CORRELATES OF WAR

IS733 DATA MINING

PROJECT REPORT | FALL 2020

Prepared By: Branden Palma, Chhaya Kulkarni, Clay Styer, Kirubel Tolosa
Instructor: Dr. James Foulds (PhD)

CONTENTS

1. ABSTRACT

The Correlates of War project has collected an abundance of datasets related to international affairs going back almost two hundred and fifty years. Their mission statement involves making quantitative data on international relations data available online to anyone who wants to analyze it. This includes a wide range of different information, from religious trends to diplomatic efforts to trade, all hosted on the website in .csv format. The centerpiece of all of this is a dataset listing out every major war in the last 250 years with considerable granularity. Our goal is to use this data in order to build a model that can determine whether conflict will occur between countries.

2. BACKGROUND AND RELATED WORK

2.1. METHODOLOGY

Data Preprocessing

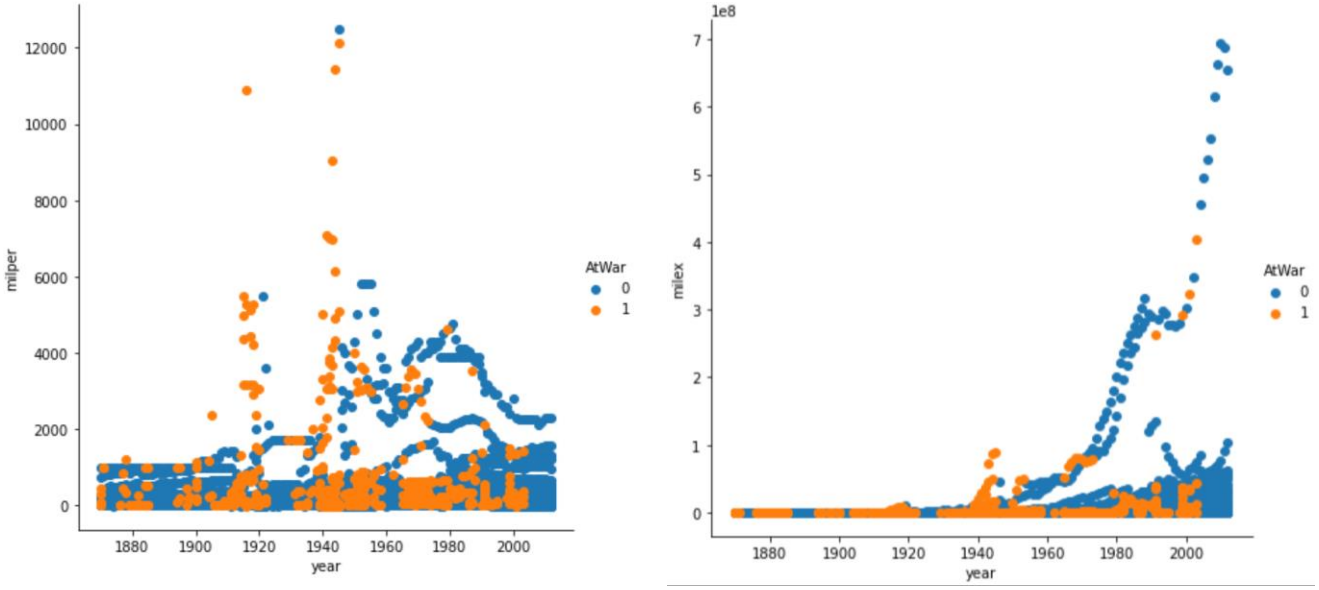
As aggregating all pertinent information from all datasets would have been too ambitious a scope to start with, we selected the two datasets we thought would be most relevant to the outbreak of war, specifically a trade dataset called 'National_COW' and an index of military capabilities by year called 'NationalMilitaryCapabilities. Using the 'Pandas' library in python we performed an inner join on the two datasets, comparing 'year' as well as 'country' to ensure that we got only instances present in both datasets. This resulted in roughly 13700 rows. We then leveraged the InterStateWar dataset, extracting a list of countries and the years they were at war (a difference in format made this a slightly more drawn out programmatic step). Using the list we generated ,we appended a column - a flag - which is a binary tag indicating whether or not a given country is at war during a given year. 0 would indicate that the country is actively engaged in conflict, where a 1 would indicate that the country is not currently at war.

