ARMED CONFLICT DATA ANALYSIS

Spring 22 Big Data CS GY 6513

TANDON SCHOOL OF ENGINEERING

NYU

Team (Shadow Recruits)

Team Members:

- Sri Sai Abhishake Gopal Dasari – sd4648
- Venkata Vyshnavi Ravella - vr2226
- Sri Ram Gowd Vuppala – sv2333

Contents

1. Introduction.

The Armed Conflict Location & Event Data Project (ACLED) is a disaggregated data collection, analysis, and crisis mapping project. ACLED collects the dates, actors, locations, fatalities, and types of all reported political violence and protest events around the world. The ACLED team conducts analysis to describe, explore, and test conflict scenarios, and makes both data and analysis open for free use by the public.

Overall, ACLED sources material in three ways: (1) information from local, regional, national, and continental media is reviewed daily; (2) NGO reports are used to supplement media reporting in hard to access cases; (3) regionally focused news reports and analyses are integrated to supplement daily media reporting. The result is the most comprehensive and wide-reaching source material presently used in disaggregated conflict event coding.

Why is this a big data problem?

ACLED dataset consists of over 1.3 Million records and each record contains around 31 categories of information, such as date and time of conflict, actors involved in the conflict, mode of conflict, region of the event, etc. Since our objective is to retrieve meaningful information from the dataset, analysis and other quantitative operations must be performed on them. A single machine takes a very long time to compute 30 million inputs. And moreover, some operations could be too heavy to be performed using a single machine. So there are two ways to handle this problem, namely vertical and horizontal scaling.

Horizontal scaling is the better approach out of the two, due to the reason being the price to performance ratio. We can spend on getting many commodity machines rather than getting a single powerful machine and performing computations. And, since the performance is now distributed among many machines, we need to also distribute data and make systems coordinate together to return an accurate output. Hence, we need big data in this scenario. Big Data concepts like MapReduce can be implemented and chunks of data can be run simultaneously in real-time using clusters of machines.

Apache Spark would also level up this task as it is very powerful and can perform the same task, multiple folds faster.

The Dataset:

The dataset contains over 1.3 Million records spanning upto 30 columns related to armed conflict events and information like actors involved, date recorded, coordinates, etc.

EVENT_DATE: Recorded as Day / Month / Year.
YEAR: The year in which an event took place.
EVENT_TYPE: The type of conflict event (one of nine possible).
ACTOR1: A named actor involved in the event.
INTER1: A numeric code indicating the type of ACTOR1.

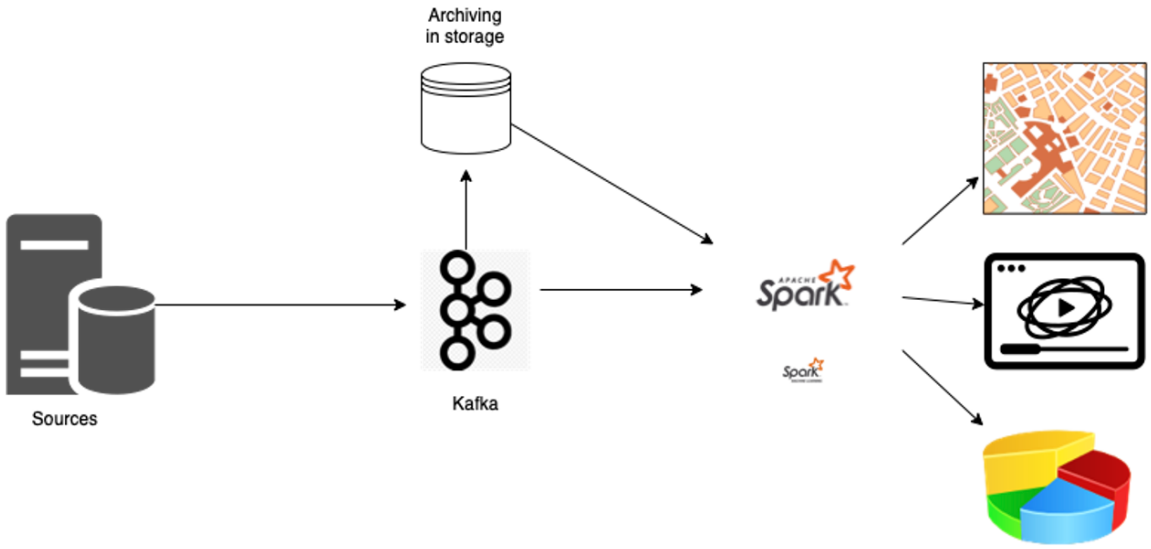
ACTOR2: The named actor involved in the event. If a dyadic event, there will also be an "Actor 1".
ASSOC_ACTOR_2: The named actor associated with or identifying with ACTOR2 in one specific event.
INTER2: A numeric code indicating the type of ACTOR2.
INTERACTION: A numeric code indicating the interaction between types of ACTOR1 and ACTOR2. Coded as an interaction between actor types and recorded as lowest joint number.
REGION: The region of the world where the event took place.
COUNTRY: The country in which the event took place.
ADMIN1: The largest sub-national administrative region in which the event took place.
LOCATION: The location in which the event took place.
LATITUDE: The latitude of the location.
LONGITUDE: The longitude of the location.
GEO_PRECISION: A numeric code indicating the level of certainty of the location coded for the event.
SOURCE: The source of the event report.
SOURCE SCALE: A note on the geographic scale of the main source (local, regional, national or international).
NOTES: A short description of the event.
FATALITIES: Number or estimate of fatalities due to event. These are frequently different across reports.

Here are a few sample images on how the dataset looks like.

ISO	EVENT_ID_CNTY	EVENT_ID_NO_CNTY	EVENT_DATE	YEAR	TIME_PRECISION	EVENT_TYPE	SUB_EVENT_TYPE	ACTOR1	ASSOC_ACTOR_1	INTER1	ACTOR2
48	BHR326	326	01-January-2016	2016	1	Riots	Violent demonstra	Rioters (Bahrain)		5	Police Forces of Bal
48	BHR324	324	01-January-2016	2016	1	Protests	Peaceful protest	Protesters (Bahrain		6	
48	BHR1978	1978	01-January-2016	2016	1	Riots	Violent demonstra	Rioters (Bahrain)	February 14 Youth	5	
48	BHR1981	1981	01-January-2016	2016	1	Riots	Violent demonstra	Rioters (Bahrain)	February 14 Youth	5	
48	BHR1980	1980	01-January-2016	2016	1	Protests	Peaceful protest	Protesters (Bahrain		6	
48	BHR323	323	01-January-2016	2016	1	Protests	Peaceful protest	Protesters (Bahrain		6	
48	BHR325	325	01-January-2016	2016	1	Protests	Peaceful protest	Protesters (Bahrain		6	
48	BHR328	328	01-January-2016	2016	1	Protests	Excessive force agai	Protesters (Bahrain		6	Police Forces of Bal
48	BHR327	327	01-January-2016	2016	1	Protests	Excessive force agai	Protesters (Bahrain		6	Police Forces of Bal
48	BHR1979	1979	01-January-2016	2016	1	Protests	Peaceful protest	Protesters (Bahrain	February 14 Youth	6	
48	BHR1982	1982	01-January-2016	2016	1	Protests	Peaceful protest	Protesters (Bahrain	February 14 Youth	6	
48	BHR1977	1977	01-January-2016	2016	1	Riots	Violent demonstra	Rioters (Bahrain)	February 14 Youth	5	

