NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

PREDICTING THE SPREAD OF TERRORIST ORGANIZATIONS USING GRAPHS

by

Anthony B. Vanderzee

June 2018

Thesis Advisor:	Samuel H. Huddleston
Co-Advisor:	Jesse R. Hammond
Second Reader:	Ruriko Yoshida

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2018	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE PREDICTING THE SPREAD OF TERRORIST ORGANIZATIONS USING GRAPHS	**5. FUNDING NUMBERS**
6. AUTHOR(S) Anthony B. Vanderzee	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

The U.S. Defense and Intelligence communities expend vast amounts of resources tracking and trying to predict the geographic spread of terrorist groups such as the Islamic State of Iraq and Syria (ISIS). Current approaches to this problem use a variety of social, demographic, and geographic data to make predictions about the spread of a terrorist organization. We demonstrate a novel approach that converts the geographic area of interest, Iraq and Syria, into a graph with the populated places as nodes and the road network as the edges of the graph. We then use this graph to compute graph-based statistics such as measures of centrality and first-order neighbor statistics on the nodes in the graph. By adding the graph-based features, we combine social, demographic, and geographic data with data that quantifies the relationships between the populated places in Iraq and Syria. This ultimately improves predictive performance for predicting future territorial gains and losses by ISIS. Furthermore, our models demonstrate that the graph-based features are the most influential variables in predicting whether or not a node will be in or out of ISIS territory.

14. SUBJECT TERMS geospatial analysis, graph analysis, network analysis, terrorism, machine learning	**15. NUMBER OF PAGES** 115
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

PREDICTING THE SPREAD OF TERRORIST ORGANIZATIONS USING GRAPHS

Anthony B. Vanderzee
Captain, United States Marine Corps
BS, University of San Diego, 2012

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 2018

Approved by: Samuel H. Huddleston
Advisor

Jesse R. Hammond
Co-Advisor

Ruriko Yoshida
Second Reader

Patricia A. Jacobs
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The U.S. Defense and Intelligence communities expend vast amounts of resources tracking and trying to predict the geographic spread of terrorist groups such as the Islamic State of Iraq and Syria (ISIS). Current approaches to this problem use a variety of social, demographic, and geographic data to make predictions about the spread of a terrorist organization. We demonstrate a novel approach that converts the geographic area of interest, Iraq and Syria, into a graph with the populated places as nodes and the road network as the edges of the graph. We then use this graph to compute graph-based statistics such as measures of centrality and first-order neighbor statistics on the nodes in the graph. By adding the graph-based features, we combine social, demographic, and geographic data with data that quantifies the relationships between the populated places in Iraq and Syria. This ultimately improves predictive performance for predicting future territorial gains and losses by ISIS. Furthermore, our models demonstrate that the graph-based features are the most influential variables in predicting whether or not a node will be in or out of ISIS territory.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AQI al-Qaeda in Iraq

CART Classification and Regression Tree

COA Course of Action

CRS Coordinate Reference System

DoD Department of Defense

ESRI Environmental Systems Research Institute

FPR false positive rate

GIS Geospatial Information Systems

GLM Generalized Linear Model

GNS GEOnet Names Server

GRASS Geographic Resources Analysis Support System

GTD Global Terrorism Database

IC Intelligence Community

ISIS Islamic State of Iraq and Syria

ISW Institute for the Study of War

JCS Joint Chiefs of Staff

JFC Joint Force Commander

JIPOE Joint Intelligence Preparation of the Operational Environment

NGA National Geospatial Intelligence Agency

OE Operational Environment

OIR Operation INHERENT RESOLVE

OSM OpenStreetMap

QGIS Quantum Geospatial Information Systems (GIS)

START Study of Terrorism and Responses to Terrorism

TPR true positive rate

Executive Summary

The U.S. Defense and Intelligence communities expend vast resources tracking and trying to predict the geographic spread of terrorist groups such as the Islamic State of Iraq and Syria (ISIS). Current approaches to this problem use a variety of social, demographic, and geographic data to make predictions about the spread of these terrorist organizations. In this thesis, we seek to improve upon current methods and demonstrate a new methodology that can accurately predict the territorial gains and losses of ISIS in Iraq and Syria between 2014 and 2017.

