AN INTERNSHIP REPORT ON

GLOBAL TERRORISM ANALYSIS AND DEATH OCCURRENCES PREDICTION

Submitted by

HARISH M (113219031150)

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

VELAMMAL ENGINEERING COLLEGE
CHENNAI -66

VELAMMAL
ENGINEERING COLLEGE
The Wheel of Knowledge rolls on!

BONAFIDE CERTIFICATE

Certified that this internship report **"GLOBAL TERRORISM ANALYSIS AND DEATH OCCURRENCES PREDICTION "** is the bonafide work of **HARISH M (113219031050)** carried out at **"MIT SQUARE, LONDON"** during 01.12.2021 to 31.01.2022

Dr. B MURUGESHWARI	**Mrs. R AMIRTHAVALLI**
PROFESSOR & HEAD	ASSISTANT PROFESSOR
Dept. of Computer Science and Engineering	Faculty Coordinator
Velammal Engineering College	Velammal Engineering College
Chennai –600 066	Chennai –600 066

MIT SQUARE

AN ISO 9001:2015 INTERNATIONAL CERTIFIED COMPANY

UK | USA | CANADA | SINGAPORE | MALAYSIA | AUSTRALIA | DUBAI | EUROPE

INTERNSHIP IN DATA SCIENCE

This certifies that

HARISH M

student from Velammal Engineering College has successfully completed two months of internship in the area of Data Science with our company, MIT Square Services Private Limited, from December 2021 to January 2022.

KNOWLEDGE IS POWER
100% CERTIFIED

S. Mithileysh

DR MITHILEYSH SATHIYANARAYANAN

FOUNDER | CEO | SCIENTIST
LONDON
WWW.MITSQUARE.COM

CERTIFICATE#202201DS37

CERTIFICATE OF EVALUATION

COLLEGE NAME : VELAMMAL ENGINEERING COLLEGE
BRANCH : COMPUTER SCIENCE & ENGINEERING
SEMESTER : VI

Sl. No	Name of the student who has done the Internship	Title of the Internship	Name of Faculty Coordinator with designation
1	**HARISH M**	**GLOBAL TERRORISM ANALYSIS AND DEATH OCCURRENCES PREDICTION**	**Mrs.R AMIRTHAVALLI** ASSISTANT PROFESSOR

This report of internship work submitted by the above student in partial fulfilment for the award of Bachelor of Computer Science & Engineering Degree in Anna University was evaluated and confirmed to be reports of the work done by the above student and then assessed.

Submitted for Internal Evaluation held on........................

Examiner 1 **Examiner 2** **Examiner 3**

ACKNOWLEDGEMENT

I wish to acknowledge with thanks to the significant contribution given by the management of our college Chairman, Dr.M.V.Muthuramalingam, and our Chief Executive Officer Thiru. M.V.M. Velmurugan, for their extensive support.

I would like to thank Dr. S. SATHISHKUMAR, Principal of Velammal Engineering College, for giving me this opportunity to do this project.

I wish to express my gratitude to our effective Head of the Department, Dr. B. Murugeshwari, for her moral support and for her valuable innovative suggestions, constructive interaction, constant encouragement and unending help that have enabled me to complete the project.

I wish to express my indebted humble thanks to our Project Coordinators,Mrs. R Amirthavalli, Department of Computer Science and Engineering for their invaluable guidance in shaping this project.

I wish to express my sincere gratitude to my Internal Guide, Dr. B. MURUGESHWARI, Head Of The Department, Professor, Department of Computer Science and Engineering for her guidance, without whom this project would not have been possible.

I am grateful to the entire staff members of the department of Computer Science and Engineering for providing the necessary facilities to carry out the project. I would especially like to thank my parents for providing me with the unique opportunity to work, and for their encouragement and support at all levels. Finally, my heartfelt thanks to The Almighty for guiding me throughout life

Table of contents

ABSTRACT

Terrorism, in its broadest sense, is the use of intentional violence to achieve political aims. The term is used in this regard primarily to refer to violence during peacetime or in the context of war against non-combatants (mostly civilians and neutral military personnel). Transnational terrorism poses a serious and prolonged threat to Every Country's national security. we have to prepare for a long-drawn campaign against terrorism, and we need to learn to live with the real prospect that a terrorist attack could occur in any country. A key challenge of understanding terrorism is both acknowledging the moral outrage at terrorist acts, while at the same time trying to understand the rationale behind terrorism. The damage may well be much wider than anything that could possibly be encapsulated in the concept of terrorism.

The U.S. Department of Defense defines Terrorism as, "The calculated use of violence or the threat of violence to inculcate fear; intended to coerce or to intimidate governments or societies in the pursuit of goals that are generally political, religious, or ideological."

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

TERMS	ABBREVIATION
ML	Machine Learning
CLF	Classifier
ROC	Receiver Operating Characteristic Curve
AUC	Area Under the curve
TPR	True Positive Rate
FRP	False Positive Rate

CHAPTER 1
INTRODUCTION

The causes of terrorism appear to be vary. There does not appear to be one lone factor that leads people to engage in acts of terror. Scholars have categorized motivations for terrorism to include psychological, ideological, and strategic.

It is impossible to say for sure what causes terrorism. A person's psychological make-up certainly will play a role, but to what extent is unclear. Some may come to terrorism, not out of any love for violence, but rather to further their ideological goals.

Others may be motivated to use terror simply because it appears to be a useful strategic alternative, or may further the state's objectives. Indeed, terrorism may occur for psychological, ideological, and strategic grounds all at once. An individual may decide terrorism fits his or her own view of the world—that it makes sense. A group may come to use terrorism because it furthers and is supported by their ideology. Finally, groups or persons may use terrorism because it fits with their strategic objectives and goals.

1.1 ABOUT THE DATA

The dataset used in this analysis is from Global Terrorism Database. The GTD is publicly available to search, browse, and download on the GTD website. In 2019, the University of Maryland began a partnership with CHC Global to manage the commercial distribution of the GTD.

The Global Terrorism Database (GTD)™ is the most comprehensive unclassified database of terrorist attacks in the world. The National Consortium for the Study of Terrorism and Responses to Terrorism (START) makes the GTD available via this site in an effort to improve understanding of terrorist violence, so that it can be more readily studied and defeated. The GTD is produced by a dedicated team of researchers and technical staff.

The GTD is an open-source database, which provides information on domestic and international terrorist attacks around the world since 1970, and now includes more than

200,000 events. For each event, a wide range of information is available, including the date and location of the incident, the weapons used, nature of the target, the number of casualties, and – when identifiable – the group or individual responsible.

#VIEWING THE DATA

```
df = pd.read_csv('/content/drive/MyDrive/3D Objects/BUILD CHATBOTS WITH PYTHON/globalterrorismdb_0718dist.csv' , encoding='latin-1',low_memory=False)
```

```
df.head()
```

```
df.head()
```

	eventid	iyear	imonth	iday	approxdate	extended	resolution	country	country_txt	region	region_txt	provstate	city	latitude	longitude	specificity	vicinity
0	197000000001	1970	7	2	NaN	0	NaN	58	Dominican Republic	2	Central America & Caribbean	NaN	Santo Domingo	18.456792	-69.951164	1.0	0
1	197000000002	1970	0	0	NaN	0	NaN	130	Mexico	1	North America	Federal	Mexico city	19.371887	-99.086624	1.0	0
2	197001000001	1970	1	0	NaN	0	NaN	160	Philippines	5	Southeast Asia	Tarlac	Unknown	15.478598	120.599741	4.0	0
3	197001000002	1970	1	0	NaN	0	NaN	78	Greece	8	Western Europe	Attica	Athens	37.997490	23.762728	1.0	0
4	197001000003	1970	1	0	NaN	0	NaN	101	Japan	4	East Asia	Fukouka	Fukouka	33.580412	130.396361	1.0	0

5 rows × 135 columns

Fig 1.1.0 DataFrame

```
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
```

#RENAMING THE REQUIRED COLUMNS

```
df.rename(columns={'iyear':'Year','imonth':'Month','iday':'Day',
'country_txt':'Country','provstate':'state',
'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)
```

MISSING VALUES

import missingno as msno

msno.matrix(df)

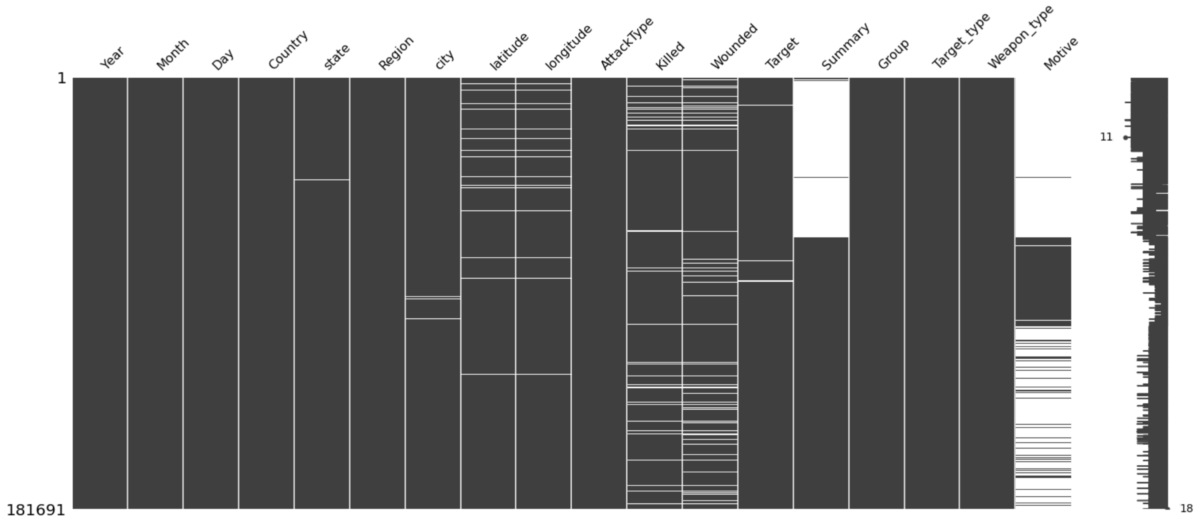


Fig 1.1.1 Null Values

df.isnull().sum()

