**基于XXX的XXX方案**

**摘要**

第一段：

首先对标题的内涵进行延伸，然后对论文一句话总括。

中间段：

分问题进行介绍，顺序依次为问题简介🡪方法🡪模型🡪算法🡪结果，具体按照下述模板：

对于XXX问题，利用XXX方法（或理论），建立了XXX模型，设计了XXX算法，得到了XXX结果。

最后一段：

可以对论文的其他内容进行介绍，也可以对论文所做的深入分析进行介绍。当然也可以使用一个有特色的结尾，将论文的模型和算法的性能之类的数据进行总结。

**关键词：** XXX XXX XXX XXX XXX（3-5个）

**一、问题重述**

**二、问题分析**

**三、符号说明**

表3.1.1 符号说明

|  |  |  |
| --- | --- | --- |
| 符号 | 含义 | 单位 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**四、模型假设**

**五、模型建立与求解**

**5.1 问题一**

**5.1.1 问题一数据预处理**

**5.1.2 问题一模型建立**

**5.1.3 问题一模型求解**

**5.1.4 问题一深入分析**

**5.2 问题二**

**5.2.1 问题二数据预处理**

数据预处理包含数据清洗和特征处理两个部分。

1.数据清洗。数据清洗旨在将原始数据进行全局清洗。其主要目的是消除数据中的错误、冗余和噪声，确保数据的完整性和标准数值化，从而有利于下一步的数据分析。数据清洗的步骤包括数据剔除，数据完整化，数据数值化和数据标准化。这些步骤的详细意义和操作如下文所述。

**数据剔除：**数据剔除的目的是将原始数据中与所有任务都不相关的错误与冗余数据全部去除。分为两个方向：

（一）纵向特征剔除。剔除与所有任务都不相关的纵向特征列。具体包含了表示数据来源的特征项（scite1, scite2,scite3,dbsource），与数值编码重复了的描述性语句的特征项（如targtype1\_txt 与targtype1意义重复了，且targtype1已经数值化了，所以targtype1\_txt可以删除），其他的一些不方便使用的描述性语句（如location, summary等）。具体见表5-2-1-1

表5-2-1-1

|  |  |
| --- | --- |
| 处理方法 | 对应属性列表 |
| 直接删除，  完全不相关属性 | approxdate,provstate,gsubname,gname2,gsubname2,gname3,gsubname3,divert,kidhijcountry,corp3,target3,resolution,scite1,scite2,scite3,dbsource' |
| 直接删除，  数据过于稀疏 | crit1,crit2,crit3,corp1,target1,corp2,target2 |
| 可删除，  描述性语句 | addnotes,ransomnote,propcomment,weaptype1, weaptype1\_txt, weapsubtype1\_txt,weaptype2\_txt, weapsubtype2\_txt, weaptype3\_txt, weapsubtype3\_txt, weaptype4\_txt, weapsubtype4\_txt,propextent\_txt,hostkidoutcome\_txt,country\_txt,alternative\_txt,attacktype1\_txt, attacktype2\_txt, attacktype3\_txt, targtype1\_txt,targsubtype1\_txt,natlty1\_txt, targtype2\_txt, targsubtype2\_txt, natlty2\_txt, targtype3\_txt, targsubtype3\_txt,natlty3\_txt,claimmode\_txt,claimmode2\_txt,claimmode3\_txt,region\_txt,location, summary, weapdetail |

(二) 横向样本剔除。将基本上认为该事件不是恐怖主义行为的样本删除。即如果某事件的疑似恐怖主义（doubtterr）标记为‘1’，则表示基本上不怀疑该事件是恐怖主义行为，所以该样本属于错误样本，可以直接删除，这样更加有利于下一步的数据分析。

**数据完整化：**所有特征项中存在数据缺失的，若该列数据中有‘Unknown’项则空缺项标记为‘unknown’，若该列数据中没有有‘Unknown’项则空缺项标记为‘NaN’。

**数据数值化：**由于在后续数据分析中主要采用的是xgboost算法，而且本案例中的非数值数据是离散的，没有数值大小意义的，所以这里的数值化编码可以采用二进制编码格式。具体做法为：

对于非数值数据，如一个单词或一段描述性语句的字符串，先统计该特征项总共有多少个分布值f\_num, 然后一一对应地编号至Num = [0…f\_num], 再按照二进制编码格式将Num编码转化为Binary\_Num。

**数据标准化**：对于‘unknown’项，统一转化为数值‘-99’。

2. 特征处理

特征处理包含数值处理、one-hot二进制编码、时间处理和文本处理。

**数值处理：**

(一) 幅度调整/归一化：

使用两种方法将数据进行归一化：

**Min-max 归一化：**



min-max标准化方法是对原始数据进行线性变换。设minA和maxA分别为属性A的最小值和最大值，将A的一个原始值x通过min-max标准化映射成在区间[0,1]中的值x’，其公式为:

新数据=(原数据-最小值)/(最大值-最小值)