This methodology converts the geographic region of Iraq and Syria into a graph where the populated places in the two countries are the nodes in the graph, and the road network is converted to a series of arcs that connect the nodes. We then use various graph and statistical techniques to extract features from the graph and add them to our social, demographic, and geographic data that we obtain from open sources. These graph features provide data that quantifies the relationships between the populated places in Iraq and Syria. We develop a rolling horizon design and use machine learning algorithms on this expanded data set over 26 time periods between 2014 and 2017. These models make predictions about which populated places in Iraq and Syria will be in or out of ISIS territory at each time period in our design. Figure 1 shows our methodology. The steps inside the orange box represent how our methodology builds upon current approaches to this problem.

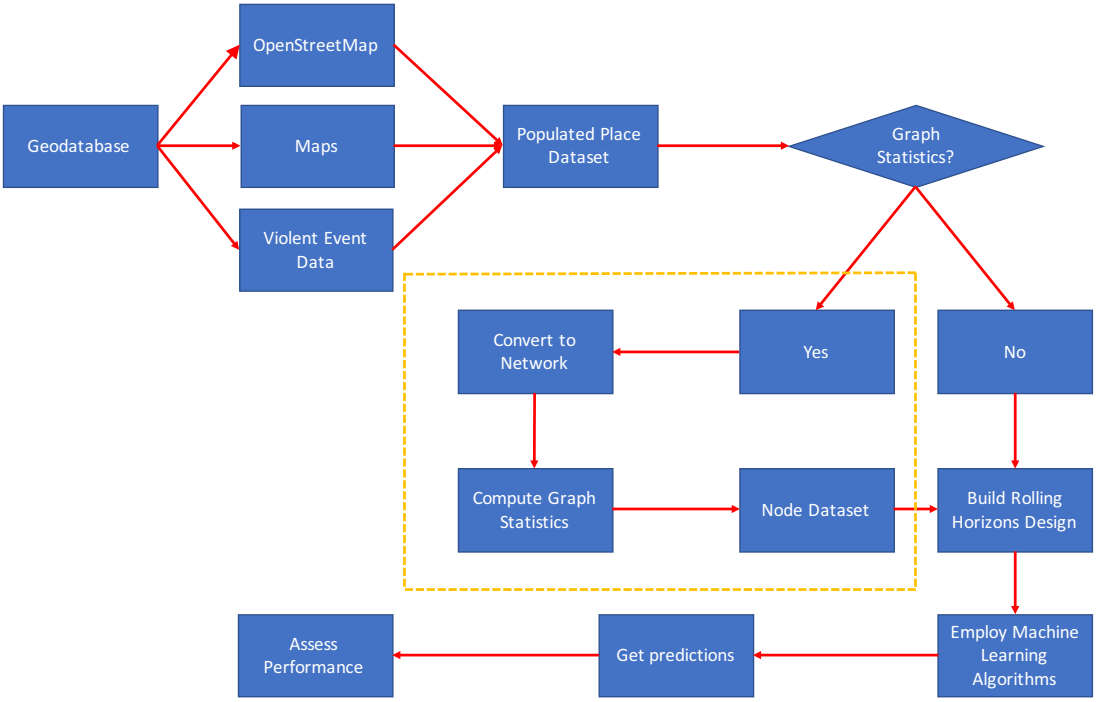


Figure 1. Flowchart of the methodology we introduce

In this thesis, we use four different machine learning algorithms to assess which, if any, performs better at predicting ISIS territorial gains and losses. We also compare each model to the naïve case. We define the naïve case as the case where territory does not change from the previous time period. We use the naïve case as the baseline result that a model must beat to be considered predictive. Our results, as shown in Figure 2, demonstrate that the Adaboost algorithm provides superior results when compared to other algorithms and the naïve case with a cumulative accuracy of 95.48%. This indicates that the Adaboost model is predictive of ISIS territorial gains and losses.

