**GLOBAL TERRORISM**

Terrorism is a complex political and social phenomenon. Terrorist attacks have a significant threat to the safety and security of the international community and have become one of the greatest obstacles to the sustainable development of global social security. Antiterrorism is an important part of global security governance, which is a sustainability issue that guarantees global security development. At present, terrorist attacks occur frequently, which leads to significant threats and poses a challenge to global security governance. According to statistics from the Global Terrorism Database (GTD), more than 200,000 terrorists’ attacks have been recorded from 1970 to the present day. Terrorist attacks typically involve high lethality and destructive power and directly cause massive casualties and property losses. In addition, they bring tremendous psychological pressure on people.

**Characteristics of the GTD**

* Contains information on over 200,000 terrorist attacks.
* Currently the most comprehensive unclassified database on terrorist attacks in the world
* Includes information on more than 95,000 bombings, 20,000 assassinations, and 15,000 kidnappings and hostage events since 1970.
* Includes information on at least 45 variables for each case, with more recent incidents including information on more than 120 variables.
* More than 4,000,000 news articles and 25,000 news sources were reviewed to collect incident data from 1998 to 2019 alone.



**Decision Tree:**

Diagram

Description automatically generated

Decision tree is a Supervised learning technique that can be used for both classification and regression problems but is generally preferred for solving classification problems. It is a tree classifier, where internal nodes represent the features of the dataset, the branches represent the decision rules, and leaf node represents the outcome.

**Support vector machine:**

Support vector mechanism is same as like Decision tree, used for both classification as well as regression problems. However, primarily, it is used for Classification problems in Machine learning.

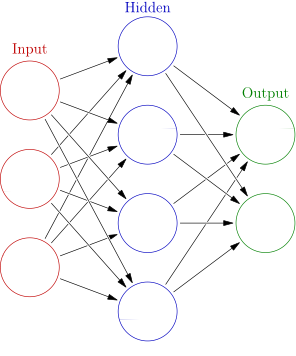
The goal of the SVM algorithm is to create the best line or decision boundary that can sort out n-dimensional space into classes. So that we can easily put the new data point in the correct category in future. This best decision boundary is called a hyperplane.

SVM chooses the extreme vectors that help in creating the hyperplane. These are nothing but support vectors.



From the above diagram, we can see that there are two different categories that are classified using a hyperplane.

**Neural Networks:**



Neural networks can be created from at least three layers of neurons:

The input layer, the hidden layer, and the output layer. As the neural networks “learns” the data, the weights , or strength of the connections between these neurons are “fine-tunes” allowing the network to come up with accurate prediction.

**Analysis of Dataset:**

**Step1: Import python libraries to analyze dataset**

Different python libraries were imported which are needed:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

import time

from IPython.display import clear\_output

%matplotlib inline

import warnings

warnings.filterwarnings('ignore')

**Step2: Import Dataset**

We used the pandas library to read the data from the dataset

terrorism = pd.read\_csv('globalterrorismdb\_0718dist.csv', encoding ='ISO-8859-1', low\_memory=False)

terrorism.head() (the head function is used to display the first few columns in the dataset)

A screenshot of a computer

Description automatically generated with medium confidence

**Step3: Data Frame Structure**

data.info() using this command we can display info of the dataset.

Graphical user interface, table

Description automatically generated with medium confidence

**PART-2:Data analysis of Global Terrorism from 1970 to 2017**

#### **Comparing the number of Terrorist Attacks and the number of Dead/Injured people from 1970 to 2017**

#Filter Data to choose Number Of Dead/Injured People In Terrorist Attacks From 1970 to 2017

people\_damage = data[["Year","Damage"]].groupby('Year').sum()

list\_year = data["Year"].unique().tolist()

#draw bar chart

fig, ax1 = plt.subplots(figsize = (20,6))

ax1.bar(people\_damage.index, [i[0] for i in people\_damage.values], color= '#0063B1' )

ax1.set\_xticklabels(np.arange(1970, 2018, step=1), rotation=90)

ax1.set\_ylabel('Number Of Dead/Injured People', size = 12)

ax1.set\_xlabel('Year', size = 12)

ax1.set\_title('Number of Terrorist Attacks vs Number of Dead/Injured people From 1970 to 2017', fontsize= 15, pad= 10, weight ='bold',

color = sns.cubehelix\_palette(8, start=.5, rot=-.75)[-3])

ax2 = ax1.twinx()

#Filter & get a number of attacked in the world from 1970 to 2017

number\_attack = []

for year in list\_year:

number\_attack.append(len(data[data['Year'] == year][["Year"]]))

number\_attack.insert(23, 0)

#draw plot chart

ax2.set\_ylabel('Number Of Terrorist Attacks', size = 12,rotation=-90)

ax2.plot(range(1970, 2018), number\_attack, 'r--o', mfc='k', label='Number Of Terrorist Attacks')

plt.xticks(np.arange(1970, 2018, step=1))

plt.legend(loc='upper left')

plt.show()

Chart, histogram

Description automatically generated

##### This graph shows the number of global terrorist attacks and number of dead/injured people by year from 1970 to 2017. Overall, the number of attacks is increasing. The number of dead/injured people also increases respectively. The number of attacks increased significantly from 2012 to 2015. The year 2015 is unlucky year with nearly 17000 reported terrorist attacks.