Missing value information: There were some records with missing information. This was expected, as the dataset covers many countries over a very long period. Additionally, many countries lacked accurate and efficient tools for data collection. We assessed these records, and ended up eliminating any and all rows in which the majority of the data was absent in our new joined dataset, which brought our total number of instances down to around 11000. In the cases that fewer records were missing, the average of the values from the previous year and the next year were used in their place (if both values were available).

Data Analysis

With the aim of establishing logical interpretations prior to running the algorithms, we did some analysis on the data. This was to primarily help our understanding of the data and confirm our assumptions. One of the early assumptions that we had about the comparative strength of the attributes was that of all features “Military-Expenditure” would be the strongest in predicting the outcome of the classification. This is based on our basic deductive reasoning that if a state is spending more on military expenses that it should be in more conflicts than others.

Thus, we went ahead and plotted the two prominent features, military expenditure and military personnel, against time. To our surprise however, what we found out from the initial analysis was contrary to our assumption. States that had more boots on the ground were in more conflicts than others.

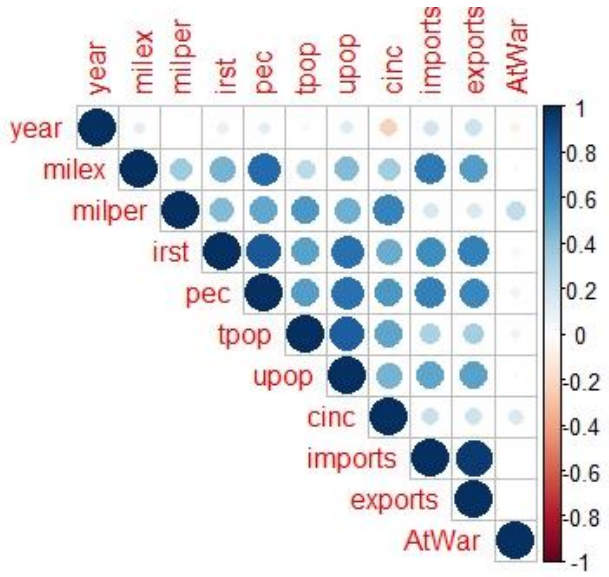


Below are a few key points that we learned from this analysis.

- States that spend more billions of dollars in military expenditure are less likely to be involved in military conflicts.
- States that comparatively have higher number military personnel are more likely to go into conflict than those that do not.
- We can see that there was a spike in military personnel recruitment during the world wars and that has died down since then but military expenditure has grown exceptionally, which may be tied to the economic advancement of nations after the world war.
- We can also see that the world is seeing much less conflicts as time passes and there’s very little conflicts registered after the 2000s.

Another key takeaway here is of all the records that were registered in the dataset, about four percent of the records were labeled “At War”. We used the “Seaborn”, matplotlib based visualization library in python to produce the scatter plots.

Correlation Matrix



A Correlation Matrix is useful for determining whether there is a strong or weak, positive or negative relationship between two variables. Analyzing the relationships between variables can help address the problem of dimensionality. If there are two variables that are strongly correlated, it may be useful to get rid of one of the variables to reduce the amount of noise in the dataset. It is important to reduce this kind of noise because redundant variables can lead to overfitting, which is an undesirable event in building predictive models. Another benefit of reducing dimensionality is that algorithms tend to run faster due to less features that are present in the dataset.

Surprisingly, the response variable “AtWar” has no strong negative or positive correlations with any predictive variable. In retrospect, the predictive variable “year” was assumed to have a strong correlation with the “AtWar” variable, but it is shown that it has little to no correlation.

In support of our previous assumptions, that military expenditures and personnel tend to have a strong effect on the “AtWar” variable, we can see that these two predictive variables have a modest positive relationship. This means that as military expenditures rise, so does military personnel and vice versa. This is more common sense because if you have more troops, you need to spend more to keep them supplied. There are many other strong correlations in the dataset, and these could identify more risk factors that influence whether or not a country is at risk of war.

