Seminar in Research and Research Methodology

INFO 5082

Research project on

Global Terrorism Database

Vinitha Ponugoti

11333896

Table of Contents:

List of Figures:

Introduction:

The Global Terrorism Dataset is the second-hand data which is available at Kaggle. The dataset consists of 135 columns and more than 1,80,000 rows with a lot of outliers, empty spaces or columns and null values. The project reflects the collection of the global database and applying visualization methods to predict the terrorism rate around the world. This Global terrorism data comprises of all the terrorist data and the original reporting of the data starts from the year 1970. The data used for the analysis is updated periodically by the National Consortium for Study of Terrorism and response to terrorism (START) by University of Maryland. We use 20 columns from this dataset and each row represents a terrorist attack. This data was collected from various countries and they were obtained and stored in different techniques.

For an event to be eligible to be in the GTD event should be an intentional incident, the event should have some level of violence or threat of violence, the event must be aimed at the political, religious, economic goal.

Statement of the problem:

Analyze data patterns and provide valuable insights from the existing data to help countries safeguard against potential terrorist attacks and their own vulnerabilities.

Review of Literature:

Global Terrorism Dataset was introduced in a paper in Terrorism and Political Violence by Gary LaFree and Laura Dugan of START, in 2007 [1]. Since then a huge amount of research has been done on this dataset, like Building terrorism knowledge graph abbreviated as TKG [2], using Clustering methods and Association rule mining to discover some interesting similarities in the data [3], Providing visual analytical approaches for identifying related entities [4].

Apart from visual research on the dataset, a lot of predictive research has also been done like predicting terrorist groups responsible of attacks [5], Predicting the types of terrorist attacks using SVM, KNN, bagging and c4.5 [6], Prediction of future terrorist attacks using deep neural networks [7], An XGBoost-based casualty prediction method for terrorist attacks [8].

While most of the above research is focused on different predictions like causality, attack types, terrorist groups. There was no research on predicting attack success, failure. Also, most of the predictive research done on the dataset has not taken into fact that the dataset is skewed towards more successful attacks which might affect operating characteristics of the model.

Objectives of the Study:

The objective of the study is to mine various patterns and information from GTD dataset through various visualization techniques and to understand factors which lead to a terrorist attack being successful thereby building a model to predict if any such attacks will be successful or not.

Data Collection (Give the link to the files, or upload your files if they are not accessible online):

The Global Terrorism Dataset is the second-hand data which is available at Kaggle. The dataset consists of 135 columns and more than 1,80,000 rows with a lot of outliers, empty spaces or columns and null values.

Global Terrorism Dataset Link : Here

Data Cleaning:

Data set consists of around 135 columns. Most of this data is sparse or unuseful so I filtered out the columns that I wanted for the data analysis and model creation. Later during the correlation analysis process, I removed null values and empty cells from the dataset and then filtered out the data from 1998 to have recent information in the analysis.

Data Dictionary:

All the columns that I used from this dataset are shown below

Name	Input	Description
iyear	Int	Year of Terrorist activity
imonth	Int	Month of Terrorist activity
iday	Int	Day of Terrorist activity
country_txt	String	Name of the country
region_txt	String	Name of the Region
city	String	Name of the City
success	Boolean	Whether incident was success or not
latitude	Float	Geographical locations
longitude	Float	Geographical locations
attacktype1_txt	String	Type of attack 1
nkill	Int	Number of kills
summary	String	Summary of the incident
gname	String	Name of the terrorist group
targtype1_txt	String	Target that terrorists aimed
target1	String	Name of the victim
weaptype1_txt	String	Type of weapon used for attack
motive	String	Motive of the attack
nwound	Int	Number of people who were wounded
natlty1	Int	Nationality of the people
suicide	Boolean	whether attack is suicide or not

Exploratory Data Analysis (EDA) and Hypotheses for the Study:

Exploratory data analysis is the significant phenomenon of managing the initial examinations on data so as to evaluate the patterns and anomalies. It is used to test hypotheses and determine the assumptions by using the graphical representations and summary statistics. It is the important process to use the data first and focus on collecting various determinations from it. In Python, I have imported all the libraries that were useful to me for visualizations such as pandas, seaborn, matplotlib, plotly, and Word Cloud. The specific columns such as Year, Month, Day, Country,

Region, City, Success, Latitude, Longitude, Attack type, Killed, Wounded, Target, Summary, Group, Target_type, Weapon_type, Motive, Casualties are filtered out for the analysis part.

```
import plotly.express as px
import pycountry_convert as pc
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud, STOPWORDS
terror=pd.read_csv('gtd.csv',encoding='ISO-8859-1', low_memory=False)
terror.rename(columns={'iyear':'Year','imonth':'Month','iday':'Day','country_txt':'Country','success' : 'Success',
       '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',
         'suicide': 'suicide', 'natlty1': 'nationality'},
       inplace=True)
terror=terror[['Year','Month','Day','Country','Region','city', 'Success','latitude','longitude','AttackType','Killed',
       'Wounded','Target','Summary','Group','Target_type','Weapon_type','Motive', 'suicide', 'nationality']]
terror['casualities']=terror['Killed']+terror['Wounded']
terror.head(3)
```

	Year	Month	Day	Country	Region	city	Success	latitude	longitude	AttackType	Killed	Wounded	Target	Summary	Group	T
0	1970	7	2	Dominican Republic	Central America & Caribbean	Santo Domingo	1	18.456792	-69.951164	Assassination	1.0	0.0	Julio Guzman	NaN	MANO-D	
1	1970	0	0	Mexico	North America	Mexico city	1	19.371887	-99.086624	Hostage Taking (Kidnapping)	0.0	0.0	Nadine Chaval, daughter	NaN	23rd of September Communist League	G (
2	1970	1	0	Philippines	Southeast Asia	Unknown	1	15.478598	120.599741	Assassination	1.0	0.0	Employee	NaN	Unknown	

First, I have imported pandas and then read the data by importing the dataset which is gtd.csv by giving the encoding type as 'ISO-8859-1'. Later I renamed all the columns that I wanted for analysis to avoid complexity by including them in the variable called 'terror'. Also, I combined both 'Killed' and 'Wounded' and named it as 'Casualties'. Then, head (3) gives us the first three rows of the dataset just to have an idea.

Data Analytics:

Below is the concise summary of my dataset.

```
terror.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 181691 entries, 0 to 181690
Data columns (total 21 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Year         181691 non-null  int64
 1   Month        181691 non-null  int64
 2   Day          181691 non-null  int64
 3   Country      181691 non-null  object
 4   Region       181691 non-null  object
 5   city         181257 non-null  object
 6   Success      181691 non-null  int64
 7   latitude     177135 non-null  float64
 8   longitude    177134 non-null  float64
 9   AttackType   181691 non-null  object
 10  Killed       171378 non-null  float64
 11  Wounded      165380 non-null  float64
 12  Target       181055 non-null  object
 13  Summary      115562 non-null  object
 14  Group        181691 non-null  object
 15  Target_type  181691 non-null  object
 16  Weapon_type  181691 non-null  object
 17  Motive       50561 non-null   object
 18  suicide      181691 non-null  int64
 19  nationality  180132 non-null  float64
 20  casualities  164817 non-null  float64
dtypes: float64(6), int64(5), object(10)
memory usage: 29.1+ MB
```

```
print('Country with Highest Terrorist Attacks:',terror['Country'].value_counts().index[0])
print('Regions with Highest Terrorist Attacks:',terror['Region'].value_counts().index[0])
print('Maximum casualities in an attack are:',terror['casualities'].max(),'that took place in',
      terror.loc[terror['casualities'].idxmax()].Country, ' - Target : ',terror.loc[terror['casualities'].idxmax()].Target )
```

```
Country with Highest Terrorist Attacks: Iraq
Regions with Highest Terrorist Attacks: Middle East & North Africa
Maximum casualities in an attack are: 9574.0 that took place in United States  - Target :  Passengers and crew members on A
merican Airlines Flight 11 and the people working in the North Tower of the World Trade Center in New York City
```

To start with I have extracted the Country and region with most terrorist attacks until now. I also looked for an attack which had most casualties to see which terrorist attack had the most impact until now.