Year	0
Month	0
Day	0
Country	0
state	421
Region	0
city	434
latitude	4556
longitude	4557
AttackType	0
Killed	10313
Wounded	16311
Target	636
Summary	66129
Group	0
Target_type	0
Weapon_type	0
Motive	131130

dtype: int64

Table 1.1 Count of Null Values

1.2 COMPANY PROFILE

MIT Square is a premier product development company headquartered in Bangalore, India and has a presence in Southampton, UK. MIT expands as "Management and Innovation for Transformation" with a tagline "We transform your life". MIT Square is an International Organisation for Standardisation, ISO 9001:2015, Certified Company. We at MIT Square, are experts in designing and developing innovative products, building start-ups, and understanding the need of the enterprises for their business growth. We offer product design, product

development, product manufacturing and patent filing services. MIT Square excels in the design, development, manufacturing and supplying of consumer products, industrial and IoT devices, education platforms, hospitality products, and healthcare technology. We offer turnkey, tooling and OEM/ODM services. From individuals, start-ups, small and medium-sized companies to international corporations, MIT Square is here to support you in all your product design & product development needs and pave the way to transform your life by turning your ideas into reality.

MIT Square offers you an unparalleled equation of value, cost and on time delivery by having our highly qualified product design-development, supply chain and product manufacturing specialists team in the UK, USA, Asia and Middle East. Our product designers, engineering developers and innovative management teams ensure your product meets the world class standard. IP protection is at the heart of our management. We follow a rigorous method to strictly protect your intellectual property rights in Asia and across the globe and offer you complete ownership of the design. We do not just stop by playing an advisory role. It is just our starting point. We walk along with you in executing these strategies end-to-end, to ensure business success.

Website: https://www.mitsquare.com

Industry: Information Technology & Services

Headquarters: London, UK

CHAPTER 2

EXECUTIVE SUMMARY

2.1 PROBLEM STATEMENT

- The ultimate aim of this analysis is to identify the trends of terrorism happened all over the world for each period.
- To Predict the death occurrences using Machine Learning.
- To look into trends of terrorism in INDIA deeply.

2.2 METHODOLOGY

For analysis I have used Tableau most because Tableau is one of the most powerful visualisation software. For predicting the death occurrences, a random forest classifier was used with help of python with google Colab.

My final model achieved **86 %** accuracy.

2.3 LIBRARIES USED IN MACHINE LEARNING

- KNeighborsClassifier()
- AdaBoostClassifier(random_state = 41)
- RandomForestClassifier(random_state = 41)
- GaussianNB()
- MLPClassifier()

KNeighborsClassifier

The K in the name of this classifier represents the k nearest neighbours, where k is an integer value specified by the user. Hence as the name suggests, this classifier implements learning based on the k nearest neighbours. The choice of the value of k is dependent on data.

AdaBoostClassifier

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

RandomForestClassifier

Assuming your dataset has “m” features, the random forest will randomly choose “k” features where k < m. Now, the algorithm will calculate the root node among the k features by picking a node that has the highest information gain.

After that, the algorithm splits the node into child nodes and repeats this process “n” times. Now you have a forest with n trees. Finally, you’ll perform bootstrapping, ie, combine the results of all the decision trees present in your forest.

It’s certainly one of the most sophisticated algorithms as it builds on the functionality of decision trees.Technically, it is an ensemble algorithm. The algorithm generates the individual decision trees through an attribute selection indication. Every tree relies on an independent random sample. In a classification problem, every tree votes and the most popular class is the end result. On the other hand, in a regression problem, you’ll compute the average of all the tree outputs and that would be your end result.

A random forest Python implementation is much simpler and robust than other non-linear algorithms used for classification problems.

GaussianNB

Naive Bayes are a group of supervised machine learning classification algorithms based on the Bayes theorem. It is a simple classification technique, but has high functionality. They find use when the dimensionality of the inputs is high. Complex classification problems can also be implemented by using Naive Bayes Classifier.

MLPClassifier

Multi-layer Perceptron classifier.

This model optimises the log-loss function using LBFGS or stochastic gradient descent.MLPClassifier trains iteratively since at each time step the partial derivatives of the loss function with respect to the model parameters are computed to update the parameters.

It can also have a regularisation term added to the loss function that shrinks model parameters to prevent overfitting.

CHAPTER 3
DEATH OCCURRENCES PREDICTION

3.1 INSIGHTS

Overview

```
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())
print("Group with the most attacks:",df['Group'].value_counts().index[1])
print("Most Attack Types:",df['AttackType'].value_counts().idxmax())
```

Output :

```
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
Group with the most attacks: Taliban
Most Attack Types: Bombing/Explosion
```

#Attack in years

```
import seaborn as sns
x_year = df['Year'].unique()
y_year = 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()
```

Output :

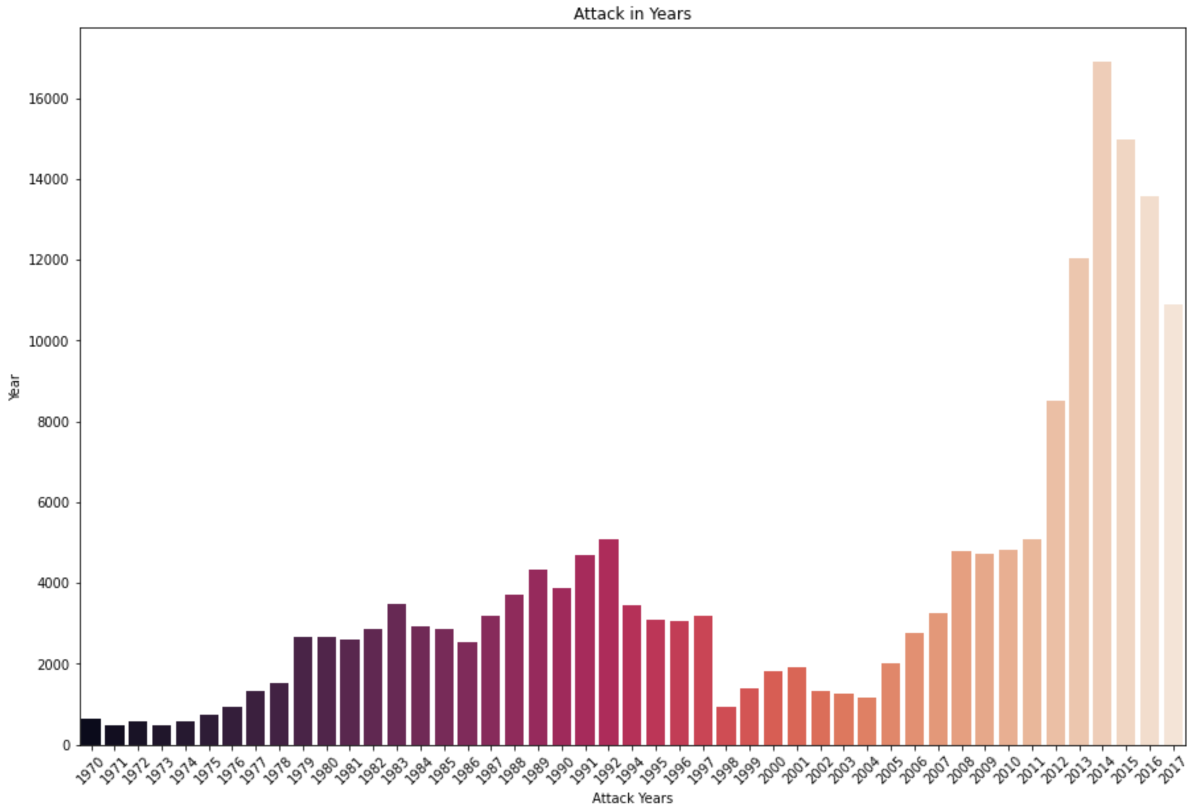


Fig 3.1.1 Attack in Years

Demonstrating attack type

```
import plotly.express as px
import plotly.graph_objs as go
from plotly import tools
from plotly.offline import iplot
from plotly.offline import init_notebook_mode
px.histogram(df, x='Year',color = 'AttackType',width=1420, height=768 )
```

