index=ids,columns=['prediction'])
results = target_test.join(results)#.
 ↪drop(['attack','target','weapon','country'],axis=1)
print("Random Forest accuracy score: " )
print(metrics.accuracy_score(results['success'],results['prediction']))
from sklearn.metrics import roc_curve, auc
false_positive_rate, true_positive_rate, thresholds = roc_curve(target_test,␣
 ↪output)
roc_auc = auc(false_positive_rate, true_positive_rate)
print('AUC = %0.4f'% roc_auc)
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate, 'b',label='AUC = %0.2f'%␣
 ↪roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.1,1.2])
plt.ylim([-0.1,1.2])
plt.ylabel('True Positive Rate')
```

```
plt.xlabel('False Positive Rate')
plt.show()
```

```
Random Forest accuracy score:
0.8694086776103008
AUC = 0.5578
```



[202]:

```
#FONTE DI RIFERIMENTO : https://developers.google.com/machine-learning/
↪crash-course/classification/roc-and-auc

#Una curva ROC (curva caratteristica operativa del ricevitore == receiver␣
↪operating characteristic curve)
#è un grafico che mostra le prestazioni di un modello di classificazione a␣
↪tutte le soglie di classificazione.
#Questa curva traccia due parametri:

#    Tasso di vero positivo --> asse ordinate
#    Tasso di falsi positivi--> asse ascisse

#True Positive Rate (TPR) è sinonimo di richiamo ed è quindi definito come␣
↪segue:
#TPR ( True Positive Rate)= SENSITIVITY =#TRUE_POSITIVES / [ #TRUE_POSITIVES +␣
↪#FALSE_NEGATIVES]
```

```
#Il tasso di falsi positivi (FPR) è definito come segue:
# FPR ( False Positive Rate)=1 - SPECIFICITY = 1- [#TRUE_NEGATIVES /␣
↪[#TRUE_NEGATIVES + #FALSE_POSITIVES]]
#Una curva ROC traccia TPR vs. FPR a diverse soglie di classificazione.␣
↪L'abbassamento della soglia di classificazione classifica più elementi come␣
↪positivi, aumentando così sia i falsi positivi che i veri positivi.
#La figura sopra mostra una tipica curva ROC.




#AUC sta per "Area sotto la curva ROC". Cioè, l'AUC misura l'intera area
#bidimensionale sotto l'intera curva ROC (si pensi al calcolo integrale) da␣
↪(0,0) a (1,1).



#Le curve ROC presentano in genere un tasso di veri positivi sull'asse Y e un␣
↪tasso di falsi
#positivi sull'asse X.
#Anche la "ripidezza" delle curve ROC è importante, poiché è l'ideale per␣
↪massimizzare
#il tasso di veri positivi riducendo al minimo il tasso di falsi positivi.


#fN=feature_cols
#cN=['AttackType', 'Country', 'Weapon_type']
#fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=800)
#tree.plot_tree(forest,feature_names = fN,class_names=cN,fontsize=10,filled =␣
↪True)

#from sklearn.datasets import load_iris
#iris = load_iris()

# Extract single tree
#estimator = forest.estimators_[5]
#tree.plot_tree(estimator,fontsize=10)
#plt.show()
#from sklearn.tree import export_graphviz
# Export as dot file
#export_graphviz(estimator, out_file='tree.dot',
#                 feature_names = feature_cols,
#                 class_names = ['AttackType', 'Country', 'Weapon_type'],
#                 rounded = True, proportion = False,
#                 precision = 2, filled = True)

# Convert to png using system command (requires Graphviz)
```

```
#from subprocess import call
#call(['dot', '-Tpng', 'tree.dot', '-o', 'tree.png', '-Gdpi=600'])

