Isabel Magnus
Advanced Data Analytics
5.15.2020

Machine Learning and Armed Conflict Data

Introduction

Predicting when armed conflict will happen and where it will occur are vital to minimizing a conflict's human cost: the death, trauma, and prolonged suffering that happen as a result. In order to predict the next instance of armed conflict, it's necessary to understand the features that most contribute to its occurrence and what kind of relationship they have. Because I have no formal background in armed conflict and I'm not associated with any formal institution, I'm taking this opportunity to expand and explore my fundamental understanding of global armed conflict data.

```
print(conflict.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13440 entries, 0 to 13439
Data columns (total 12 columns):
Country Name                                        13440 non-null object
Year                                                13440 non-null int64
Adjusted_net_national_income_per_capita__current_US__    6909 non-null float64
Death_rate__crude__per_1_000_people_                11975 non-null float64
GDP__current_US__                                   10040 non-null float64
GDP_per_capita__current_US__                        10037 non-null float64
Military_expenditure__current_USD_                  7448 non-null float64
Population_density__people_per_sq__km_of_land_area_    12568 non-null float64
Population__total                                   13108 non-null float64
Rural_population                                    12990 non-null float64
Urban_population                                    12990 non-null float64
Battle_related_deaths__number_of_people_            13440 non-null int64
dtypes: float64(9), int64(2), object(1)
memory usage: 1.2+ MB
None
```

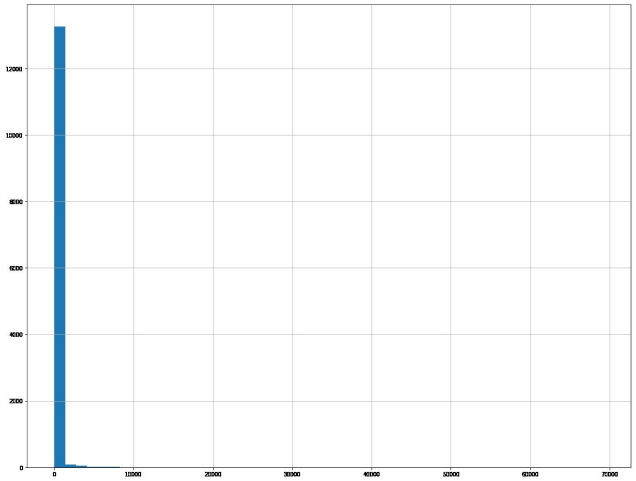
To begin my exploration, I turned to the World Bank's Database, specifically its World Development Indicators collection. With over 1,000 features to choose from, geographic and economic groups of countries in addition to each individual country, and over 50 years of data, I struggled to narrow my focus. Thinking of the problems presented by introducing too many insignificant features into machine learning models, I decided to work with some basic and decidedly random aspects of country and conflict, listed with their original data type in the chart to the left.

I will discuss specific variables in depth in the following sections: Pre-processing, Visualization and Transformation, Supervised Learning, Unsupervised Learning, and Conclusion.

Pre-Processing

This section entails my initial exploration of the features, their distribution, and how I've handled the missingness of data.

First, I have to identify and understand the dependent variable, which should be a direct measurement of armed conflict. While there is little consensus within the realm of armed conflict research, the Uppsala Conflict Data Program (UCDP) defines armed conflict as "the use of armed force by a group in order to get what they want". The UCDP uses battle-related deaths as the feature that classifies a conflict as an armed conflict,



which is why ‘battle-related deaths’ is the dependent variable I’m going to model on. In the histogram above, notice how the majority of observations occur between 0 and 10,000, while outliers stretch to just under 70,000. ‘Battle-related deaths’ is heavily right-skewed because I made a mistake at the beginning of this class and turned all of the NAs into 0s. This is a really good example of how the complexity of the data is often sacrificed for the ease of workability. In the Unsupervised Learning section, I introduce a new dataset with a more appropriately handled ‘battle-related deaths’ variable. In the meantime, Project 1 and Project 2 were based on the ‘battle-related deaths’ depicted by the histogram to the right. Because of it’s abnormal distribution, it may be beneficial to transform this variable, scaling it or making it logarithmic in order to make it a more normally-distributed shape, which is necessary for certain models.