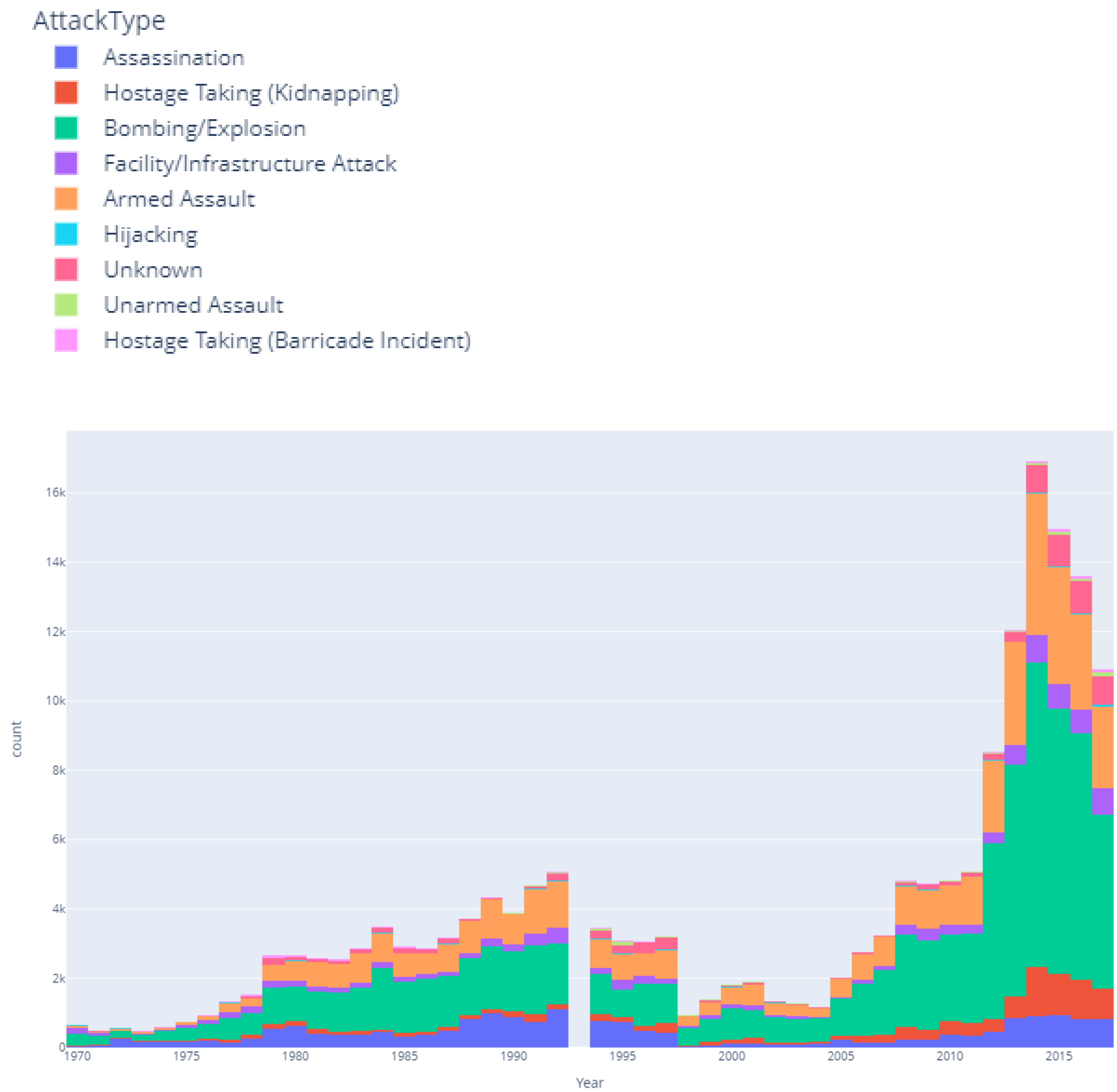
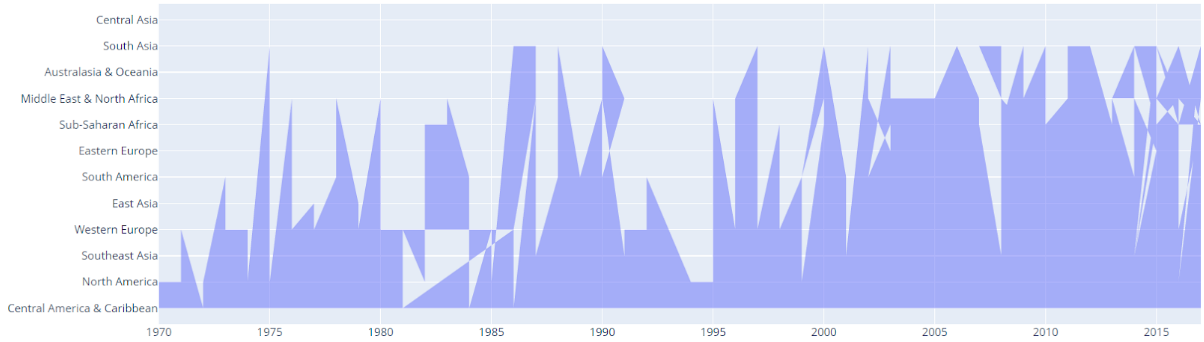


Fig 3.1.2 Attack type

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['Year'], y=df['Region'], fill='tozeroy',
                    mode='none' # override default markers+lines
                    ))
fig.add_trace(go.Scatter(x=df['Year'], y=df['Region'], fill='tonexty',
                    mode= 'none'))

fig.show()
```



Correlation between features

plt.figure(figsize=[15,8])

sns.heatmap(df.corr(),cmap='icefire', linewidths=0.4)

plt.show()

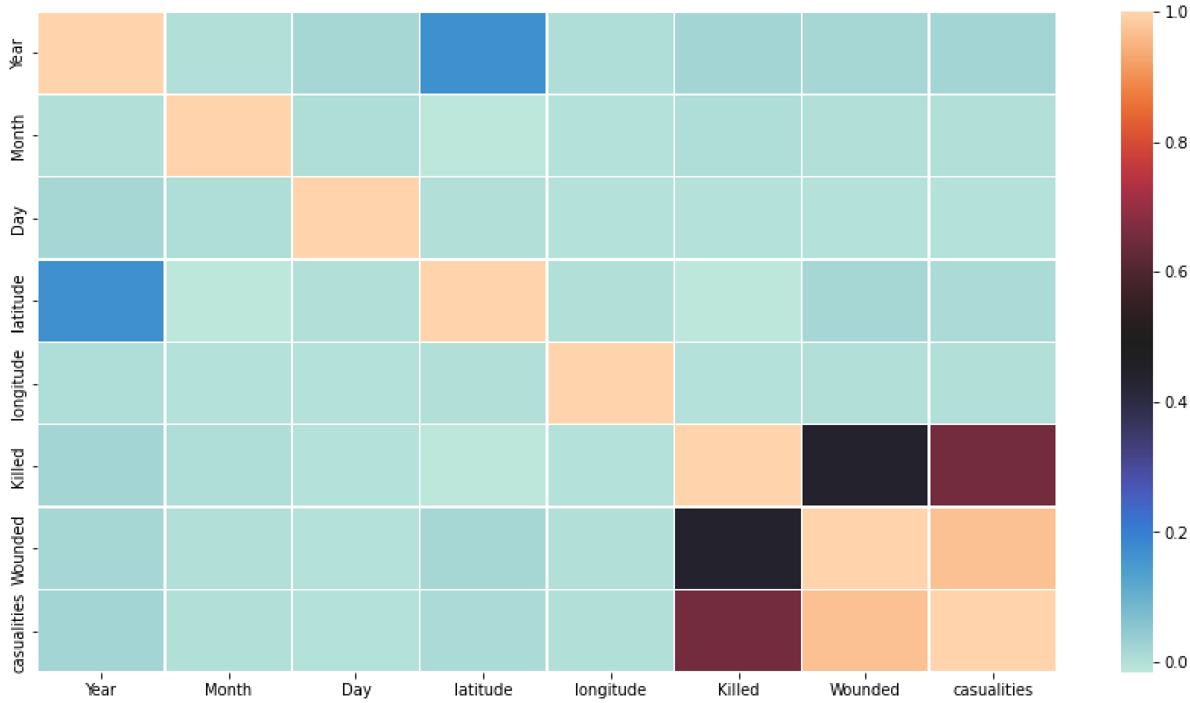


Fig 3.1.3 Correlation between features

#The most attacks happened in which country

attack_country = df.Country.value_counts()[:20]

attack_country

```
Iraq              24636
Pakistan          14368
Afghanistan       12731
India             11960
Colombia           8306
Philippines        6908
Peru               6096
El Salvador        5320
United Kingdom     5235
Turkey             4292
Somalia            4142
Nigeria            3907
Thailand           3849
Yemen              3347
Spain              3249
Sri Lanka          3022
United States      2836
Algeria            2743
France             2693
Egypt              2479
Name: Country, dtype: int64
```

#**Pictorial representation of top countries affected**