# Display in jupyter notebook
#from IPython.display import Image
#Image(filename = 'tree.png')
```

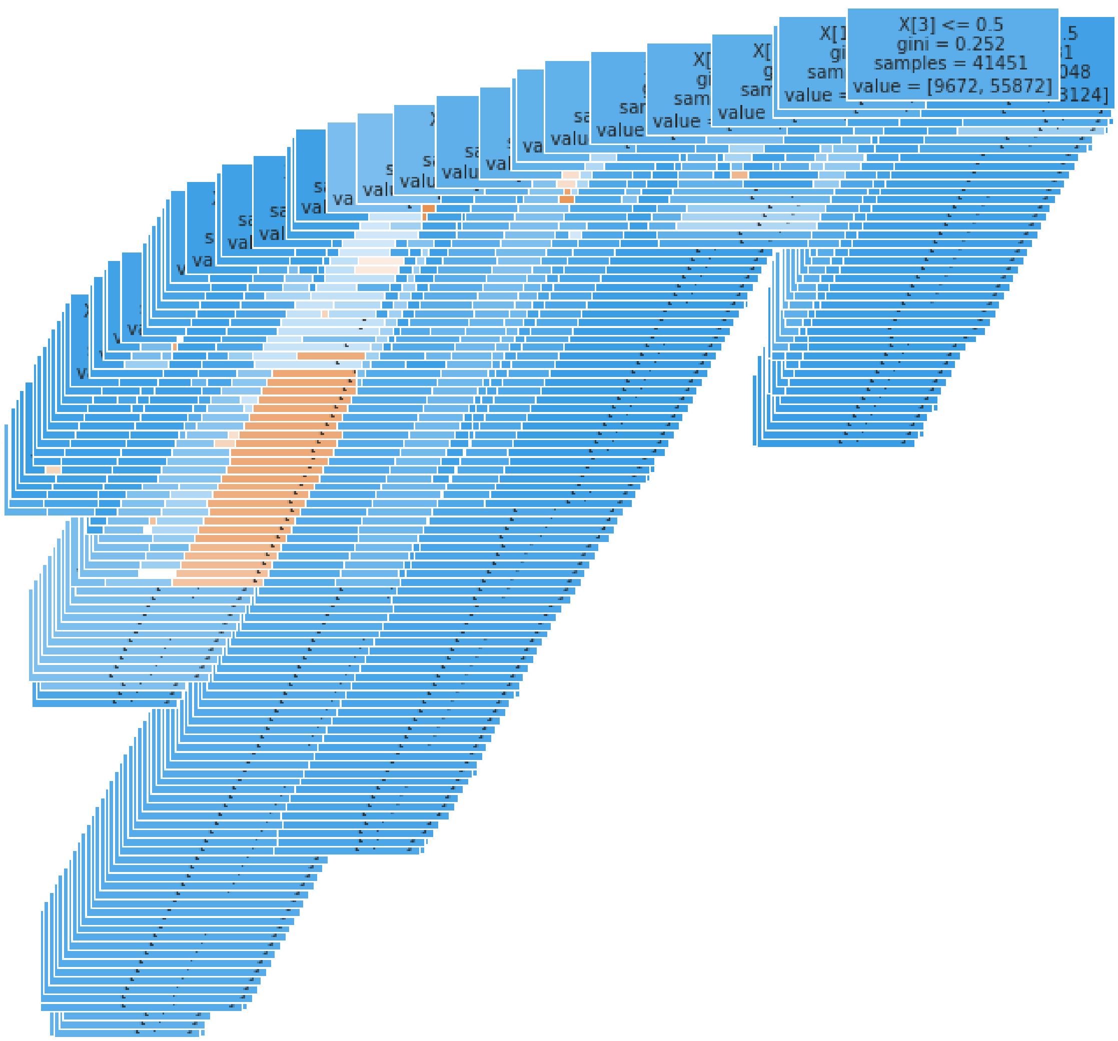
[203]:

```
len(forest.estimators_)
```

[203]: 15

[204]:

```
fig, ax = plt.subplots(figsize=(10,10))
tree.plot_tree(forest.estimators_[0],fontsize=10, filled=True)
plt.show()
```



[205]:
```
print(forest.estimators_[0].max_depth)
```

None

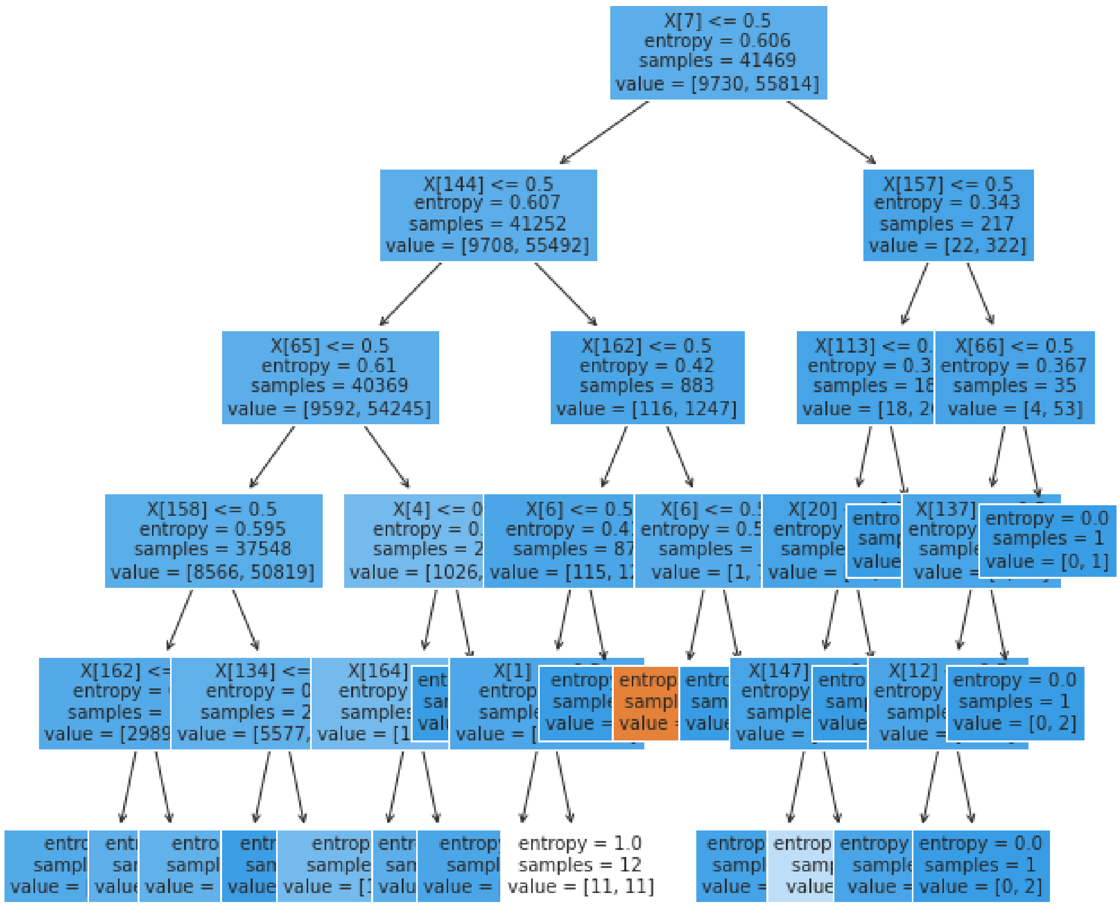
[206]:
```
#Random Forest
# calcoliamo l'accuratezza del classificatore