REGION	COUNTRY	ADMIN1	ADMIN2	ADMIN3	LOCATION	LATITUDE	LONGITUDE	GEO_PRECISION	SOURCE
Middle East	Bahrain	Capital			Sitrah	26.155	50.621	1	Press TV
Middle East	Bahrain	Capital			Al Maamir	26.133	50.609	1	Press TV
Middle East	Bahrain	Capital			Jid Ali	26.179	50.563	1	14 February Revolu
Middle East	Bahrain	Northern			Karzakkan	26.116	50.482	1	14 February Revolu
Middle East	Bahrain	Muharraq			Samahij	26.28	50.635	1	Revolution Bahrair
Middle East	Bahrain	Northern			Al Diraz	26.218	50.471	1	Press TV
Middle East	Bahrain	Capital			Nabih Saleh	26.183	50.585	1	Press TV
Middle East	Bahrain	Northern			Ash Shakhurah	26.215	50.507	1	Middle East Eye
Middle East	Bahrain	Northern			Abu Saybi	26.218	50.507	1	Middle East Eye
Middle East	Bahrain	Capital			Karbabad	26.23	50.529	1	14 February Revolu
Middle East	Bahrain	Northern			Shahrakkan	26.075	50.501	1	14 February Revolu
Middle East	Bahrain	Capital			Bilad al Qadim	26.212	50.562	1	14 February Revolu
Middle East	Bahrain	Northern			Al Malikiyah	26.098	50.487	1	AFP
Middle East	Bahrain	Northern			Abu Saybi	26.218	50.507	1	AFP
Middle East	Bahrain	Capital			Sitrah	26.155	50.621	1	Press TV

Architecture:



The components used in the above architecture and their use is described below:

Kafka

Apache Kafka is commonly used to create real-time streaming data pipelines and adaptable applications. It mixes communications, storage, and stream processing to enable for both historical and real-time data storage and analysis.

ACLED website which updates data very frequently in batches. We propose the use of Kafka for when we extend our data sources outside of ACLED like twitter and news websites. The current application doesn't need Kafka as it gets updated in batches.

Spark

Spark is at the center of this architecture's processing. Spark's in-memory processing allows for quick computation of big datasets. We can expect thousands or millions of reports per day in the scalable environment, and Spark can handle all processes from pre-processing through publishing this data. For reporting and monitoring, data that has been filtered using Spark processing is used.

Spark.ML

spark.ml is a new package introduced in Spark 1.2, which aims to provide a uniform set of high-level APIs that help users create and tune practical machine learning pipelines.

Many processes in machine learning are computationally heavy. Distributing these processes via Apache Spark is the easiest, fastest and most efficient way. In industrial applications there is a need for an engine which is powerful enough to process data in real time and can perform in batch mode, as well as an engine that can perform in-memory processing. Apache Spark provides real-time streaming, interactive processing, graph processing, in-memory processing and batch processing with a very fast and simple interface.

Data Visualization

Data monitoring and visualization is very important in our project. Tools like Plotly, Folium can be used to show relevant information to the end-user. This visualization is useful when people want to make decision about strategic deployment of resources or aid. They can monitor and view the data which will give them meaningful information.

2 Analysis:

Preprocessing:

Data preprocessing involves transforming raw data to well-formed data sets so that data mining analytics can be applied. Preprocessing is done here in order to remove the unwanted columns in the data. It's a crucial process that can affect the success of data mining and machine learning projects. It makes knowledge discovery from datasets faster and can ultimately affect the performance of machine learning models.

Further, some of the data in the columns needed to be standardized and indexed, so that the computation process was made smoother. Some of the required columns also had unnecessary data present and some of the columns are unnecessary for analysis we can remove these. These entire steps account for data cleaning.

Exploratory Data Analysis:

Exploratory Data Analysis (EDA) is the crucial process of using summary statistics and graphical representations to perform preliminary investigations on data in order to uncover patterns, detect anomalies and verify assumptions.

EDA is a data exploration technique to understand the various aspects of the data. EDA is often used to see what data may disclose outside of formal modelling and to learn more about the variables in a data collection and how they interact. It could also help us figure out if the statistical procedures we are considering for data analysis are appropriate. Before modelling the data, it gives insight into all of the data and the numerous interactions between the data elements.

```
df.select('fatalities').describe().show()
cnts = df.groupBy("year").count()

mode = cnts.join(
    cnts.agg(max("count").alias("max_")), col("count") == col("max_")
).limit(1).select("year")
mode.show()
cnts.show()
country = df.groupBy("country").count().sort(desc("count"))
country.show()
```

EDA Samples:
Summary on fatalities column of the data set

summary	fatalities
count	1315502
mean	0.8352104367762269
stddev	4.822280616611522
min	0
max	750

Number of events per year, it can be noticed that there has been great change in first half of the last decade and after 2016. This could be due to the increased availability to the information with the advent of social media and technology.

year	count
2018	198200
2022	84473
2019	215689
2020	249360
2021	266737
2015	34410
2016	72602
2017	111328
2013	24750
2014	22068
2012	20280
2011	15605

Here is a cross table of region and event type which shows that the Caucasus have the large number of battles.

region_event_type	Battles	Explosions/Remote violence	Protests	Riots	Strategic developments	Violence against civilians
East Asia	7	10	40461	834	1730	1724
Northern Africa	8538	6972	28439	5220	2470	6981
Oceania	29	0	1482	119	96	30
North America	5964	63	63686	4592	3665	24956
South Asia	11456	6654	155678	30193	3263	11235
Western Africa	9331	3096	11054	5900	3236	12325
Southeast Asia	12643	8014	22426	1928	7924	13820
Central America	3178	32	6918	1019	954	5244
Caucasus and Cent...	60217	17820	10674	607	2106	3344
Southern Africa	183	26	9165	7263	338	2226
Europe	23220	25402	105469	4702	3809	1598
Caribbean	1063	3	2732	1336	602	3004
South America	19488	1108	56353	9848	4167	16794
Eastern Africa	22958	7485	8409	5819	4562	18461
Middle East	55327	126481	62229	15400	22041	16068
Antarctica	0	0	6	0	0	0

We can perform the same kind of analysis on each country, here is an example result on an analysis in Yemen.

event_type	count
Explosions/Remote...	41965
Violence against ...	2900
Strategic develop...	3798
Protests	2642
Riots	367
Battles	20689

summary	fatalities
count	72361
mean	2.1543372811321015
stddev	5.072934557579858
min	0
max	150

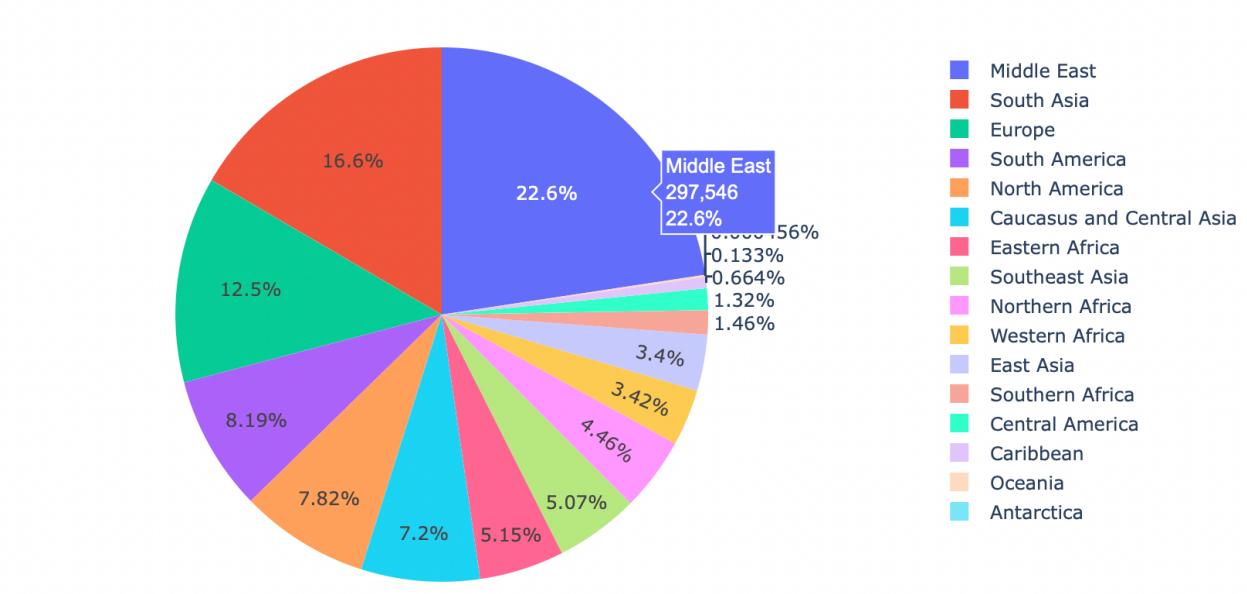
year	count
2018	11484
2022	2640
2019	11875
2020	11914
2021	8788
2015	8058
2016	9079
2017	8523

Summary of fatalities of Yemen and events count by year

Visualizations

Data visualization can help by delivering data in the most efficient way possible. As one of the essential steps in the business intelligence process, data visualization takes the raw data, models it, and delivers the data so that conclusions can be reached. In advanced analytics, data scientists are creating machine learning algorithms to better compile essential data into visualizations that are easier to understand and interpret.