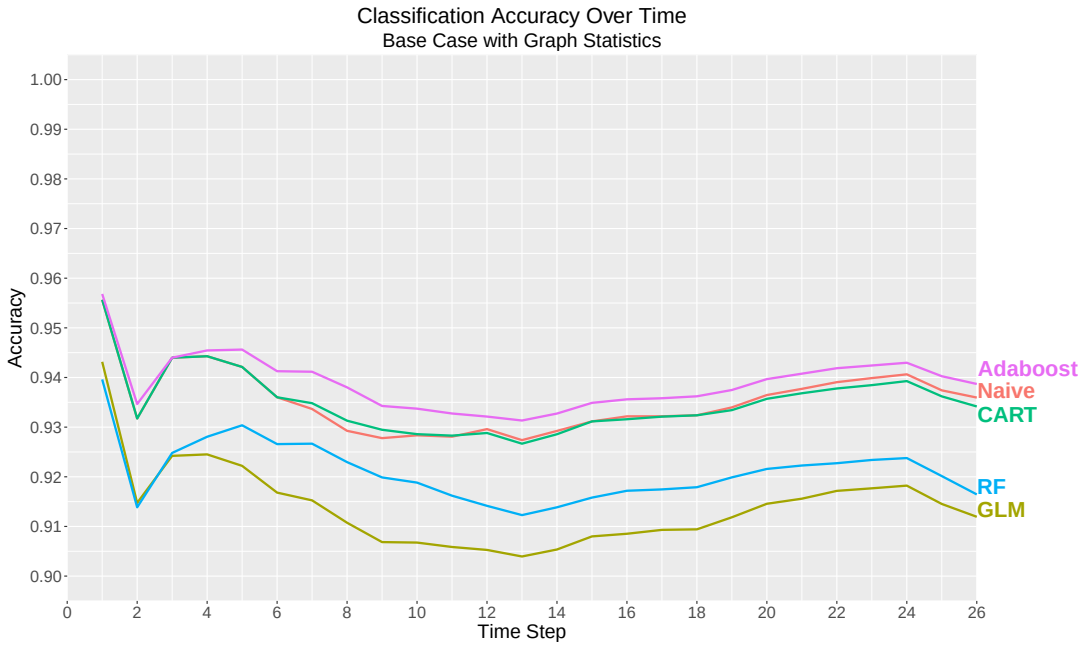


Figure 2. Base case performance

While we find that the predictive accuracy of a model with graph statistics is similar to the predictive accuracy of a model without graph statistics in a binomial classification setting, in a multinomial classification setting we see that a model with graph statistics performs better than a model without graph statistics. Additionally in the multinomial setting, when the Adaboost model asserts a change in territory, we find the assertion to be correct 54.84% of the time throughout the course of our design. This finding confirms the value of adding graph statistics to this problem. Furthermore, we also see that when graph statistics are added as features to the data set, they have greater influence on the models' predictions compared to the non graph statistics. All of this demonstrates that graph features, derived from a graph of Iraq and Syria, can both help to describe ISIS territorials gains and losses and improve upon our ability to accurately predict the ISIS territorial gains and losses.

This thesis shows the merit of our methodology and how it can be applied to predict what territory a group may hold in a future state. Furthermore, this methodology can also be applied in theater to provide military decision makers an improved understanding of the battlespace. Once military operations researchers and data scientists build and run machine learning models using our design, intelligence analysts can leverage the machine learning results to focus their research on specific areas based on the machine's predictions. While

also leveraging their own knowledge, these results may allow the analyst, and intelligence section as a whole, to develop an understanding of the battlespace that is more accurate and in a shorter period of time. Furthermore, the use of machine learning models provides insight into why certain predictions were made. This information is invaluable to the intelligence analyst as they are developing their understanding of the battlespace.

The combination of the experience and expertise of the human analyst with the accurate predictions made by the machine creates a human-machine team that can perform better than either entity by themselves. This team can deliver a superior understanding of the battlespace that will allow U.S. forces to quickly generate and exploit a superior tempo when countering organizations like ISIS. The successes of this methodology, as demonstrated in this thesis, provides an example of how developing a human-machine team can increase our accuracy, efficiency, and productivity while directly improving our ability to fight and win the wars of the future.

Acknowledgments

I would like to express my gratitude to my family and friends for their support throughout this academic endeavor.

I owe a special thanks to my future wife, Anna, for her endless patience and constant love and support throughout this journey.

I would also like to acknowledge all of my fellow classmates in my time at NPS. I learned more from them throughout this experience than any textbook and will take those lessons with me throughout my life.

Finally, I am indebted to the faculty of the Operations Research department at NPS for all that I have learned at NPS, specifically, my two thesis advisors, LTC Sam Huddleston and Professor Jesse Hammond. Their knowledge, passion, patience, and guidance has facilitated my successes at NPS.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

1.1 Background

The Islamic State of Iraq and Syria (ISIS), also known as the Islamic State, is a Sunni Islamic terrorist group that succeeded al-Qaeda in Iraq (AQI). Abu Bakr al-Baghdadi established ISIS in 2013 with the goal of establishing a worldwide caliphate (Sprusansky 2014). In January of 2014, ISIS took control of Fallujah and Ramadi in Iraq and Raqqa in Syria, marking the beginning of their territorial conquest in the region (Glenn 2017). By the summer of 2014, ISIS captured Mosul and Tikrit, establishing their caliphate (Cronin 2015). This rapid acquisition of territory gave ISIS a lot of visibility and publicity while increasing tensions in the Middle East, and directly affecting the U.S. military and diplomatic strategies in the region.