**z-score 标准化：**

这种方法基于原始数据的均值(mean)和标准差(standard deviation)进行数据的标准化。将A的原始值x使用z-score标准化到x’。z-score标准化方法适用于属性A的最大值和最小值未知的情况，或有超出取值范围的离群数据的情况。将数据按其属性(按列进行)减去其均值，然后除以其方差。最后得到的结果是，对每个属性/每列来说所有数据都聚集在0附近，方差值为1。



其中μ为数据的均值，δ为方差。

（二）统计值：包括max, min, mean, std等。使用pandas库序列化数据后，可以得到数据的统计值。

（三）离散化：把连续值转成非线性数据。例如从0.03到100的连续值，用99维的向量代表每一个维度所处的区间，即1.2元和1.6的向量都是 [0,1,0,…,0]。

**One-hot二进制编码：**

由于在后续数据分析中主要采用的是xgboost算法，而且本案例中的非数值数据是离散的，没有数值大小意义的，所以这里的数值化编码可以采用二进制编码格式。具体做法为：

对于非数值数据，如一个单词或一段描述性语句的字符串，先统计该特征项总共有多少个分布值f\_num, 然后一一对应地编号至Num = [0…f\_num], 再按照二进制编码格式将Num编码转化为Binary\_Num。

**时间处理：**

使用calendar的weekday方法将iyear, imonth, iday三列数据转换为星期数据并构造’xingqi’列，将日期转换为星期。例如iyear=2015,imonth=1,iday=1构造xingqi=5。

**文本处理：**

（一）词袋：文本数据预处理后，去掉停用词，剩下的词组成的list，在词库中的映射稀疏向量。

（二）把词袋中的词扩充到n-gram：n-gram代表n个词的组合。

（三）使用TF-IDF特征：TF-IDF是一种统计方法，用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。TF(t) = (词t在当前文中出现次数) / (t在全部文档中出现次数)，IDF(t) = ln(总文档数/ 含t的文档数)，TF-IDF权重 = TF(t) \* IDF(t)。

**5.2.2 问题二模型建立**

**5.2.3 问题二模型求解**

**5.2.4 问题二深入分析**

**六、模型优缺点及推广**

**七、参考文献**

[1] 姜启源、谢金星，《数学模型》，北京：高等教育出版社，2007

[2]

[3]

[4]

[5]

**附录：**

Q1预处理代码：

#读取原数据

import pandas as pd

import numpy as np

terror=pd.read\_csv('../input/raw/globalterrorismdb.csv',encoding='ISO-8859-1')

# 全局数据清洗的步骤包括数据剔除，数据完整化，数据数值化，数据标准化。

# 1.数据剔除 纵向特征剔除。

#删除一些无关列

print("数据原columns")

print(terror.columns.tolist())

#terror.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)

#terror=terror[['Year','Month','Day','Country','Region','city','latitude','longitude','AttackType','Killed','Wounded','Target','Summary','Group','Target\_type','Weapon\_type','Motive']]

#直接删除，完全不相关

terror.drop(['approxdate','scite1','scite2','scite3','dbsource'], axis=1,inplace=True)

#可删除，描述性语句

terror.drop(['addnotes','ransomnote','propcomment','weaptype1', 'weaptype1\_txt', 'weapsubtype1\_txt',

'weaptype2\_txt', 'weapsubtype2\_txt', 'weaptype3\_txt',

'weapsubtype3\_txt', 'weaptype4\_txt', 'weapsubtype4\_txt','propextent\_txt','hostkidoutcome\_txt','country\_txt',

'alternative\_txt','attacktype1\_txt', 'attacktype2\_txt', 'attacktype3\_txt', 'targtype1\_txt','targsubtype1\_txt',

'natlty1\_txt', 'targtype2\_txt', 'targsubtype2\_txt', 'natlty2\_txt', 'targtype3\_txt', 'targsubtype3\_txt',

'natlty3\_txt','claimmode\_txt','claimmode2\_txt','claimmode3\_txt','region\_txt','location', 'summary'

], axis=1,inplace=True)

#可能有用的，描述性语句，暂时删除

terror.drop('weapdetail', axis=1,inplace=True)

print("删除后columns")

print(terror.columns.tolist())

# 1.数据剔除 横向样本剔除。

terror.doubtterr.dtypes

print(terror.size)

doubtterrlist = terror[(terror.doubtterr==1)].index.tolist()

terror = terror.drop(doubtterrlist)

print(terror.size)

terror.head(20)

# 数据完整化

terror.dtypes

# terror['hostkidoutcome'].head(10)

# terrorlist = terror.columns.tolist()

print(terrorlist)

def nan2neg99(x):

if np.isnan(x):

return -99

else:

return x

for column\_str in terrorlist:

colum\_type = terror[column\_str].dtypes

# print(colum\_type)

if(colum\_type=='float64' or colum\_type=='int64'):

# print("here")

terror[column\_str] = terror[column\_str].map(lambda x:nan2neg99(x))