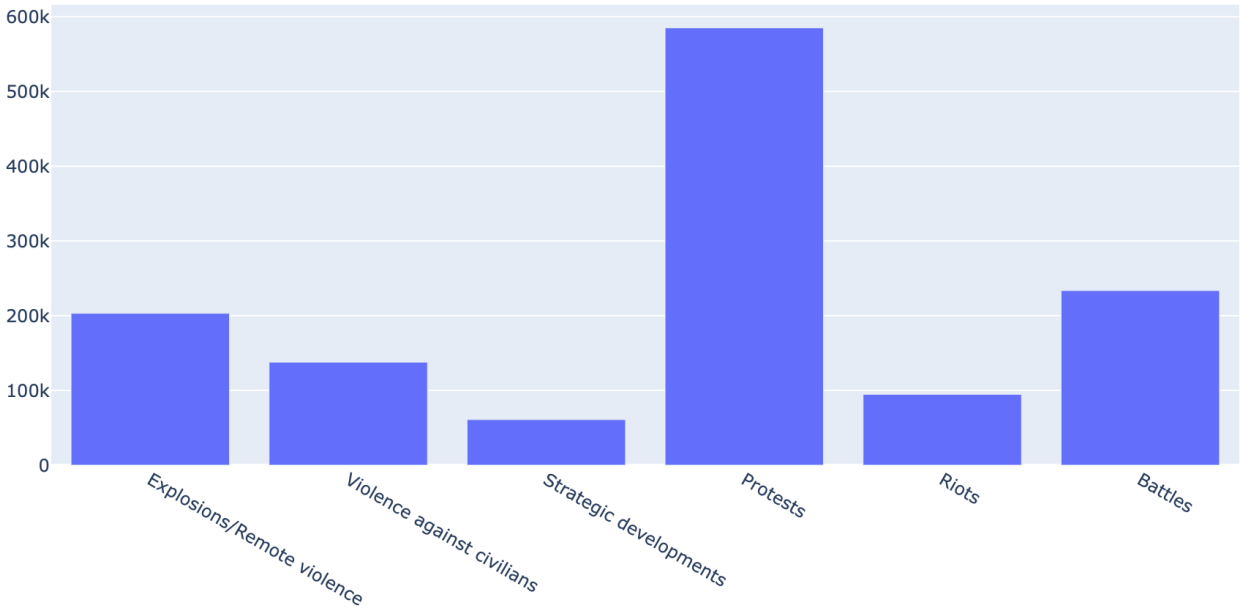
In our first visualization we are trying to display the number of events by regions across the globe, we can see that there are number of conflicts is very high in the Middle east and then in South Asia given that the 1/3 of the global population lives in the region.



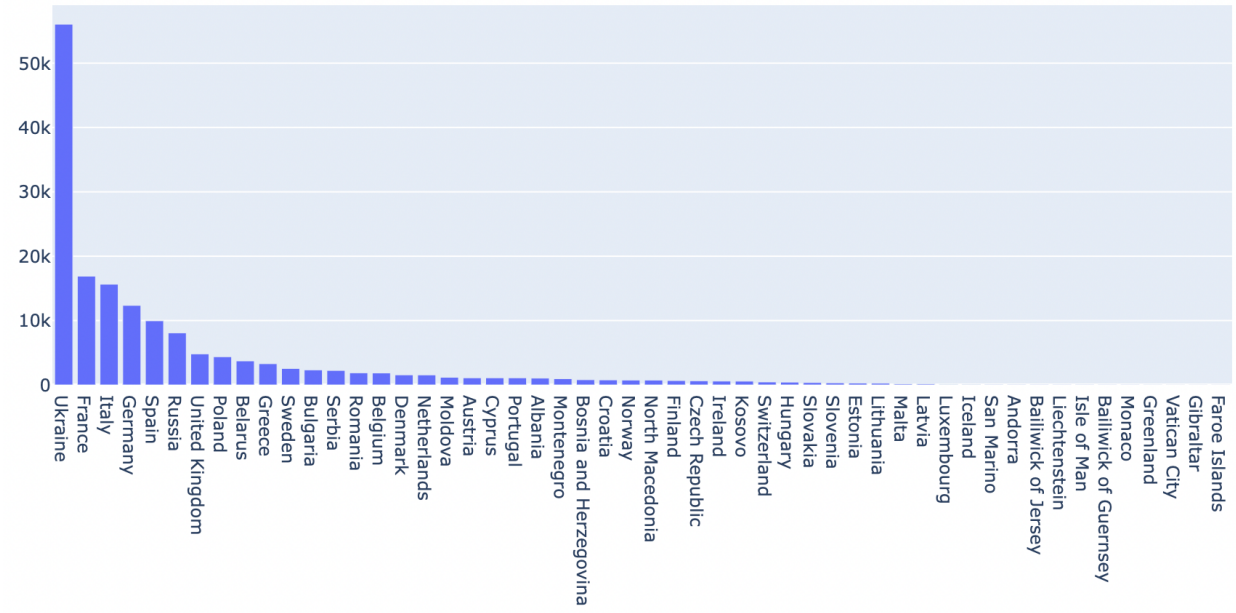
We can plot different types using the plotly graphical objects.

```
df_event_type = df.groupBy("event_type").count()
eventType = df_event_type.select(collect_list('event_type')).first()[0]
eventTypeCount = df_event_type.select(collect_list('count')).first()[0]
fig1 = go.Figure(
    data=[go.Bar(x = eventType , y = eventTypeCount)],
    layout_title_text="Event Type vs Count of  Armed Conflict Events"
)
fig1.show()
```