```
fig = px.bar(df, x=attack_country.index, y=attack_country.values,
colour=attack_country.index, title="Top Countries Affected")
fig.update_traces( marker_line_color='rgb(8,48,107)',
                  marker_line_width=1.5, opacity=0.6)
fig.show()
```

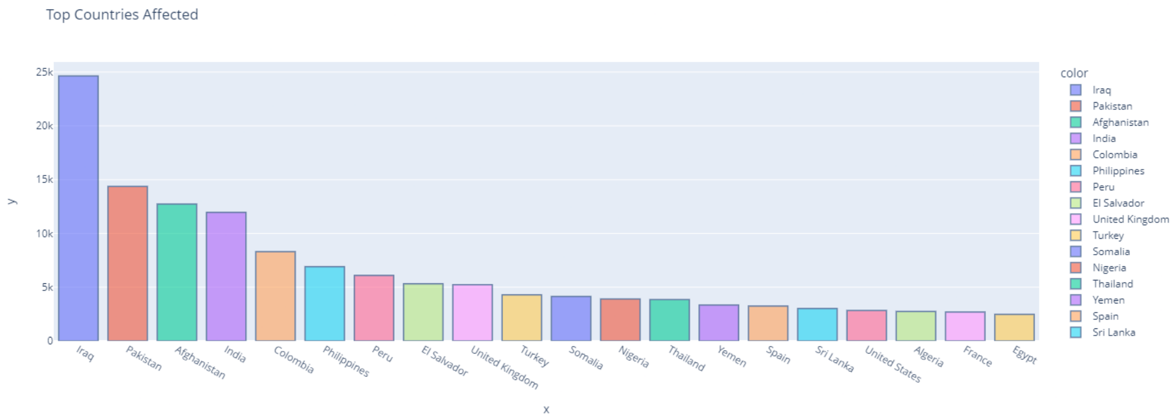


Fig 3.1.4 Top Countries Affected

#Groups which attacked most

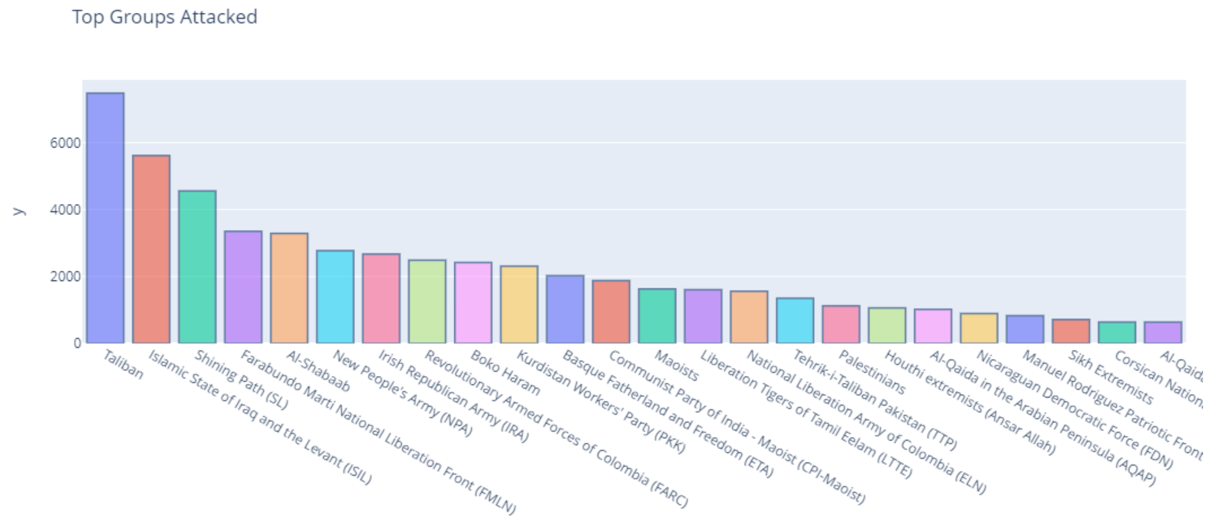
attack_groups = df.Group.value_counts()[1:25]

fig = px.bar(df, x=attack_groups.index, y=attack_groups.values, color=attack_groups.index, title="Top Groups Attacked")

fig.update_traces(marker_line_color='rgb(8,48,107)',

marker_line_width=1.5, opacity=0.6)

fig.show()



Kill count of each groups

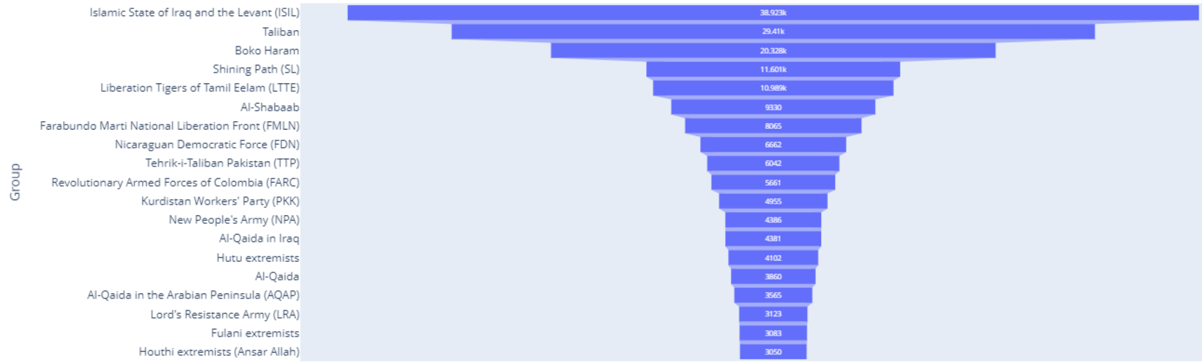
group_killed=df[['Group','Killed','Year','Country','Region']].groupby(['Group'],axis=0).sum().sort_values('Killed', ascending=False)[1:20]

```
group_killed
```

Group	Killed
Islamic State of Iraq and the Levant (ISIL)	38923
Taliban	29410
Boko Haram	20328
Shining Path (SL)	11601
Liberation Tigers of Tamil Eelam (LTTE)	10989
Al-Shabaab	9330
Farabundo Marti National Liberation Front (FMLN)	8065
Nicaraguan Democratic Force (FDN)	6662
Tehrik-i-Taliban Pakistan (TTP)	6042
Revolutionary Armed Forces of Colombia (FARC)	5661
Kurdistan Workers' Party (PKK)	4955
New People's Army (NPA)	4386
Al-Qaida in Iraq	4381
Hutu extremists	4102
Al-Qaida	3860
Al-Qaida in the Arabian Peninsula (AQAP)	3565
Lord's Resistance Army (LRA)	3123
Fulani extremists	3083
Houthi extremists (Ansar Allah)	3050

Table 3.1 Kill Count of top groups

```
fig = px.funnel(group_killed, x='Killed', y=group_killed.index)
fig.show()
```



Count of attacks and number of people killed by each country

```
count_terror = df['Country'].value_counts()[:15].to_frame()
count_terror.columns=['Attacks']
count_kill=df.groupby('Country')['Killed'].sum().to_frame()
count_kill.columns
```

```
Index(['Killed'], dtype='object')
```

```
count_terror = count_terror.merge(count_kill,left_index = True,right_index =True,how='left')
```

```
address = ['Iraq','Pakistan','Afghanistan','India','Colombia','Philippines','Peru','El Salvador','United Kingdom','Turkey','Somalia','Nigeria','Thailand','Yemen','Spain']
count_terror['Country'] = address
Count_terror
```

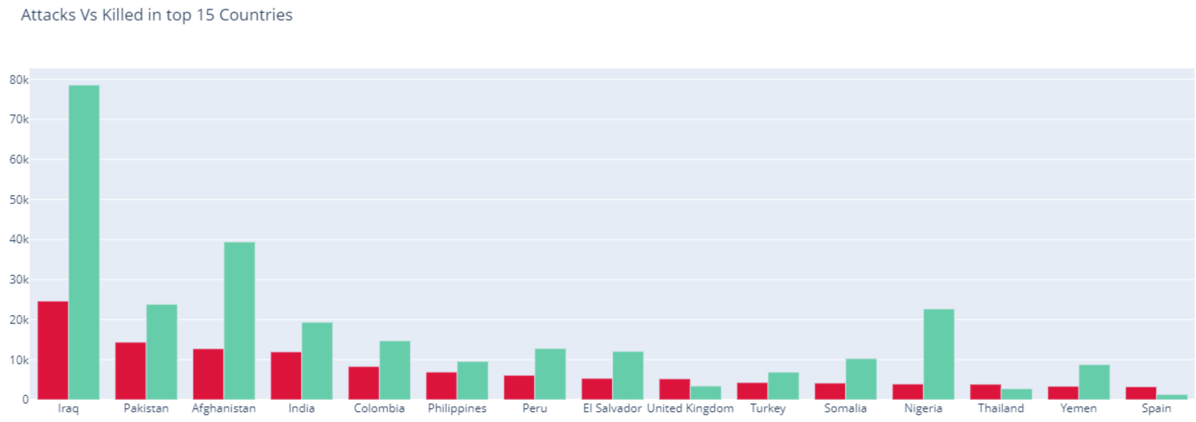
	Attacks	Killed
Iraq	24636	78589
Pakistan	14368	23822
Afghanistan	12731	39384
India	11960	19341
Colombia	8306	14698
Philippines	6908	9559
Peru	6096	12771
El Salvador	5320	12053
United Kingdom	5235	3410
Turkey	4292	6888
Somalia	4142	10273
Nigeria	3907	22682
Thailand	3849	2742
Yemen	3347	8776
Spain	3249	1288

Table 3.1.2 Count of attacks and number of people killed by each country

Difference between attack and killed in top 15 countries

```
fig = go.Figure(data=[
    go.Bar(name='Attacks', x=count_terror.Country,
y=count_terror.Attacks,marker_color='crimson'),
    go.Bar(name='Killed', x=count_terror.Country,
y=count_terror.Killed,marker_color='mediumaquamarine')])

fig.update_layout(barmode='group')
fig.update_layout(title_text='Attacks Vs Killed in top 15 Countries')
```

fig.show()



Terrorist attacks by lattitude and longitude in india