Then, I have to prepare features with text for machine learning, which requires the creation of binary variables. I use Pandas get_dummies to encode both the ‘country’ and the ‘year’ features; I do have to wonder more largely, though, about using ‘country’ and ‘year’ as categorical, independent predictors. At best, perhaps, they stand as simple proxies for more complicated realities such as histories of colonialism, neighbors at conflict, and valuable natural resources. I will note that an important predictor of armed conflict touted among researchers is a country’s past history with armed conflict - countries that have had armed conflict before are more likely to have armed conflict again, and this is weighted more or less heavily based on how recently a country experienced armed conflict. If I had coded ‘year’ as a timedate feature, I might have been able to use that data type’s numerous analytical strengths to go further into this type of analysis. With categorical features separated and properly encoded, I then examine my continuous features.

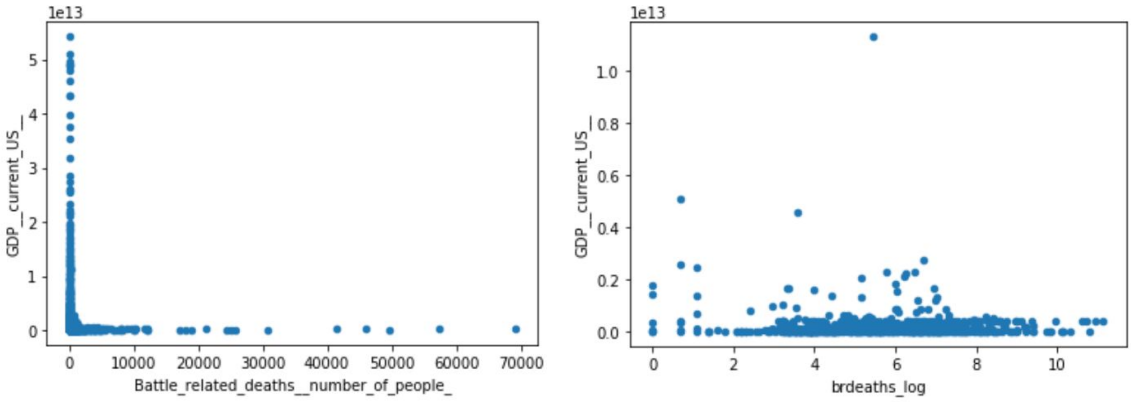
The missingness of data in the form of nulls and NAs pose a challenge in that they leave gaps in the larger story that the data depicts. NAs in data are meaningful in that they exist for some reason; in this case, missingness may result because certain countries have less developed infrastructure or because they have less access to the tools and resources necessary for capturing and tracking data. Within the context of armed conflict, it may also be interesting to explore the frequency of NAs in other features as they relate to ‘battle-related deaths’ - is there less information known about countries that have more conflict? I removed some of the features that I had initially included in the dataset because there were seemingly too many NAs, with NAs for at least half of the observations. With the remaining features, I decided to use ScikitLearn’s Imputer function, with strategy=mean. This fills all of the NAs in each feature with that feature’s mean value. It’s really important to acknowledge here that because of the way I bungled my ‘battle-related deaths’ variable by changing NAs to 0s, I changed the meaning of the data; imputing a mean or median value for ‘battle-related deaths’ might have brought me closer to valuable analysis.