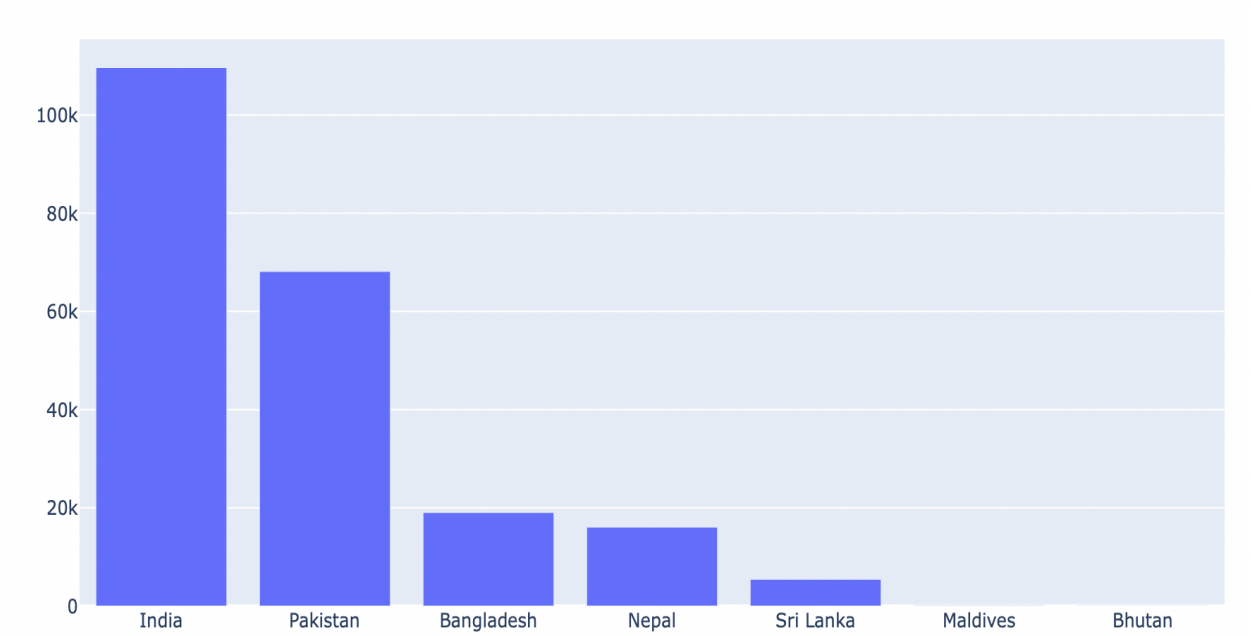
We can notice that there are very high number of protests that could have been caused by pandemic, inflation, raise in authoritarian powers in recent years.



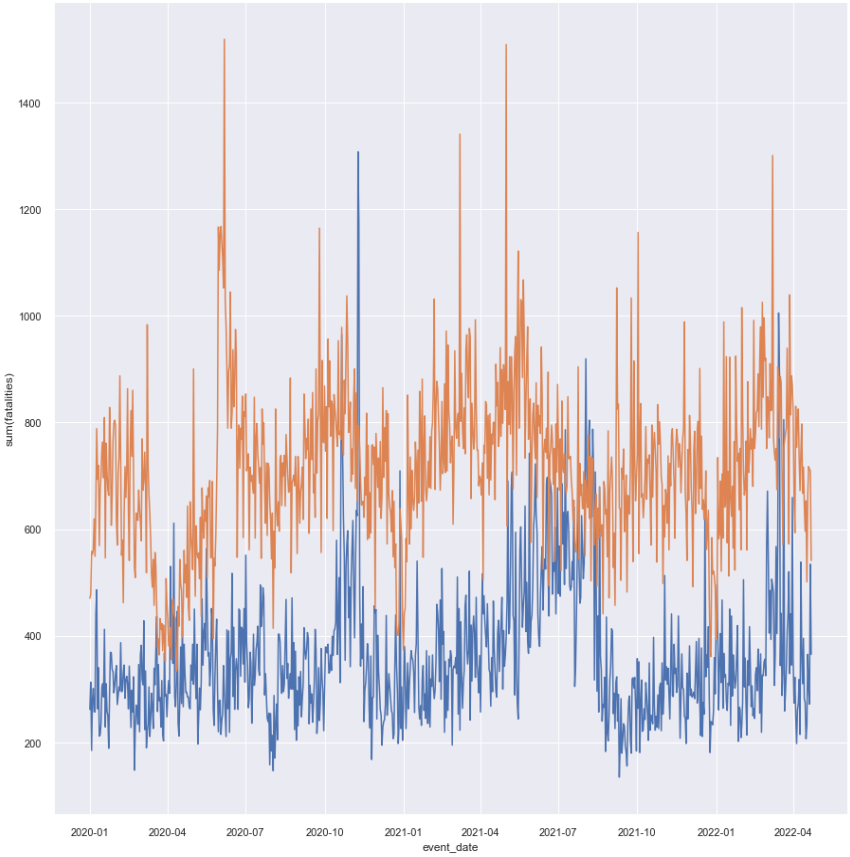
While there are relatively a smaller number of armed conflicts in developed economies of Europe this year has been abnormal and caused Ukraine to be an outlier.



In south Asia India has highest number of conflict events, its population and the transparency of the democracy are some of the key factors that one should consider.



Lets plot the number of events and the number of fatalities across the globe from the year 2020 – 2022



We have chosen to do an in-depth analysis on two countries: Ukraine and Afghanistan.
For two reasons,

USECASE 1 - Ukraine is facing a lot of military conflict.

Armed conflict in eastern Ukraine erupted in early 2014 following Russia's annexation of Crimea. The previous year, protests in Ukraine's capital Kyiv against Ukrainian President Viktor Yanukovych's decision to reject a deal for greater economic integration with the European Union (EU) were met with a violent crackdown by state security forces. The protests widened, escalating the conflict, in 2022 Putin announced the beginning of a full-scale land, sea, and air invasion of Ukraine targeting Ukrainian military assets and cities across the country.

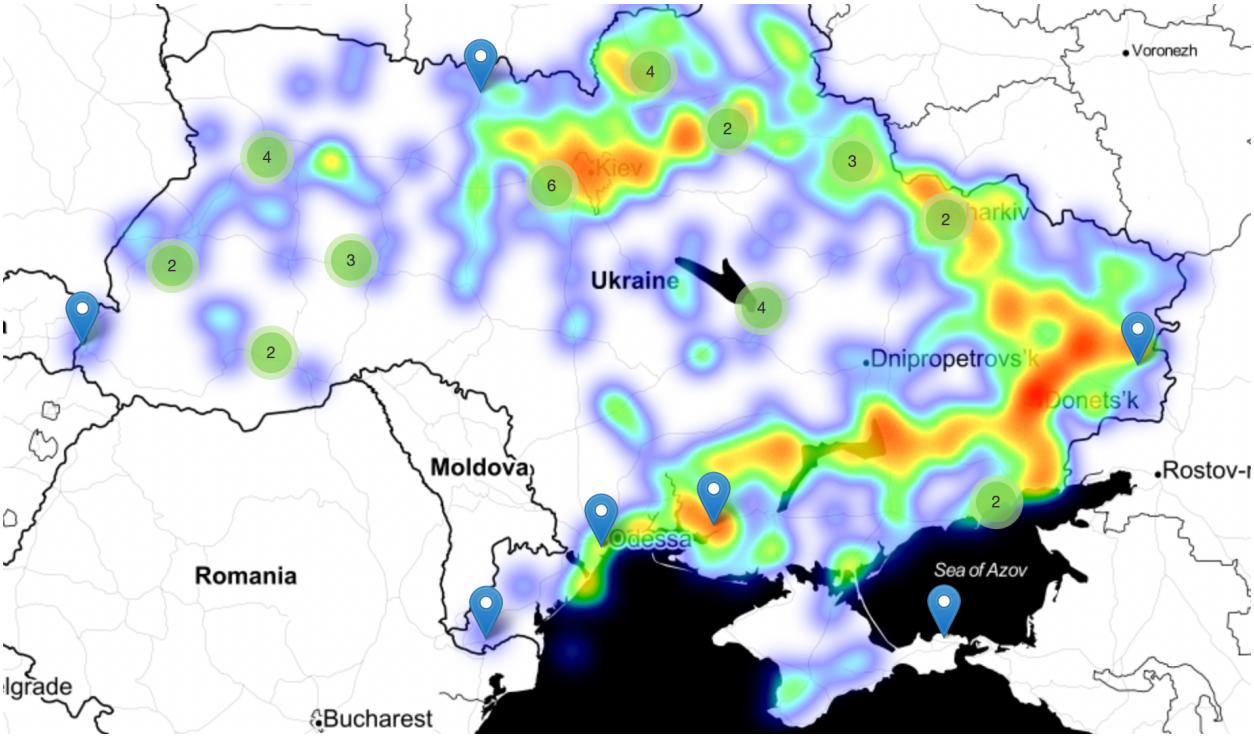
USECASE 2 - Afghanistan is an active conflict region.