In response, President Barack Obama outlined the American strategy in a statement to the American people in September of 2014. "Our objective is clear: We will degrade, and ultimately destroy [ISIS] through a comprehensive and sustained counterterrorism strategy." His speech showed that he planned to use U.S. military forces and the Intelligence Community (IC), combined with coalition forces, to systematically target ISIS and take back the territory that they had gained (Obama 2014).

Since the beginning of Operation INHERENT RESOLVE (OIR) in 2014, U.S. supported forces in Iraq and Syria have successfully retaken Fallujah, Ramadi, Mosul, and Raqqa. In total, ISIS has lost 95% of its territory and Iraqi Prime Minister Haider al Abadi has declared victory over ISIS (Glenn 2017). However, ISIS remains a global threat.

1.2 Motivation

U.S. Military Forces and the IC have expended vast resources tracking the spread of and combating ISIS forces since 2014 (Frantzman 2017). Considering ISIS' ability to acquire new territory, we see an inherent need to provide the ability to predict these kinds of territorial expansions that ISIS and other terrorist organizations aim to achieve. Given that,

"geographic research has shifted from a data-scarce to a data-rich environment" (Miller 2015), we propose that analysts in the military and IC use social, demographic, and geographic data to predict how territorial control of terrorist organizations will evolve over time. These predictions can provide insight to the forces that drive territorial expansion of terrorist groups. This insight should improve our ability to employ military forces and intelligence assets to combat terrorism.

1.3 Objective

This thesis seeks to predict the territorial gains and losses of ISIS during the period 2014-2017. More broadly, we seek to identify analytical methods that can be used to accurately predict territorial gains and losses of armed groups in general. Additionally, we seek to understand the impact of various predictors, or features, on our ability to predict the actions of armed groups.

1.4 Approach

To achieve our objective, we use an approach that first converts the social, demographic, geographic, and terrorism data that we already have into a graph. We assign this data as attributes to the nodes and arcs of the graph and then compute node level graph statistics creating additional node attributes. We then use this node level dataset and various machine learning algorithms to predict the territorial gains and losses of ISIS in the specified timeframe. Finally, we assess our performance compared to the truth and determine what node level attributes influence the ability to accurately predict ISIS territorial gains and losses.

1.5 Research Questions

This thesis aims to answer the following research questions:

1. Can we use machine learning methods to predict the territorial gains and losses of ISIS during the period 2014-2017?
2. Do graph statistics, derived from representing Iraq and Syria as a graph, improve our ability to predict ISIS territorial gains and losses?

3. What predictors, or features, are most important for predicting the territorial gains and losses of ISIS during the period 2014-2017?

1.6 Scope

The scope of this thesis is limited to ISIS in Iraq and Syria between 2014 and 2017. We also only use open source data for our analysis. We understand that there may be some limitations with only covering a small timeframe and using publicly available data. However, if our approach proves worthwhile, it can be applied using classified data sets. Also, this methodology could be used to study other terrorist groups in other parts of the world.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Literature Review and Background Information

The purpose of this chapter is to orient the reader on the relevance of this thesis and provide a background on the concepts we use. We describe the importance of this research by discussing its role in military decision making, address similar research that has already been done, and state the gaps in the literature that this thesis fills. Then, we provide a broad overview on spatial analysis, graphs, and machine learning, as they are the disciplines we use in this thesis.

2.1 Relevance

In the context of military operations, there is an inherent need to understand where the enemy presently is and where they intend to go in the future. Intelligence drives the development of this understanding and is critical to military decision making. Commanders use intelligence to understand the Operational Environment (OE) and provide focus and direction to operations. Furthermore, geospatial intelligence has significant influence on the ability to understand the OE. Advances in technology and increasing amounts of geospatial data allow for unique visualizations and analysis to support the understanding of the OE (Joint Chiefs of Staff (JCS) 2017).