Transformation and Visualization

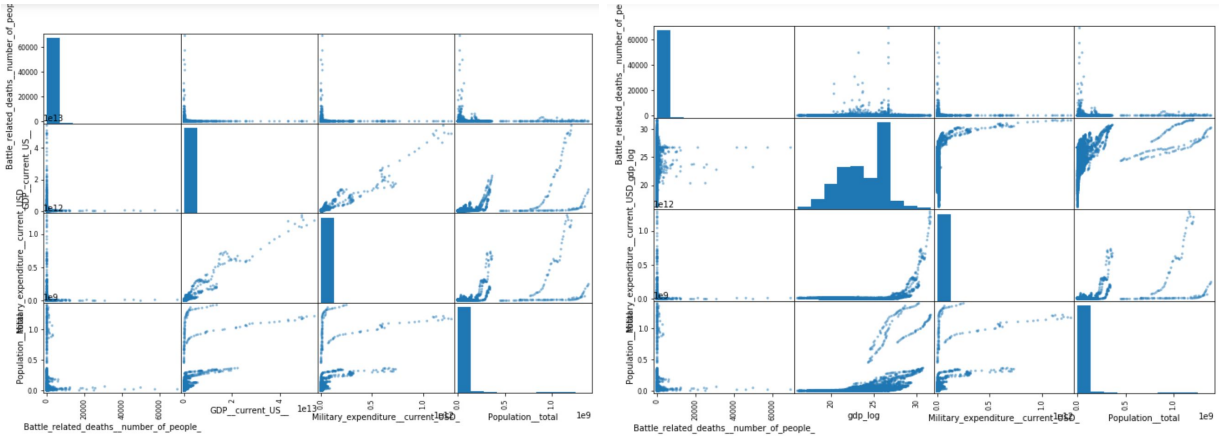
So much of the process of data analysis within machine learning depends on the minimization of certain errors, specifically: bias, variance, and the irreducible kind. Reducing one kind of error inevitably leads to an increase in another - while I understood this conceptually I did not

understand it practically until I began to run my supervised learning models. While I did experiment with the logarithmic variables of ‘GDP’ and ‘battle-related deaths’ as I visualized the data, I did not use these variables when running my models, which leads to reduced accuracy scores. I also unintentionally assumed that the data is linear, inviting in a lot of bias error. I may have had much greater success had I explored a cubed or even quadratic relationship, which may have helped me with the problem of underfitting. In addition, I didn’t scale my data appropriately by using a function like Standard Scaler that would have made my data more accessible to models like Linear Regression. I could have also binned certain variables, creating categories to reduce noisiness. Indeed, had I greater subject knowledge, I could have better identified and worked with outliers to address some of the irreducible errors. It’s very possible that instead of minimizing errors, I have maximized every possible error.

I could really appreciate the effect of logarithmic variables when I compared both GDP and battle-related deaths with their logged variables (see the charts below).



In the scatter plots above, ‘gdp’ is graphed by ‘battle-related deaths’ and the log of ‘battle-related deaths’. In the charts below, the only feature that has been changed between the two is ‘gdp’, from its original feature to the logged.



```
Battle_related_deaths__number_of_people_                1.000000
Syrian Arab Republic                                    0.253381
Afghanistan                                             0.172926
Ethiopia                                                0.118631
Iraq                                                    0.066494
                                                            ...
1963                                                   -0.010439
1981                                                   -0.010472
Population_density__people_per_sq__km_of_land_area_    -0.011285
GDP_per_capita__current_US__                           -0.025537
Adjusted_net_national_income_per_capita__current_US__  -0.029146
Name: Battle_related_deaths__number_of_people_, Length: 295, dtype: float64
```

Another important piece of exploratory data analysis is obtaining a preliminary understanding of how your features relate to one another, or how they are correlated. Some machine learning models struggle to produce accurate results when features are strongly correlated. I've provided a snapshot of the correlation matrix; there are no strong correlations between 'battle-related deaths' and the other features. However, it would be valuable to make matrices based on 'country', considering their relations to the other features included in the dataset. It would also be good to do a correlation matrix without countries included. I now know to use what I learn in exploratory data analysis to inform the way I transform the data and prepare if for machine learning.