forest2=RandomForestClassifier(n_estimators=10,max_depth=5,criterion='entropy')
forest2 = forest2.fit( features_train, target_train )
output2 = forest.predict(features_test).astype(int)
output2_tilde=forest.predict(features_test)
results2 = pd.DataFrame(data=output2,index=ids,columns=['prediction'])
results2 = target_test.join(results2)
print('Accuracy del Decision Tree Classifier (Splitting criterion: ENTROPY):')
print(metrics.accuracy_score(results2['success'],results2['prediction']))
```

Accuracy del Decision Tree Classifier (Splitting criterion: ENTROPY):
0.8694086776103008

[207]:
```
print(forest2.estimators_[0].max_depth)
```

5

[208]:
```
fig, ax = plt.subplots(figsize=(10,10))
tree.plot_tree(forest2.estimators_[0],fontsize=10, filled=True)
plt.show()
```

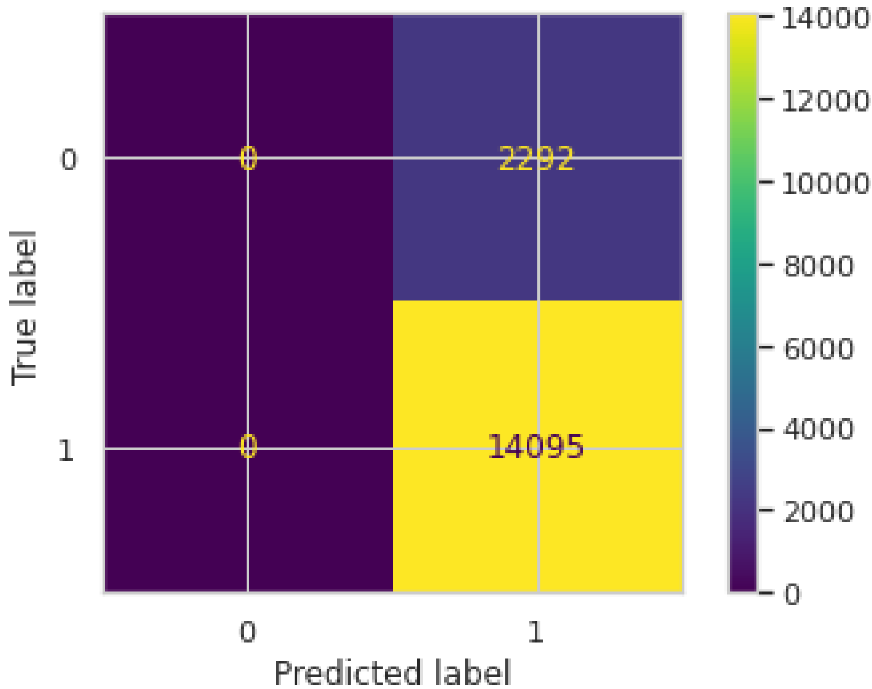