Joint Intelligence Preparation of the Operational Environment (JIPOE) is the process used in joint intelligence operations to produce intelligence in support of the joint force. This process, as outlined in Joint Publication 2-01.3, has four steps:

1. Define the OE
2. Describe the impact of the OE
3. Evaluate the adversary and other relevant actors
4. Determine the Course of Action (COA) for adversary and other relevant actors (JCS 2014)

By conducting JIPOE, joint intelligence organizations can provide military decision makers with an improved understanding of the battlespace and how it is likely to evolve:

> A holistic understanding of all relevant components within the OE helps the [Joint Force Commander (JFC)] to know how the OE constrains or shapes options, how the OE affects capabilities, and how friendly, adversary, and neutral actors' actions affect or shape the conflict. Of greatest significance, understanding relevant aspects of the OE enables the JFC to leverage aspects of the OE to achieve the objectives and attain the desired end states of the operation. (JCS 2014)

With a goal of providing a better understanding of the OE, our research aims to improve the ability of the joint intelligence organizations to complete the four steps of JIPOE and ultimately support a commander's decision making.

Moreover, Secretary of Defense James Mattis states in the United States National Defense Strategy that advances in computing complemented with "big data" analytics will "ensure that we are able to fight and win wars of the future" (Mattis 2018). By using geospatial data and machine learning algorithms to make predictions about the OE, this analysis directly supports the National Defense Strategy and joint intelligence operations.

2.2 Previous Studies

There has been a significant amount of academic research conducted on the location and spread of terrorism and political violence. This research provides a foundation for this thesis, and we have split this research into four different categories. The first discusses factors that lead to violence. The second category examines how and why violence may spread through a geographic region. The third category reviews the importance of controlling territory. The final category encompasses the prediction of future violent or terrorist events.

2.2.1 Factors that Lead to Violence

Recent research has focused on a variety of factors that can lead to violence. These factors are broken down into economic, geographic, and demographic factors. Understanding these factors is vital to the process of defining the OE.

The first set of factors deals with the economic status of a location. Research has shown that violence is more frequent in areas that have a higher economic status (Hegre et al. 2009) as violent groups are incentivized to target rich locations where violent actions will pay off. On the other end of the spectrum, poor economic conditions has been shown to increase

the likelihood of terrorist events occurring in the area (Nemeth et al. 2014) as the cost of violent actions may be lower. These competing views are likely a result of the violent group itself. If the violent group is well established, it may be more likely for them to target a richer location. Whereas, if the violent group needs to become established, they may target a poor location first to build a following. Nonetheless, we see that economic factors play a role in both the frequency and likelihood of violence in an area.

Geographic factors play a role in violence as well. Much of the current literature discussing geographic factors focuses on a location's proximity to borders, capitals, infrastructure and resources. Border regions and locations far from a national capital are more likely to experience territorial conflict (Buhaug and Rod 2006). However, locations that are close to a national capital have an increased likelihood of experiencing terrorism (Nemeth et al. 2014). Additionally, locations in close proximity to highways have been shown to experience an increased number of violent events (O'Loughlin and Witmer 2011). Similarly, locations in close proximity to natural resources are more likely to experience violence as well (Buhaug and Rod 2006). Furthermore, different types of terrain have been shown to play a factor in the occurrence of violence (Buhaug and Rod 2006, Nemeth et al. 2014, O'Loughlin and Witmer 2011), but the type of terrain is largely dependent on the region being studied.

Lastly, demographic attributes of a location also play a role in the likelihood and frequency of violent events. We find that the population size of a location can affect the likelihood of violence in that location (Buhaug and Rod 2006, Raleigh and Hegre 2009) and population density likewise affects this likelihood (Nemeth et al. 2014). Population density has also been linked to an increased number of violent events (Medina et al. 2011). Other research has found that ethnic minorities in areas that are largely dominated by a different ethnicity increases the likelihood of violence in an area (Di Salvatore 2016).

2.2.2 Diffusion and the Spread of Violence

Another segment of conflict research has focused on how political violence spreads during civil conflict and terror campaigns. If we understand how violence, or organizations that intend to commit violence, will spread, then we can improve the development of adversary COAs within the JIPOE.

Research has shown that violence in civil wars tends to diffuse, or expand, from the

original conflict site opposed to an actual relocation of the violence that may be exhibited in conventional maneuver warfare (Schutte and Weidmann 2011). This fact is important to consider when planning operations in different wartime environments. Other research within this area has examined the role of road networks in the spread of violence. It can be shown that the relocation of insurgent activity, via the road network at a local level, is more likely than diffusion. Additionally, road network and logistical constraints can factor into the ability for an insurgency to spread (Zhukov 2012). In the context of this thesis, we see that ISIS initially exhibited a diffusion of violence as they gained additional territory while generally maintaining the territory they already captured. However, when U.S. and coalition forces confronted ISIS in Iraq and Syria, they showed a relocation of activity to other strongholds that were not under attack.