Supervised Learning

I want to first acknowledge that my planning process was flawed: I did not intentionally ask myself which models would best serve my research objective and therefore did not transform and normalize my data accordingly. I also would have changed the order in which I ran models, starting with Decision Trees and letting their outputs guide the regression models.

Linear Regression

My first attempt at linear regression resulted in poor accuracy scores, with a training set score of 0.12 and a test set score of 0.09. I haven't scaled my inputs and my features' distributions are not all Gaussian, either - two factors that affect regression models. There's also extra noise in this data because certain variables (like total population, urban population, and rural population) are correlated, lacking independence. Other model measurements, like MAE, RMSE, and MSE also indicated the poor quality of the linear regression as a model for the conflict data. The large MAE, 250, indicates that my model is not very good at making predictions. The MSE and RMSE are both measures of fit. The RMSE, as the standard deviation of the prediction errors, is a measure of how far the data points are from the regression line. With an RMSE = 1739, we can safely say this is a very bad fit.

I used cross validation with the intention of minimizing the bias I've introduced by not using a balanced train/test set. Because of the size of the dataset, I might have been well served by using shuffle split validation but I used scikit learn's cross_val_score due to its simplicity. I tried cross validation with both 3 and 5 folds, and the model performs even more poorly when I increase the number of folds to 5. The average of scores (3-folds) = -0.19 while the average of scores (5-folds) = -16.03.

Generally, linear regressions can be improved upon by exploring the difference in results between L1 and L2 Penalty. I'm going to try a Ridge regression (L2 Penalty) to explore more/less robust regularization in the model. Because I am already doing so poorly (underfitting, which means the model is too simple), I don't believe Ridge is going to help me. Increasing alpha moves more coefficients towards 0. My ridge models perform almost identically to my

plain linear regression model, with a higher alpha slightly worsening the test set score. I'm now going to try using Lasso regression (L1 Penalty). With Lasso, I can adjust both the alpha values (significance of coefficients) and the max_iteration value. In these models, if alpha is too high it will zero out too many coefficients; an alpha value that is too low will result in a model that is effectively unregularized. I have not achieved train/test scores that are any better by using Lasso. I still have a model that's really underfitting. When I decreased alpha, the model used almost all of the features (i.e. it's almost completely unregularized when it's considering 292 out of 294 features). When I dramatically increase the alpha to 100, the model zeroes out too many coefficients, using only 8 features and returning the worst possible train/test scores of 0. I didn't expect to do better with Lasso than I did with Ridge, but it has been interesting to see the slight differences in resulting scores - it would appear that a completely unregularized Lasso performs better than a Lasso with an alpha as high as 10!

Decision Trees, Random Forests, and Gradient Boosted Forests

Decision trees don't require scaling and normalization, although they do tend to overfit. With my first tree of max_depth 3, I get accuracy on the training set=0.346 and accuracy on the test set=0.504. I'm imagining this would be a very interpretable visualization, but I'm not able to to figure out the tree visualization code, unfortunately. With an R^2 of 0.50, the model explains half of the variance of the dependent variable. However, the RMSE being so high at 1651270.46 also makes it seem that this isn't a good fit. When I set a larger max depth=5, I give my tree more room to grow in complexity before it finds the best leaves. Accuracy improves for both the training set and the test set, with scores 0.787 and 0.801 respectively. With more depth, I have better accuracy scores, a higher R^2 (0.80), and a lower RMSE (it was >1,000,000 before).

The Forest is less easily interpretable but creates many trees and averages their scores. The forest increased the accuracy on the training set, with 0.804, and decreased accuracy on the test set=0.578, which means it overfit to the training set. My first forest didn't perform as well as my tree with max_depth = 5, perhaps because I had a small number of estimators. I made a second forest with a much higher number of estimators (100) that did a bit better on the training set but even worse on the test set. I then ran Gradient Boosted Trees (GBT) to see how well they would model the data, with each shallow tree correcting the mistakes of the previous tree. In the first GBT, the accuracy score on the training set was 1 and the accuracy on the test set was 0.921. While the model overfit, it still produced the highest test score accuracy so far! I ran three GBRTs and would like to note the following: n_estimators=100, random_state=0, (and a default learning_rate=0.1) produced the highest accuracy scores for both the train and test set. The learning_rate controls how strongly each tree tries to correct the previous tree, so I could have used a higher learning rate. While GBRTs function on the concept that many shallower (i.e weaker trees) will be able to correct one another and provide the strongest accuracy, a max_depth of 1 is too shallow and results in worse scores. The higher n_estimators worked well because it added more trees to the ensemble and increased model complexity.