```
[209]: #calcolo delle altre metriche per valutare le prestazioni del classificatore
target_names = ['class0','class1']
#features_train, target_train --> X_train , y_train
#features_test -> X_test
#target_test -> y_test
#results2['success'],results2['prediction'] --> y_test , Y_pred
print(classification_report(results2['success'], results2['prediction'],
    target_names =target_names))
```

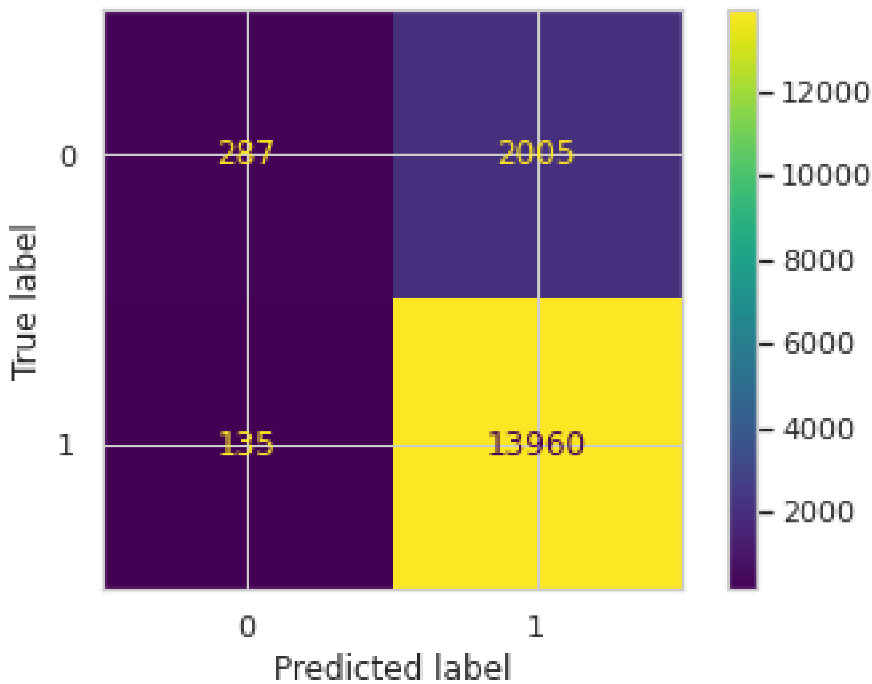
```
              precision    recall  f1-score   support

      class0       0.68      0.13      0.21      2292
      class1       0.87      0.99      0.93     14095

    accuracy                           0.87     16387
   macro avg       0.78      0.56      0.57     16387
```

```
weighted avg       0.85      0.87      0.83     16387
```

[210]:
```python
#plottiamo la matrice di confusione
plot_confusion_matrix(forest2, features_test, target_test)
plt.show()
```



[211]:
```python
#plottiamo la matrice di confusione
plot_confusion_matrix(forest, features_test, target_test)
plt.show()
```



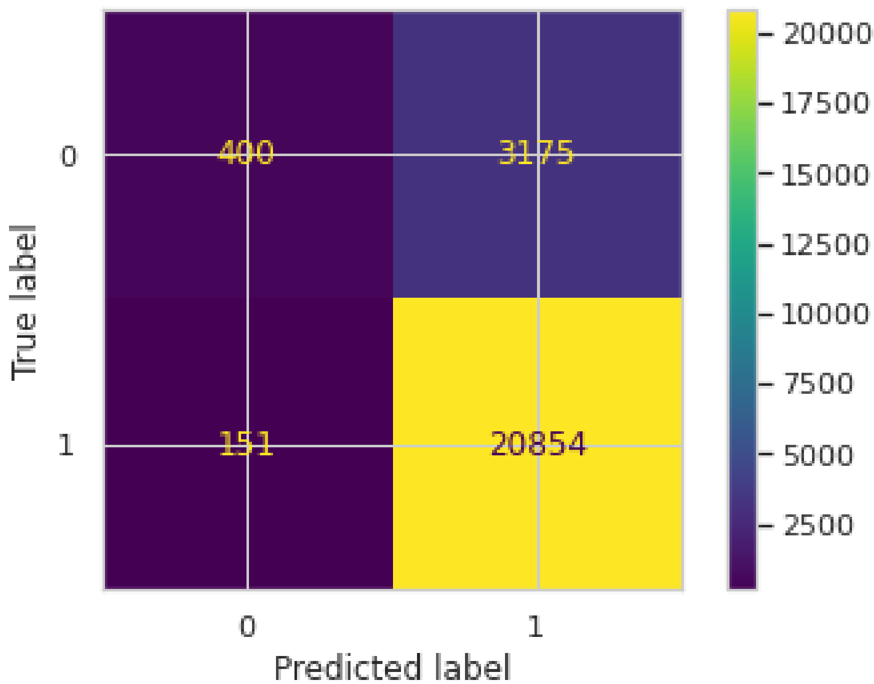
[212]:
```
#PROMEMORIA
#-success Success of a terrorist strike is defined according to the tangible␣
 ↪effects of the attack.
#Success is not judged in terms of the larger goals of the perpetrators. For␣
 ↪example, a bomb that exploded in a
#building would be counted as a success even if it did not succeed in bringing␣
 ↪the building down or inducing government
#repression.
#Sulla base di questo valore di probabilità, è possibile allora capire se un␣
 ↪albero decisionale
#possa prevedere se avviene o meno l'attacco ? Anche il #killed di quel␣
 ↪determinato attacco ?