```
df1['fatalities'] = df1['fatalities'].fillna(0).astype(int)
df1['injuries'] = df1['injuries'].fillna(0).astype(int)
# terrorist attacks in india only (11,960 rows)
terror_india = df1[(df1.country == 'India')]
terror_india
```

id	year	month	day	country	state	latitude	longitude	target	weapon	fatalities	injuries	
1186	197202220004	1972	2	22	India	Delhi	28.585836	77.153336	Airports & Aircraft	Unknown Explosive Type	0	0
2764	197501190004	1975	1	2	India	Bihar	25.863042	85.781004	Government (General)	Unknown Explosive Type	0	0
3857	197605260001	1976	5	26	India	Delhi	28.585836	77.153336	Airports & Aircraft	Unknown Explosive Type	0	0
5327	197709280004	1977	9	28	India	Maharashtra	19.075984	72.877656	Airports & Aircraft	Unknown Gun Type	0	0

7337	197901130004	1979	1	13	India	Assam	26.200605	92.937574	Police	Automatic or Semi-Automatic Rifle	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
181663	201712300021	2017	12	30	India	Kerala	11.831902	75.565432	Police	Other Explosive Type	0	0
181665	201712300023	2017	12	30	India	Chhattisgarh	18.802725	81.497666	Business	Unknown Gun Type	0	0
181672	201712310005	2017	12	31	India	Jammu and Kashmir	33.966527	74.964225	Police	Grenade	3	0
181684	201712310019	2017	12	31	India	Assam	25.180162	93.015788	Government (General)	Automatic or Semi-Automatic Rifle	0	0
181689	201712310031	2017	12	31	India	Manipur	24.798346	93.940430	Government (General)	Grenade	0	0

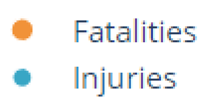
11960 rows × 12 columns

Table 3.1.3 Terrorist attacks by lattitude and longitude in india

```
pd.options.mode.chained_assignment = None
terror_india['day'][terror_india.day == 0] = 1
terror_india['date'] = pd.to_datetime(terror_india[['day', 'month', 'year']])
terror_india = terror_india[['id', 'date', 'year', 'state', 'latitude', 'longitude',
                    'target', 'weapon', 'fatalities', 'injuries']]
terror_india = terror_india.sort_values(['fatalities', 'injuries'], ascending = False)
```

```
terror_india = terror_india.drop_duplicates(['date', 'latitude', 'longitude', 'fatalities'])
terror_india['text'] = terror_india['date'].dt.strftime('%B %-d, %Y') + ' , ' +\
                terror_india['fatalities'].astype(str) + ' Killed, ' +\
                terror_india['injuries'].astype(str) + ' Injured'
fatality = dict(
        type = 'scattergeo',
        locationmode = "ISO-3",
        lon = terror_india[terror_india.fatalities > 0]['longitude'],
        lat = terror_india[terror_india.fatalities > 0]['latitude'],
        text = terror_india[terror_india.fatalities > 0]['text'],
        mode = 'markers',
        name = 'Fatalities',
        hoverinfo = 'text+name',
        marker = dict(
           size = terror_india[terror_india.fatalities > 0]['fatalities'] ** 0.255 * 8,
           opacity = 0.95,
           color = 'rgb(240, 140, 45)')
        )
injury = dict(
       type = 'scattergeo',
       locationmode = 'USA-states',
       lon = terror_india[terror_india.fatalities == 0]['longitude'],
       lat = terror_india[terror_india.fatalities == 0]['latitude'],
       text = terror_india[terror_india.fatalities == 0]['text'],
       mode = 'markers',
       name = 'Injuries',
       hoverinfo = 'text+name',
       marker = dict(
          size = (terror_india[terror_india.fatalities == 0]['injuries'] + 1) ** 0.245 * 8,
          opacity = 0.85,
          color = 'rgb(20, 150, 187)')
       )
layout = dict(
```

```
    title = 'Terrorist Attacks by Latitude/Longitude in India (1970-2015)',
    showlegend = True,
    legend = dict(
        x = 0.85, y = 0.4
    ),
    geo = dict(
        scope = 'asia',
        projection = dict(type = 'mercator'),
        showland = True,
        landcolor = 'rgb(250, 250, 250)',
        subunitwidth = 1,
        subunitcolor = 'rgb(217, 217, 217)',
        countrywidth = 1,
        countrycolor = 'rgb(217, 217, 217)',
        showlakes = True,
        lakecolor = 'rgb(255, 255, 255)')
    )

data = [fatality, injury]
figure = dict(data = data, layout = layout)

iplot(figure)
```



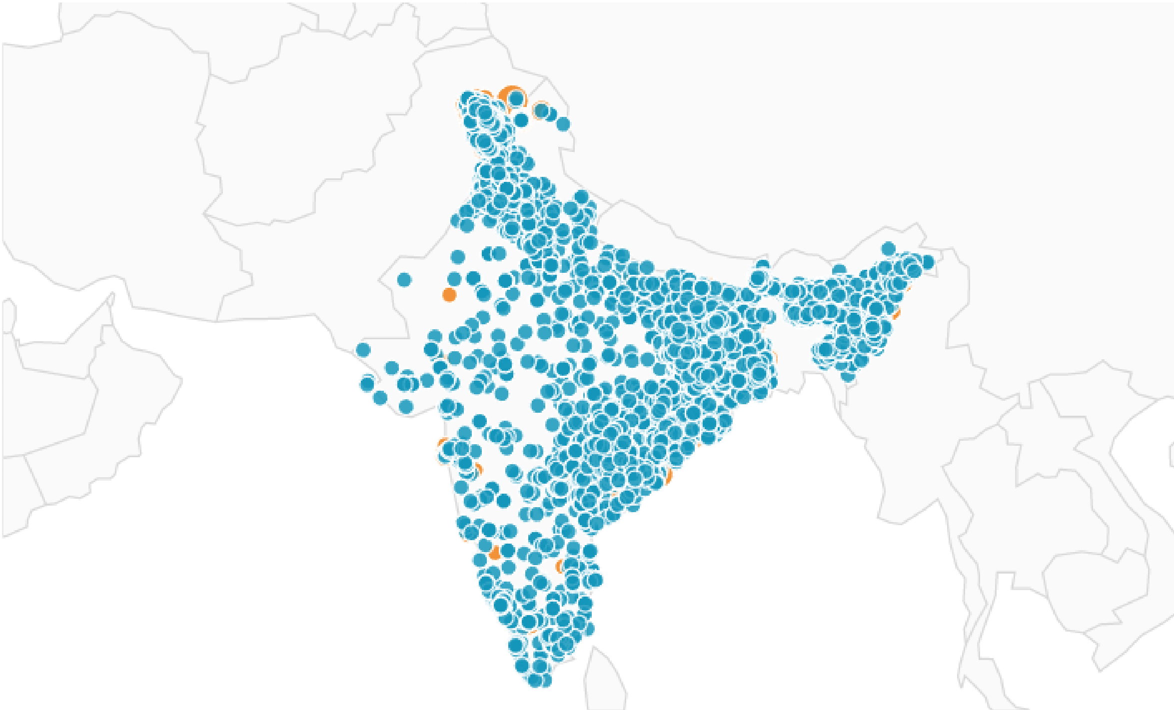


Fig 3.1.5 Attack Vs Kill count in india

Terrorists Attacks By Year In India

```
import numpy as np
terror_peryear = np.asarray(terror_india.groupby('year').year.count())

terror_years = np.arange(1970, 2016)
# terrorist attacks in 1993 missing from database
terror_years = np.delete(terror_years, [23])

trace = [go.Scatter(
        x = terror_years,
        y = terror_peryear,
        mode = 'lines',
        line = dict(
            color = 'rgb(240, 140, 45)',
            width = 3)
        )]
```

```
layout = go.Layout(
        title = 'Terrorist Attacks by Year in INDIA (1970-2015)',
        xaxis = dict(
            rangeslider = dict(thickness = 0.05),
            showline = True,
            showgrid = False
        ),
        yaxis = dict(
            range = [0.1, 425],
            showline = True,
            showgrid = False)
        )

figure = dict(data = trace, layout = layout)
iplot(figure)
```

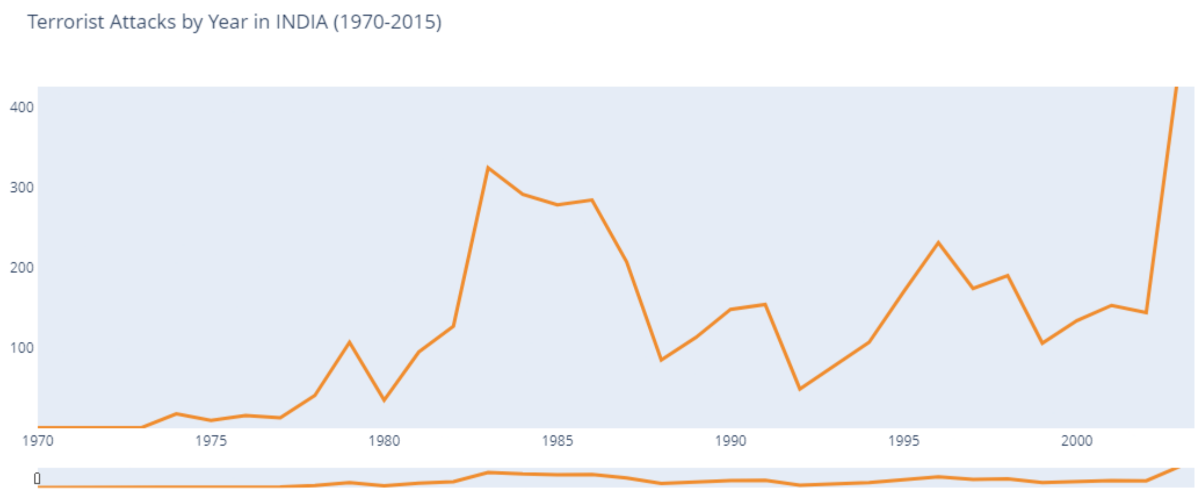


Fig 3.1.6 Terrorist attacks by year in INDIA

Terrorists Attack By Target Type In India