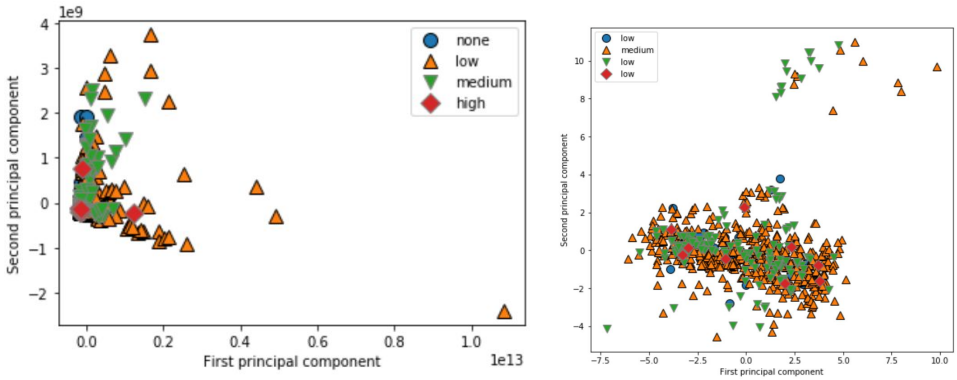
Unsupervised Learning

The dataset I used for Project 1 and Project 2 was not appropriate for Principal Component Analysis: there were far too many classes within 'battle-related deaths'. Instead, I created a

dataset including only non-null observations for ‘battle-related deaths’ and then binned those observations into four categories: none, low, medium, and high. I also was unable to apply PCA to the clusters in Step Three, so the comparison is between clusters on scaled data and clusters on unscaled data.

Principal Component Analysis

The graph to the bottom left shows PCA on data that has not been scaled, while the graph on the bottom right shows PCA on data that has been scaled. The scaled PCA graph is much better spread because of the whitening effect of scaling, or the reduction of collinearity.



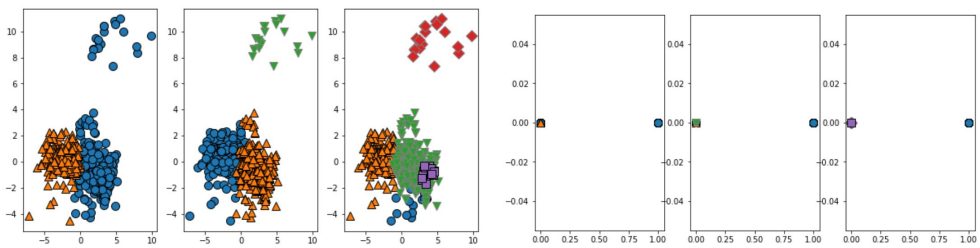
When I perform logistic regression with PCA, the accuracy score on the training set is 0.85 and the accuracy score on the test set is 0.69. I can't necessarily compare these scores to the models from Project 2, however, because I've changed datasets and model types. There is overfitting happening though, which is odd because PCA should counter overfitting. My data isn’t normally-distributed and this might cause the model to perform less well, as PCA assumes linearity.

It was very helpful to directly compare the three types of clustering -KMeans, Agglomerative, and DBSCAN- on data with PCA and data without PCA.