If you can see the output, you can clearly understand that the country with highest terrorist attacks is Iraq, regions with highest terrorist attacks are Middle east and North Africa, Also the maximum casualties in an attack are 9574 which took place in United States targeting passengers and crew members on American Airlines Flight and the people working in the North tower of the World Trade center in New York City. In short, it is called the 9/11 attack that happened in 2001 which shook the whole world.

Data Visualization and Results Report:

1. Number of terrorist activities each year

```
#plt.style.use('fivethirtyeight')
plt.subplots(figsize=(15,6))
sns.countplot(x='Year',data=terror.loc[terror['Year'] > 1997],palette='RdYlGn_r',edgecolor=sns.color_palette('dark',7))
plt.xticks(rotation=90)
plt.title('Number Of Terrorist Activities Each Year')
plt.show()
USterror = terror.loc[(terror['Country'] == 'United States') & (terror['Year'] > 1997)]
plt.subplots(figsize=(15,6))
sns.countplot(x='Year',data=USterror,palette='RdYlGn_r',edgecolor=sns.color_palette('dark',7))
plt.xticks(rotation=90)
plt.title('Number Of Terrorist Activities in US Each Year')
plt.show()
```

From the below first graph, the highest number of terrorist attacks happened in the year 2014 with around 17000 attacks across the world; later it decreased a bit. In general we can see a gradual increase in terrorist attacks in the world.

If you observe the second graph for Terrorist attacks in the United States, the maximum number of terrorist attacks were 2017 with almost 70 attacks. We can discern from the graph that after the 9/11 attack in 2001, the attacks decreased rapidly until 2013 and then increased eventually with higher numbers than before.

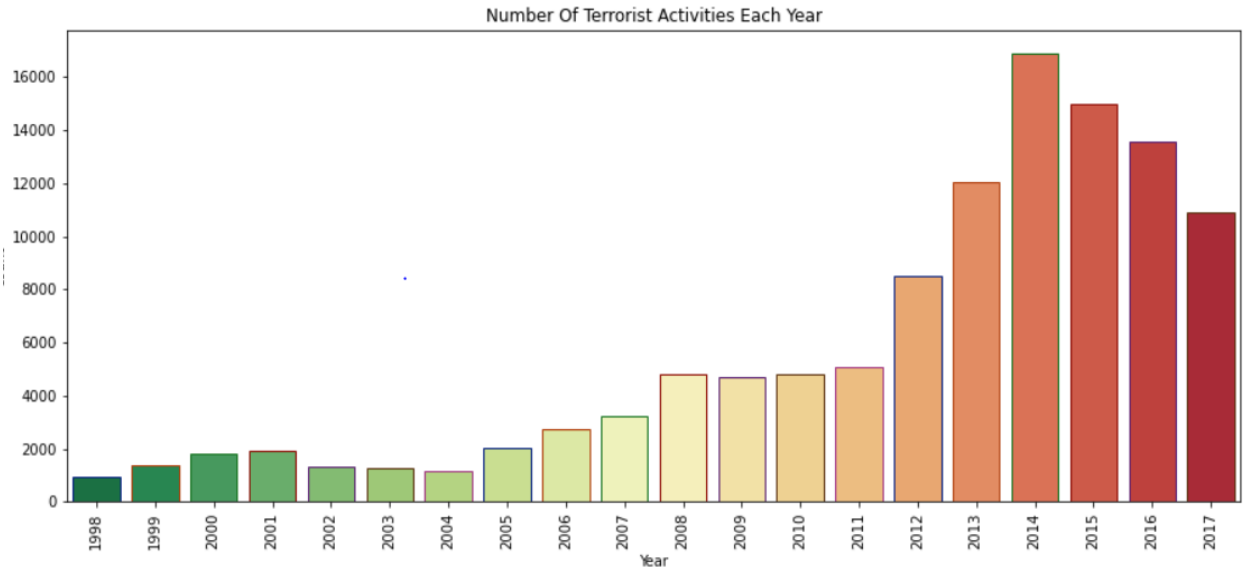

Fig 1: Number of Terrorist Activities Each Year

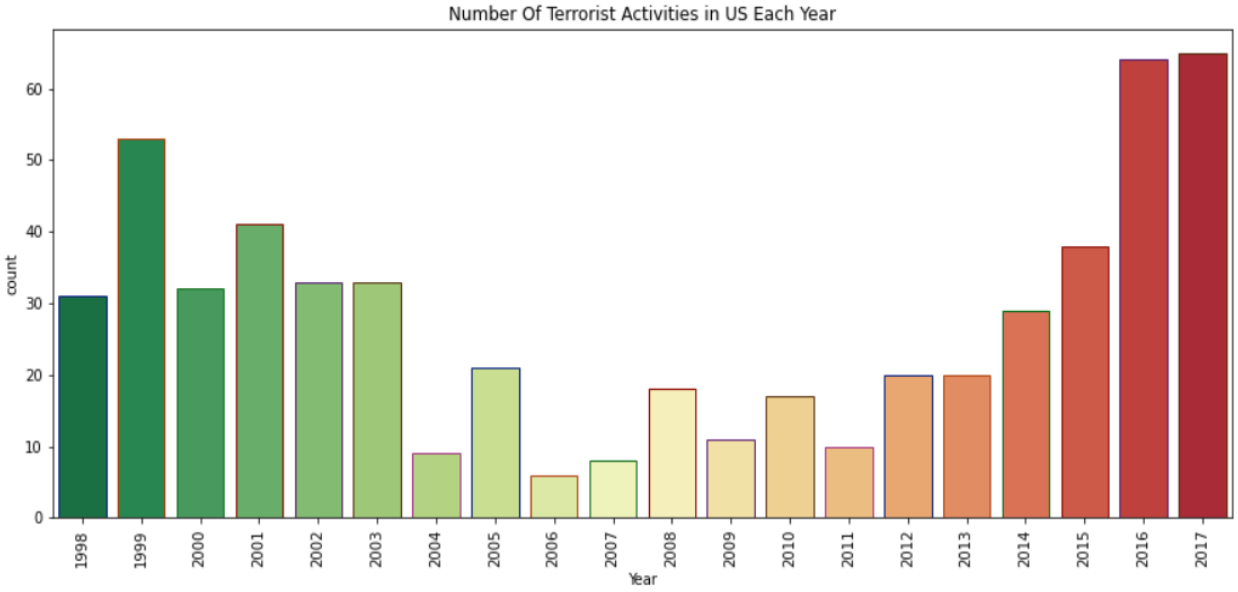

Fig 2: Number of Terrorist Activities in US Each Year

2. Attacking methods by terrorists:

```
plt.subplots(figsize=(15,6))
sns.countplot(x='AttackType',data=terror,palette='inferno',order=terror['AttackType'].value_counts().index)
plt.xticks(rotation=90)
plt.title('Attacking Methods by Terrorists')
plt.show()
```

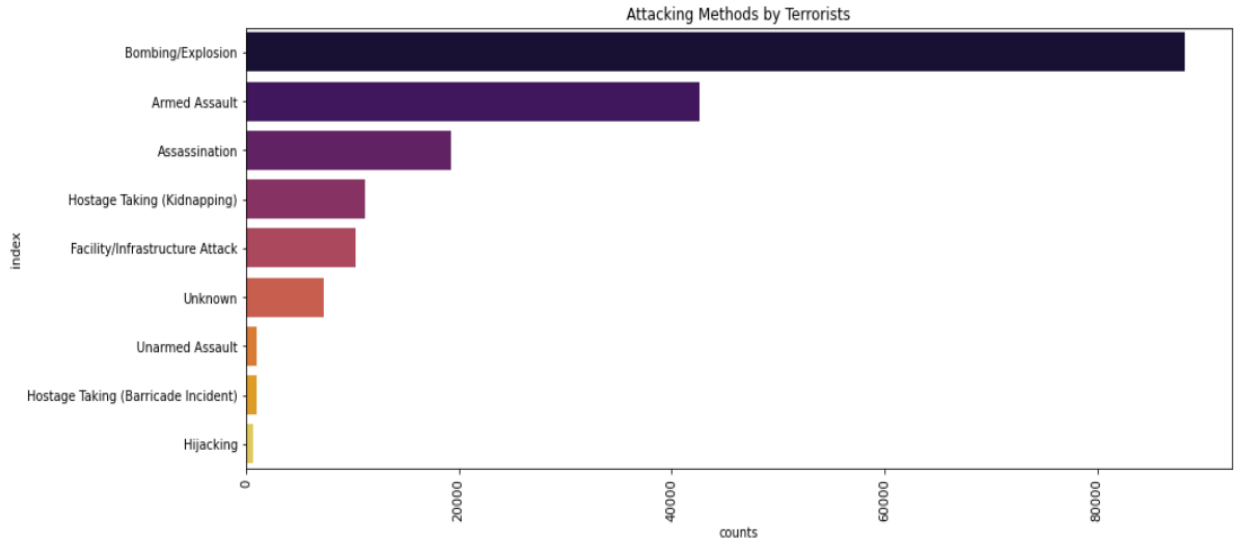

Fig 3: Attacking Methods by Terrorists

The above visualization explains the attack methods that have been used by terrorists, the highest number of terrorist attacks has happened because of Bombing / Explosive with a count of more than 80000. The second highest method is Armed assault with a count of 40000 and the most unused method by terrorists is hijacking. This graph says that the terrorists preferred bombing to attack where there is a chance of destroying more people and the area, followed by Armed assault and assassination.