# terror[terrorlist].

terror['hostkidoutcome'].head(10)

#数据数值化

# terror['gname'].head(100)

count = 1

gname\_dict = {}

gnamelist = terror['gname'].tolist()

for gname in gnamelist:

if gname not in gname\_dict:

string\_front = ''

for \_ in range(11 - len(bin(count)[2:])):

string\_front += '0'

gname\_dict[gname] = string\_front+bin(count)[2:]

# print(string\_front+bin(count)[2:])

count+=1

print(gname\_dict)

print(count)

terror['gname'] = terror['gname'].map(lambda x:gname\_dict[x])

#新增列

terror['casulity']=terror['nkill']+terror['nwound']

region\_level = {1:4, 2:1, 3:3, 4:3, 5:1, 6:1, 7:1, 8:4, 9:3, 10:2, 11:1, 12:4 }

# targtype1划分为四类：公共环境类（1 5 6 8 10 14 21）生活设施类（9 11 16 18 19）

# 政府组织类（2 3 4 7）其它机构类（12 13 15 17 20 22），分别赋值为4,3,2,1

targtype1\_level = {1:4, 5:4, 6:4, 8:4, 10:4, 14:4, 21:4, 9:3, 11:3, 16:3, 18:3, 19:3, 2:2, 3:2, 4:2, 7:2, 12:1, 13:1,

15:1, 17:1, 20:1, 22:1}

# terror['regionlevel']=terror[region\_level[terror['region']]]

terror['regionlevel'] = terror['region'].map(lambda x:region\_level[x])

terror['targtype1level'] = terror['targtype1'].map(lambda x:targtype1\_level[x])

def nan2four(x):

if np.isnan(x):

return 4

else:

return x

def nan2zero(x):

if np.isnan(x):

return 0

else:

return x

# nan2zero

terror['casulity'] = terror['casulity'].map(lambda x:nan2zero(x))

terror['nhostkid'] = terror['nhostkid'].map(lambda x:nan2zero(x))

# nan2four

terror['propvalue'] = terror['propvalue'].map(lambda x:nan2four(x))

terror['propextent'] = terror['propextent'].map(lambda x:nan2four(x))

#terror['propextent'] reverse

propextent\_reverse = {4.0:1.0,3.0:2.0,2.0:3.0,1.0:4.0}

terror['propextent'] = terror['propextent'].map(lambda x:propextent\_reverse[x])

# min-max normalize

terror['propextent'] = (terror['propextent'] - terror['propextent'].min()) / (terror['propextent'].max() - terror['propextent'].min())

terror['regionlevel'] = (terror['propextent'] - terror['propextent'].min()) / (terror['propextent'].max() - terror['propextent'].min())

terror['nhostkid'] = (terror['propextent'] - terror['propextent'].min()) / (terror['propextent'].max() - terror['propextent'].min())

terror['targtype1level'] = (terror['propextent'] - terror['propextent'].min()) / (terror['propextent'].max() - terror['propextent'].min())

# Z-score normalize

terror['casulity'] = (terror['casulity'] - terror['casulity'].mean()) / (terror['casulity'].std())

terror.head(5)

Q2 = terror[['eventid', 'iyear','propextent', 'casulity', 'regionlevel', 'nhostkid',

'targtype1level','gname']]

Q2.head(100)

# Q2.to\_csv('../input/raw/Q2.csv')

Q2预处理代码：

#读取原数据

import pandas as pd

import numpy as np

terror=pd.read\_csv('../input/raw/globalterrorismdb.csv',encoding='ISO-8859-1')

# 全局数据清洗的步骤包括数据剔除，数据完整化，数据数值化，数据标准化。

# 1.数据剔除 纵向特征剔除。

#删除一些无关列

print("数据原columns")

print(terror.size)

print(terror.columns.tolist())

#terror.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)

#terror=terror[['Year','Month','Day','Country','Region','city','latitude','longitude','AttackType','Killed','Wounded','Target','Summary','Group','Target\_type','Weapon\_type','Motive']]

#直接删除，完全不相关

terror.drop(['approxdate','provstate','city','crit1','crit2','crit3','corp1', 'target1','corp2', 'target2',

'gsubname','gname2','gsubname2','gname3','gsubname3','divert','kidhijcountry',

'corp3', 'target3','resolution','scite1','scite2','scite3','dbsource'], axis=1,inplace=True)

#可删除，描述性语句

terror.drop(['addnotes','ransomnote','propcomment','weaptype1', 'weaptype1\_txt', 'weapsubtype1\_txt',

'weaptype2\_txt', 'weapsubtype2\_txt', 'weaptype3\_txt',

'weapsubtype3\_txt', 'weaptype4\_txt', 'weapsubtype4\_txt','propextent\_txt','hostkidoutcome\_txt','country\_txt',

'alternative\_txt','attacktype1\_txt', 'attacktype2\_txt', 'attacktype3\_txt', 'targtype1\_txt','targsubtype1\_txt',