KMeans Clustering

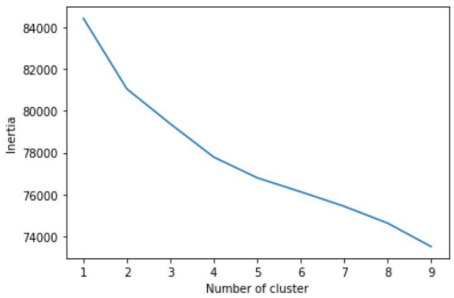
KMeans assumes the clusters have the same width/diameter. There tends to be issues with non-spherical/complex shape clusters but my shapes are rather spherical so that shouldn't be a problem; however, because there are data points that overlap, the random initialization of KMeans may cause me to have different clusters each time that it's run. In the sets of graphs below (with n_clusters = 2, 3, and 5 respectively), it’s clear to see that KMeans with scaled data creates a much more easily understood visual representation of distinct and separate clusters.

KMeans with Scaled Data **KMeans without Scaled Data**

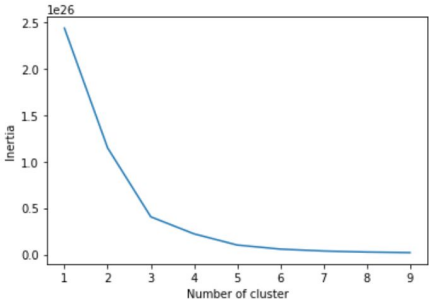


I compare the elbow graphs below and find the KMeans with scaled data elbow graph to have a relatively smooth curve all the way down to 9. The elbow graph for KMeans without scaled data shows the most dramatic decrease in slope happening between 1 and 2 clusters, with the slope plateauing somewhere after 5.

KMeans with Scaled Data



KMeans without Scaled Data

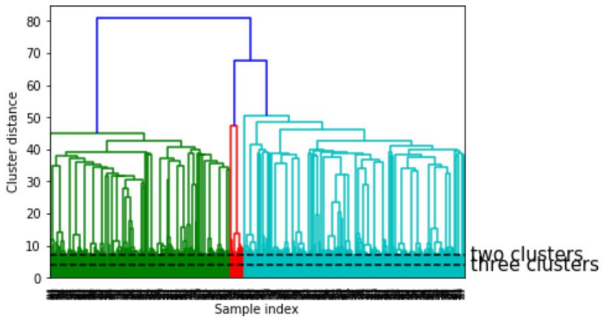


Agglomerative Clustering

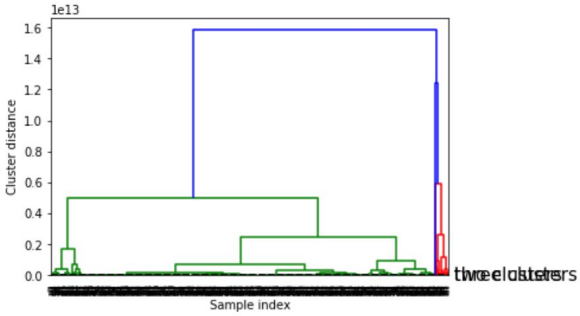
When I explored agglomerative clustering, I experimented with the stopping point: between 3 to 4, only one point is changed and classified as its own individual cluster, which led me to believe that n=3 is the better stopping point. I didn't run the clustering with different linkage types; ward as the default seemed most appropriate.

Dendrograms provide visualization of multidimensional datasets and information about cluster distances. With scaled data, it's much easier to see the spread along features even with a very small distance between two and three clusters. Without scaled data, there is even less space between the second and third cluster.