2.2.3 Importance of Territory

Much of the historical conflict literature discusses the importance of territory in the development of insurgencies. The capture and control of territory is imperative for an insurgency to be able to build its strength and can determine the success or failure of an insurgency (McColl 1969). Once controlled, the territory provides "a military base, a source of recruitment, food, supplies, and revenue from which needs for arms, ammunition, influence and personal enrichment [can] be satisfied" (Hills 1997).

With this knowledge, it follows that we need to understand what territory is important. That is, what territory does a group seek to control. Using network analysis, measures of importance can be developed to determine what territory is more important in a geographic region. Network attributes like efficiency (Latora and Marchiori 2004) can allow us to find what locations are critical, and centrality measures can quantify the strategic importance of an area (Hammond 2018). These critical locations can then help us better define the OE and develop additional adversary COAs.

2.2.4 Prediction of Violent or Terrorist Events

The final category of reviewed research revolves around spatial predictions of violent or terrorist events. There are a variety of methods that can be used to make these predictions. Spatial forecasting and machine learning have beaten naïve approaches at predicting terrorist events (Brown et al. 2004, Ding et al. 2017). Network analysis can be used to predict target

locations of violence, which can lead to direct improvements in defense priorities (Ferenc 2008). Models have also been built to learn the crime site selection preferences of an organization and identify locations that have high probabilities of selection in the near future (Huddleston and Brown 2009). It is crucial for a joint intelligence organization to have the capabilities to make predictions about violent events in the future, and research shows that there are data driven approaches to making these predictions that can improve military decision making.

2.3 Gaps in the Literature

This research provides us with a significant amount of information about the study of violence and methods we can use to improve our ability to understand the OE and our potential adversaries. However, a combination of this research and methods is not evident in current literature. Combining the spatial analysis of economic, geographic, and demographic factors with network analysis allows for the determination of important territory. Machine learning techniques can then be employed to learn how an adversary operates in an area. Finally, with these models, predictions about the state of the region in future periods can be made and can directly inform and improve JIPOE processes allowing superior understanding of the environments that a joint force operates in.

This thesis intends to fill this gap in the literature and will provide products and methodologies that are of great value to the Department of Defense (DoD) and IC. Using ISIS as a test case, we will answer the following research questions:

1. Can we use machine learning methods to predict the territorial gains and losses of ISIS during the period 2014-2017?
2. Do network statistics, derived from representing Iraq and Syria as a graph, improve our ability to predict ISIS territorial gains and losses?
3. What predictors, or features, are most important for predicting the territorial gains and losses of ISIS during the period 2014-2017?

2.4 Spatial Analysis

This section provides a background on spatial analysis, Geospatial Information Systems (GIS) and associated concepts used in this thesis.

2.4.1 Overview

Spatial analysis uses geographic information in order to better understand the processes that drive spatial observations. Generally speaking, spatial analysis is broken down into four main types:

1. Reducing large data sets using summary statistics and visualizations to allow a more simplified understanding of a spatial observation.
2. Developing hypotheses on spatial processes by using exploratory data analysis and visual displays of spatial data.
3. Examining randomness in observed spatial patterns and using statistical models to test hypotheses on the observed patterns.
4. Techniques that involve the mathematical modeling and prediction of spatial processes (Fotheringham and Rogerson 2009).

This thesis will primarily focus on the first and fourth type of spatial analysis, but will touch on all four.

2.4.2 Geographic Information Systems

GIS are software tools that provide modeling techniques for spatial data that allow analysts the ability to visualize and explain spatial relationships. Using GIS to conduct spatial analysis is also called geospatial analysis (de Smith et al. 2009). GIS uses different types of spatial data models to store and display data in the system. We primarily use vector data models in this thesis. Vector data models use sets of coordinates and associated attributes to define objects such as points, lines, and polygons. For example, the location of populated places can be modeled as a point vector, the boundaries of a country can be modeled as a line vector, and the region of territory that a group controls can be modeled as a polygon vector. Figure 2.1 depicts the three kinds of vector data models.