### Analyze the Terrorist attack types and damage

### atk\_filtered =data['Attacktype'].apply(lambda x: x if x in ['Bombing/Explosion','Armed Assault','Assassination', 'Hostage Taking', 'Facility/Infrastructure Attack'] else 'Others')

### attack\_type = atk\_filtered.value\_counts().tolist()

#Pie chart representation of Terrorist attack types

labels = ['Bombing/Explosion','Armed Assault','Assassination',

'Hostage Taking','Facility/Infrastructure Attack','Others']

sizes = []

for i in attack\_type:

percent = i\*100/len(data['Attacktype'])

sizes.append(percent)

fig, ax = plt.subplots(figsize=(10,10))

patches, texts, autotexts = ax.pie(sizes, labels=labels, autopct='%1.1f%%',

startangle = -20, shadow = True,

explode = (0.05, 0, 0, 0, 0, 0),

colors = sns.color\_palette("Set2", 8)[:5]+[(0.5843137254901961, 0.6470588235294118, 0.6509803921568628)],

textprops={'fontsize':15,'weight':'light','color':'k'})

ax.axis('equal')

plt.title('Terrorist attack types', fontsize= 25, pad= -70, weight ='bold',

color = sns.cubehelix\_palette(8, start=.5, rot=-.75)[-3]) #pad change the distance from title to graph

plt.tight\_layout()

ax.legend(loc='lower right',framealpha = 0.5,bbox\_to\_anchor=(1.8,0.5,0.1,1), prop={'size': 14})

fig.show()

Chart, pie chart

Description automatically generated

**Attack types and number of Dead/Injured People**

cat = ['Bombing/Explosion','Armed Assault','Assassination','Hostage Taking','Facility/Infrastructure Attack']

color\_cat = sns.color\_palette("Set2", 8)[:5]

color\_cat\_dict = dict(zip(cat, color\_cat))

table\_1 = data[['Attacktype','Damage']].groupby('Attacktype',as\_index = False).sum().sort\_values(by='Damage', ascending=False)

table\_1 = table\_1.reset\_index()

inci = [88255,42669,7276,19312,12149,659,1015,10356]

table\_1["Incidents"] = np.array(inci)

table\_1["Damage\_rate"] = table\_1["Damage"]/table\_1["Incidents"]

table\_1

Table

Description automatically generated

**chart of Terrorist Attack Types and the Damage**

labels = table\_1['Attacktype'].tolist()

x = np.arange(len(labels))

dmg = table\_1['Damage'].tolist()

gray = (0.5843137254901961, 0.6470588235294118, 0.6509803921568628)

color\_list = [color\_cat\_dict[ter\_type] if ter\_type in color\_cat\_dict.keys() else gray for ter\_type in labels]

d\_rate = table\_1["Damage\_rate"].tolist()

fig, ax1 = plt.subplots(figsize=(15,8))

# Bar chart

ax1.bar(labels, dmg,

color = color\_list,

align='center')

# Number in bar chart

for i,v in enumerate(dmg):

ax1.text(i-0.3, v-13000 if v==37209 else v+3000, str(round(v)),

color='w' if v==37209 else 'k',

fontweight='bold')

# Insert a second plot -line plot

ax2 = ax1.twinx()

ax2.plot(labels, d\_rate, linestyle='--', linewidth =4, marker ='o',

markerfacecolor='black', markersize =10,

label='Mean Of Dead/Injured People',

color = '#C44D51')

plt.title('Terrorist Attack Types and Damage', fontsize= 25, pad= 20, weight ='bold',

color = sns.cubehelix\_palette(8, start=.5, rot=-.75)[-3])

ax1.set(xlabel='Types Of Terrorist Attacks', ylabel='Number Of Dead/Injured People')

ax1.set\_xticklabels(labels, rotation=45)

plt.yticks(fontsize=10)

ax2.legend(loc='upper center')

fig.show()

**Chart

Description automatically generated**

##### *The total number of dead/injured people caused by Bombing/Explosion is the highest (514.233 people) since 48.6% of terrorist attacks belong to this type. However, Hijacking is actually the type of attacks which kills and injures the most number of people per attack (>30 people per attack). Facility/Infrastructure Attack type kills the least number of people both in total and per case.*