2.2. LIBRARIES AND TOOLS

Python and Libraries:

Pandas has been our preferred choice for data manipulation. Pandas was helpful to visualize our data in different ways such as the first 5 records, last 5 records or the entire dataset. We could easily add and drop a column or a record too. Pandas was extremely useful for handling the missing value treatment for our data.

We used the Scikit-Learn package(sklearn) for performing cross validation. Sklearn was used to run different machine learning algorithms such as Naïve Bayes, K- Nearest Neighbor algorithms. Label encoding feature of sklearn definitely made our life easier when we had to replace the country names with numerals. Confusion matrices were created using sklearn package.

Jupyter Lab:

Platform on which we ran python elements of the project.

Google Colab:

Platform used to run the neural network algorithm for our project.

R:

Utilization of the "corrplot" package and library in RStudio allowed for the creation of a correlation matrix.

Weka(Waikato Environment for Knowledge Analysis)**:**

Weka tool was developed at the University of Waikato, New Zealand in the early 1990's. It is an open source machine learning software. It can be accessed either via a Graphical User Interface or via Java API. It works with the unique file format 'ARFF'. However, it supports other file formats as well. Greatest advantage of WEKA is that it is great for trying out various machine learning algorithms on data in a matter of a few minutes. Results of each machine learning algorithm can also be analyzed using their 'Visualize' feature. Data scientists highly prefer working with Weka. We have had an extremely positive experience using Weka in our project. We used Weka to build our machine learning models using logistic regression and decision tree classifier. We used the 'extra tree classifier' algorithm in Weka to determine the ranking of each attribute of our dataset. Weka has proven useful to generate ROC and AUC curves as well. We are now staunch supporters of Weka.

3. ALGORITHMS

KNN

Justification:

We decided to try using a SciKitLearn instantiation of the K Nearest Neighbor algorithm because it has considerable predictive power. It also functions well when working with datasets with few outliers, and indeed especially after scaling this dataset had little in the way of particularly large outliers. Also, it seems that it functions well on small to mid sized datasets with good labelling, which is what we are working with.

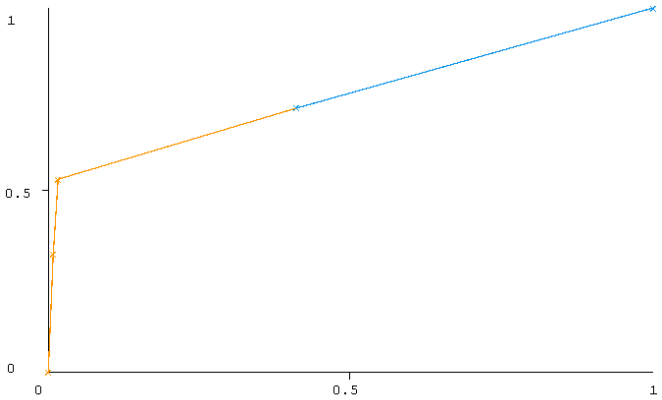
Hyperparameter tuning:

Starting from the recommendation of using the square root of the total size of the dataset (which was right around 100), we tried a variety of different values for the number of neighbors for the algorithm, using primarily odd numbers as we had an even number of classes. We found that the model had the best performance with a considerably smaller number than the one we tried initially, finding best classification performance at around 5 neighbors.

Experimental results

With our tuning taken care of, the model performed rather well, correctly classifying 95.88% of instances. This puts it in a similar category with Logistic Regression, Neural Networks, and Decision trees. That said the following AUC curve puts this high accuracy score somewhat into doubt.

AUC Curves



Confusion Matrix (Weighted Avg.)

```
  a     b     <-- classified as
3868   7 |    a = 0
160    5 |    b = 1
```

True Positive Rate (TP) : **99.819%**
False Positive Rate (FP) : **95.808%**

Confusion Matrix for KNN

Naive Bayes

Justification

Naive Bayes seems like a good starting point for exploration, so it was the first algorithm we tried. Specifically, we used the SciKitLearn 'ComplementNB' library as the documentation said that it performed particularly well with unbalanced datasets.