Agglomerative with Scaled Data



Agglomerative without Scaled Data



DBSCAN

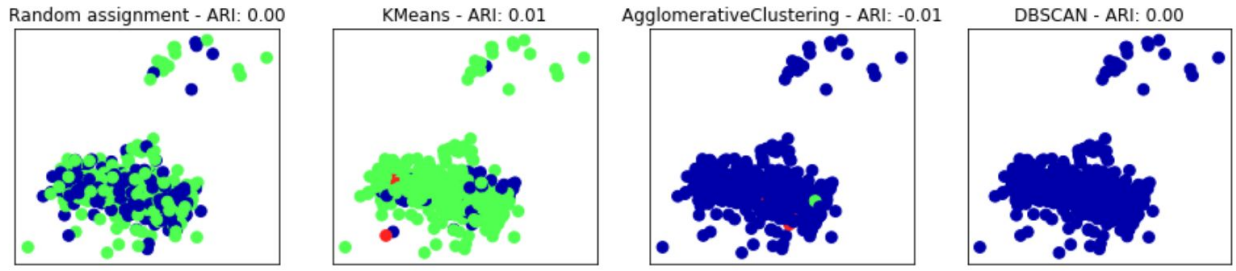
I didn't learn very much from this method of clustering. In its attempt to distinguish between densely and sparsely populated data, DBSCAN found two clusters. However, the clusters cannot necessarily be distinguished in the graph provided and I'm surprised that it didn't identify outliers. There was very little, if any, difference between DBSCAN with and without scaled data.

ARI scores

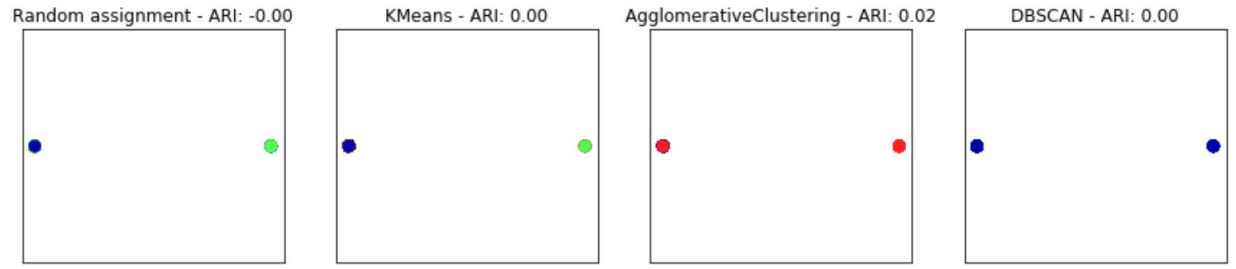
ARI scores demonstrate how close our clustering methods are to approaching ground truth, or how well clustering methods do compared to randomly assigned clusters for data that will be

used for supervised learning. None of the methods are close to aligning with ground truth at ARI = 1.00, and scaled data does very little to improve ARI scores, except for Agglomerative clustering which scores the highest out of both sets of clusters. With PCA, I may have had a better result in each of my clustering models.

With Scaled Data



Without Scaled Data



Conclusion

Having finished and submitted Projects 1, 2, and 3, I can safely conclude that I know far more now than I did when I started - hindsight is both a blessing and a curse. There is so much that I would change were I to do these projects all over again, starting with the original dataset. I should not have changed the 'battle-related deaths' nulls to 0s; instead, I would prefer to work with only non-null observations, deleting the nulls from the downloaded dataset. From there, I would have liked to work with two datasets from Project 1 all the way through Project 3: one dataset with the continuous 'battle-related deaths' dependent variable and another dataset with 'battle-related deaths' feature binned appropriately. This would allow me to compare different types of machine learning models based on how they perform on categorization and regression models. In addition, I'm curious to see how my models would have changed were I to delete outliers, or better yet, if I were to have used the logged 'battle-related deaths' variable and other normalized features. I would also like to play around more with scaling and transformation. It was very helpful to see the effects of scaled versus unscaled data in Project 3. This kind of direct comparison would have served me well throughout the first and second projects.

I did not go about researching my objective in the appropriate way, and I believe that I have learned more by doing things incorrectly over the span of these three projects than I have learned by doing them properly. While I can by no means say that I have advanced the field of armed conflict research, I have expanded upon my understanding of conflict data, the challenges in working with conflict data, and the ample opportunities that exist for improved machine learning and analysis in armed conflict analysis.