In April 2021, President Joe Biden announced that U.S. military forces would leave Afghanistan by September 2021. The Taliban, which had continued to capture and contest territory across the country despite ongoing peace talks with the Afghan government, ramped up attacks on Afghan National Defense and Security Forces (ANDSF) bases and outposts and began to rapidly seize more territory. In May 2021, the U.S. military accelerated the pace of its troop withdrawal. By the end of July 2021, the United States had completed nearly 95 percent of its withdrawal, leaving just 650 troops to protect the U.S. embassy in Kabul.

Use case 1 – Ukraine

The Idea here is to show the density of armed conflicts near each city and airbases in Ukraine. The heat maps here are generated using folium. We initially plotted the conflicts recorded based on their geo coordinate details. And then we used airbases location to point them on the map using the various sources we found on internet.

The heap map is for the concentration of conflicts, blue pointer is for an individual air base and the green circle represents a cluster of airbases (the number indicates the actual number of bases in the cluster)

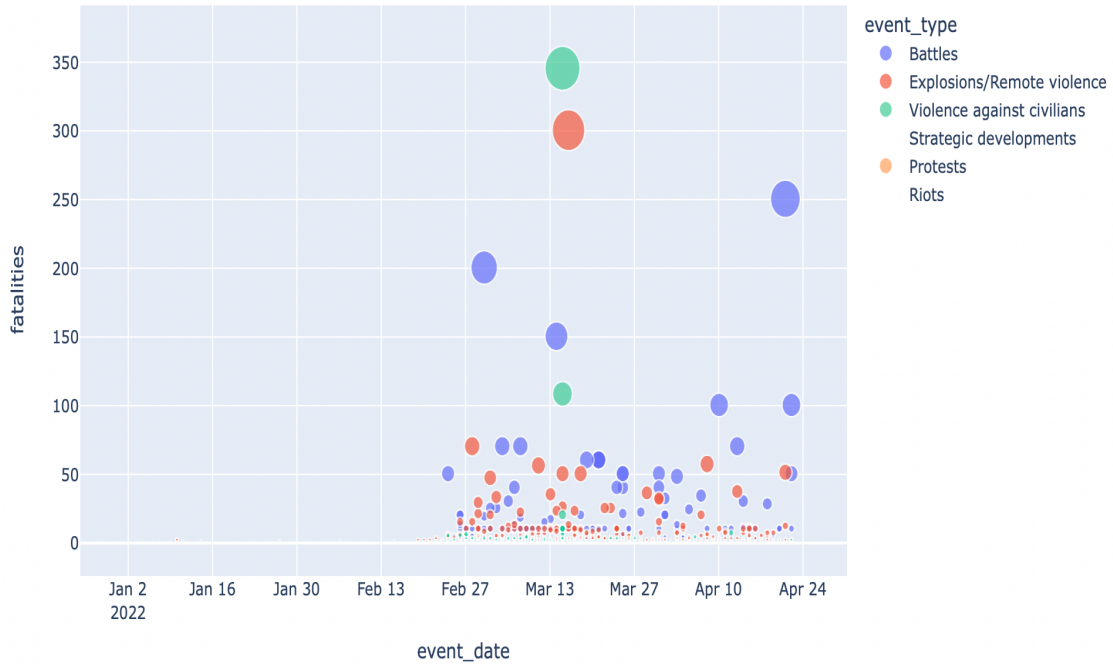


```
ukraine=df[df["country"]=='Ukraine']
ukraine=ukraine[ukraine["year"]==2022]
events=np.array(ukraine.select('latitude','longitude').collect())
airbases=np.array(airbase.select('latitude','longitude').collect())
trailmap= folium.Map([50.4501, 30.5234], zoom_start=11,tiles ='Stamen Toner')

plugins.MarkerCluster(airbases).add_to(trailmap)
trailmap.add_children(plugins.HeatMap(events, radius=15))
```

Code for the above heat map

The above heatmap shows that major amount of the conflict is concentrated in the eastern front of Donbas where there are more pro-Russian people and the Russian army. We can also see the scale of incidents surrounding the major cities of Kiev, Kharkiv. Russia wants to cut Ukrainian access to ocean we can see the concentration of forces by the southern (down) part of the country. We can also notice the incidents around the airbases in Ukraine which were an attempt by Russian air force to cripple Ukrainian Air power

Here is another visualization to understand the scale and reasons of fatalities that have happened since the war started in Feb 2022



Preparing for Machine Learning Tasks:

StringIndexer:

StringIndexer encodes a string column of labels to a column of label indices. The indices are in [0, numLabels), and four ordering options are supported: "frequencyDesc": descending order by label frequency (most frequent label assigned 0), "frequencyAsc": ascending order by label frequency (least frequent label assigned 0), "alphabetDesc": descending alphabetical order, and "alphabetAsc": ascending alphabetical order (default = "frequencyDesc"). The unseen labels will be put at index numLabels if user chooses to keep them. If the input column is numeric, we cast it to string and index the string values. When downstream pipeline components such as Estimator or Transformer make use of this string-indexed label, you must set the input column of the component to this string-indexed column name.

StandardScaler

transforms a dataset of Vector rows, normalizing each feature to have unit standard deviation and/or zero mean.

It takes parameters:
withStd: True by default. Scales the data to unit standard deviation.
withMean: False by default. Centers the data with mean before scaling. It will build a dense output, so take care when applying to sparse input.
StandardScaler is an Estimator which can be fit on a dataset to produce a

StandardScalerModel; this amounts to computing summary statistics. The model can then transform a Vector column in a dataset to have unit standard deviation and/or zero mean features.

```
ukraine=df[df['country']=='Ukraine']
indexer= StringIndexer(inputCol='actor1',outputCol='actor1_idx')
indexer=indexer.fit(ukraine)
ukraine=indexer.transform(ukraine)
ukraine=StringIndexer(inputCol='event_type',outputCol='event_type_idx').fit(ukraine).transform(ukraine)
ukraine=StringIndexer(inputCol='source_scale',outputCol='source_scale_type_idx').fit(ukraine).transform(ukraine)
```

VectorAssembler

VectorAssembler is a transformer that combines a given list of columns into a single vector column.
It is useful for combining raw features and features generated by different feature transformers into a single feature vector, in order to train ML models like logistic regression and decision trees.
VectorAssembler accepts the following input column types: all numeric types, Boolean type, and vector type. In each row, the values of the input columns will be concatenated into a vector in the specified order.

```
assembler=VectorAssembler(inputCols=['actor1_idx','event_type_idx','source_scale_type_idx','interaction',
ukraine=assembler.transform(ukraine)


scale=StandardScaler(inputCol='features',outputCol='standardized')
data_scale=scale.fit(ukraine)
ukraine=data_scale.transform(ukraine)
```

Clustering

Clustering is an unsupervised learning technique, in short, you are working on data, without having any information about a target attribute or a dependent variable. The general idea of clustering is to find some intrinsic structure in the data, often referred to as groups of similar objects. The algorithm studies the data to identify these patterns or groups such that each member in a group is closer to another member in the group (lower intracluster distance) and farther from another member in a different group (higher inter-cluster distance).

PySpark uses the concept of Data Parallelism or Result Parallelism when performing the K Means clustering.