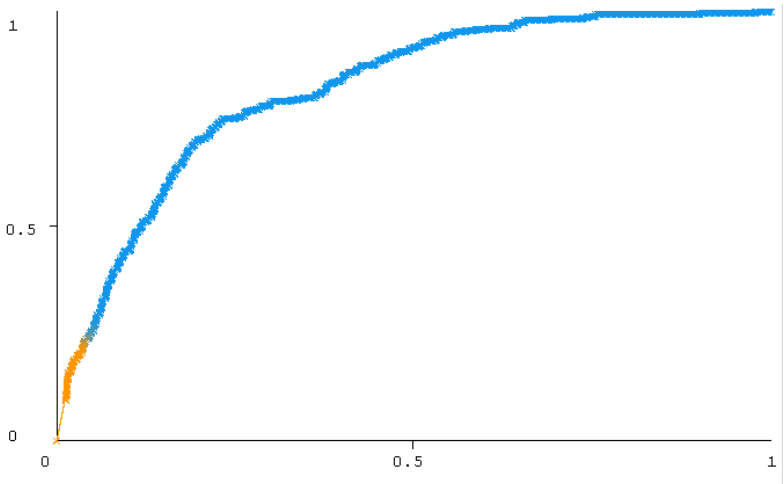
Hyperparameter tuning

There were no hyperparameters associated with the ComplementNB library.

Experimental results

Though it was the worst of the methods we used , Naive Bayes still managed to achieve an accuracy of more than 88%. With that in mind, it also had the same issue KNN did with respect to what is indicated by the AUC plot.

AUC Curves



AUC Plot for classes that tested Positive - Naive Bayes.

Colors: Orange = Positive , Blue = Negative)

- The area under the curve measures **0.8157** indicating a great accuracy.

Confusion Matrix (Weighted Avg.)

```
  a     b     <-- classified as
3666  16 |   a = 0
137   21 |   b = 1
```

True Positive Rate (TP) : **99.565%**
False Positive Rate (FP) : **89.542%**

Confusion Matrix for Neural Network

Neural Network

Justification

Neural networks are popular because they eliminate the major time spent on feature engineering. We used PyTorch library to build the neural network. PyTorch library is well known for its simplistic approach. It is widely used in Academia. PyTorch also favors a class based programming which makes segmenting the code an easier task. PyTorch was especially easy to understand as we were familiar and comfortable with Python. PyTorch is flexible to use.

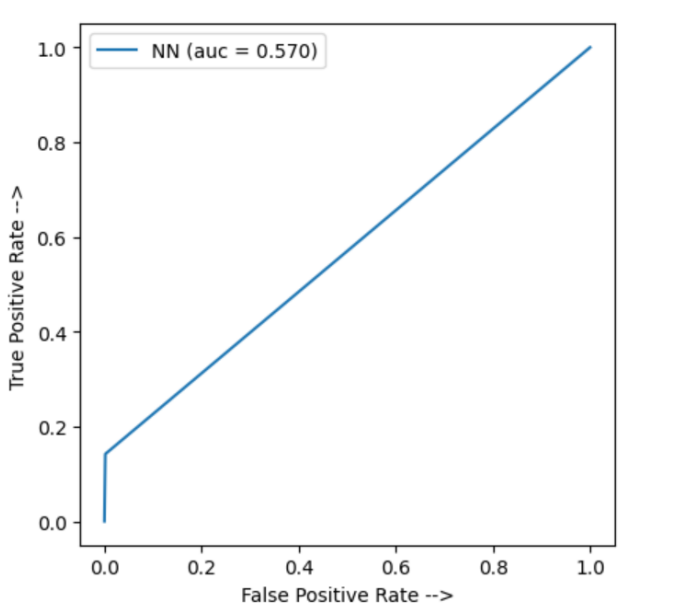
Scikit learn, Pandas and NumPy packages were also utilized to construct the neural network. The field country_name which repeated in many of the records in the dataset was imputed. For imputation, the label encoding feature of the Scikit learn package was used.

Hyperparameter tuning

Data was split into training and testing sets. Training dataset included 70% of the dataset and the test dataset comprised 30% of the entire data. Before building the model, data was standardized using the Standard Scaler option available in Scikit Learn package. Neural network model consisted of three layers viz input layer, hidden layer and output layer. The first layer consisted of 13 nodes as we had 13 input features. We used the Rectified Linear Unit(ReLU) activation function and the loss function we used was Binary Cross Entropy Logits Loss. Adam Optimizer was used. We set the number of epochs to 50. Batch size was set to 64 and learning rate was 0.05.