3. Distribution of attack failure and success over the years

```
import warnings
warnings.filterwarnings("ignore")
plt.subplots(figsize=(15,6))
sns.violinplot(x = terror['Success'], y = terror['Year'], data = terror, hue =terror['Success'])
plt.title('Distribution of Attack Failure and Success over the years')
plt.show()
```

From the below graph, you can observe both the failure and success rates from the years 1970 to 2020. Failure rate is given in blue and considered as ‘0’ and success rate is given in orange and considered as ‘1’. You can observe that the failure rate is high between the years 2010 and 2020 and the success rate is also high in the years 2010 to 2020. By this we can understand that both failure and success rates are in between 2010 and 2020 which means most of the attacks happened from 2010 to 2020. Looking at the width of the graph during the years 1970 to 2000 we can see that successful attacks were much more consistent that Failed attacks.

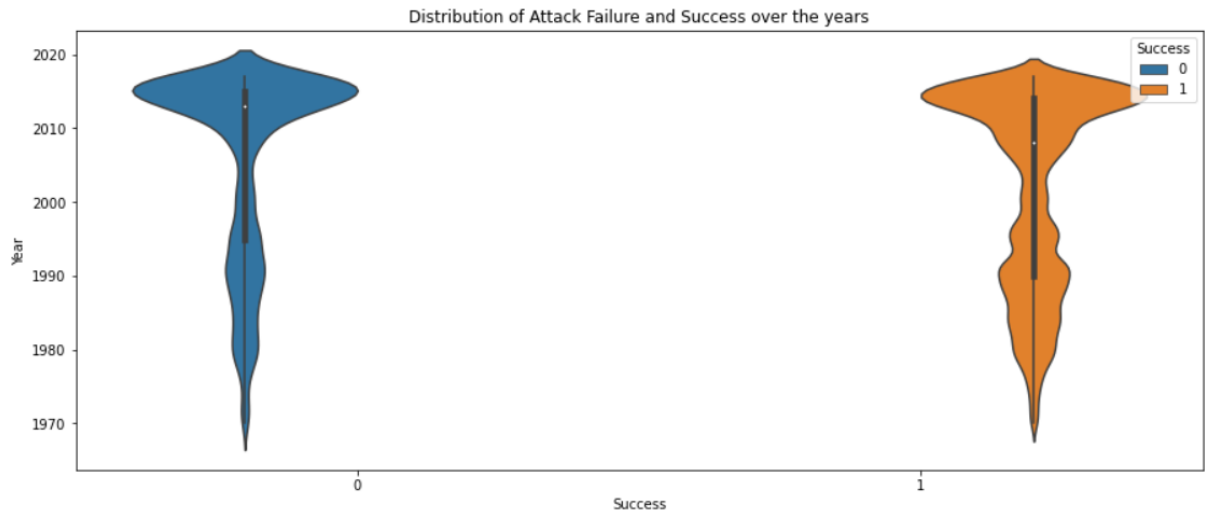

Fig 4: Distribution of Attack Failure and Success over the years

4. Number of terrorist activities by Region:

```
plt.subplots(figsize=(15,6))
sns.countplot(x='Region',data=terror,palette='RdYlGn',edgecolor=sns.color_palette('dark',7),
              order=terror['Region'].value_counts().index)
plt.xticks(rotation=90)
plt.title('Number Of Terrorist Activities By Region')
plt.show()
```

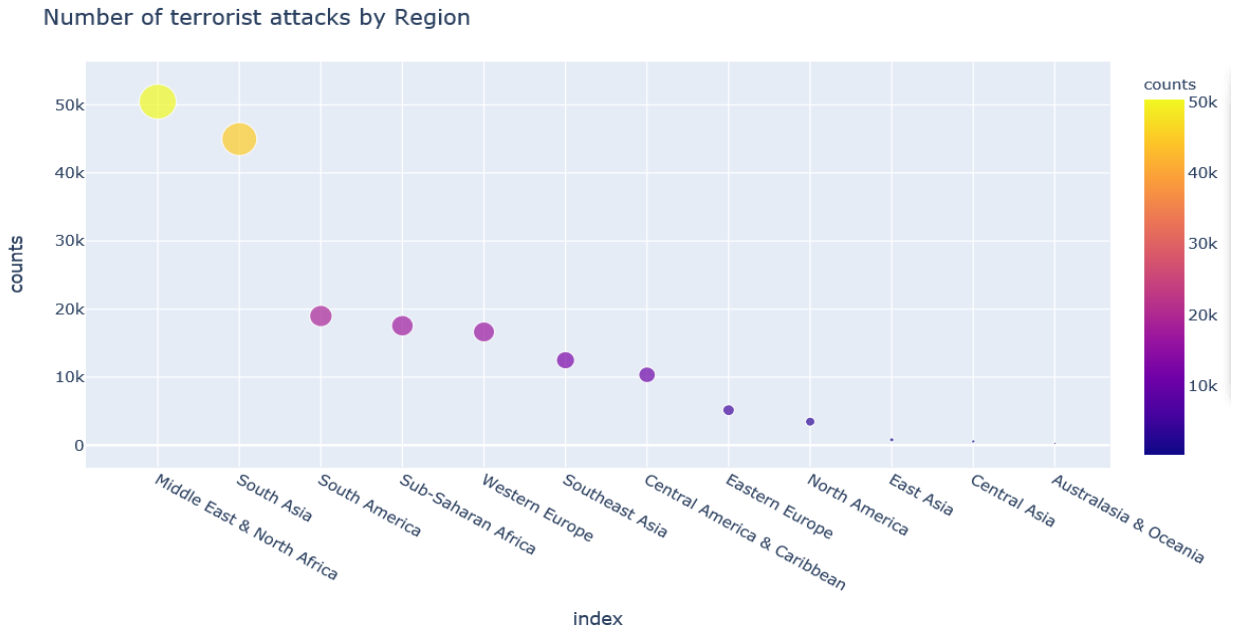

Fig 5: Number of terrorist attacks by Region

From the above graph, count of attacks was shown region wise. The highest number of terrorist attacks happened in the region Middle East and North Africa with count of 50000, followed by South Asia with almost 45,000 and the region with least attacks is Australasia and Oceania. The sharp drop in bubble size from top 2 regions indicates the regions Middle East & North Africa, South Asia have been the center for terrorist activities throughout the world.