'natlty1\_txt', 'targtype2\_txt', 'targsubtype2\_txt', 'natlty2\_txt', 'targtype3\_txt', 'targsubtype3\_txt',

'natlty3\_txt','claimmode\_txt','claimmode2\_txt','claimmode3\_txt','region\_txt','location', 'summary'

], axis=1,inplace=True)

#可能有用的，描述性语句，暂时删除

terror.drop('weapdetail', axis=1,inplace=True)

print("删除后columns")

print(terror.columns.tolist())

# 1.数据剔除 横向样本剔除。

# terror.doubtterr.dtypes

print(terror.size)

doubtterrlist = terror[(terror.doubtterr==1)].index.tolist()

terror = terror.drop(doubtterrlist)

print(terror.size)

# terror.head(20)

terror.drop('doubtterr', axis=1,inplace=True)

# 去除nan

terror['casulity']=terror['nkill']+terror['nwound']

def numoflist(x):

# print(x)

# print(type(x))

if(isinstance(x,float)):

return 0

else:

return len(x.split(","))

terror['related\_num'] = terror['related'].map(lambda x:numoflist(x))

terror.drop('related', axis=1,inplace=True)

def nan2neg99(x):

if np.isnan(x):

return -99

else:

return x

terrorlist = terror.columns.tolist()

for column\_str in terrorlist:

colum\_type = terror[column\_str].dtypes

# print(colum\_type)

if(colum\_type=='float64' or colum\_type=='int64'):

# print("here")

terror[column\_str] = terror[column\_str].map(lambda x:nan2neg99(x))

# terror[terrorlist].

# terror.head(20)

terror = terror

print(terror.size)

terror.head(20)

# 时间

# chen=pd.read\_csv('/home/xyx/Downloads/shumo/input/raw/globalterrorismdb.csv',encoding='ISO-8859-1')

import calendar

from datetime import datetime

# terror=terror[list(range(21))]

# maxindex=19

# test=terror.head(maxindex+1)

# xingqi=[]

# for i in range(maxindex+1):

# year=test.iat[i,1]

# month=test.iat[i,2]

# day=test.iat[i,3]

# xingqi.append(calendar.weekday(year,month,day))

# test['xingqi']=xingqi

# test.head(3)

terror.head(5)

# calendar.weekday(int(row['iyear']), int(row['imonth']), int(row['iday'])), axis=1)

# terror['xingqi'] = terror.apply(lambda row: my\_test(row[imonth],row[iday]))

terror['xingqi'] = terror.apply(lambda x: calendar.weekday(x.imonth,x.iday))

terror['xingqi'].head(5)

# terror.head(5)

#数据数值化

# ##########################

# gname

# terror['gname'].head(100)

count = 1

gname\_dict1 = {}

gname\_dict2 = {}

gnamelist = terror['gname'].tolist()

# print(gnamelist)

for gname in gnamelist:

if gname not in gname\_dict2:

string\_front = ''

for \_ in range(11 - len(bin(count)[2:])):

string\_front += '0'

gname\_dict2[gname] = string\_front+bin(count)[2:]

gname\_dict1[gname] = count

# print(string\_front+bin(count)[2:])

count+=1

print(gname\_dict1)

print(count)

terror['gname\_strencode'] = terror['gname'].map(lambda x:gname\_dict2[x])

for i in range(11):

terror['gname\_2encode\_'+str(i)] = terror['gname\_strencode'].map(lambda x:x[i])

##########################

motive

def nan2unknow(x):

if x=='NaN' or x=='nan':

return 'Unknown'

else:

return x

# for motive in motivelist:

# colum\_type = terror[motive].dtypes

# # print(colum\_type)

# if(colum\_type=='float64' or colum\_type=='int64'):

# # print("here")

# terror[column\_str] = terror[column\_str].map(lambda x:nan2neg99(x))

# terror['motive'] = terror['motive']

terror['motive'] = terror['motive'].map(lambda x:str(x))

terror['motive'] = terror['motive'].map(lambda x:nan2unknow(x))

# terror['motive'].head(40)

count = 0

motive\_dict1 = {}

motive\_dict2 = {}

motivelist = terror['motive'].tolist()

# print(motivelist)

# print(len(motivelist))

for motive in motivelist:

if motive not in motive\_dict2:

string\_front = ''

for \_ in range(11 - len(bin(count)[2:])):

string\_front += '0'

motive\_dict2[gname] = string\_front+bin(count)[2:]

motive\_dict1[gname] = count

# print(string\_front+bin(count)[2:])

count+=1

print(motive\_dict2)

print(motive\_dict1)

print(count)

terror['motive\_strencode'] = terror['motive'].map(lambda x:motive\_dict2[x])

for i in range(11):

terror['motive\_2encode\_'+str(i)] = terror['motive\_strencode'].map(lambda x:x[i])