Experimental results

We were able to achieve an accuracy of 96% with this model, with performance second only to the logistic regression approach. The AUC curve illustrated below does not justify the accuracy that we have obtained using our neural network model.



AUC Curve for Neural Network

Confusion Matrix (Weighted Avg.)

```
  a     b     <-- classified as
3874   10 |    a = 0
138    17 |    b = 1
```

True Positive Rate (TP) : **96.56**
False Positive Rate (FP) : **37.03**

Confusion Matrix for Neural Network

Logistic Regression

Justification

Logistic Regression is a classification algorithm that is easy to train, efficient, and easy to implement. Simple algorithms like logistic regression tend to work just as well or better than complicated ones, a phenomenon that is known as occam's razor. At the very least, a baseline of performance can be gained to provide insight for other algorithms. Additionally, our response variable uses the number 1 to represent at war and 0 to represent not at war. Therefore, a binary classifier such as logistic regression is appropriate. We used Weka to implement the algorithm by importing the data and selecting the "logistic" classifier. Weka has an easy to use GUI and the experience gained from prior assignments allowed us to use the program without many issues.

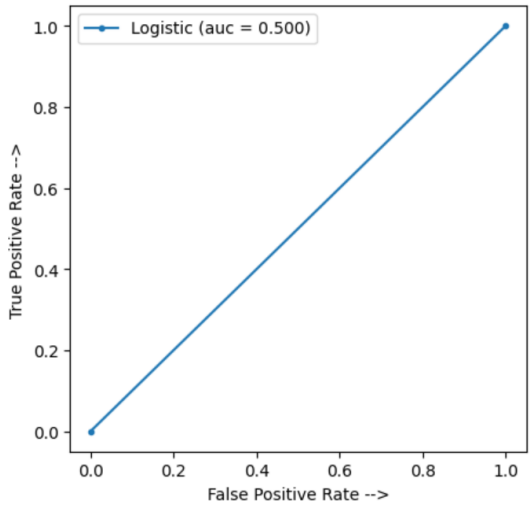
Hyperparameter tuning

Logistic regression does not have any explicit hyperparameters to tune (Brownlee, 2019). Differences in solvers and regularization can help display useful differences(Brownlee, 2019). These alternative methods were not implemented because they were not applicable for our project.

Experimental results

Logistic regression performed very well, with an accuracy of 96.52% of correctly classified instances. However, a 87% false positive rate obscures the meaningfulness of this accuracy.

AUC Curves



AUC Curve for Logistic Regression

Confusion Matrix (Weighted Avg.)

a	b	<-- classified as
10862	9	a = 0
384	39	b = 1

True Positive Rate (TP) : **0.965**
False Positive Rate (FP) : **0.874**
F-measure : **0.952**

Confusion Matrix for Logistic Regression

With the AUC equaling .5, the model cannot efficiently discriminate between the true positivity rate and false positive rate.

Decision trees

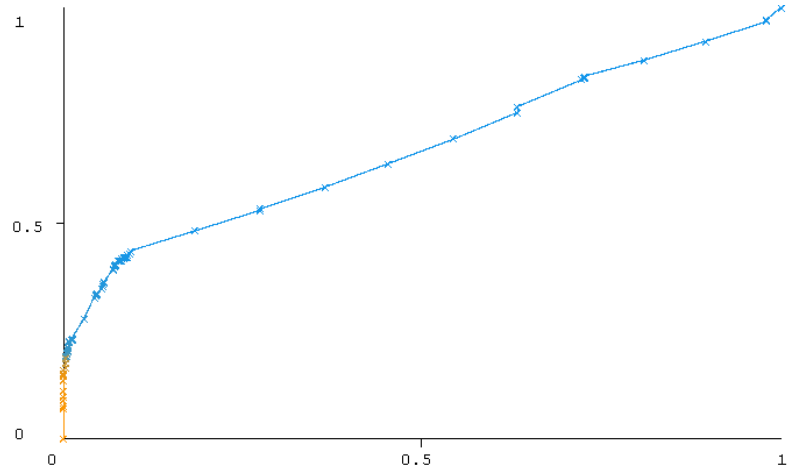
Justification

Decision tree being a popular algorithm for classification tasks, we used it as one of the algorithms for our data considering the ease of data preprocessing it requires as well as the simplicity of its implementation. A decision tree doesn't require scaling or normalization of data, which is important particularly for our dataset as our pre-processing did not involve normalization and scaling and we do have extremely varying values for some of our attributes. Moreover, decision trees are very intuitive and easy to explain. One of the other advantages we aspired to get out of using decision trees is the handling redundant predictive attribute values which we observed in the correlation matrix.