#2nd come prefisso dava errore --> accorciato come 'nd'


#FEATURES DEFINITIONS
feature_cols =pd.concat( [pd.get_dummies(recent_df['AttackType'],prefix='atk'),
    pd.get_dummies(recent_df['Country'],prefix='ctry'),
    pd.get_dummies(recent_df['Weapon_type'],prefix='wpn') ],axis=1)
#no: feature_cols = ['AttackType', 'Country', 'Weapon_type']
#DEFINITION OF FEATURES MATRIX AND VECTOR TO PREDICT
# NO: X = df[feature_cols]
y = df['success']
```

```
success=pd.DataFrame(recent_df['success'])#y vector --> expected
#PARAMETERS DEFINITIONS
ndX_train, ndX_test, ndy_train, ndy_test =␣
 ↪train_test_split(feature_cols,success, test_size = 0.3)
ids=ndX_test.index
clf_tree = DecisionTreeClassifier(criterion='entropy',max_depth=5)
# addestro il classificatore sul dataset di training
clf_tree = clf_tree.fit(ndX_train, ndy_train)
y_pred_2 = clf_tree.predict(ndX_test).astype(int)
results = pd.DataFrame(data=y_pred_2,index=ids,columns=['prediction'])
results = target_test.join(results)#Join columns with other DataFrame either on␣
 ↪index or on a key column.
 # calcoliamo l'accuratezza del classificatore
print('Accuracy del Decision Tree Classifier (Splitting criterion: Entropy):
 ↪',metrics.accuracy_score(y_pred_2, ndy_test))
 # plottiamo l'albero decisionale
fig, ax = plt.subplots(figsize=(10,10))
tree.plot_tree(clf_tree, fontsize=10)
plt.show()
print(clf_tree.tree_.max_depth)
#calcolo delle altre metriche per valutare le prestazioni del classificatore
target_names = ['class0','class1']
print(classification_report(ndy_test, y_pred_2, target_names =target_names))
 #plottiamo la matrice di confusione
plot_confusion_matrix(clf_tree, ndX_test, ndy_test)
plt.show()
```

```
Accuracy del Decision Tree Classifier (Splitting criterion: Entropy):
0.864686737184703
```

5

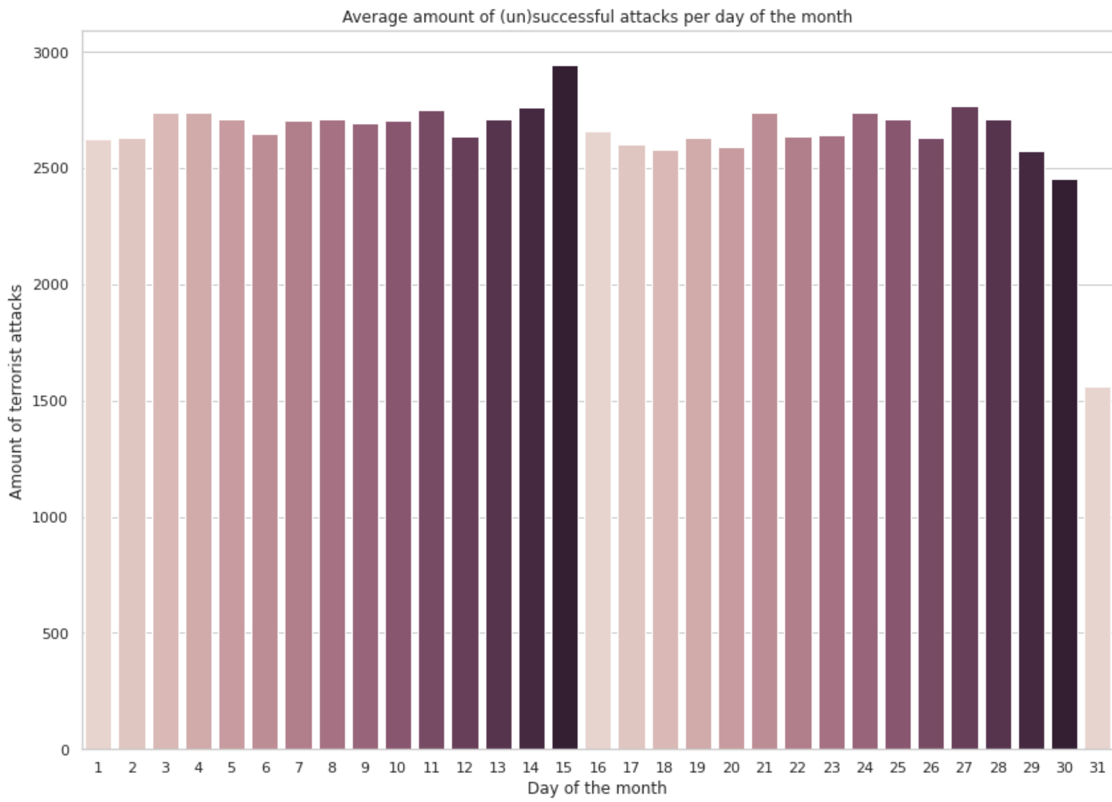
	precision	recall	f1-score	support
class0	0.73	0.11	0.19	3575
class1	0.87	0.99	0.93	21005
accuracy			0.86	24580
macro avg	0.80	0.55	0.56	24580
weighted avg	0.85	0.86	0.82	24580



[213]:
```
df_day_coords = recent_df[['Month', 'Day', 'longitude', 'latitude', 'success']].
    ↪copy()[(recent_df['Day'] != 0) & (recent_df['Month'] != 0)]
```

[214]:
```
fig, ax = plt.subplots(figsize=(14,10))
sns.countplot(x="Day", data=df_day_coords, ax=ax, palette=sns.
    ↪cubehelix_palette(15, start=.3, rot=.3))
ax.set_xlabel('Day of the month')
ax.set_ylabel('Amount of terrorist attacks')
plt.title('Average amount of (un)successful attacks per day of the month')
```

[214]: `Text(0.5, 1.0, 'Average amount of (un)successful attacks per day of the month')`



[215]:
```
fig, axs = plt.subplots(nrows=12)
fig.set_size_inches(15, 100, forward=True)

