<u>**Visualizing Conflicts in Indonesia**</u>

Group Members:

Radisa Hussien Rachmadi - 2301891752

Rayhan Ali Darmawan - 2301891683

I. Problem Introduction & Hypothesis

Indonesia has a long history of being affected by many serious conflicts that arose due to factors such as ethnicity, regional tensions, certain radical groups, and even religion. And as an effect from this, Indonesia has witnessed many deaths, both guilty and innocent. From 1989 until 2018 there were a lot of conflicts that were happening in Indonesia, most notably from GAM (Gerakan Aceh Mandiri) and OPM (Organisasi Papua Merdeka)[1]. We want to make a visualization to show where the conflicts happen with the detailed information such as how the conflicts happen and how long the conflicts last, also how many casualties that the conflicts produce. We want to show the areas where conflicts often occur in Indonesia, and who is held responsible or which party is the one who started the conflicts.

If we can manage to give a detailed visualization about Indonesia's past conflicts, it may raise awareness to Indonesia's citizens and citizens outside of Indonesia. And people have more knowledge in areas where conflicts happen more often. With further data analyzing, we hopefully can try to find any specific traits or pattern that each organization does when dealing in violent conflicts. Our hypothesis is that each organization has a specific set of rules, that they stick to whenever they are caught up in violent conflicts. For example each organization will have rules on where they start conflict, when they initiate, and how many casualties there will be.

II. Dataset

For our data, we will be using data provided from the Humanitarian Data Exchange[2]. This dataset covers individual events of organized violence collected from 1989 until 2018. The reason we chose this dataset is that it provides comprehensive details regarding the location, date, status, and also all the parties involved.

```
Data columns (total 45 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   id                  1725 non-null   float64
 1   year                1726 non-null   object 
 2   active_year         1725 non-null   float64
 3   start_year          1726 non-null   object 
 4   end_year            1726 non-null   object 
 5   type_of_violence    1725 non-null   float64
 6   conflict_new_id     1725 non-null   float64
 7   conflict_name       1725 non-null   object 
 8   dyad_new_id         1725 non-null   float64
 9   dyad_name           1725 non-null   object 
 10  side_a_new_id       1725 non-null   float64
 11  gwnoa               1478 non-null   float64
 12  side_a              1726 non-null   object 
 13  side_b_new_id       1725 non-null   float64
 14  gwnob               0 non-null      float64
 15  side_b              1726 non-null   object 
 16  number_of_sources   1725 non-null   float64
 17  source_article      1717 non-null   object 
 18  source_office       517 non-null    object 
 19  source_date         517 non-null    object 
 20  source_headline     518 non-null    object 
 21  source_original     1227 non-null   object 
 22  where_prec          1725 non-null   float64
 23  where_coordinates   1726 non-null   object 
 24  adm_1               1720 non-null   object 
 25  adm_2               1475 non-null   object 
 26  latitude            1726 non-null   object 
 27  longitude           1726 non-null   object 
 28  geom_wkt            1725 non-null   object 
 29  priogrid_gid        1725 non-null   float64
 30  country             1726 non-null   object 
 31  country_id          1725 non-null   float64
 32  iso3                1726 non-null   object 
 33  region              1726 non-null   object 
 34  event_clarity       1725 non-null   float64
 35  date_prec           1725 non-null   float64
 36  date_start          1726 non-null   object 
 37  date_end            1726 non-null   object 
 38  deaths_a            1725 non-null   float64
 39  deaths_b            1725 non-null   float64
 40  deaths_civilians    1725 non-null   float64
 41  deaths_unknown      1725 non-null   float64
 42  low                 1725 non-null   float64
 43  best                1726 non-null   object 
 44  high                1725 non-null   float64
dtypes: float64(21), object(24)
```

III. Data Preparation and Processing

There are 45 columns in this dataset. Most of these columns are irrelevant as they have an enormous amount of missing values that it is not worth to conduct fillna queries onto them. The most important values to note are the following:

- Year
- Type_of_vioelnce
- Conflict_name
- Side_a
- Deaths_a
- Side_b

- Deaths_b
- Deaths_civillians
- Region
- Country
- Latitude & longitude

These columns are pretty self explanatory, as the name of the column suggests. Type_of_violence refers to the classification for each conflict. There are 3 possible values ranging from 1 to 3. For this project, we will be trying to create a predictive model. Using the columns “deaths_civillians”,”year”,”latitude”,”longitude”, . Hopefully with this model, investigators and police are able to utilize this, by using the facts easily available to them to recognize violence patterns from different organizations.