5. Attack count over the years by region:

```
terror_region=pd.crosstab(terror.Year,terror.Region)
terror_region.plot(color=sns.color_palette('Set2',12))
fig=plt.gcf()
fig.set_size_inches(18,6)
plt.title('Attack count over the years by region')
plt.show()
```

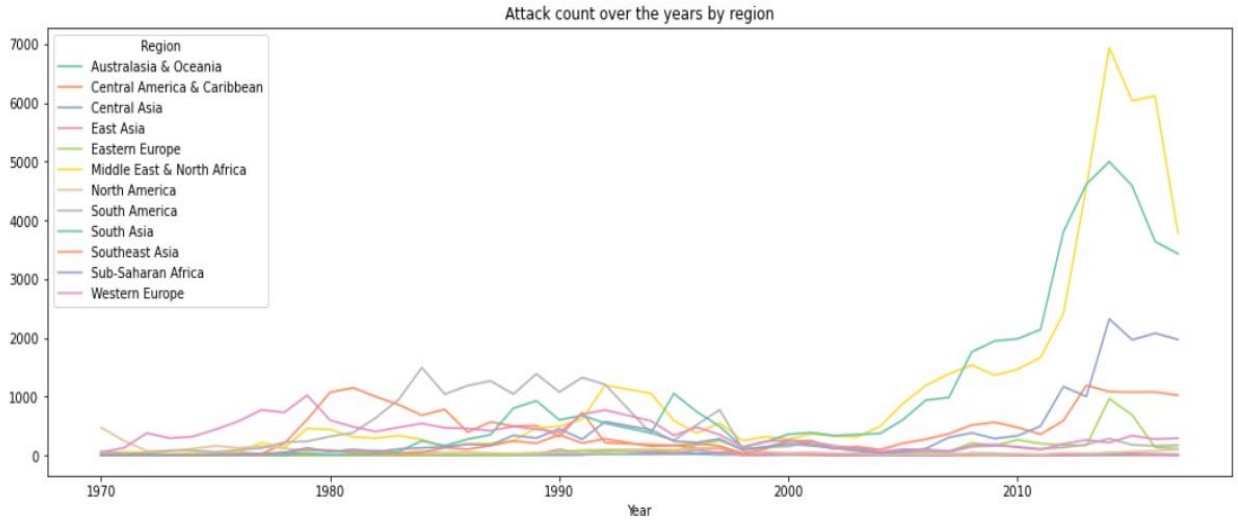

Fig 6: Attack count over the years by region

If you observe from the above line graph, the attack count was shown for every region from 1970 to 2017. The highest number of terrorist attacks happened in the region Middle East and North Africa with a yellow line which is at a very low level in the beginning and it eventually increased during 2010 and 2017. The second highest is South Asia with a green line which also increased between the years 2010 and 2017. From this we can understand that the attack count is more in recent years. We can also see that there was a sharp increase in attack count in the Middle East and North Africa region post 2010.

6. Count of favorite targets:

```
plt.subplots(figsize=(15,6))
sns.countplot(terror['Target_type'],palette='inferno',order=terror['Target_type'].value_counts().index)
plt.xticks(rotation=90)
plt.title('Count of Favorite Targets')
plt.show()
```

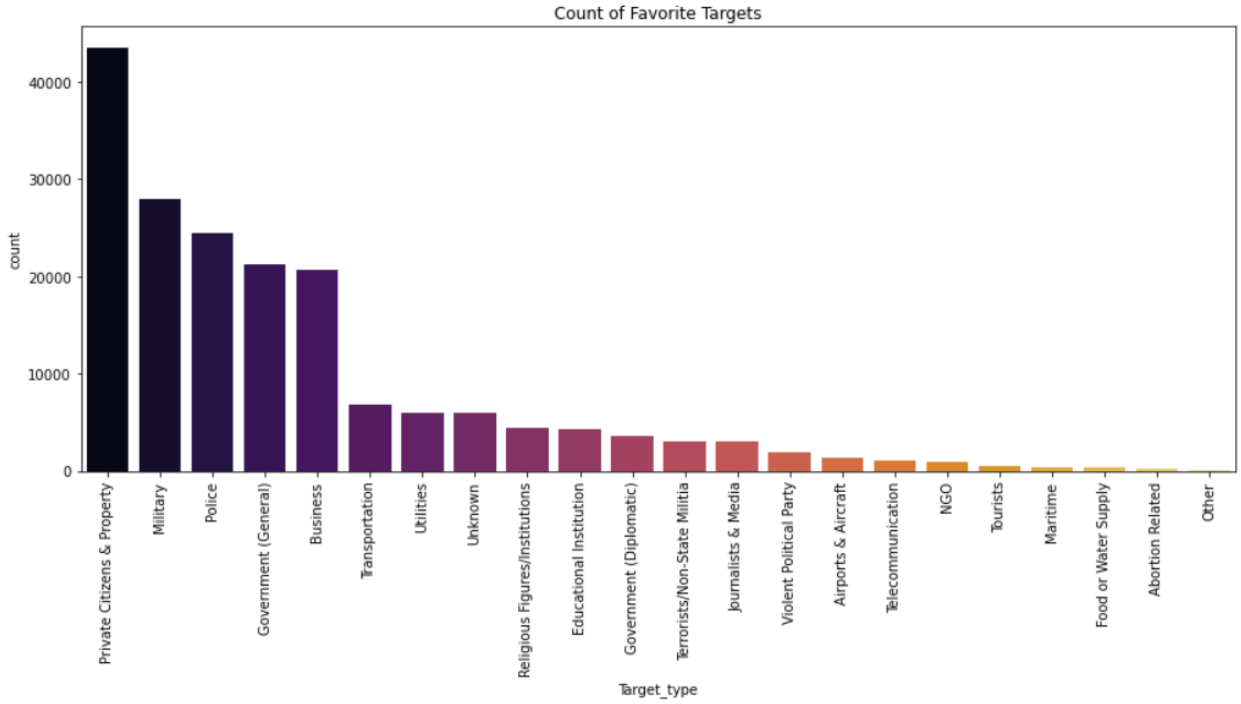

Fig 7: Count of Favorite Targets

From the above graph you can understand that the most favorite targets for the terrorists are private citizens and property with more than 40000 count of attacks. After citizens, terrorists preferred to attack the Military and police with a respective count of 28000 and 25000. We can think that the reason for attacking the police and military is that they are protecting the nation with world peace and act as a barrier for intruders. Later, Government and business are getting attacked with around 20000 count.

7. Groups with highest number of kills:

```
plt.subplots(figsize=(15,6))
df2 = terror.groupby(['Group']).sum()['Killed'].sort_values(ascending = False).head(10)
#d = dict(df2)
df2.plot(kind = 'pie')
plt.title("Groups with highest number of kills")
```

From the pie chart below, we can observe that the group with the highest number of kills is unknown surprisingly with almost 40% of the people killed. Next place is taken by ISIL with 20% of the kills. Later comes the Taliban which is a popular group that the whole world knows with 15% kills.

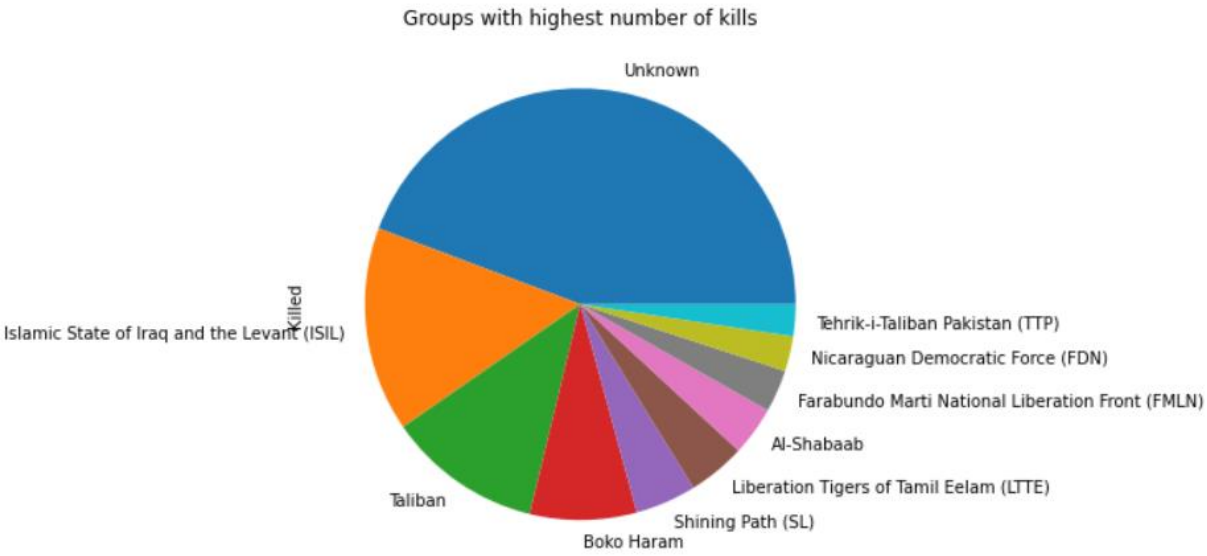

Fig 8: Groups with highest number of kills

8. Intensity of terrorist attacks from 1970 to 2015

```
import plotly.express as px
import pycountry_convert as pc

fdf = terror.groupby(['Country']).size().reset_index(name = 'counts')
def convert(name):
    try:
        name = pc.country_name_to_country_alpha3(name, cn_name_format="default")
        return name
    except Exception as e :
        #print(e)
        return 'Unknown'

fdf['Country'] = fdf['Country'].apply(lambda x: convert(x))

fig = px.choropleth(fdf, locations="Country",
                    color="counts",
                    hover_name="Country", # column to add to hover information
                    color_continuous_scale=px.colors.sequential.Plasma
                   )
fig.update_layout(
    title_text = 'Intensity of terrorist attacks from 1970 to 2015',

)
fig.show()
```