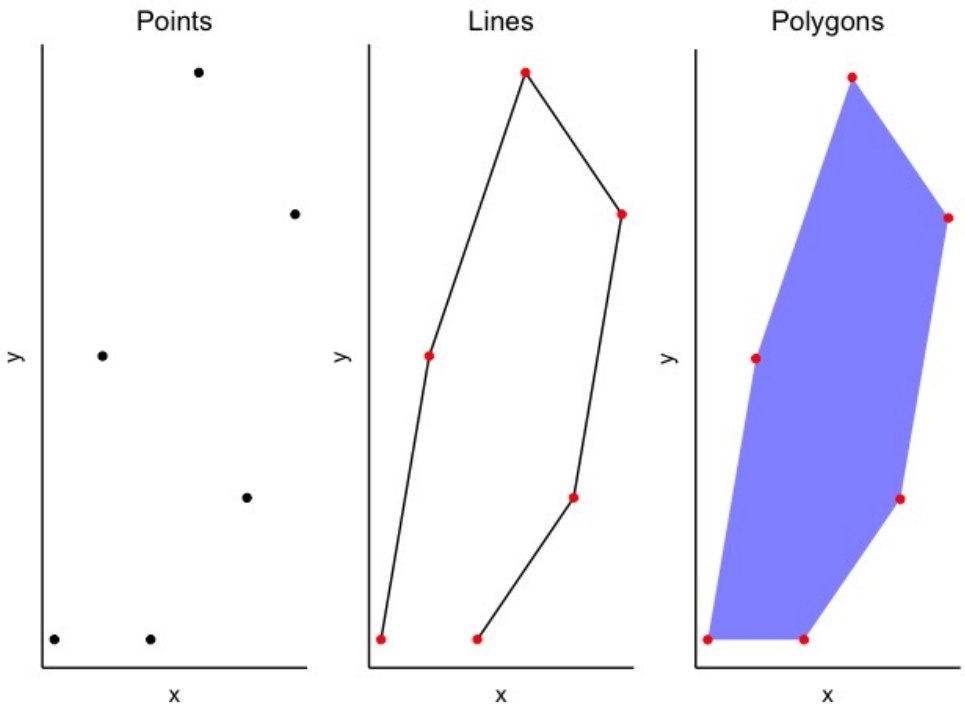


Figure 2.1. Types of vector data models

We also use raster data models which define a set of cells in a grid-like pattern. They are used to represent continuous spatial features such as elevation and land cover. Raster data models can also be used to georeference, or convert, images and maps into formats that can be used in a GIS. These models also come in point, line and polygon formats. Figure 2.2 shows the types of raster data models.

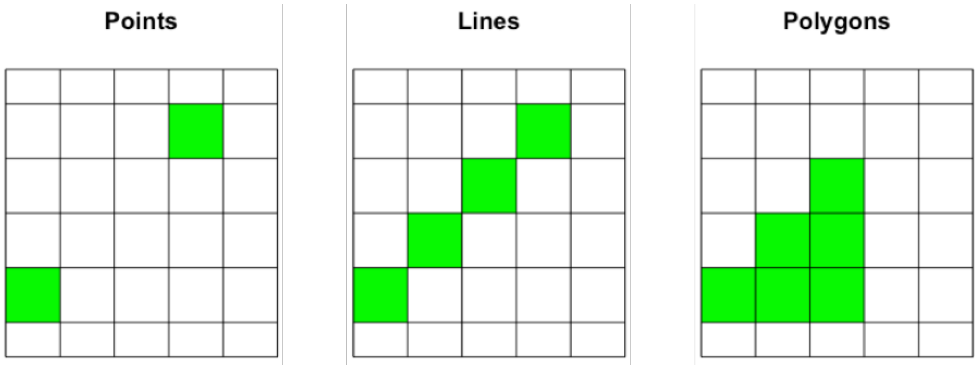


Figure 2.2. Types of raster data models

In this thesis, we use Quantum GIS (QGIS) (QGIS Development Team 2017) and the

Geographic Resources Analysis Support System (GRASS) (GRASS Development Team 2017) to conduct our geospatial analysis tasks. Both systems offer a large array of functions that can be used to analyze spatial data. In the next section, we define some of the main GIS related concepts that we use in our analysis.

2.4.3 GIS Definitions

This section provides definitions for GIS functions that are applied and discussed in this thesis. All definitions are sourced from the Environmental Systems Research Institute (ESRI) (ESRI Technical Support 2018).