As can seen from the pictures above, the dataself itself has a lot of missing values. Which is why before continuing, we will have to clean and prepare the data by filling in the missing values, and selecting which columns we need for this project. And then we can start to make a visualization and model out of the data that we clean. The step by step process of the data preparation goes as follows:

1. After loading the data into Jupyter notebook, we look for the missing data

```
sum(df.apply(lambda x:sum(x.isnull().values),axis=1)>0)
```

```
1725
```

2. Then we pinpoint in which column does the missing data come from

```
print(df.isnull().sum())
```

```
id                      0
year                    0
active_year             0
start_year              0
end_year                0
type_of_violence        0
conflict_new_id         0
```

```
conflict_name          0
dyad_new_id            0
dyad_name              0
side_a_new_id          0
gwnoa                247
side_a                 0
side_b_new_id          0
gwnob               1725
side_b                 0
number_of_sources      0
source_article         9
source_office       1208
source_date         1208
source_headline     1208
source_original      498
where_prec             0
where_coordinates      0
adm_1                  6
adm_2                251
latitude               0
longitude              0
geom_wkt               0
priogrid_gid           0
country                0
country_id             0
iso3                   0
region                 0
event_clarity          0
date_prec              0
date_start             0
date_end               0
deaths_a               0
deaths_b               0
deaths_civilians       0
deaths_unknown         0
low                    0
best                   0
high                   0
dtype: int64
```

3. After analyzing each column and discussing the necessary columns for our model, we drop the columns we don’t need, mostly columns that have a lot of missing data because we decided that filling the missing value will be pointless for our goal

```python
dropcolumns =
['gwnoa','gwnob','source_office','source_date','source_headline','source_original','adm_1','adm_2','source_article']
for c in dropcolumns:
  df.drop(c,axis=1,inplace=True)

print(df.isnull().sum())
```

```
id                  0
year                0
active_year         0
start_year          0
end_year            0
type_of_violence    0
conflict_new_id     0
conflict_name       0
dyad_new_id         0
dyad_name           0
side_a_new_id       0
side_a              0
side_b_new_id       0
side_b              0
number_of_sources   0
where_prec          0
where_coordinates   0
latitude            0
longitude           0
geom_wkt            0
priogrid_gid        0
country             0
country_id          0
iso3                0
region              0
event_clarity       0
date_prec           0
date_start          0
date_end            0
deaths_a            0
deaths_b            0
deaths_civilians    0
deaths_unknown      0
low                 0
```

```
best                    0
high                    0
dtype: int64
```

4. Then from the remaining columns, we look for the unique values in order to find which column will suit best for our model

```
uniquecolumns = []
for columns in df:
  uniquecolumns.append(columns)

for c in uniquecolumns:
  print("\n",c,": ")
  uniquevals = df[c].unique()
  print(np.sort(a=uniquevals))
```

A snippet of the results:

```
id :
[ 68299.  68300.  68319. ... 273583. 274960. 276379.]

 year :
['1989' '1990' '1991' '1992' '1993' '1994' '1995' '1996' '1997'
'1998'
 '1999' '2000' '2001' '2002' '2003' '2004' '2005' '2006' '2007'
'2008'
 '2011' '2012' '2013' '2014' '2015' '2017' '2018']

 active_year :
[0. 1.]

 start_year :
['1989' '1990' '1991' '1992' '1993' '1994' '1995' '1996' '1997'
'1998'
 '1999' '2000' '2001' '2002' '2003' '2004' '2005' '2006' '2007'
'2008'
 '2011' '2012' '2013' '2014' '2015' '2017' '2018']

 end_year :
['1989' '1990' '1991' '1992' '1993' '1994' '1995' '1996' '1997'
'1998'
 '1999' '2000' '2001' '2002' '2003' '2004' '2005' '2006' '2007'
'2008'
 '2011' '2012' '2013' '2014' '2015' '2017' '2018']
```

```
 type_of_violence :
[1. 2. 3.]

 conflict_new_id :
[ 291.  330.  366.  493.  524.  539.  630. 4755. 4903. 4929.]

 conflict_name :
['Christians (Indonesia) - Muslims (Indonesia)' 'Dayak - Madurese'
 'Dayak, Malay (Indonesia) - Madurese' 'GAM - Civilians'
 'Government of Indonesia - Civilians' 'Indonesia: Aceh'
 'Indonesia: East Timor' 'Indonesia: West Papua'
 'Jemaah Islamiya - Civilians' 'Laskar Jihad - Civilians']
```

5. After selecting which columns we want, we perform operations to convert the data type. For example, we will be using the columns "side_a" and "side_b", however they are object types. To convert, we simply created mappings for each unique value and reassigned them in order to give them numerical values.

```
a_map = {
    'Christians (Indonesia)':1,
    'Dayak':2,
    'Dayak, Malay (Indonesia)':3,
    'GAM':4,
    'Government of Indonesia':5,
    'Jemaah Islamiya':6,
    'Laskar Jihad':7

}

b_map = {
    'CNRT':1,
    'Civilians':2,
    'GAM':3,
    'Madurese':4,
    'Muslims (Indonesia)':5,
    'OPM':6
}

final_df['side_a'] = final_df['side_a'].map(a_map)
```

A snippet of the final dataframe after mapping:

	side_a	deaths_b	year
1	5	3.0	2004
2	3	0.0	1999
3	3	0.0	1999
4	3	0.0	1999
5	1	0.0	2000
...	...	...	...