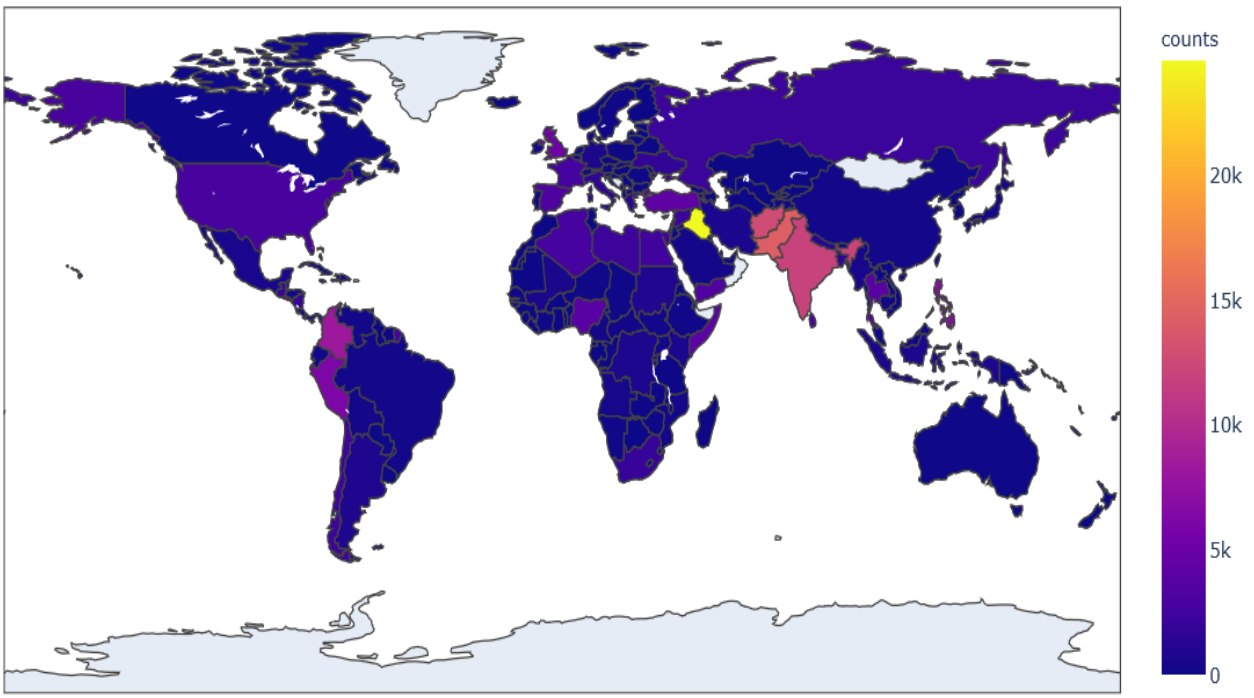

Fig 9: Intensity of terrorist attacks from 1970 to 2015

The above geographical map shows us the intensity of terrorist attacks from 1970 to 2015 with a color scale in the right to understand the count. Iraq is the country in yellow with more intensity of attacks with more than 24k people. The second place is taken by Pakistan with around 14k people, then comes Afghanistan with approximately 13k count and India with almost 11k people.

9. People killed over the years

```
yeardf = terror.groupby(['Year'])['Killed'].sum().reset_index(name='counts')

fig = px.treemap(yeardf, path=['Year'], values='counts',
                 color='counts', hover_data=['counts'],)
fig.update_layout(
    title_text = 'People killed over the years',

)
fig.show()
```

From the below tree map we can analyze that the highest number of people were killed in the year 2014 with more than 40,000 kills observing from the color scale and the tree map gives us the years depending upon the size of the tile from highest to lowest. The least number of people killed is in the year 1971 with 173 count.

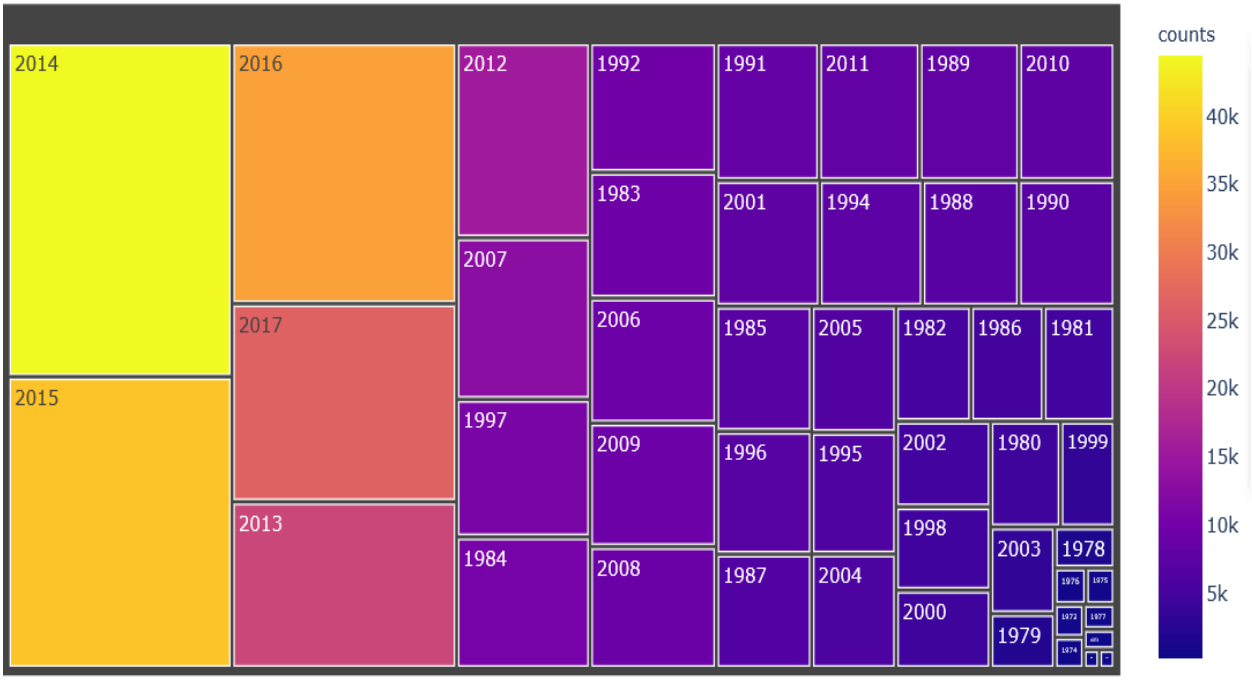

Fig 10: People killed over the years

10. Word Cloud on Motive:

```
stopwords = set(STOPWORDS)
stopwords.add('unknown')
stopwords.add('motive')
stopwords.add('specific')
stopwords.add('.')
stopwords.add(',')
stopwords.add(';')
terror.Motive.fillna("", inplace = True)
comment_words = ''
# iterate through the csv file
for val in terror.Motive:
    if val == '':
        continue
    val = str(val)
    val = val.lower()
    comment_words = comment_words + ' ' + val



wordcloud = WordCloud(width = 800, height = 800,
            background_color ='white',
            stopwords = stopwords,
            min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.title('Wordcloud on Motive')
plt.show()
```

Fig 11: Word Cloud on Motive

To understand key reasons for a terrorist attack, I have plotted a word cloud on the motive given for each terrorist. To build this world cloud I have combined all the motives to form a document while removing commonly used stop words in English. A word cloud shows the words in a document with sizes based on frequency of the word in the document. As we examine the word cloud, we can see words like attack, intimidate, retaliation, Islamic, part, responsibility etc. which inform us about key motives for attacks. For example, words like intimidate, attack, retaliation point that some of the terrorist's attacks are either a response to an incident or an act to intimidate the people. Other words like Islamic, minority, Sunni, Shiite etc. point towards some religious beliefs which may be the cause for the attack.