```
target_codes = []
```

```
for attack in terror_india['target'].values:
    if attack in ['Business', 'Journalists & Media', 'NGO']:
        target_codes.append(1)
    elif attack in ['Government (General)', 'Government (Diplomatic)']:
        target_codes.append(2)
    elif attack == 'Abortion Related':
        target_codes.append(4)
    elif attack == 'Educational Institution':
        target_codes.append(5)
    elif attack == 'Police':
        target_codes.append(6)
    elif attack == 'Military':
        target_codes.append(7)
    elif attack == 'Religious Figures/Institutions':
        target_codes.append(8)
    elif attack in ['Airports & Aircraft', 'Maritime', 'Transportation']:
        target_codes.append(9)
    elif attack in ['Food or Water Supply', 'Telecommunication', 'Utilities']:
        target_codes.append(10)
    else:
        target_codes.append(3)

terror_india['target'] = target_codes
target_categories = ['Business', 'Government', 'Individuals', 'Healthcare', 'Education',
                     'Police', 'Military', 'Religion', 'Transportation', 'Infrastructure']
target_count = np.asarray(terror_india.groupby('target').target.count())
target_percent = np.round(target_count / sum(target_count) * 100, 2)
# terrorist attack fatalities by target
target_fatality = np.asarray(terror_india.groupby('target')['fatalities'].sum())
target_yaxis = np.asarray([1.33, 2.36, 2.98, 0.81, 1.25, 1.71, 1.31, 1.53, 1.34, 0])
# terrorist attack injuries by target
target_injury = np.asarray(terror_india.groupby('target')['injuries'].sum())
```

```
target_xaxis = np.log10(target_injury)
target_text = []
for i in range(0, 9):
   target_text.append(target_categories[i] + ' (' + target_percent[i].astype(str) + '%)<br>' +
target_fatality[i].astype(str) + ' Killed, ' + target_injury[i].astype(str) + ' Injured')
data = [go.Scatter(
        x = target_injury,
        y = target_fatality,
        text = target_text,
        mode = 'markers',
        hoverinfo = 'text',
        marker = dict(
            size = target_count / 6.5,
            opacity = 0.9,
            color = 'rgb(240, 140, 45)')
        )]

layout = go.Layout(
         title = 'Terrorist Attacks by Target in INDIA (1970-2015)',
         xaxis = dict(
             title = 'Injuries',
             type = 'log',
             range = [1.36, 3.25],
             tickmode = 'linear',
             nticks = 2,
             showline = True,
             showgrid = False
         ),
         yaxis = dict(
             title = 'Fatalities',
             type = 'log',
             range = [0.59, 3.45],
             tickmode = 'auto',
```

```
            nticks = 4,
            showline = True,
            showgrid = False)
        )

annotations = []
for i in range(0, 9):
    annotations.append(dict(x=target_xaxis[i], y=target_yaxis[i],
                            xanchor='center', yanchor='middle',
                            text=target_categories[i], showarrow=False))

layout['annotations'] = annotations
figure = dict(data = data, layout = layout)
iplot(figure)
```

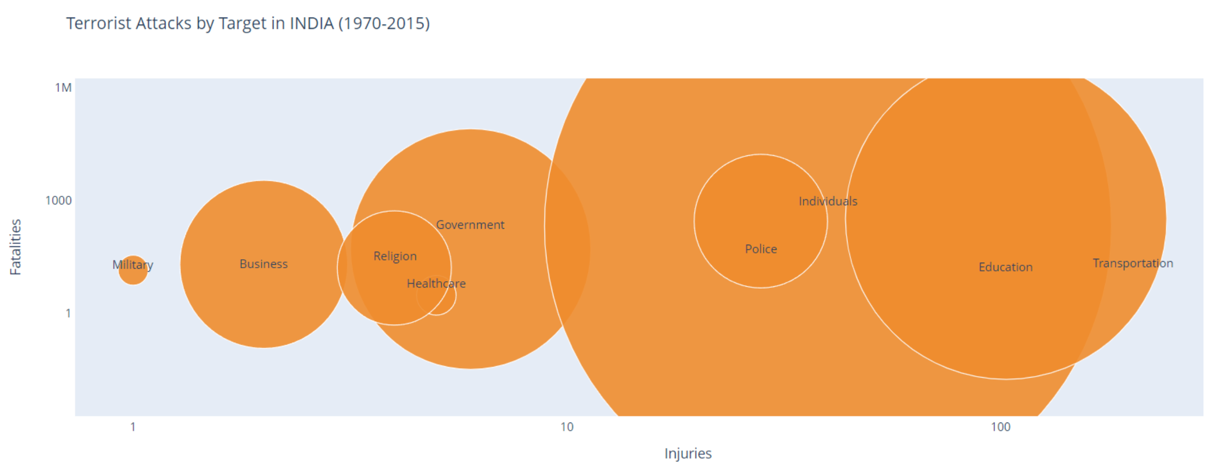


Fig 3.1.7 Terrorist Attack by target type in INDIA

3.2 TRAIN_TEST_SPLIT

```
from sklearn.model_selection import train_test_split

income = df1['death']

# Split the 'features' and 'income' data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df1.drop(['death'], axis=1),
                                                    income,
                                                    shuffle=True,
                                                    test_size = 0.2,
                                                    random_state = 43)

# Show the results of the split
print("Training set has {} samples.".format(X_train.shape[0]))
print("Testing set has {} samples.".format(X_test.shape[0]))
```

Training set has 137102 samples.
Testing set has 34276 samples.

First let's view the relation of each variable .
Then we use a benchmark model to predict.
Then we split and fit the data into ML models . Using feature selection we can reduce the number of features and improve our model.

Correlation and most important features

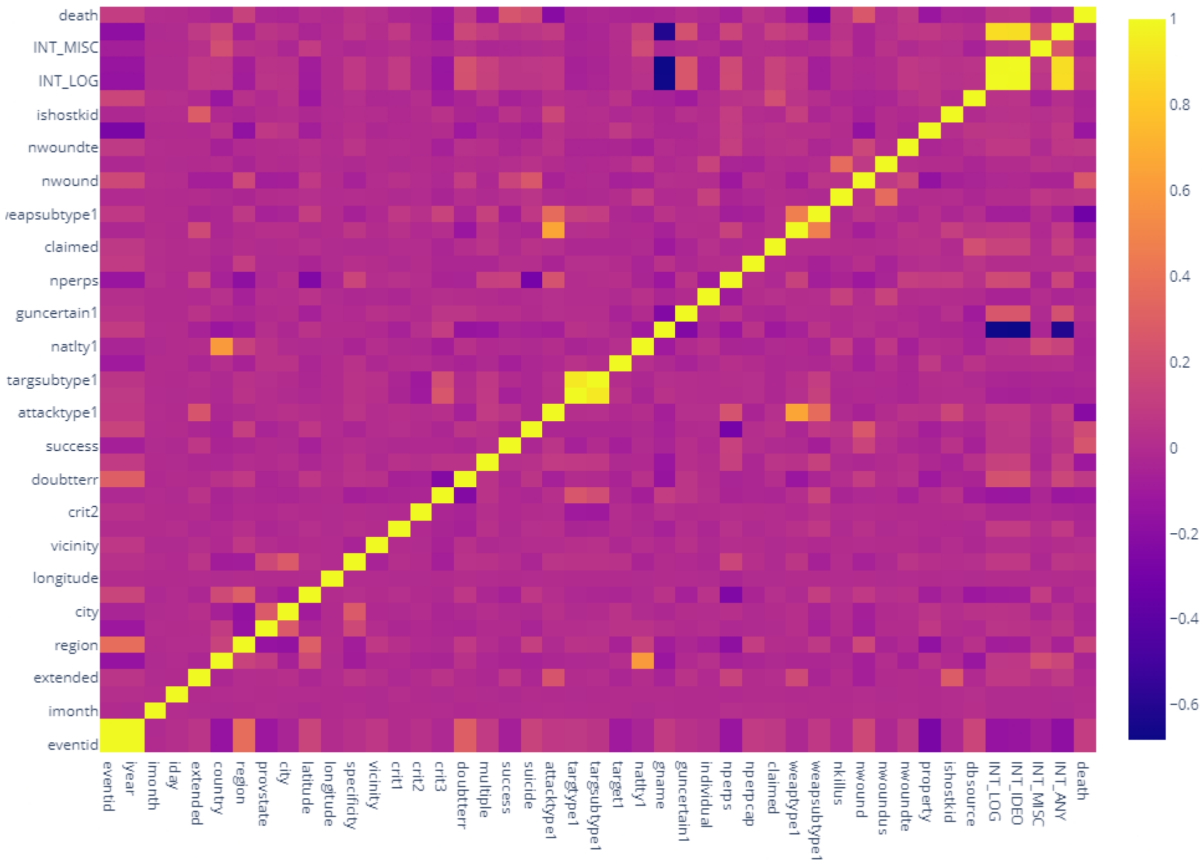


Fig 3.3.1 Most Important Features

Evaluating model's performance with Benchmark model

While it is important to benchmark the predictive performance of a machine learning model, there are also important operational constraints that require consideration. These include the training and runtime performance characteristics of a model. That is, how long it takes to train a model, how long it takes to score new data, and the compute resources required to accomplish both of these.

Benchmarking operational characteristics may often be as important as evaluating the predictive characteristics of a model. For example, consider the problem of real-time bidding on an ad-buying network. The total latency budget for a bid, covering network access, database queries and prediction, is 100ms. The 80ms spent on calculating a prediction in a deep learning model may mean that the model does not meet the business requirements. A logistic regression model, which returns a result in 5ms, may be more suitable, despite the deep learning model's increased predictive power.

A benchmark that only focuses on the predictive performance of a model would ignore this important aspect of putting a model into production.

The model used for benchmarking is the Decision Tree Classifier which is focused on supervised learning. The Decision Tree tries to ramify all the data .
The classifier is fitted with x_train , y_train which has 137102 samples , 151386 samples respectively.

Benchmark Model:

Accuracy: 80.79

F_score: 80.03

Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition:
Accuracy = Number of correct prediction
Total Number of Prediction
Our bench mark model has achieved 80.79% accuracy with a decision tree classifier.
The beta parameter determines the weight of recall in the combined score. beta < 1 lends more weight to precision, while beta > 1 favours recall
In our case the f-beta score is 0.80 which is 80.03 percent.
The F-beta score is the weighted harmonic mean of precision and recall, reaching its optimal value at 1 and its worst value at 0.

3.3 Training And Predicting Pipelines

Let's define a function to fit , train and predict the model.