6. Finally, all that's left to do is to select the columns we want for our features and target before performing the data modelling and fitting.

IV. Model and Techniques

For the technique, we will be using several methods in order to visualize the data from this dataset. We noticed the dataset has latitude and longitude coordinates, so we will be doing a map visualization and a few other visualization. We will also be doing classification with the type of conflicts. We can also create a predictive model that will be able to predict the parties involved based on the civilian deaths, year, and location.

There are several libraries that we used for this project:

1. Numpy
2. Pandas
3. SciKit Learn

Numpy and Pandas are used to clean and process the data, as well as any other operations related to the csv file. SciKit Learn, or SKLearn, is used to model and fit the data. This includes classifiers, training, and metric libraries.

```
#Libraries for Classifiers
from sklearn import tree, neighbors, svm

#Libraries for training and metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, ConfusionMatrixDisplay,
```

```
classification_report, RocCurveDisplay
```

We also use Tableau, for creating the visualization.

V. Evaluation Method

Since our model is a multi-class classification, we will have to modify the evaluation metrics. There are 2 different ways to compute this, the first one is Macro Averaged F1, where the F1 score of every class is calculated and is then averaged. The second one is Micro Averaged F1, where we calculate the macro-averaged precision score and macro-averaged recall score and then take the harmonic mean.

For our evaluation method, we decided to use 3 different evaluation metrics. Those include Accuracy, Micro Averaged F1, Macro Averaged F1. These 3 evaluation metrics will help us evaluate the model itself after training and fitting. The reason why we chose 3 different metrics is to ensure that this data does produce good results and that it can be made reliable. For this model, we decided to use a 70-30 split, with a random state of 1752.

VI. Results and Discussion

After training and fitting the model, we evaluate each model to produce the following results:

Accuracy

	SVC	Decision Trees	Neural Networks
SIDE A			
Train Accuracy	88.82%	98.92%	85.17%
Test Accuracy	89.19%	95.17%	86.87%
SIDE B			
Train Accuracy	86.5%	97.76%	46.64%
Test Accuracy	85.91%	88.42%	45.17%

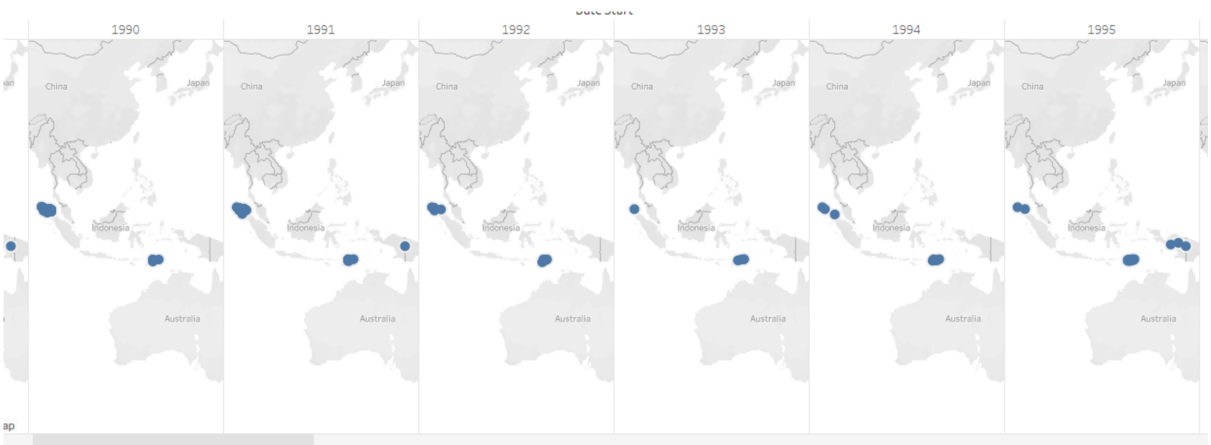
F1 Score (MICRO & MACRO)

	SVC	Decision Trees	Neural Networks
SIDE A			
MICRO	0.891891	0.950724	0.867376
MACRO	0.129166	0.156862	0
SIDE B			
MICRO	0.859073	0.891891	0.646717
MACRO	0.159203	0.154589	0