# #########################

# terror.head(20)

terror['motive\_10encode'] = terror['motive'].map(lambda x:motive\_dict1[x])

terror.drop('motive', axis=1,inplace=True)

terror['gname\_10encode'] = terror['gname'].map(lambda x:gname\_dict1[x])

terror.drop('gname', axis=1,inplace=True)

terror.head(20)

terror = terror.sort\_values(by="gname\_strencode")

terror.to\_csv('../input/raw/Q2\_1.csv')

Q3预处理代码：

print('受到恐怖袭击次数最多的国家:',terror['country\_txt'].value\_counts().index[0])

print('受到恐怖袭击次数最多的地区:',terror['region\_txt'].value\_counts().index[0])

print('在一次恐怖袭击中死亡人数最多的事件:',terror['nkill'].max(),'that took place in',terror.loc[terror['nkill'].idxmax(),'country\_txt'])

plt.subplots(figsize=(15,6))

sns.countplot('iyear',data=terror,palette='RdYlGn\_r',edgecolor=sns.color\_palette('dark',7))

plt.xticks(rotation=90)

plt.title('Number Of Terrorist Activities Each Year')

plt.show()

plt.subplots(figsize=(15,6))

sns.countplot('attacktype1\_txt',data=terror,palette='inferno',order=terror['attacktype1\_txt'].value\_counts().index)

plt.xticks(rotation=90)

plt.title('Attacking Methods by Terrorists')

plt.show()

plt.subplots(figsize=(15,6))

sns.countplot(terror['targtype1\_txt'],palette='inferno',order=terror['targtype1\_txt'].value\_counts().index)

plt.xticks(rotation=90)

plt.title('Favorite Targets')

plt.show()

m3 = Basemap(projection='mill',llcrnrlat=-80,urcrnrlat=80, llcrnrlon=-180,urcrnrlon=180,lat\_ts=20,resolution='c',lat\_0=True,lat\_1=True)

lat\_100=list(terror[terror['casualities']>=75].latitude)

long\_100=list(terror[terror['casualities']>=75].longitude)

x\_100,y\_100=m3(long\_100,lat\_100)

m3.plot(x\_100, y\_100,'go',markersize=5,color = 'r')

lat\_=list(terror[terror['casualities']<75].latitude)

long\_=list(terror[terror['casualities']<75].longitude)

x\_,y\_=m3(long\_,lat\_)

m3.plot(x\_, y\_,'go',markersize=2,color = 'b',alpha=0.4)

m3.drawcoastlines()

m3.drawcountries()

m3.fillcontinents(lake\_color='aqua')

m3.drawmapboundary(fill\_color='aqua')

fig=plt.gcf()

fig.set\_size\_inches(10,6)

plt.title('Global Terrorist Attacks')

plt.legend(loc='lower left',handles=[mpatches.Patch(color='b', label = "< 75 casualities"),

mpatches.Patch(color='red',label='> 75 casualities')])

plt.show()

terror\_fol=terror.copy()

terror\_fol.dropna(subset=['latitude','longitude'],inplace=True)

location\_fol=terror\_fol[['latitude','longitude']][:5000]

country\_fol=terror\_fol['country'][:5000]

city\_fol=terror\_fol['city'][:5000]

killed\_fol=terror\_fol['nkill'][:5000]

wound\_fol=terror\_fol['nwound'][:5000]

def color\_point(x):

if x>=30:

color='red'

elif ((x>0 and x<30)):

color='blue'

else:

color='green'

return color

def point\_size(x):

if (x>30 and x<100):

size=2

elif (x>=100 and x<500):

size=8

elif x>=500:

size=16

else:

size=0.5

return size

map2 = folium.Map(location=[30,0],tiles='CartoDB dark\_matter',zoom\_start=2)

for point in location\_fol.index:

info='<b>Country: </b>'+str(country\_fol[point])+'<br><b>City: </b>: '+str(city\_fol[point])+'<br><b>Killed </b>: '+str(killed\_fol[point])+'<br><b>Wounded</b> : '+str(wound\_fol[point])

iframe = folium.IFrame(html=info, width=200, height=200)

folium.CircleMarker(list(location\_fol.loc[point].values),popup=folium.Popup(iframe),radius=point\_size(killed\_fol[point]),color=color\_point(killed\_fol[point])).add\_to(map2)

map2

plt.subplots(figsize=(15,6))

sns.countplot('region\_txt',data=terror,palette='RdYlGn',edgecolor=sns.color\_palette('dark',7),order=terror['region\_txt'].value\_counts().index)

plt.xticks(rotation=90)

plt.title('Number Of Terrorist Activities By Region')

plt.show()

terror\_region=pd.crosstab(terror.iyear,terror.region\_txt)

terror\_region.plot(color=sns.color\_palette('Set2',12))

fig=plt.gcf()

fig.set\_size\_inches(18,6)

plt.show()

pd.crosstab(terror.region,terror.attacktype1\_txt).plot.barh(stacked=True,width=1,color=sns.color\_palette('RdYlGn',9))

fig=plt.gcf()