K-means is one of the most commonly used clustering algorithms for grouping data into a predefined number of clusters. The KMeans function from pyspark.ml.clustering includes the following parameters:

- k is the number of clusters specified by the user
- maxIterations is the maximum number of iterations before the clustering algorithm stops. Note that if the intracluster distance doesn't change beyond the epsilon value mentioned, the iteration will stop irrespective of max iterations

```
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import ClusteringEvaluator
```

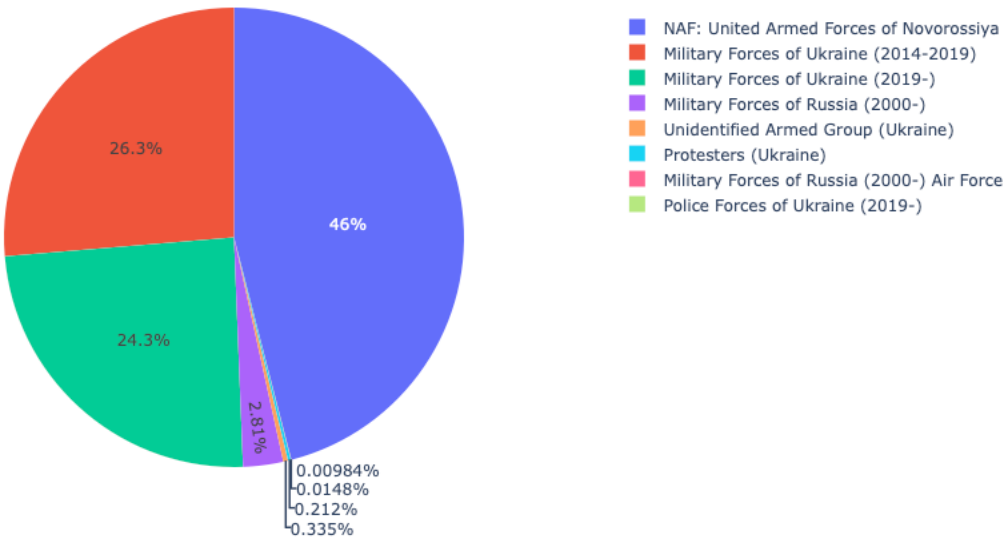
```
silhouette_score=[]
evaluator = ClusteringEvaluator(predictionCol='prediction', featuresCol='standardized', \
                                metricName='silhouette', distanceMeasure='squaredEuclidean')

for i in range(2,6):

    KMeans_algo=KMeans(featuresCol='standardized', k=i)
    KMeans_fit=KMeans_algo.fit(ukraine)
    output=KMeans_fit.transform(ukraine)
    score=evaluator.evaluate(output)
    silhouette_score.append(score)
    print("Silhouette Score:",score)
```

We have built the clustering algorithm using k=5 dividing the data into 5 clusters. Here is how the data divides into the following clusters

prediction	count
1	259
3	5137
4	22959
2	5
0	27694

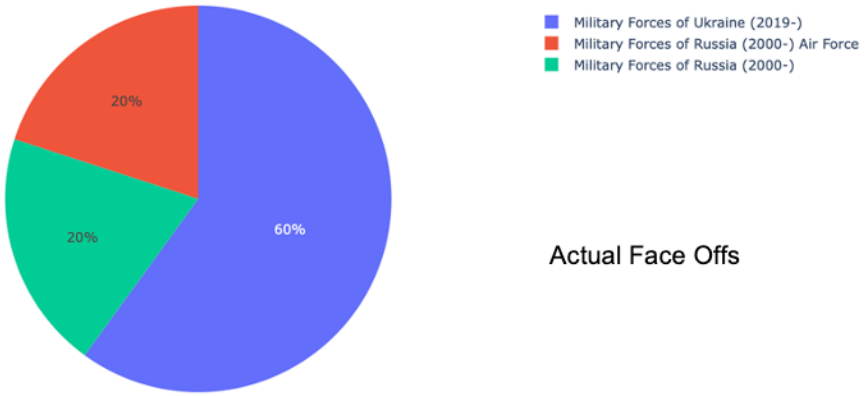
Let's try to understand each cluster:

Cluster 0, This cluster contains mostly battle events, these are mostly against the separatists from the eastern Donbas region. The share of major actors can be seen from the pie chart.



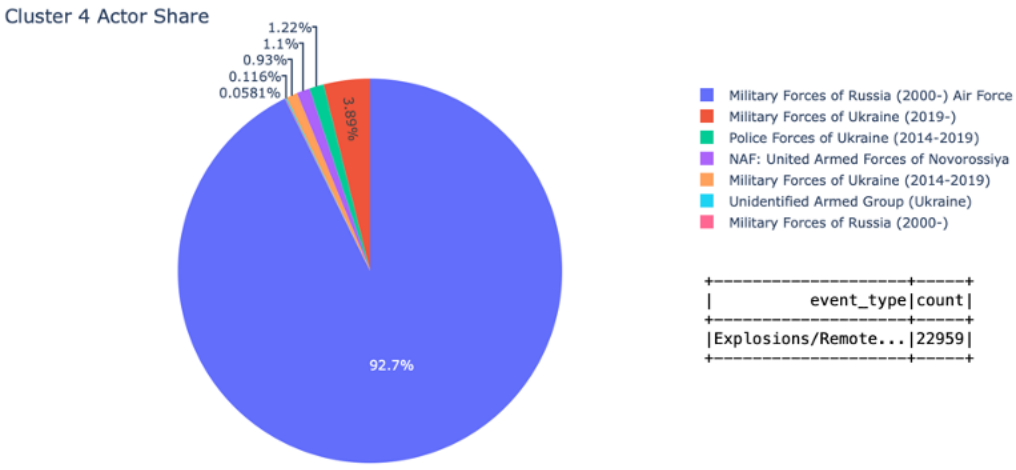
Cluster 1, This cluster contains mostly battle events, these are mostly against the Russian by the Ukrainian forces . The share of major actors can be seen from the pie chart.

event_type	count
Violence against ...	6
Strategic develop...	313
Protests	43
Battles	19966



Actual Face Offs

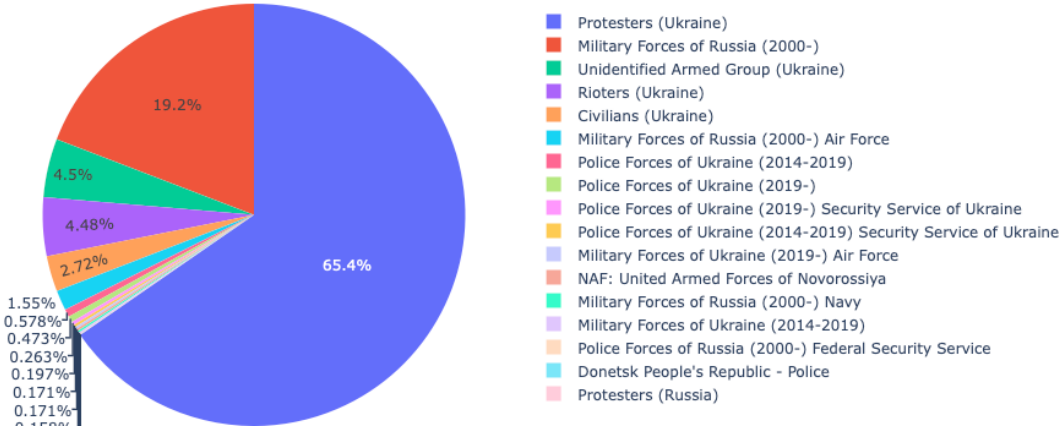
Cluster 2, This cluster contains only explosion events, these are mostly done by the Russian against the Ukrainian forces and citizens. The share of major actors can be seen from the pie chart.



event_type	count
Explosions/Remote...	22959

Cluster 3, This cluster contains pro and anti-Russian Protests, Could be done by people, Militias and consists mostly protests and riots involving violence.

actor1	count
Donetsk People's ...	1
Military Forces o...	10
Military Forces o...	13
Police Forces of ...	3
Protesters (Russia)	1
Military Forces o...	118
Civilians (Ukraine)	207
Police Forces of ...	36
Protesters (Ukraine)	4972
Military Forces o...	12
Police Forces of ...	15
Military Forces o...	1458
Police Forces of ...	20
Rioters (Ukraine)	341
Unidentified Arme...	342
Police Forces of ...	44
NAF: United Armed...	13



Cluster 4, This cluster contains Miscellaneous events that occur in a normal country, These events are diverse and cannot be plotted in pie chart.