```
def train_predict(learner, sample_size, X_train, y_train, X_test, y_test):
results = {}
learner = learner.fit(X_train[:sample_size], y_train[:sample_size])
predictions_test = learner.predict(X_test)
predictions_train = learner.predict(X_train[:300])
```

```
results['acc_train'] = accuracy_score(y_train[:300], predictions_train)
results['acc_test'] = accuracy_score(y_test, predictions_test)
results['f_train'] = fbeta_score(y_train[:300], predictions_train, beta = 0.5, average = 'weighted')
results['f_test'] = fbeta_score(y_test, predictions_test, beta = 0.5, average = 'weighted')
print("{} trained on {} samples.".format(learner.__class__.__name__, sample_size))
return results
```

Results of the Models

The models used in this research are the KNeighborsClassifier , AdaBoostClassifier , RandomForestClassifier , GaussianNB , MLPClassifier which focused on supervised learning.

After defining the function we have to define the classifiers by importing necessary modules from sklearn.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import fbeta_score
from sklearn.svm import SVC


clf_A = KNeighborsClassifier()
clf_B = AdaBoostClassifier(random_state = 41)
clf_C = RandomForestClassifier(random_state = 41)
clf_D = GaussianNB()
clf_E = MLPClassifier()
samples_100 = len(y_train)
samples_10 = int(samples_100/10)
samples_1 = int(samples_100/100)
```

Then we have to Calculate the number of samples for 1%, 10%, and 100% of the training data for effective results.

After defining classifiers we can fit the model in our previously defined train_predict function to fit , train and predict from the data. And this is how our model is trained on each classifiers,

```
results = {}
for clf in [clf_A, clf_B, clf_C , clf_D , clf_E ]:
    clf_name = clf.__class__.__name__
    results[clf_name] = {}
    for i, samples in enumerate([samples_1, samples_10, samples_100]):
        results[clf_name][i] = train_predict(clf, samples, X_train, y_train, X_test, y_test)
    print('-'*40)
accuracy = bench_acc
fbeta = bench_fsc
```

```
KNeighborsClassifier trained on 1371 samples.
KNeighborsClassifier trained on 13710 samples.
KNeighborsClassifier trained on 137102 samples.
----------------------------------------
AdaBoostClassifier trained on 1371 samples.
AdaBoostClassifier trained on 13710 samples.
AdaBoostClassifier trained on 137102 samples.
----------------------------------------
RandomForestClassifier trained on 1371 samples.
RandomForestClassifier trained on 13710 samples.
RandomForestClassifier trained on 137102 samples.
----------------------------------------
GaussianNB trained on 1371 samples.
GaussianNB trained on 13710 samples.
GaussianNB trained on 137102 samples.
----------------------------------------
MLPClassifier trained on 1371 samples.
MLPClassifier trained on 13710 samples.
MLPClassifier trained on 137102 samples.
----------------------------------------
```

3.4 Results of the Classification Models

The models used in this research are the KNeighborsClassifier , AdaBoostClassifier , RandomForestClassifier , GaussianNB , MLPClassifier which focus on supervised learning.

```
for k, v in results.items():
    print('-'*40)
    print("Model: %s" %(k))
    for i in v:
        print("Accuraccy: %.4f \n Fscore_test: %.4f" %(v[i]['acc_test'], v[i]['f_test']))
```

And their results are as follows ,

```
----------------------------------------
Model: KNeighborsClassifier
Accuraccy: 0.5233
Fscore_test: 0.5239
Accuraccy: 0.5433
Fscore_test: 0.5431
Accuraccy: 0.5857
Fscore_test: 0.5856
----------------------------------------
Model: AdaBoostClassifier
Accuraccy: 0.7920
Fscore_test: 0.7923
Accuraccy: 0.8069
Fscore_test: 0.8070
Accuraccy: 0.8099
Fscore_test: 0.8099
----------------------------------------
Model: RandomForestClassifier
```

Accuraccy: 0.8031

Fscore_test: 0.8034

Accuraccy: 0.8357

Fscore_test: 0.8357

Accuraccy: 0.8634

Fscore_test: 0.8634

Model: GaussianNB

Accuraccy: 0.5463

Fscore_test: 0.5457

Accuraccy: 0.5475

Fscore_test: 0.5475

Accuraccy: 0.5177

Fscore_test: 0.2966

Model: MLPClassifier

Accuraccy: 0.5177

Fscore_test: 0.2966

Accuraccy: 0.4823

Fscore_test: 0.2595

Accuraccy: 0.4823

Fscore_test: 0.2595

From the result we can see that our Random forest classifier works better than the others with accuracy of 86 percent . AdaBoost Classifier is also performing well but not as efficient as random forest classifier.

Feature selection

Feature selection is also called variable selection or attribute selection.
It is the automatic selection of attributes in your data (such as columns in tabular data) that are most relevant to the predictive modelling problem you are working on.
feature selection… is the process of selecting a subset of relevant features for use in model construction
Feature selection is different from dimensionality reduction. Both methods seek to reduce the number of attributes in the dataset, but a dimensionality reduction method does so by creating new combinations of attributes, whereas feature selection methods include and exclude attributes present in the data without changing them.
For feature selection we can use select from model class from sklearn's feature selection module as follows,
from sklearn.feature_selection import SelectFromModel
For this task I am going to select a random forest model as it performs best.
And these are the results,

```
from sklearn.feature_selection import SelectFromModel


feat_labels = df1.columns
sfm = SelectFromModel(clf_C, threshold=0.02)
sfm.fit(X_train, y_train)
important_feat = [feat_labels[col] for col in sfm.get_support(indices=True)]
#for feature_list_index in sfm.get_support(indices=True):
#    print(feat_labels[feature_list_index])
important_feat.append('death')

X_important_test = sfm.transform(X_test)
X_important_train = sfm.transform(X_train)

print('-'*40)
print('Feature Selection:')
print('before')
```

```
print(X_train.shape)
print('after')
print(X_important_train.shape)
print('-'*40)
```

```
----------------------------------------
Feature Selection:
before (137102, 44)
after (137102, 21)
----------------------------------------
```

After fitting the above selected features the model achieved ,

Accuracy after Feature Selection: 85.98

F_score after Feature Selection: 85.70

Then again splitting the model as train set validation set and test set ,

```
X_intermediate, X_test, y_intermediate, y_test = train_test_split(df1.drop(['death'], axis=1),
                                                    income,
                                                    shuffle=True,
                                                    test_size = 0.2,
                                                    random_state = 43)

X_train, X_validation, y_train, y_validation = train_test_split(X_intermediate,
                                                  y_intermediate,
                                                  shuffle=False,
                                                  test_size=0.2,
                                                  random_state=43)
del X_intermediate, y_intermediate

print('train: {}% | validation: {}% | test
{}%'.format(round(float(len(y_train))/len(income),2),
                                         round(float(len(y_validation))/len(income),2),
                                         round(float(len(y_test))/len(income),2)))
```