fig.set\_size\_inches(20,8)

plt.show()

plt.subplots(figsize=(18,6))

sns.barplot(terror['country\_txt'].value\_counts()[:15].index,terror['country\_txt'].value\_counts()[:15].values,palette='inferno')

plt.title('Top Affected Countries')

plt.show()

coun\_terror=terror['country\_txt'].value\_counts()[:15].to\_frame()

coun\_terror.columns=['Attacks']

coun\_kill=terror.groupby('country\_txt')['nkill'].sum().to\_frame()

coun\_terror.merge(coun\_kill,left\_index=True,right\_index=True,how='left').plot.bar(width=0.9)

fig=plt.gcf()

fig.set\_size\_inches(18,6)

plt.show()

l1=list(['Afghanistan', 'Albania', 'Algeria', 'American Samoa', 'Andorra','Angola', 'Anguilla', 'Antigua and Barbuda', 'Argentina', 'Armenia','Aruba', 'Australia', 'Austria', 'Azerbaijan', 'Bahamas, The',

'Bahrain', 'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize','Benin', 'Bermuda', 'Bhutan', 'Bolivia', 'Bosnia and Herzegovina',

'Botswana', 'Brazil', 'British Virgin Islands', 'Brunei','Bulgaria', 'Burkina Faso', 'Burma', 'Burundi', 'Cabo Verde','Cambodia', 'Cameroon', 'Canada', 'Cayman Islands',

'Central African Republic', 'Chad', 'Chile', 'China', 'Colombia','Comoros', 'Congo, Democratic Republic of the','Congo, Republic of the', 'Cook Islands', 'Costa Rica',"Cote d'Ivoire", 'Croatia', 'Cuba', 'Curacao', 'Cyprus','Czech Republic', 'Denmark', 'Djibouti', 'Dominica','Dominican Republic', 'Ecuador', 'Egypt', 'El Salvador',

'Equatorial Guinea', 'Eritrea', 'Estonia', 'Ethiopia','Falkland Islands (Islas Malvinas)', 'Faroe Islands', 'Fiji','Finland', 'France', 'French Polynesia', 'Gabon', 'Gambia, The',

'Georgia', 'Germany', 'Ghana', 'Gibraltar', 'Greece', 'Greenland','Grenada', 'Guam', 'Guatemala', 'Guernsey', 'Guinea-Bissau','Guinea', 'Guyana', 'Haiti', 'Honduras', 'Hong Kong', 'Hungary','Iceland', 'India', 'Indonesia', 'Iran', 'Iraq', 'Ireland','Isle of Man', 'Israel', 'Italy', 'Jamaica', 'Japan', 'Jersey','Jordan', 'Kazakhstan', 'Kenya', 'Kiribati', 'Korea, North',

'Korea, South', 'Kosovo', 'Kuwait', 'Kyrgyzstan', 'Laos', 'Latvia','Lebanon', 'Lesotho', 'Liberia', 'Libya', 'Liechtenstein',

'Lithuania', 'Luxembourg', 'Macau', 'Macedonia', 'Madagascar','Malawi', 'Malaysia', 'Maldives', 'Mali', 'Malta','Marshall Islands', 'Mauritania', 'Mauritius', 'Mexico','Micronesia, Federated States of', 'Moldova', 'Monaco', 'Mongolia',

'Montenegro', 'Morocco', 'Mozambique', 'Namibia', 'Nepal','Netherlands', 'New Caledonia', 'New Zealand', 'Nicaragua','Nigeria', 'Niger', 'Niue', 'Northern Mariana Islands', 'Norway',

'Oman', 'Pakistan', 'Palau', 'Panama', 'Papua New Guinea','Paraguay', 'Peru', 'Philippines', 'Poland', 'Portugal','Puerto Rico', 'Qatar', 'Romania', 'Russia', 'Rwanda','Saint Kitts and Nevis', 'Saint Lucia', 'Saint Martin','Saint Pierre and Miquelon', 'Saint Vincent and the Grenadines',

'Samoa', 'San Marino', 'Sao Tome and Principe', 'Saudi Arabia','Senegal', 'Serbia', 'Seychelles', 'Sierra Leone', 'Singapore','Sint Maarten', 'Slovakia', 'Slovenia', 'Solomon Islands',

'Somalia', 'South Africa', 'South Sudan', 'Spain', 'Sri Lanka','Sudan', 'Suriname', 'Swaziland', 'Sweden', 'Switzerland', 'Syria','Taiwan', 'Tajikistan', 'Tanzania', 'Thailand', 'Timor-Leste',

'Togo', 'Tonga', 'Trinidad and Tobago', 'Tunisia', 'Turkey','Turkmenistan', 'Tuvalu', 'Uganda', 'Ukraine','United Arab Emirates', 'United Kingdom', 'United States','Uruguay', 'Uzbekistan', 'Vanuatu', 'Venezuela', 'Vietnam',