for i in range(1,13):
    monthly_data = df_day_coords[df_day_coords['Month'] == i]

    axs[i-1].set_xlabel('Day of the month')
    axs[i-1].set_ylabel('Amount of terrorist attacks')
    if i==1:
        print('Month observed:January')
        sns.countplot(x="Day", data=monthly_data, hue="success", ax=axs[i-1])
    elif i==2:
        print('Month observed:February')
        sns.countplot(x="Day", data=monthly_data, hue="success", ax=axs[i-1])
    elif i==3:
        print('Month observed:March')
        sns.countplot(x="Day", data=monthly_data, hue="success", ax=axs[i-1])
    elif i==4:
        print('Month observed:April')
        sns.countplot(x="Day", data=monthly_data, hue="success", ax=axs[i-1])
    elif i==5:
        print('Month observed:May')
```

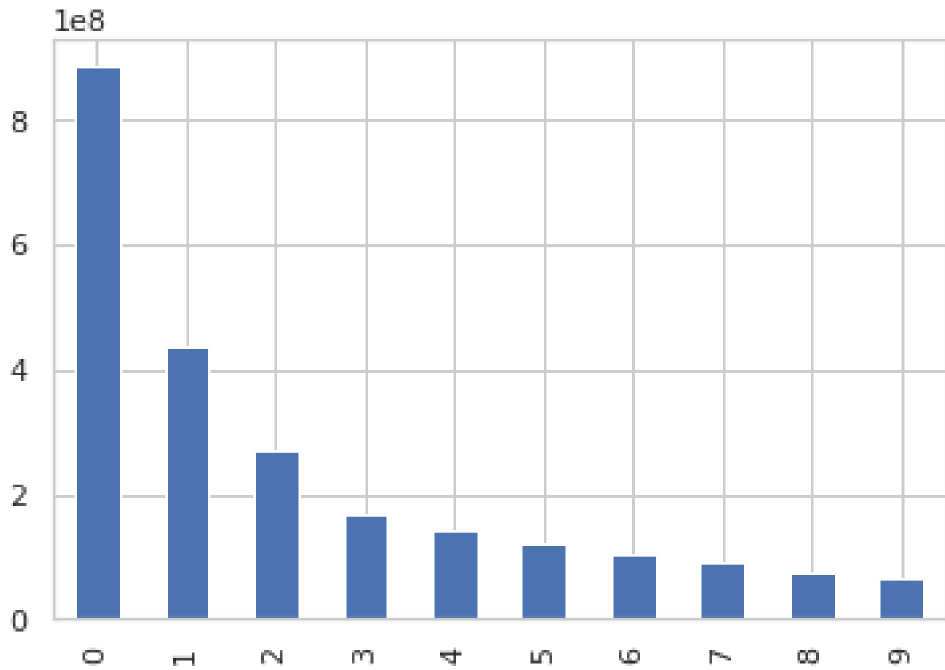
```python
            sns.countplot(x="Day", data=monthly_data, hue="success", ax=axs[i-1])
        elif i==6:
            print('Month observed:June')
            sns.countplot(x="Day", data=monthly_data, hue="success", ax=axs[i-1])
        elif i==7:
            print('Month observed:July')
            sns.countplot(x="Day", data=monthly_data, hue="success", ax=axs[i-1])
        elif i==8:
            print('Month observed:August')
            sns.countplot(x="Day", data=monthly_data, hue="success", ax=axs[i-1])
        elif i==9:
            print('Month observed:September')
            sns.countplot(x="Day", data=monthly_data, hue="success", ax=axs[i-1])
        elif i==10:
            print('Month observed:October')
            sns.countplot(x="Day", data=monthly_data, hue="success", ax=axs[i-1])
        elif i==11:
            print('Month observed:November')
            sns.countplot(x="Day", data=monthly_data, hue="success", ax=axs[i-1])
        elif i==12:
            print('Month observed:December')
            sns.countplot(x="Day", data=monthly_data, hue="success", ax=axs[i-1])
```