Predictive Analysis:

(a) Feature selection:

Our aim for the predictive analysis is to predict whether an attack will be successful or not. For our first step to this we need to figure out which of the columns in the dataset should we choose as our features to our model.

1. Correlation Analysis:

Correlation means association - more precisely it is a measure of the extent to which two variables are related.

```
d = pd.read_csv('gtd.csv', encoding='ISO-8859-1')
d = d.rename(
columns={ 'iyear':'year',
         'country':'country', 'natlty1':'nationality', 'success':'success','targtype1':'target', 'region' : 'region',
         'weaptype1':'weapon', 'attacktype1':'attack',
         'suicide':'suicide', 'imonth' : 'month'})

dff=d[d['year']>1998]
```

Here, I filtered out some columns which have sparse data by giving the encoding type as 'ISO-8859-1'. The columns that I filtered now are different from the columns that I used for visualizations. They are country, nationality, success, target, region, weapon, attack, suicide, and month. I wanted to take recent data, so I have given the year as >1998.

```
corr = dff[['attack', 'country', 'weapon', 'latitude','longitude','nationality','target','region',
            'suicide','success', 'month']].corr()
plt.figure(figsize=(15,6))
sns.heatmap(corr, annot = True)

plt.title('Correlation Analysis')
plt.show()
```

I wanted to know if any of the columns have direct correlation with attack success or failure. So, I have chosen columns with most relevance and did a correlation analysis on these columns. I have generated a heatmap for this.

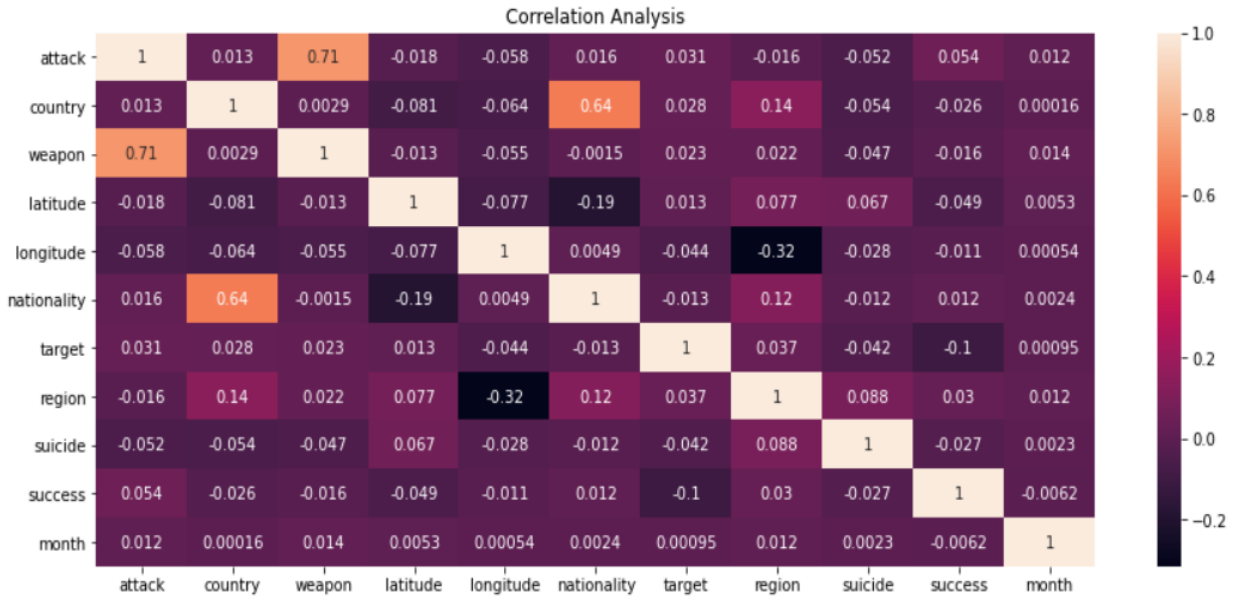

Fig 12: Correlation Analysis between success and other columns

If you see there is a very good correlation between attack and weapon but no correlation for success with any of the columns. To have a correlation it should be above 0.5. To be more accurate it should be 1 for a perfect correlation. If you observe all the success values are closer to 0.03. This means that there is no direct correlation between the columns.

There is no direct correlation in the data. According to an article by *Yi Feng, Dujuan Wanf and Yunqiang Yin [1]*, it was found out that a feature count of ten results in a very good classification outcome. So, I chose the following features that had continuous data to determine whether an attack is successful or not.

Columns: attack, country, weapon, latitude, longitude, nationality, target, region, suicide, success

(b) Data Analysis:

Before training the model, I wanted to understand the distribution of attack failure and success.

```
#count of success and failures in dataset
dff['success'].astype('str')
dff.groupby('success').describe()
```

	eventid			
	count	mean	std	min
success				
0	14327.0	2.013358e+11	4.034638e+08	1.999010e+11
1	98923.0	2.011976e+11	4.391617e+08	1.999010e+11

This is written to find out the count of success and failures in the dataset. We understood that the failure count is 14327 and the success count is 98923. Our data has 85 % Successful attacks and 15% Attacks which failed. By training our models on this data we get 90% Accuracy but our True Negatives are very low, that means our model tends to predict that every attack is successful and will not get which factors are responsible for an attack to be successful.

To avoid this we filter data so that we have 60% successful attacks and 40% attacks which failed so that the model correctly understands the factor contributing to it.

(c) Data Preprocessing:

As mentioned in data analysis we filter our feature data to have 60% Successful and 40% Failed attacks.

```
dff['success'].astype('float')
newdf = dff[dff['success']==1].sample(n=22000)
df = pd.concat([newdf, dff[dff['success'] == 0]])
#DF is Final data set after filteration
success=pd.DataFrame(df['success'])
features = pd.DataFrame(df[['attack', 'country', 'weapon', 'latitude',
          'longitude','nationality','target','region','suicide', 'month']])
features.fillna(features.mean(), inplace=True)
features['latitude'].astype(float)
features['longitude'].astype(float)
features['nationality'].astype(float)
features['month'].astype(float)
print(features)
```

If you see the data, the dataset has all the data in both string and number format. For the model to understand, the data I chose is number data.

Before we train our model, I split the data into 80% training and 20% test data. We train our model on 80% Train data and test the accuracy of our model on 20% test data. We avoid training our data on 100% of the data to avoid overfitting the model.

We also split our test data to contain 50% Successful and 50% Failed attacks to gauge our True Positives more accurately to True Negatives ratios.

```
features_train, features_test,target_train, target_test = train_test_split(features,success,
                                                          test_size = 0.2,random_state=0)
#feature_train - 80% of feature data, feature_test - 20% feature data
#target_train - 80% of result(successful or not) data, target_test - 20%
target_test['success'].astype('str')
print(target_test.groupby('success').size())
target_test['success'].astype('float')
```

```
success
0    2886
1    4380
dtype: int64
```

```
ds = features_test.join(target_test)
#totalset.groupby('success').size()
sds = ds[ds['success']==1].sample(n=2886)
fds = pd.concat([sds, ds[ds['success'] == 0]])

target_test = fds['success']
print(target_test)
features_test = fds.drop(['success'], axis=1)
features_test
```

```
84662     1
101408    1
128262    1
97318     1
116294    1
         ..
158701    0
97581     0
82170     0
174421    0
111402    0
Name: success, Length: 5772, dtype: int64
```

Model Creation:

This is a supervised classification problem i.e. we want to classify whether an attack will be successful or not based on a set of predetermined features. We can solve this classification problem using many algorithms/techniques. We test out a couple of Decision Tree based models for our classification problem and determine the best among them using Accuracy and other characteristics

(a) XGBoost Model:

Gradient boosting refers to a class of ensemble machine learning algorithms that can be used for classification or regression predictive modeling problems.

Ensembles are constructed from decision tree models. Trees are added one at a time to the ensemble and fit to correct the prediction errors made by prior models. This is a type of ensemble machine learning model referred to as boosting.