'Virgin Islands', 'West Bank', 'Yemen', 'Zambia', 'Zimbabwe']) #Country names

l2=list(['AFG', 'ALB', 'DZA', 'ASM', 'AND', 'AGO', 'AIA', 'ATG', 'ARG','ARM', 'ABW', 'AUS', 'AUT', 'AZE', 'BHM', 'BHR', 'BGD', 'BRB',

'BLR', 'BEL', 'BLZ', 'BEN', 'BMU', 'BTN', 'BOL', 'BIH', 'BWA','BRA', 'VGB', 'BRN', 'BGR', 'BFA', 'MMR', 'BDI', 'CPV', 'KHM',

'CMR', 'CAN', 'CYM', 'CAF', 'TCD', 'CHL', 'CHN', 'COL', 'COM','COD', 'COG', 'COK', 'CRI', 'CIV', 'HRV', 'CUB', 'CUW', 'CYP','CZE', 'DNK', 'DJI', 'DMA', 'DOM', 'ECU', 'EGY', 'SLV', 'GNQ',

'ERI', 'EST', 'ETH', 'FLK', 'FRO', 'FJI', 'FIN', 'FRA', 'PYF','GAB', 'GMB', 'GEO', 'DEU', 'GHA', 'GIB', 'GRC', 'GRL', 'GRD','GUM', 'GTM', 'GGY', 'GNB', 'GIN', 'GUY', 'HTI', 'HND', 'HKG',

'HUN', 'ISL', 'IND', 'IDN', 'IRN', 'IRQ', 'IRL', 'IMN', 'ISR','ITA', 'JAM', 'JPN', 'JEY', 'JOR', 'KAZ', 'KEN', 'KIR', 'KOR',

'PRK', 'KSV', 'KWT', 'KGZ', 'LAO', 'LVA', 'LBN', 'LSO', 'LBR','LBY', 'LIE', 'LTU', 'LUX', 'MAC', 'MKD', 'MDG', 'MWI', 'MYS','MDV', 'MLI', 'MLT', 'MHL', 'MRT', 'MUS', 'MEX', 'FSM', 'MDA',

'MCO', 'MNG', 'MNE', 'MAR', 'MOZ', 'NAM', 'NPL', 'NLD', 'NCL','NZL', 'NIC', 'NGA', 'NER', 'NIU', 'MNP', 'NOR', 'OMN', 'PAK','PLW', 'PAN', 'PNG', 'PRY', 'PER', 'PHL', 'POL', 'PRT', 'PRI','QAT', 'ROU', 'RUS', 'RWA', 'KNA', 'LCA', 'MAF', 'SPM', 'VCT','WSM', 'SMR', 'STP', 'SAU', 'SEN', 'SRB', 'SYC', 'SLE', 'SGP',

'SXM', 'SVK', 'SVN', 'SLB', 'SOM', 'ZAF', 'SSD', 'ESP', 'LKA','SDN', 'SUR', 'SWZ', 'SWE', 'CHE', 'SYR', 'TWN', 'TJK', 'TZA',

'THA', 'TLS', 'TGO', 'TON', 'TTO', 'TUN', 'TUR', 'TKM', 'TUV','UGA', 'UKR', 'ARE', 'GBR', 'USA', 'URY', 'UZB', 'VUT', 'VEN',

'VNM', 'VGB', 'WBG', 'YEM', 'ZMB', 'ZWE']) #Country Codes

sns.barplot(terror['gname'].value\_counts()[1:15].values,terror['gname'].value\_counts()[1:15].index,palette=('inferno'))

plt.xticks(rotation=90)

fig=plt.gcf()

fig.set\_size\_inches(10,8)

plt.title('Terrorist Groups with Highest Terror Attacks')

plt.show()

top\_groups10=terror[terror['gname'].isin(terror['gname'].value\_counts()[1:11].index)]

pd.crosstab(top\_groups10.iyear,top\_groups10.gname).plot(color=sns.color\_palette('Paired',10))

fig=plt.gcf()

fig.set\_size\_inches(18,6)

plt.show()

top\_groups=terror[terror['gname'].isin(terror['gname'].value\_counts()[:14].index)]

m4 = Basemap(projection='mill',llcrnrlat=-80,urcrnrlat=80, llcrnrlon=-180,urcrnrlon=180,lat\_ts=20,resolution='c',lat\_0=True,lat\_1=True)

m4.drawcoastlines()

m4.drawcountries()

m4.fillcontinents(lake\_color='aqua')

m4.drawmapboundary(fill\_color='aqua')

fig=plt.gcf()

fig.set\_size\_inches(22,10)

colors=['r','g','b','y','#800000','#ff1100','#8202fa','#20fad9','#ff5733','#fa02c6',"#f99504",'#b3b6b7','#8e44ad','#1a2b3c']

group=list(top\_groups['gname'].unique())

def group\_point(group,color,label):

lat\_group=list(top\_groups[top\_groups['gname']==group].latitude)

long\_group=list(top\_groups[top\_groups['gname']==group].longitude)