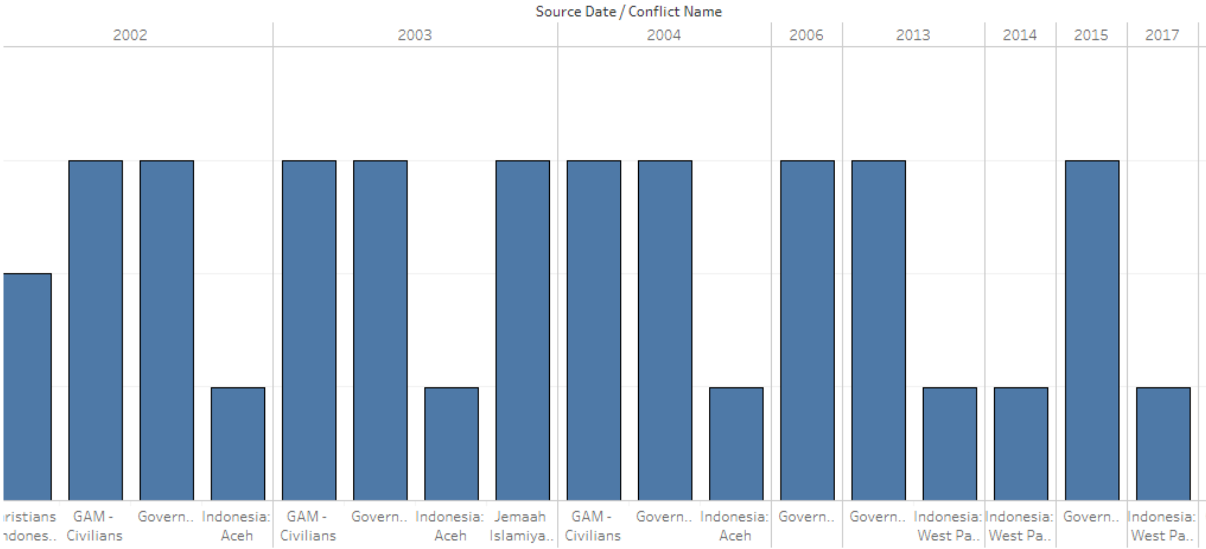
As seen from the data above, the model seems to be quite accurate, and shows promising potential to be used consistently, and effectively.

We choose to make three data visualization, The output we use is : Year of start, Latitude, Longitude, Conflict name, deaths_a, deaths_b, deaths civilians, death unknown, and event clarity.

the first one we use is Maps visualization.



The second visualization that we use is Bar



And the third visualization that we use is Area Charts

VII. Conclusion and Recommendations

As seen above from the evaluation results, our model was best using the Decision Tree, which produced the highest results compared to the 3. Although SVC came a close second place, this shows the power of Decision Trees. It might be also due to the fact that SVC can’t handle outliers like decision trees and k-nearest neighbors. From our evaluation metrics, we see that our model is actually quite accurate, which means that our hypothesis is true. That each organization or crime syndicate, has its own set of rules / standards that they deal with when involved in violent crimes. Each organization has its own region, and has a pattern that they conform to.

We definitely think that ultimately our model is not perfect. We require more data, and more time in order to collect the necessary amount of data to create a predictive model like ours. This model definitely does look promising, it has a pretty good accuracy on most cases, and hopefully can be reliable for future use especially in real world scenarios. The ability to predict a perpetrator based on the facts you have without having to do much work will revolutionized Indonesia's new hope, an efficient task force with resourceful investigators. Hopefully if this system is developed for real use application, Indonesia will be able to solve a majority of its problems and can start growing for a better country.

References :

[1] Sundberg, R., & Melander, E. (2013). Introducing the UCDP georeferenced event dataset. *Journal of Peace Research*, *50*(4), 523–532. https://doi.org/10.1177/0022343313484347

[2] Sundberg, R. & Melander, E. (2013). *Introducing the UCDP Georeferenced Event Dataset*. Humanitarian Data Exchange. Retrieved 2021, from https://data.humdata.org/dataset/ucdp-data-for-indonesia#data-resources-0

Github Link: https://github.com/radisahussein/DatSciFinalSem5

Questions :

1. How big is your Data?
   Our data has 1734 lines of data.

2. What is the most surprising insight that you find?
   From the Data we realize that each preparator have a signature style when dealing in violent conflicts.
   From the model and the evaluation we found that we can almost accurately predict the perpetrator based on the number of civillian deaths, location, and the year

3. What is your Label/Output?
   Our labels consist of side_a, side_b, deaths, year, latitude, longitude.
   Our target is side_a or side_b
   Our features are deaths_civillian, year, latitude, and longitude

4. Why is it multi-class classification?

It is a multi-class classification because our model is not looking to classify a binary option, such as yes or no, but instead will be classifying out of the different parties / organizations. Hence will require different functions when applying the evaluation metrics.