Models are fit using any arbitrary differentiable loss function and gradient descent optimization algorithm. This gives the technique its name, "gradient boosting," as the loss gradient is minimized as the model is fit, much like a neural network.

```
train = xgb.DMatrix(features_train, label = target_train)
test = xgb.DMatrix(features_test, label = target_test)
```

We convert our data frame into Dmatrix which is an internal data structure that is used by XGBoost, which is optimized for both memory efficiency and training speed.

Hyperparameter Optimization:

With XGBoost there are a huge number of parameters which can affect our model accuracy. To determine the best values for this, we can do a manual trial and error method, but this will take a huge amount of time due to the number of possibilities. So, we are using Grid Search and get the optimum parameter results for our model. Here is an example code which yielded output as below.

```
from sklearn.model_selection import GridSearchCV
gsc = GridSearchCV(
    estimator=xgb.XGBClassifier(),
    param_grid={

                'gamma': [0.5, 1, 1.5, 2, 5],
                'subsample': [0.6, 0.8, 1.0],

                'max_depth': [3, 4, 5,6],

                },
    n_jobs=-1,
    cv=5,
    scoring='neg_mean_squared_error',
    verbose=100
  )

grid_result = gsc.fit(features_train, target_train)
best_params = grid_result.best_params_
print(best_params)
```

We get the following best hyperparameters after running grid search

{'max_depth': 6, 'subsample': 0.5}

```
param = { 'max_depth': 6 , 'subsample': 0.5, 'num_class' : 2}

epoch = 10
model = xgb.train(param, train, epoch)
```

Parameters:

1. max_depth: Maximum depth of a tree
2. subsample: Ratio of data to be given to each tree
3. num_class: count of Classification outcomes that a model can give, in our case whether it is pass or fail.

```
xgpred = model.predict(test)
#print(xgpred)
print("XGBoost accuracy score: " )
print(accuracy_score(target_test, xgpred))
```

```
XGBoost accuracy score:
0.7307692307692307
```

We train our model with training data and hyper parameters. Then we ask the model to predict on test data. We calculate accuracy based on the actual outcome and predicted outcome. We got the accuracy of 73%.

(b) Random Forest Model:

Random forest consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.

```
forest= RandomForestClassifier(n_estimators=10)
forest = forest.fit( features_train, target_train.values.ravel()  )
print("Random Forest accuracy score: " )
print(forest.score(features_test,target_test))
predict = forest.predict(features_test)
```

```
Random Forest accuracy score:
0.7524255024255024
```

Random Forest has a hyper parameter called n_estimators i.e., the number of trees that are built in a model. By trial and error, we finalize our n_estimators to be 10. Then we train the model on our trained data and calculate our accuracy which turns out to be 75.2%

(C) Evaluation and Comparison of models:

Our XGBoost model yields an accuracy of 73%, whereas the Random Forest model yields 75.2%. To understand which model is better for our use case we analyze more on the prediction outcomes for the models.

1. Confusion Matrix:

Confusion matrix is a tabular summary of the number of correct and incorrect predictions made by a model.

As our classification outcomes are only 3, we get a confusion matrix of size 2*2;

Each Cell in our confusion matrix signifies one of the following scenarios.

What is the confusion matrix?

Actual - Fail, predicted - Fail **True Negative**	Actual - Fail, Predicted - Success **False Positive**
Actual - Success, Predicted - Fail **False Negative**	Actual - Success, Predicted - Success **True Positive**

XG Boost Confusion Matrix:

```
xgconfm = confusion_matrix(target_test, xgpred)
print(xgconfm)
plt.figure(figsize=(15,6))
sns.heatmap(xgconfm, annot = )
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('CM XGBoost')
plt.show()
```

```
[[1579 1307]
 [ 247 2639]]
```

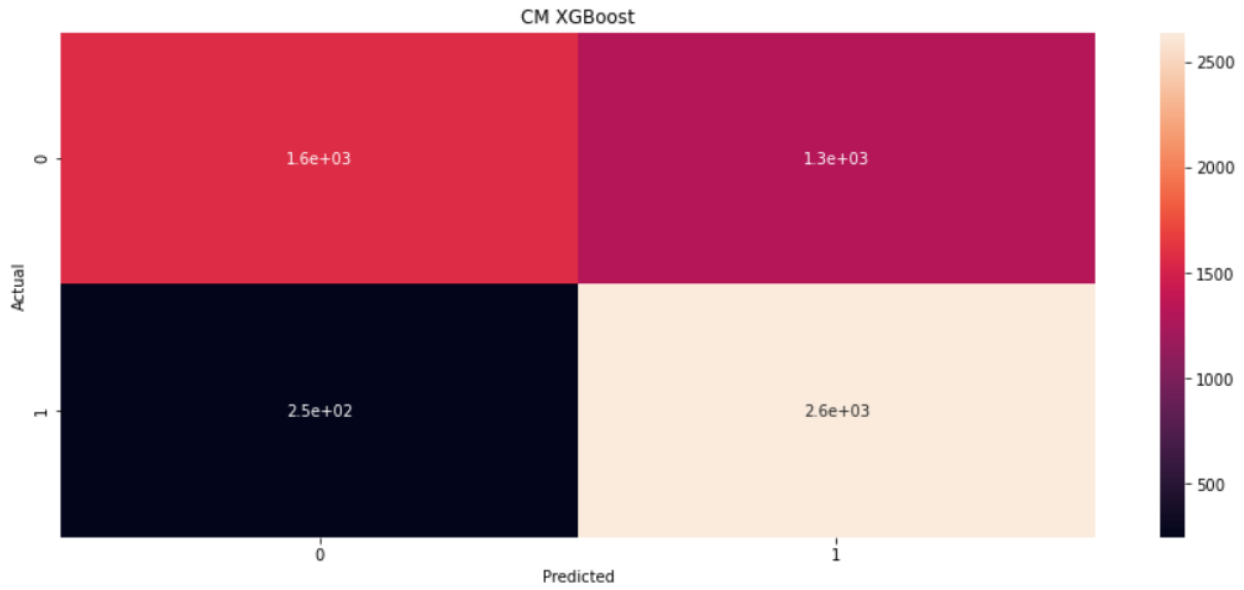

Fig 13: XGBoost Confusion Matrix

Random Forest Confusion Matrix:

```
confm = confusion_matrix(target_test, predict)
print(confm)
plt.figure(figsize=(15,6))
sns.heatmap(confm, annot = True)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('CM Random forest')
plt.show()
```

```
[[2026  860]
 [ 569 2317]]
```

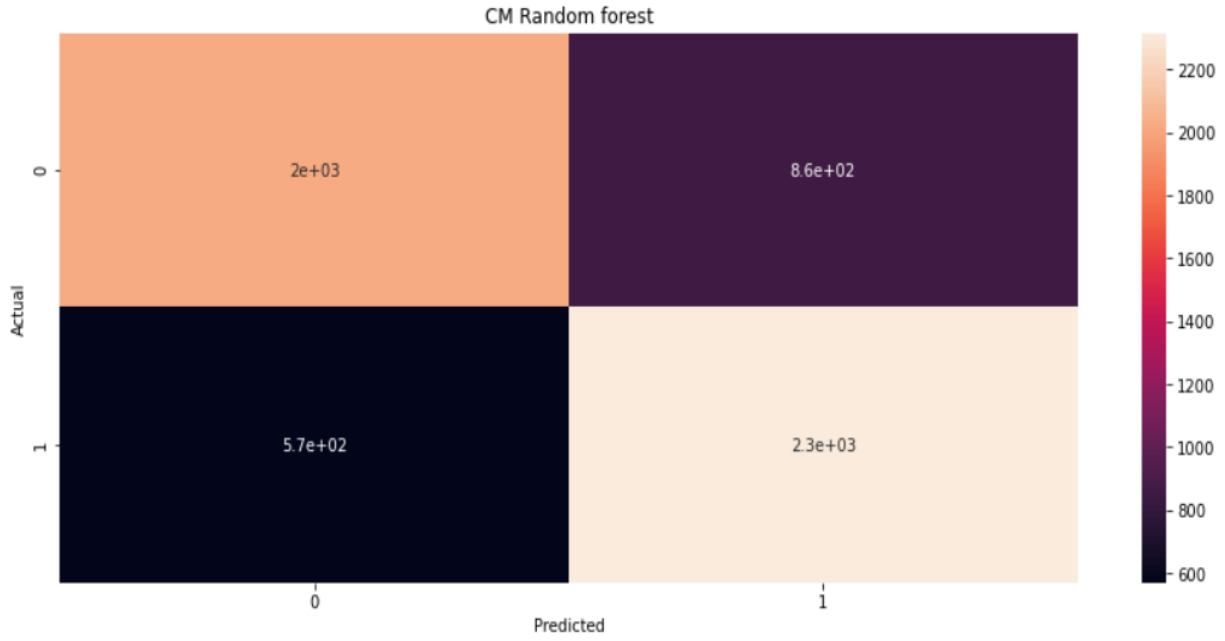

Fig 14: Random Forest Confusion Matrix

After comparing the two matrices of XGBoost and Random Forest we can see that both the models have good True Positives i.e. 2639 and 2317. But XGBoost yields mediocre True Negative values than Random Forest i.e. 1579 to 2026, respectively. This means that our XGBoost model tends to give more False positive results than Random forest which will cause false alarms.