### Analyze the Weapon types and Damage

### Chart representation of weapons types

### weap\_labels = ['Explosives','Firearms','Unknown','Incendiary','Others']

### weap\_sizes = []

### for j in weaptype:

### percent = j\*100/len(data['Weaptype'])

### weap\_sizes.append(percent)

### fig, ax = plt.subplots(figsize=(10,10))

### patches, texts, autotexts = ax.pie(weap\_sizes, labels=weap\_labels, autopct='%1.1f%%',

### startangle = -20, shadow = True,

### explode = (0.05, 0, 0, 0, 0),

### colors = sns.color\_palette("Set2", 8)[:4:1]+

### [(0.5843137254901961, 0.6470588235294118, 0.6509803921568628)],

### textprops={'fontsize':15,'weight':'light','color':'k'})

### ax.axis('equal')

### plt.title('Weapon types', fontsize= 25, pad= 20, weight ='bold',

### color = sns.cubehelix\_palette(8, start=.5, rot=-.75)[-3]) #pad change the distance from title to graph

### ax.legend(loc='lower right',framealpha = 0.5,bbox\_to\_anchor=(1.2,0.5,0.1,1), prop={'size': 14})

### fig.show()

### Chart, pie chart Description automatically generated

### Weapon types and number of Dead/Injured People

### table\_2 = data[['Weaptype','Damage']].groupby('Weaptype',as\_index = False).sum().sort\_values(by='Damage',

### ascending=False)

### df\_count = data['Weaptype'].value\_counts()

### df\_count = df\_count.reindex(table\_2['Weaptype'].values)

### table\_2['WeapCount'] = df\_count.values

### table\_2["Weap\_damage\_rate"] = table\_2["Damage"]/table\_2["WeapCount"]

### table\_2

### Table Description automatically generated

### Weapon types and number of Dead/Injured People

### weap\_labels = table\_2['Weaptype'].tolist()

### y = np.arange(len(weap\_labels))

### weap\_dmg = table\_2['Damage'].tolist()

### w\_cat = ['Explosives','Firearms','Unknown','Incendiary']

### w\_color\_cat = sns.color\_palette("Set2", 8)[:4:1]

### w\_color\_cat\_dict = dict(zip(w\_cat, w\_color\_cat))

### w\_gray = (0.5843137254901961, 0.6470588235294118, 0.6509803921568628)

### w\_color\_list = [w\_color\_cat\_dict[ter\_type] if ter\_type in w\_color\_cat\_dict.keys()

### else w\_gray for ter\_type in weap\_labels]

### w\_rate = table\_2["Weap\_damage\_rate"].tolist()

### fig, ax1 = plt.subplots(figsize=(15,8))

### # Bar chart

### ax1.bar(weap\_labels, weap\_dmg,

### color = w\_color\_list,

### align='center')

### # Number in bar chart

### for i,v in enumerate(weap\_dmg):

### ax1.text(i-0.3, v-13000 if v==37209 else v+3000, str(round(v)),

### color='w' if v==37209 else 'k',

### fontweight='bold')

### # Insert a second plot -line plot

### ax2 = ax1.twinx()

### ax2.plot(weap\_labels, w\_rate, linestyle='--', linewidth =4, marker ='o',

### markerfacecolor='black', markersize =10,

### label='Mean Of Dead/Injured People',

### color = '#C44D51')

### plt.title('Weapon Types And Damage', fontsize= 25, pad= 20, weight ='bold',

### color = sns.cubehelix\_palette(8, start=.5, rot=-.75)[-3])

### ax1.set(xlabel='Types Of Weapons', ylabel='Number Of Dead/Injured People')

### ax1.set\_xticklabels(weap\_labels, rotation=45)

### plt.yticks(fontsize=10)

### ax2.legend(loc='upper center')

### fig.show()

### Chart, line chart, histogram Description automatically generated

#### Explosives and Firearms are the two most popular choice of terrorist attacks. However, the damage they cause per case are less than Vehicle, Chemical, and Biological weapons.

## Part 3: Data analysis of Global Terrorism from 2000 to 2017

##### *If we want to avoid Terrorist Attacks and look for the best place to live, we should focus on the situation of each country in the last 2 decades*