actor1	count
Military Forces o...	11
Donetsk People's ...	18
Unidentified Arme...	5
Military Forces o...	1
Military Forces o...	2
Sentianivka Commu...	1
Police Forces of ...	3
Unidentified Mili...	6
National Corps Party	4
Government of Ukr...	11
Donetsk People's ...	5
Rioters (Greece)	1
Opposition Platfo...	1
Government of Ukr...	14
Police Forces of ...	4
Fatherland Party	1
Police Forces of ...	14
Bucha Communal Mi...	1
Police Forces of ...	15
Luhansk People's ...	3

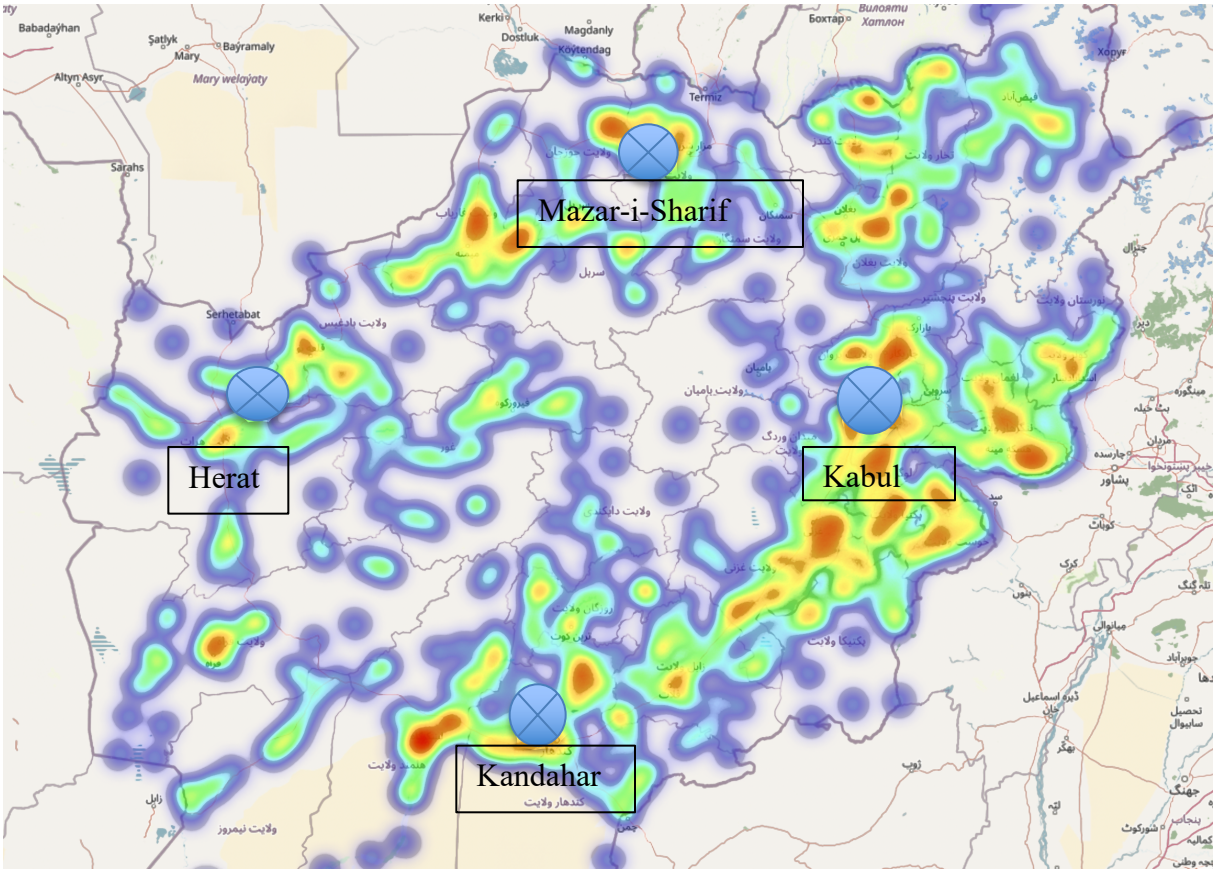
event_type	count
Explosions/Remote...	97
Violence against ...	124
Strategic develop...	259
Protests	498
Riots	77
Battles	65

Use case 2 – Afghanistan

The Ring Road

When US wanted to rebuild Afghanistan, it poured billions of dollars into the task. One of their major priorities was to rebuild the Ring Road that would run a loop around the Afghanistan Connecting the cities of Kabul, Kandahar, Herat, Sheberghan.

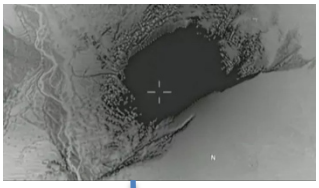
When we look at the conflict activity in the region from 2011 through 2022, we can notice that the Ring Road has been the hot bed of these activities.

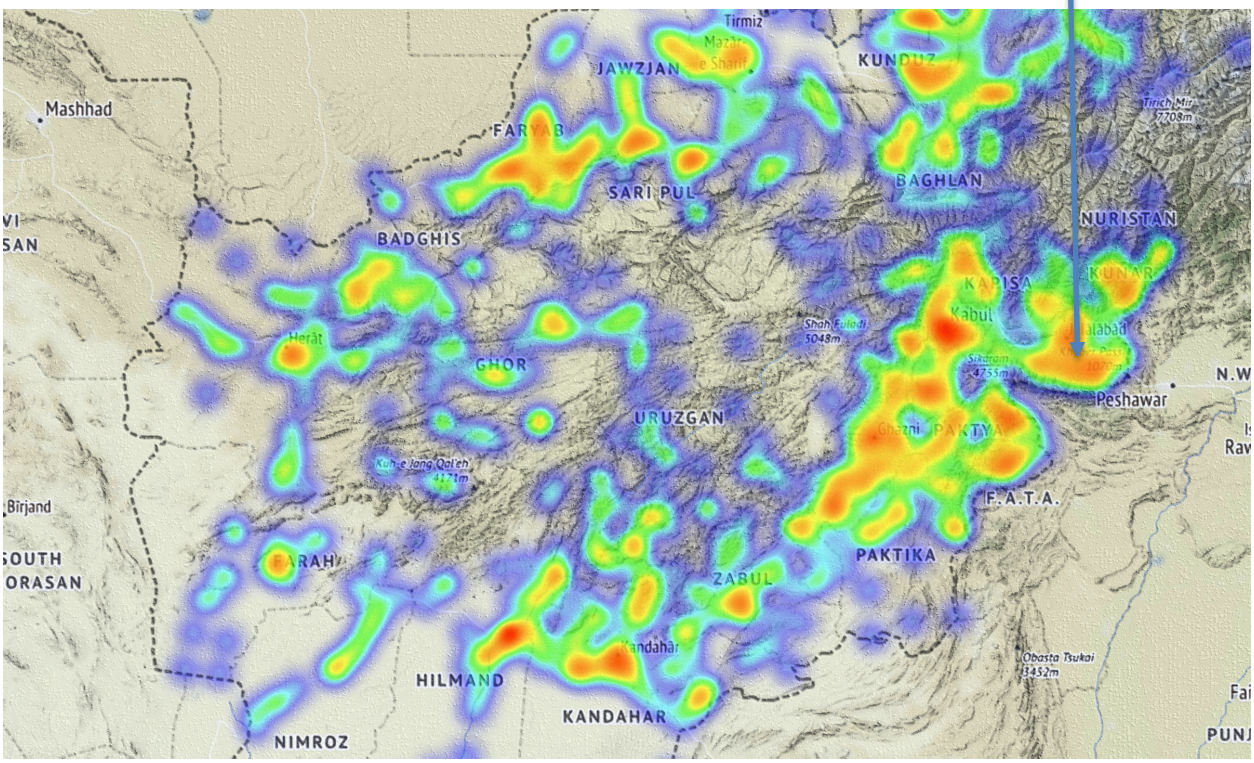


Geography:

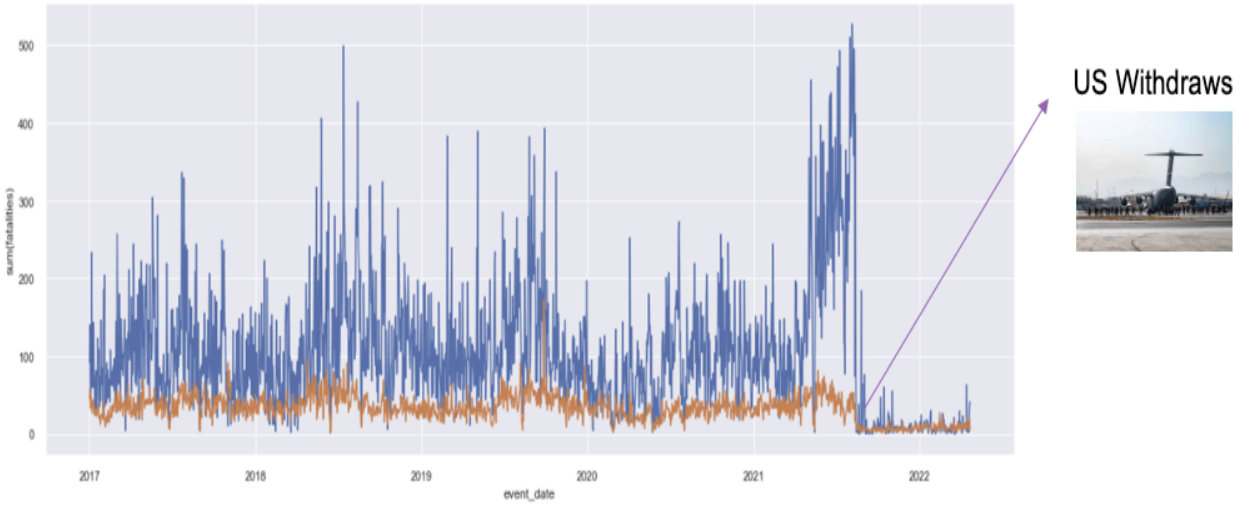
Another correlation that we noticed is that the most of these events are concentrated near the plains and cities, more in the plains bordering the Neighboring country of Pakistan. The mountainous region in the central part of country has seen relatively a smaller number of incidents/ conflicts over the period of observation. However, These mountainous regions can be held as strong holds to recuperate.

In 2017 US drops its most powerful non-nuclear bomb on suspected ISIS Cave Complex





We used to have a approximated number of fatalities in Afghanistan before the withdrawal of US forces, After the withdrawal of those forces we can see a steep decline in the reported fatalities, this doesn't mean that US army presence has been the reason for the higher numbers but there has been a lack of transparency after they left. We can use machine learning models to try and bring out the actual numbers. Regression come to our aid in out attempt.
Linear regression is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable



We trained 3 different regression models using spark.ml in our attempt to find the real figures using the other parameters we have on the events

- Linear Regression
- Decision Tree Regression
- FM Regression

Linear regression:

Linear regression is most widely used predictive modelling. It use the line equation of Y = mX + c.
In this equation Y = mX + c ,
X is the independent variable,
Y is the dependent variable and
m is slope/inclination of straight line which is the graph of this equation.
Linear regression equation is also similar

Y = a + m*X + ε
a is the intercept
m is the slope of the line or coefficient
ε is the error term or residual
In linear regression analysis we need to get the best fit line with minimum error. The main objective of the linear regression analysis is to check whether independent variable explains the dependent variable.

```
from pyspark.ml.regression import LinearRegression
train=afghan[afghan["year"]<=2020]
test=afghan[afghan["year"]>=2021].sample(False, 0.1, seed=0).limit(20)
#train,test = afghan.randomSplit([0.75, 0.25])
lin_reg = LinearRegression(featuresCol = 'standardized', labelCol='fatalities',maxIter=1000,regParam=0.4,
                           loss='squaredError', elasticNetParam=0.9)
linear_model = lin_reg.fit(train)
print("Coefficients: " + str(linear_model.coefficients))
print("\nIntercept: " + str(linear_model.intercept))
```

Decision Tree:

Decision Tree is a decision-making tool that uses a flowchart-like tree structure or is a model of decisions and all of their possible results, including outcomes, input costs.

Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables.
Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values.

```
from pyspark.ml.regression import DecisionTreeRegressor
dt = DecisionTreeRegressor(featuresCol = 'standardized', labelCol = 'fatalities')
dt_model = dt.fit(train)
```

Factorization machine:

Factorization machine (FM) is a predictor model that estimates parameters under the high sparsity. The model combines advantages of SVM and applies a factorized parameters instead of dense parametrization like in SVM . FM is a supervised learning algorithm and can be used in classification, regression, and recommendation system tasks in machine learning. PySpark MLLib API provides a FMRegressor class to implement factorization machines for regression tasks.

```
from pyspark.ml.regression import FMRegressor
fm = FMRegressor(featuresCol="standardized",labelCol = 'fatalities', stepSize=0.001)
fm_model=fm.fit(train)
```

Here are the results on a random sample from all three regression models, While the linear regression and decision tree regression model converged on the dataset, they failed to recognize outliers for example in one instance there are 15 causalities however these models couldn't converge on the number correctly, The factorization machine regression could converge well on the numbers.

.

Linear Regressor

prediction	fatalities
7.665947240819182	4
4.260148489269746	1
8.886408261596689	0
8.743165482488832	1
2.1803750549449408	2
2.0198376678795684	15
3.083931710266037	0
3.515286037908896	2
4.191424498076552	1
3.88288894952386	2
3.257449035514756	3
9.144245263990829	0
8.260968347587585	0
8.886408261596689	0
3.5659845840674484	3
2.4214014918456463	2
3.287867842755456	0
3.7093716242751404	0
2.4214014918456463	1
3.7093716242751404	0

Decision Tree Regressor

prediction	fatalities
7.81058495821727	4
5.2774266365688485	1
7.81058495821727	0
7.81058495821727	1
1.901749663526245	2
3.180058651026393	15
1.5217391304347827	0
5.2774266365688485	2
7.467296844031102	1
5.2774266365688485	2
5.2774266365688485	3
7.81058495821727	0
7.81058495821727	0
7.81058495821727	0
7.467296844031102	3
3.180058651026393	2
5.2774266365688485	0
1.5217391304347827	0
3.180058651026393	1
1.5217391304347827	0

Factor Machines Regressor

prediction	fatalities
3.8611575093695447	4
5.86977098597857	1
4.556868000126847	0
4.339703650008339	1
2.6966521498654443	2
13.271164650495042	15
2.9965668367145257	0
3.836442870022784	2
2.5817334066041475	1
2.679934469628627	2
2.0070744225914714	3
7.107929458241366	0
5.93617631059427	0
4.556868000126847	0
2.583717608926285	3
4.256692507779152	2
1.3587862984561359	0
2.9861118256374914	0
2.5527709096402624	1
4.281803749769108	0

Conclusion

The dataset would continue to increase exponentially with the coming years as, violence keeps occurring. The operations that can be performed on this dataset, become less and less feasible by a single machine. To level things up a notch, a CI/CD (continuous integration/ continuous deployment) method can be implemented to ensure that the ML model is updated with the latest data. This can make the model get trained better and can mitigate future conflicts. For example, the model can predict red zones and other unsafe areas, where patrol and surveillance can be reinforced. In such way, analysis of conflict events can be useful in the long run.

We can expand on the data sources mining data from twitter and by scraping the web we can build a stream of data and use various big data tools to analyze this large amounts of data.

References:

https://acleddata.com/data-export-tool/

https://python-visualization.github.io/folium/

https://spark.apache.org/docs/2.3.1/api/python/pyspark.ml.html

https://www.cfr.org/global-conflict-tracker/conflict/war-afghanistan

https://www.vox.com/2022/2/23/22948534/russia-ukraine-war-putin-explosions-invasion-explained