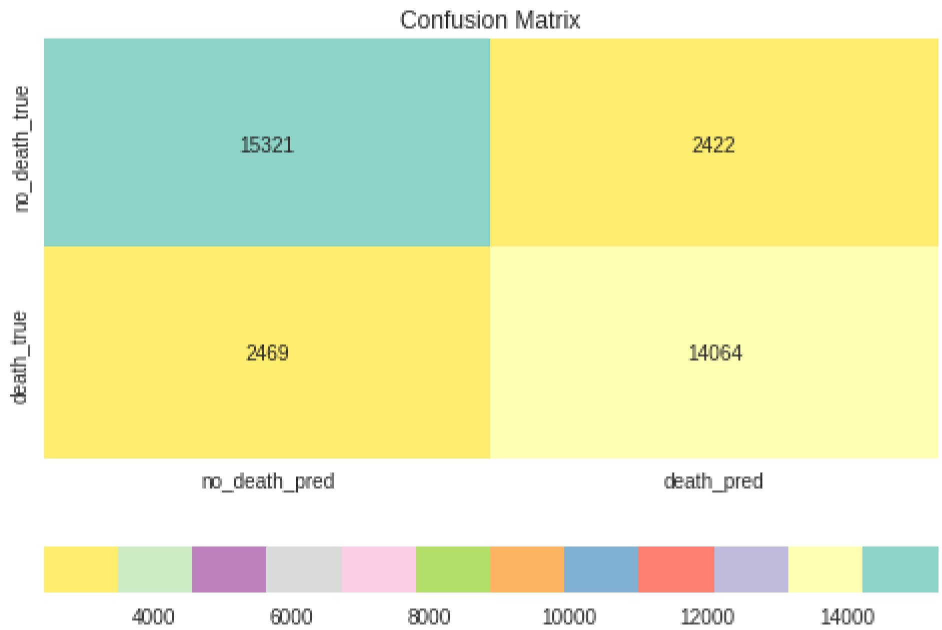
```
train: 0.64% | validation: 0.16% | test 0.2%
```

```
rf.fit(X_train, y_train)

print('-'*40)
print("Train:")
```

```
print("Accuracy: %.2f" %round((accuracy_score(y_train, rf.predict(X_train))*100),2))
print("F_score: %.2f" %round((fbeta_score(y_train, rf.predict(X_train), beta =0.5)*100),2))
print('-'*40)
print("Validation:")
print("Accuracy: %.2f" %round((accuracy_score(y_validation,
rf.predict(X_validation))*100),2))
print("F_score: %.2f" %round((fbeta_score(y_validation, rf.predict(X_validation), beta =
0.5)*100),2))
print('-'*40)
print("Test:")
print("Accuracy: %.2f" %round((accuracy_score(y_test, rf.predict(X_test))*100),2))
print("F_score: %.2f" %round((fbeta_score(y_test, rf.predict(X_test), beta = 0.5)*100),2))
print('-'*40)
```

```
----------------------------------------
Train: Accuracy: 99.94

F_score: 99.93

----------------------------------------
Validation: Accuracy: 86.07

F_score: 85.72

----------------------------------------
Test: Accuracy: 86.09

F_score: 85.65

----------------------------------------
```

How often does the model predict no deaths?



<table>
<tr><td></td><td></td><td colspan="2">Predicted</td></tr>
<tr><td></td><td></td><td>Negative (N)
-</td><td>Positive (P)
+</td></tr>
<tr><td rowspan="2">Actual</td><td>Negative
-</td><td>True Negative (TN)</td><td>False Positive (FP)
Type I Error</td></tr>
<tr><td>Positive
+</td><td>False Negative (FN)
Type II Error</td><td>True Positive (TP)</td></tr>
</table>

Given death occurrences in attacks, the model mistakenly predicted no deaths in 14.93% cases.

- TP: True Positive: Predicted values correctly predicted as actual positive
- FP: Predicted values incorrectly predicted an actual positive. i.e., Negative values predicted as positive
- FN: False Negative: Positive values predicted as negative
- TN: True Negative: Predicted values correctly predicted as an actual negative

3.5 ROC curve

The ROC curve shows the trade-off between sensitivity (or TPR) and specificity (1 – FPR). Classifiers that give curves closer to the top-left corner indicate a better performance. As a baseline, a random classifier is expected to give points lying along the diagonal (FPR = TPR). The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.

Note that the ROC does not depend on the class distribution. This makes it useful for evaluating classifiers predicting rare events such as diseases or disasters. In contrast, evaluating performance using accuracy (TP +

TN)/(TP + TN + FN + FP) would favour classifiers that always predict a negative outcome for rare events.

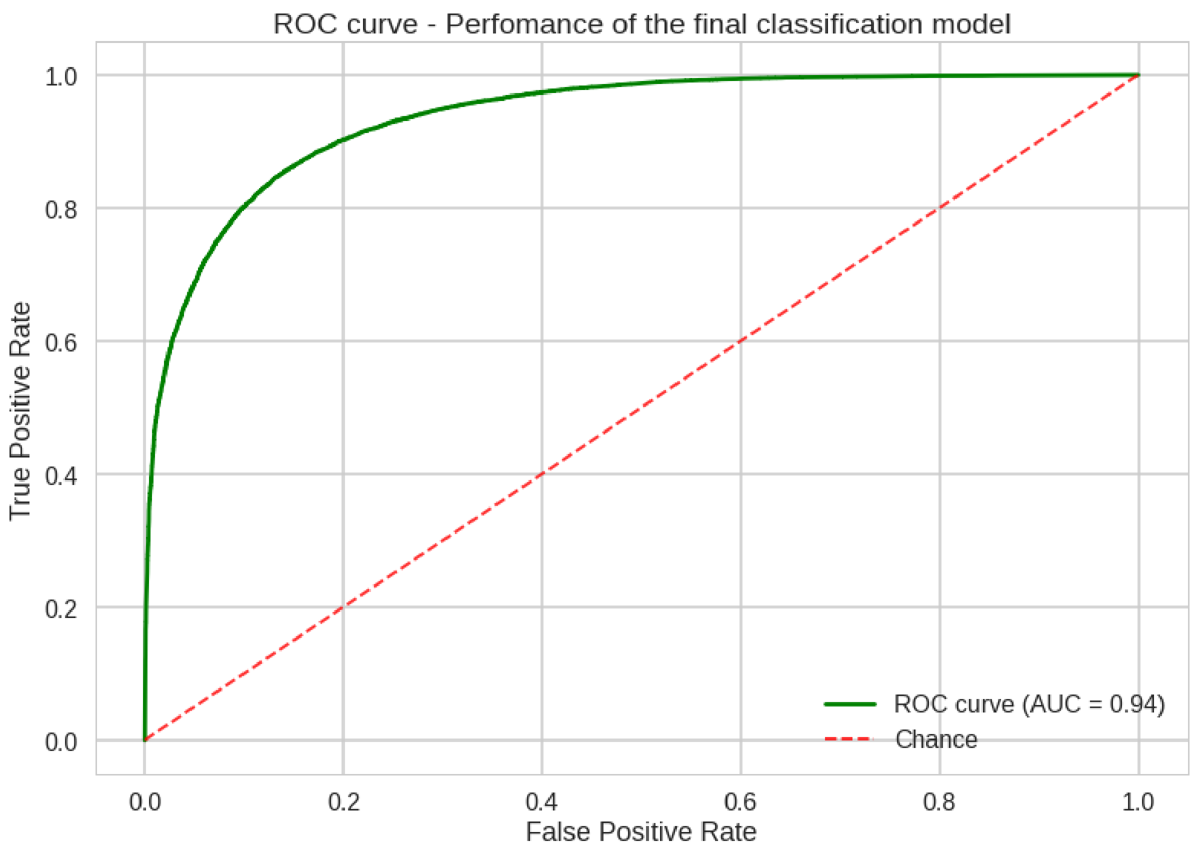
The AUC performs well as a general measure of predictive accuracy.

```
from sklearn.model_selection import learning_curve
from sklearn.model_selection import ShuffleSplit



def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None,
                        n_jobs=None, train_sizes=np.linspace(.1, 1.0, 5)):
    plt.figure()
    plt.title(title)
```

```
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    plt.grid()

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
             label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation score")

    plt.legend(loc="best")
    return plt
plt.style.use(['seaborn-whitegrid','seaborn-poster'])
y_pred_proba = new_rf.predict_proba(X_test)[::,1]
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr, tpr,color='g', label = "ROC curve (AUC = "+str(round(auc,2))+')')
plt.plot([0,1],[0,1], linestyle = '--', lw=2, color ='r', label='Chance', alpha=.8)
plt.legend(loc=4)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

plt.title("ROC curve - Performance of the final classification model")

plt.show()



Obviously the higher the AUC score, the better the model is able to classify observations into classes and our model achieves 0.94 descrimination.

CHAPTER 4
CONCLUSION

Terrorist attacks are among the causes of national instability. A clear understanding of how this event is occurring will help us to conduct more in-depth investigations.

Through this research, it is possible to conclude that the use of Machine Learning techniques was able to visualise and predict Death occurrences. The results section shows that there has been a considerable growth in terrorist attacks since 2010 and that due to the classification models, it was possible to determine the probability of which region and type of attack may occur.

Concerning the number of attacks by region in the visualisation, it was obtained that there is a probability that they will happen in the Middle East & North Africa and followed by South Asia.

Regarding the types of attacks, there is still the probability that bombs and explosions are involved, followed by armed assault.

The results have been successfully achieved by using the historical data collected from the GTD. The models that were made through Random Forest give the same probabilistic results from 86% of assertiveness. The robustness of the proposed prediction has been evaluated using various performance metric like accuracy, specificity, sensitivity, f1 score and ROC curve

Taliban , ISIL , Al - shabaab , Boko Haram are the most notorious groups that cause threats.

Most the terrorist attacks happen for the various factors and motivation as well, thus even with using powerful deep learning model over the global terrorist database this not sufficient to determine what are the next attacks instead rising suspicions. To achieve such goal we have to study the behavior of the terrorist over social networking.