### Top 10 countries with the highest Number of Dead/Injured people in the world from 2000 to 2017

### #Choose comlumn Country and Damage from data

### country\_damage = data[data['Year'] > 1999][["Country", "Damage"]].groupby('Country',as\_index=False).sum()

### data\_paint = country\_damage.sort\_values(by='Damage', ascending = False).head(10)

### #Paint the bar chart

### fig, ax = plt.subplots(figsize=(12, 6))

### ax.barh(data\_paint["Country"][::-1], data\_paint["Damage"][::-1], color = 'red')

### plt.xticks(rotation=-45)

### ax.set\_ylabel('Countries', size=16)

### ax.set\_xlabel('Number Of Dead/Injured People', size=16)

### plt.title("Top 10 Attacked Countries From 2000 To 2017", fontsize= 20, pad= 10, weight ='bold',

### color = sns.cubehelix\_palette(8, start=.5, rot=-.75)[-3])

### plt.show()

Chart

Description automatically generated

#### **Top 10 targeted nationalities with the highest number of Dead/Injured People from 2000 to 2017**

#Choose comlumn Country and Damage from data

nalty\_damage = data[["Natlty1", "Damage"]].groupby('Natlty1', as\_index=False).sum()

data\_paint\_natlty = nalty\_damage.sort\_values(by='Damage', ascending = False).head(10)

fig, ax = plt.subplots(figsize=(12, 6))

#Paint bar chart

ax.barh(data\_paint\_natlty["Natlty1"][::-1], data\_paint\_natlty["Damage"][::-1], color = 'blue')

plt.xticks(rotation=-45)

ax.set\_ylabel('Nationality', size=16)

ax.set\_xlabel('Number Of Dead/Injured People', size=16)

plt.title("Top 10 Targeted Nationalitieslty From 2000 To 2017", fontsize= 20, pad= 40, weight ='bold',

color = sns.cubehelix\_palette(8, start=.5, rot=-.75)[-3])

plt.show()

**Chart

Description automatically generated**

##### *Before taking a look at the top 10 safest countries, we should take a look at top 10 most targerted countries and nationalities. Top 4 most targeted countries and nationalities are still Iraq, Pakistan, Afganistan and India. The situation have not been improved in the last 2 decadeds for these countries.*

### Identify the safest countries from 2000 to 2017 which has 0 dead/injured people caused by Terrorist Attacks

country\_damage[country\_damage["Damage"]==0]

Graphical user interface

Description automatically generated with low confidence

**Classification using different Machine Learning Algorithms**

1. **Using Decision Tree Approach**

**Import the decision tree classifier and fit the training data and test data to the classifier**

**# import the regressor**

from sklearn.tree import DecisionTreeClassifier

**# create a regressor object**

decision\_Tree\_Classifier = DecisionTreeClassifier (random\_state = 0)

**# import necessary libraries for building model**

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

from sklearn import tree , metrics, preprocessing

from sklearn.metrics import classification\_report, confusion\_matrix

from sklearn.preprocessing import MinMaxScaler

**# train test split**

X\_train\_DT, X\_test\_DT, y\_train\_DT, y\_test\_DT = train\_test\_split( X, Y, test\_size = 0.3, random\_state = 100)

# fit the train and test data to the Decision Tree Classifier

decision\_Tree\_Classifier.fit(X\_train\_DT, y\_train\_DT)

Confusion Matrix using Decision Tree Algorithm

print(confusion\_matrix(y\_test\_DT, y\_pred\_DT))

print(classification\_report(y\_test\_DT, y\_pred\_DT))

Table

Description automatically generated

Using the Decision Tree algorithm, we got the accuracy as 90%. This means that the algorithm is very good to analyze and predict the data.

Text

Description automatically generated

1. **Using Support Vector Machine**

print(confusion\_matrix(y\_test\_SVC, y\_pred\_SVC))

print(classification\_report(y\_test\_SVC, y\_pred\_SVC))

print("-----------------------------------------------------------------------------------------------------------------------")

print("-----------------------------------------------------------------------------------------------------------------------")

accuracy\_SVC=metrics.accuracy\_score( final\_model\_predictions\_SVC.Actual, final\_model\_predictions\_SVC.predictions )\*100

accuracy\_SVC='{:.2f}'.format(accuracy\_SVC)

print( 'Total Accuracy : ',accuracy\_SVC)

recall\_SVC=metrics.recall\_score(final\_model\_predictions\_SVC.Actual, final\_model\_predictions\_SVC.predictions,average='micro' )

print('recall',recall\_SVC)

Precision\_SVC=metrics.precision\_score(final\_model\_predictions\_SVC.Actual, final\_model\_predictions\_SVC.predictions,average='micro' )

print('Precision',Precision\_SVC)

Table

Description automatically generated

Using Support Vector Machine Algorithm, we got accuracy as 85%.

**Comparing Decision Tree and Support Vector Machine**

**Table

Description automatically generated**

1. **Using Deep Learning**

**#import the necessary packages**

from keras.models import Sequential

from keras.layers import Dense, Activation, Flatten

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

#split the dataset into train and test data

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.1, random\_state=0)

#calculate the Precision and accuracy of data

from sklearn.metrics import classification\_report

predictions = model.predict(X\_test, verbose=0)

pred = np.argmax(predictions, axis=1)

y\_train = np.argmax(Y\_test, axis=1)

print(classification\_report(y\_train, pred))

Table

Description automatically generated

By using Neural Networks, we got the accuracy as 97%, which is excellent.