```
Month observed:January
Month observed:February
Month observed:March
Month observed:April
Month observed:May
Month observed:June
Month observed:July
Month observed:August
Month observed:September
Month observed:October
Month observed:November
Month observed:December
```

```
    wcss.append(kmeans.inertia_)
    #wcss[k]=KMeans(n_clusters = k, init='k-means++', max_iter=300).fit(x).
 ↪score(x)
pd.Series(wcss).plot.bar()
```

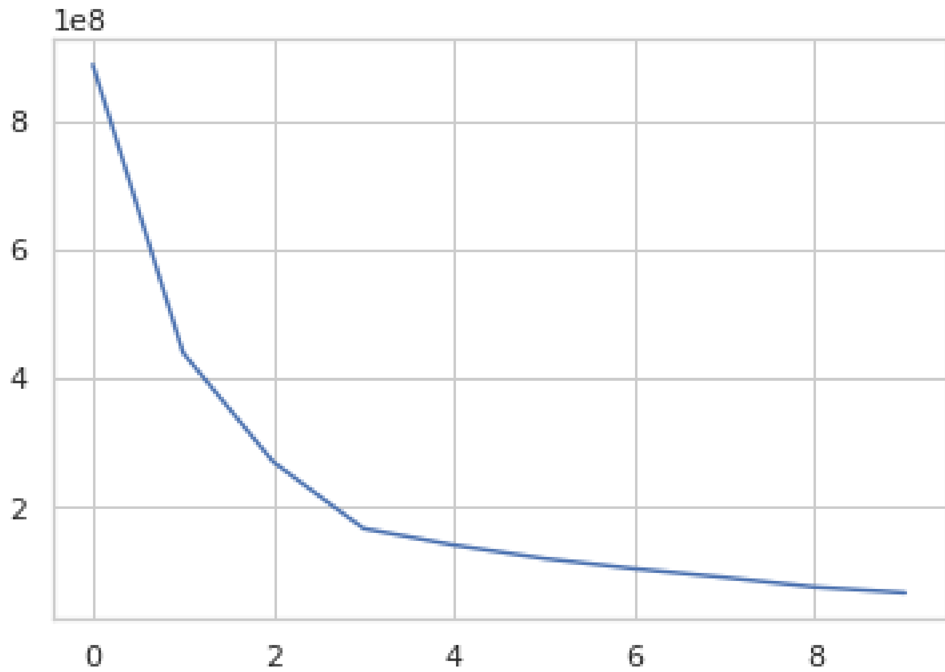
2, 3, 4, 5, 6, 7, 8, 9, 10, 11,

[239]: <AxesSubplot:>



[240]:
```
#pd.Series(wcss).plot.bar(x='k=#clusters', y='Total sum of square error')
pd.Series(wcss).plot()
```

[240]: <AxesSubplot:>



[241]:
```
testing_df=pd.DataFrame(wcss)
```

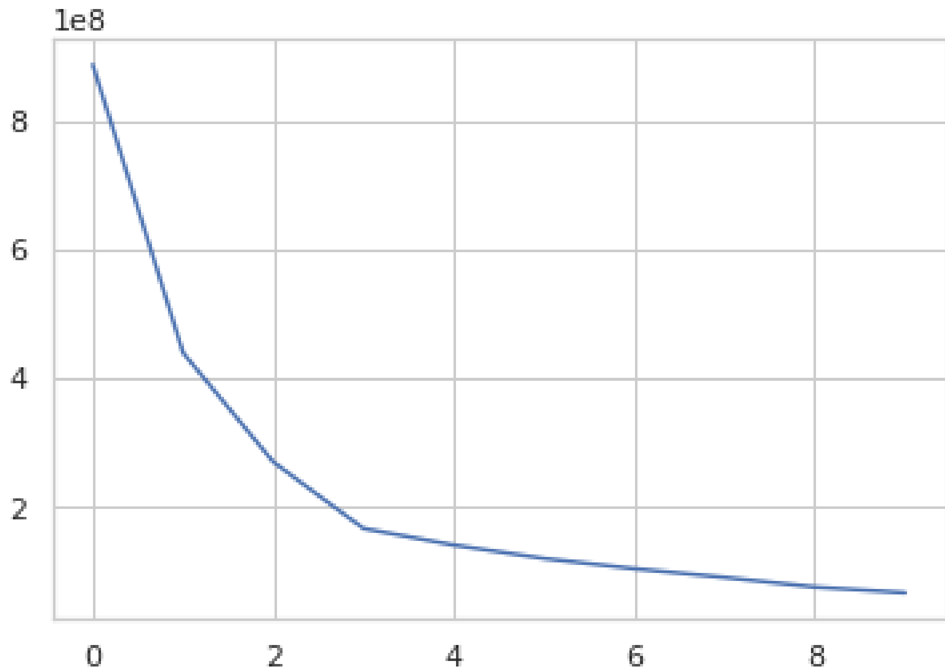
[242]:
```
testing_df
```

[242]:

	0
0	8.865459e+08
1	4.394324e+08
2	2.698899e+08
3	1.667261e+08
4	1.409684e+08
5	1.201605e+08
6	1.042696e+08
7	9.077022e+07
8	7.578023e+07
9	6.699148e+07

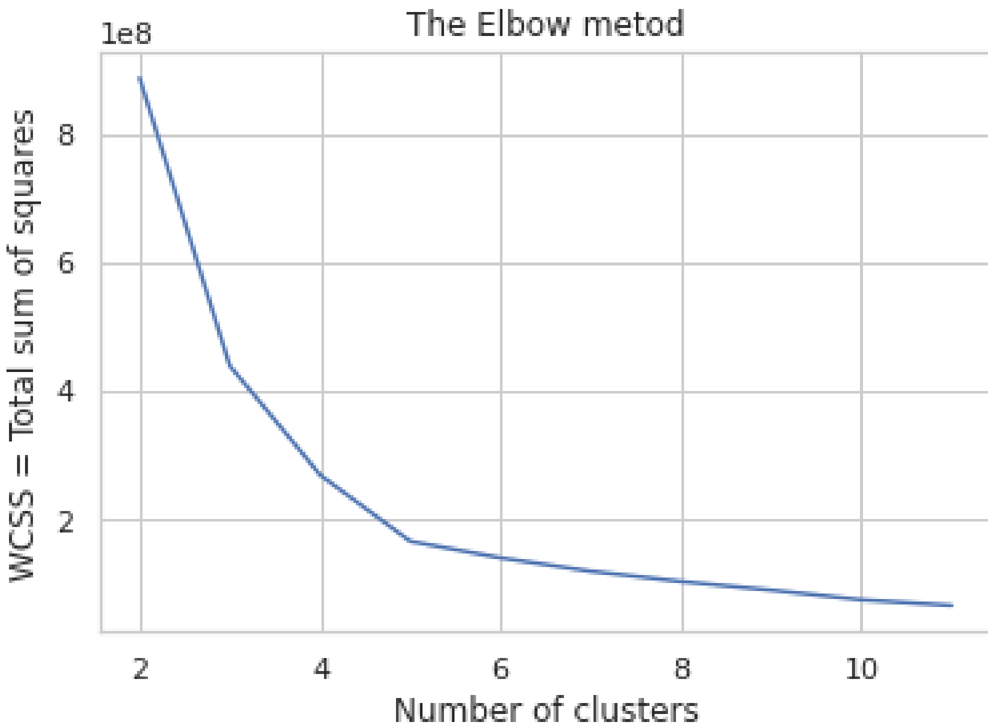
[243]:
```
#pd.Series(wcss).plot.bar(x='k=#clusters', y='Total sum of square error')
pd.Series(wcss).plot()
plt.title('')
```

[243]: Text(0.5, 1.0, '')



[244]:

```
# visualizziamo graficamente come varia wcss rispetto a k
plt.plot(range(2,12),wcss)
plt.title('The Elbow metod')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS = Total sum of squares')
plt.grid('on')
plt.show()
```



[245]:
```
kmeans = KMeans(n_clusters = 15, init='k-means++', max_iter=300)
```

[246]:
```
y_kmeans = kmeans.fit_predict(x)
print(y_kmeans)
```

```
[10 14  3 …  5  6  5]
```

[247]:
```
df_clusters_copy = df.copy()
```

[248]:
```
df_clusters_copy['Cluster'] = KMeans(n_clusters=6).fit_predict(x) + 1
#print('Silhouette Score:', silhouette_score(x, df_clusters_copy['Cluster'],␣
↪sample_size=10000) * 10000 // 1 / 100, '%')


#Silhouette Score: 45.26 %
#L'analisi della sagoma può essere utilizzata per studiare la distanza di␣
↪separazione tra i cluster risultanti. Il grafico della silhouette mostra una␣
↪misura di quanto è vicino ogni punto in un cluster ai punti nei cluster␣
↪vicini e fornisce quindi un modo per valutare visivamente parametri come il␣
↪numero di cluster. Questa misura ha un intervallo di [-1, 1].

#I coefficienti di sagoma (come vengono chiamati questi valori) vicino a +1
```