2. ROC Curve and AUC:

Receiver Operating Characteristics (ROC) curve is a graph between True Positive Rate to False Positive Rate.

True Positive Rate = Sensitivity = TP/TP+FN

False Positive Rate = 1 - Specificity = 1 - FP/TN+FP = TN/TN+FP

ROC Curve is a performance measurement for classification problems at various thresholds settings.

AUC - Area Under the Curve - It Signifies the capability of a classification model at predicting 0s as 0s and 1s as 1s. The higher the AUC the better the model is.

XGBoost AUC - ROC Curve:

```
false_positive_rate, true_positive_rate, thresholds = roc_curve(target_test, xgpred)
roc_auc = auc(false_positive_rate, true_positive_rate)
print('AUC = %0.4f'% roc_auc)
plt.title('Receiver Operating Characteristic - XG Boost')
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 = 0.7308
```

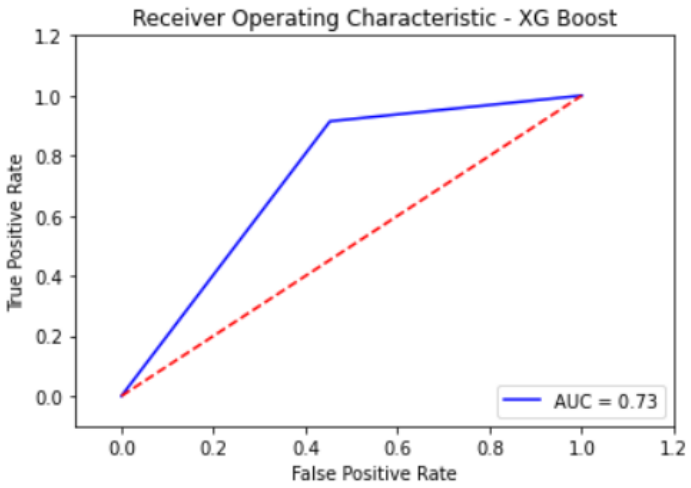

Fig 15: XGBoost ROC Curve

Random Forest AUC - ROC Curve:

```
false_positive_rate, true_positive_rate, thresholds = roc_curve(target_test, predict)
roc_auc = auc(false_positive_rate, true_positive_rate)
print('AUC = %0.4f'% roc_auc)
plt.title('Receiver Operating Characteristic - Random Forest')
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 = 0.7524
```

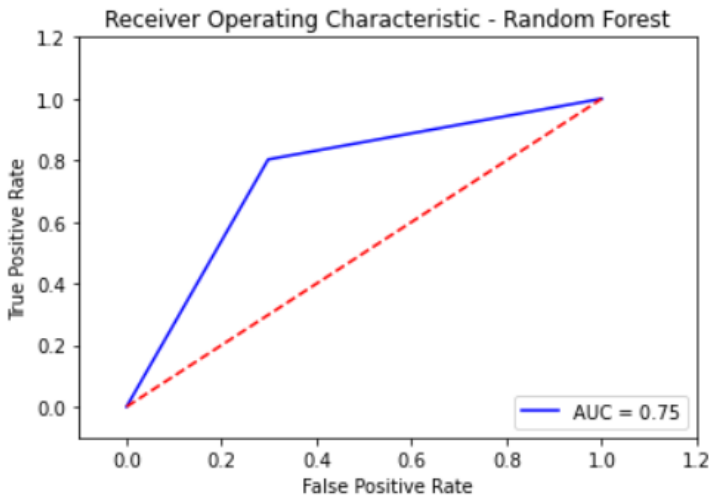

Fig 16: Random Forest ROC Curve

Looking at AUC-ROC Curve we can see that Random Forest has slightly better AUC of 0.75 than XGBoost model's 0.73. This signifies that the Random Forest Model is better at Classification for the dataset than the XGBoost model.

3. Precision:

Precision or true positive rate says how well did my model catch positive cases. This is the most important metric for the evaluation of our models as predicting positive results correctly is important because predicting a successful attack as a failure will result in potential loss of life.

XGBoost Precision:

```
xgprecision = xgconfm[1][1]/(xgconfm[1][1]+xgconfm[0][1])
xgprecision
```

```
0.6687785098834262
```

Random Forest Precision:

```
precision = confm[1][1]/(confm[1][1]+confm[0][1])
precision
```

```
0.7293043751967264
```

With higher precision, the Random forest model is better suited overall.

Using the Prediction Model:

With this prediction model we can choose a likely scenario of a terrorist attack and check whether the attack may be successful or not with various input combinations:

```
forest.predict([[3,217,6,33.21659, -97.13162 ,217 ,14,1,1,12]])
#Bombing-  US -  Explosive Denton_Latitude Denton_longitude  US Private_citizen North America Yes December
# Will a bombing in Denton US in December month targeted at a Private Citizen Will be Successful or Not
# Will Not be successful
```

```
array([0], dtype=int64)
```

Conclusion:

With such classification models, Law enforcement officers and Countries can try out different combinations of attacks on a sensitive region to gauge which area is vulnerable to which combinations of attacks and what the most likely outcome of such an attack might be. This will help officials in finding out improvement areas against certain types of attacks.

In future we can improve the accuracy of our models by optimizing different hyper parameters for the XGBoost models and also to try some other models like KNN and SVM with an emphasis on precision, we can also improve our feature selection using regression methods to find out which features have good relation with success.

Bibliography:

[1] LaFree, Gary; Dugan, Laura (2007). "Introducing the Global Terrorism Database" (PDF). Terrorism and Political Violence. 19 (2): 181–204. doi:10.1080/09546550701246817.

[2] T. Xia and Y. Gu, "Building Terrorist Knowledge Graph from Global Terrorism Database and Wikipedia," 2019 IEEE International Conference on Intelligence and Security Informatics (ISI), Shenzhen, China, 2019, pp. 194-196, doi: 10.1109/ISI.2019.8823450.

[3] J. Górecki, K. Slaninová and V. Snášel, "Visual investigation of similarities in Global Terrorism Database by means of synthetic social networks," 2011 International Conference on Computational Aspects of Social Networks (CASoN), Salamanca, 2011, pp. 255-260, doi: 10.1109/CASON.2011.6085954.

[4] Alex Godwin, Remco Chang, Robert Kosara, William Ribarsky, "Visual analysis of entity relationships in the Global Terrorism Database," Proc. SPIE 6983, Defense and Security 2008: Special Sessions on Food Safety, Visual Analytics, Resource Restricted Embedded and Sensor Networks, and 3D Imaging and Display, 69830G (15 April 2008); https://doi.org/10.1117/12.778084

[5] Tolan, Ghada & Abou-El-Enien, Tarek & Khorshid, Motaz. (2015). HYBRID CLASSIFICATION ALGORITHMS FOR TERRORISM PREDICTION in Middle East and North Africa. International Journal of Emerging Trends & Technology in Computer Science (IJETTCS). 4. 23-29.

[6] Meng X, Nie L, Song J (2019) Big data-based prediction of terrorist attacks. Comput Electr Eng 77:120–127

[7] M. I. Uddin, N. Zada, F. Aziz et al., "Prediction of future terrorist activities using deep neural networks," Complexity, vol. 2020, pp. 1–16, 2020.

[8] Feng, Y., Wang, D., Yin, Y. et al. An XGBoost-based casualty prediction method for terrorist attacks. Complex Intell. Syst. 6, 721–740 (2020). https://doi.org/10.1007/s40747-020-00173-0