Experimental results

We used the J48 Decision Tree Classifier from Weka for this particular classification task. The percentage split between test and training data was 66% to 34%. The algorithm resulted in 95.02% correctly classified instances, leaving about 4% incorrectly classified. We observed the following measures of accuracy for our test.

AUC Curves



AUC Plot for classes that tested Positive. (Colors: Orange = Positive , Blue = Negative)

- We have a weighted average of 0.669 from the area under the ROC curve for the class labels.
- We observe from the AUC Curve above that the plot leans very close towards the true positive Y-axis which indicates the good performance of our model.

Confusion Matrix (Weighted Avg.)

a	b	<-- classified as
3666	16	a = 0
137	21	b = 1

True Positive Rate (TP) : **0.960**
False Positive Rate (FP) : **0.832**
F-measure : **0.948**

Confusion matrix for Decision Tree

4. FORECASTING

Through our data analysis efforts for this project, we were able to figure out which are the two major risk factors one being military personnel(milper) and the military expenses(milex) of a country. However, we realize that there might be other factors too that could lead to war like tensions developing between countries. In order to build a forecasting model we would need to be clear about the possible risk factors and then predict the % danger for warlike conditions to develop in a country. We would ideally have to build a regression model.

As the model and risk factors currently stand, the most recent datasets can be acquired from the correlates of war website, and this data can be plugged into the model to predict whether or not a country will be in conflict. This is a form of forecasting because a country can increase troops and military expenditures without explicitly being in war. However, these patterns usually correlate to war in the near future.

5. CONCLUSION AND FUTURE SCOPE

We noted that KNN, Logistic Regression and Neural network models achieved an accuracy of 96%. Based on the AUC plots which were important to check the accuracy of our prediction due to the disproportionate composition of the class labels, we found Naive Bayes to have performed better than the other algorithms we tried out.

Different features such as Military Expenditures, Military personnel, Energy consumption, Iron and Steel production have been useful in predicting which countries are most likely to engage in wars. Strongest indicator of a country being at war in the current year is tied mainly to the number of military personnel a state has. In the future, we hope to retrofit this model and improve it by trying different variables and classification techniques.

REFERENCES

Aflak, O. (2018, November 14). Neural Network from scratch in Python. *Towards Data Science*. Retrieved on December 9, 2020 from https://towardsdatascience.com/math-neural-network-from-scratch-in-python-d6da9f29ce65

Bhandari, A. (2020, June 16). AUC-ROC Curve in Machine Learning Clearly Explained. *Analytics Vidhya*. Retrieved on December 10, 2020 from https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/

Bronshtein, A. (2017, May 17). Train/Test Split and Cross Validation in Python. *Towards Data Science.* Retrieved on December 9, 2020 from https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

Brownlee, J. (2019, December 13). Tune Hyperparameters for Classification Machine Learning Algorithms. *Machine Learning Mastery*. Retrieved on December 10, 2020 from https://machinelearningmastery.com/hyperparameters-for-classification-machine-learning-algorithms/

Correlates of War Project. (2020). COW War Data (v5.0). Retrieved from https://correlatesofwar.org/data-sets

Correlates of War Project. (2020). National Material Capabilities (v5.0). Retrieved from https://correlatesofwar.org/data-sets

Gupta, A. (2020, January 7). ML | Extra Tree Classifier for Feature Selection. *GeeksforGeeks.* Retrieved on December 9, 2020 from https://www.geeksforgeeks.org/ml-extra-tree-classifier-for-feature-selection/

Rout, A. (2020, February 9). Advantages and Disadvantages Of Logistic Regression. *GeeksforGeeks*. Retrieved on December 9, 2020 from https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/