x\_group,y\_group=m4(long\_group,lat\_group)

m4.plot(x\_group,y\_group,'go',markersize=3,color=j,label=i)

for i,j in zip(group,colors):

group\_point(i,j,i)

legend=plt.legend(loc='lower left',frameon=True,prop={'size':10})

frame=legend.get\_frame()

frame.set\_facecolor('white')

plt.title('Regional Activities of Terrorist Groups')

plt.show()

f,ax=plt.subplots(1,2,figsize=(25,12))

usa\_groups=terror\_usa['gname'].value\_counts()[1:11].index

usa\_groups=terror\_usa[terror\_usa['gname'].isin(usa\_groups)]

sns.countplot(y='gname',data=usa\_groups,ax=ax[0])

sns.countplot(y='attacktype1\_txt',data=terror\_usa,ax=ax[1])

plt.subplots\_adjust(hspace=0.3,wspace=0.6)

ax[0].set\_title('Top Terrorist Groups')

ax[1].set\_title('Favorite Attack Types')

ax[0].tick\_params(labelsize=15)

ax[1].tick\_params(labelsize=15)

plt.show()

fig = plt.figure(figsize = (10,8))

def animate(Year):

ax = plt.axes()

ax.clear()

ax.set\_title('Terrorism In USA '+'\n'+'Year:' +str(Year))

m6 = Basemap(llcrnrlon=-119,llcrnrlat=22,urcrnrlon=-64,urcrnrlat=49,

projection='lcc',lat\_1=33,lat\_2=45,lon\_0=-95)

lat\_gif1=list(terror\_usa[terror\_usa['iyear']==Year].latitude)

long\_gif1=list(terror\_usa[terror\_usa['iyear']==Year].longitude)

x\_gif1,y\_gif1=m6(long\_gif1,lat\_gif1)

m6.scatter(x\_gif1, y\_gif1,s=[killed+wounded for killed,wounded in zip(terror\_usa[terror\_usa['iyear']==Year].nkill,terror\_usa[terror\_usa['iyear']==Year].nwound)],color ='r')

m6.drawcoastlines()

m6.drawcountries()

m6.fillcontinents(color='coral',lake\_color='aqua', zorder = 1,alpha=0.4)

m6.drawmapboundary(fill\_color='aqua')

ani = animation.FuncAnimation(fig,animate,list(terror\_usa.iyear.unique()), interval = 1500)

ani.save('animation.gif', writer='imagemagick', fps=1)

plt.close(1)

filename = 'animation.gif'

video = io.open(filename, 'r+b').read()

encoded = base64.b64encode(video)

HTML(data='''<img src="data:image/gif;base64,{0}" type="gif" />'''.format(encoded.decode('ascii')))

import nltk

from wordcloud import WordCloud, STOPWORDS

motive=terror['motive'].str.lower().str.replace(r'\|', ' ').str.cat(sep=' ')

words=nltk.tokenize.word\_tokenize(motive)

word\_dist = nltk.FreqDist(words)

stopwords = nltk.corpus.stopwords.words('english')

f1=open("kaggle.png", "wb")

f1.write(codecs.decode(kaggle,'base64'))

f1.close()

img1 = imread("kaggle.png")

hcmask1 = img1

words\_except\_stop\_dist = nltk.FreqDist(w for w in words if w not in stopwords)

wordcloud = WordCloud(stopwords=STOPWORDS,background\_color='black',mask=hcmask1).generate(" ".join(words\_except\_stop\_dist))

plt.imshow(wordcloud)

fig=plt.gcf()

fig.set\_size\_inches(10,6)

plt.axis('off')

plt.show()

fig = plt.figure(figsize = (10,6))

def animate(iyear):

ax = plt.axes()

ax.clear()

ax.set\_title('Animation Of Terrorist Activities'+'\n'+'Year:' +str(iyear))

m6 = Basemap(projection='mill',llcrnrlat=-80,urcrnrlat=80, llcrnrlon=-180,urcrnrlon=180,lat\_ts=20,resolution='c')

lat6=list(terror[terror['iyear']==iyear].latitude)

long6=list(terror[terror['iyear']==iyear].longitude)

x6,y6=m6(long6,lat6)

m6.scatter(x6, y6,s=[(kill+wound)\*0.1 for kill,wound in zip(terror[terror['iyear']==iyear].nkill,terror[terror['iyear']==iyear].nwound)],color = 'r')

m6.drawcoastlines()

m6.drawcountries()

m6.fillcontinents(zorder = 1,alpha=0.4)

m6.drawmapboundary()

ani = animation.FuncAnimation(fig,animate,list(terror.iyear.unique()), interval = 1500)

ani.save('animation.gif', writer='imagemagick', fps=1)

plt.close(1)

filename = 'animation.gif'

video = io.open(filename, 'r+b').read()

encoded = base64.b64encode(video)

HTML(data='''<img src="data:image/gif;base64,{0}" type="gif" />'''.format(encoded.decode('ascii')))