Coordinate Reference System (CRS)

A reference system that defines the positions of points, lines, and/or shapes in space in either two or three dimensions (ESRI Technical Support 2018).

Geographic Coordinate System

A reference system that uses latitude and longitude to define the locations of points on the surfaces of a sphere. Figure 2.3 shows a geographic coordinate system.

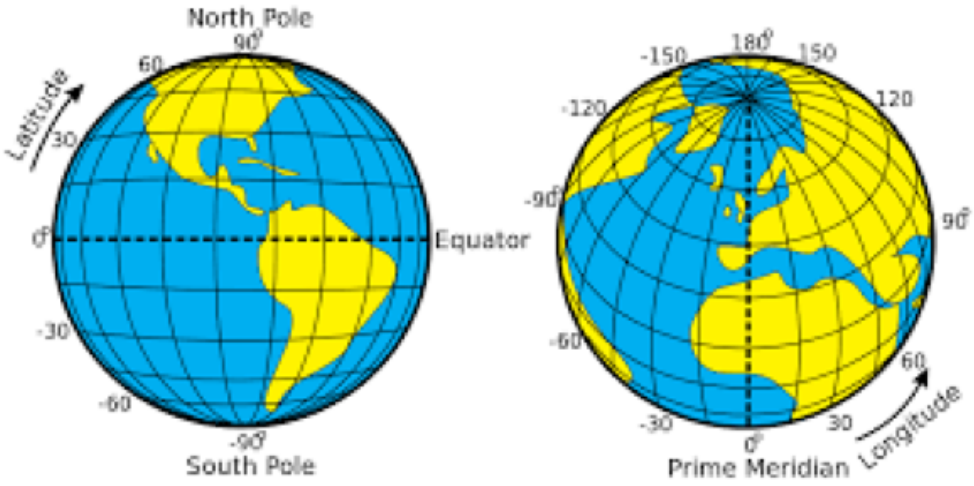


Figure 2.3. Geographic coordinate system. Source: Djexplo (2018).

Projected Coordinate System

A projected coordinate system is a reference system that projects maps of the earth's surface onto a two-dimensional coordinate plane. Figure 2.4 shows an example of a projected coordinate system.

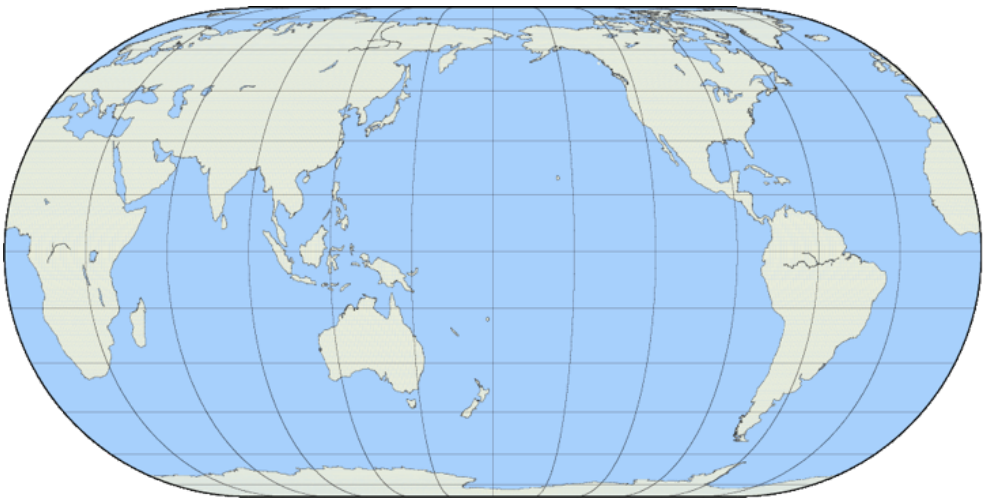


Figure 2.4. Projected coordinate system. Source: Reisio (2018).

Shapefile

Shapefiles are vector data models that represent the location, shape, and attributes of various geographic features. A shapefile is generally stored in a set that contains one feature class (ESRI Technical Support 2018).

Geodatabase

A geodatabase is used to store geographic data. Geodatabases store geometry, coordinate reference systems, and attributes for various types of vector and raster data formats.

Georeferencing

The process of georeferencing includes aligning, shifting, rotating, and scaling geographic data, or images, to a coordinate system so it can analyzed with other geographic data from the same coordinate system (ESRI Technical Support 2018).

2.5 Graphs

This section first provides a brief overview of the basic concepts associated with graphs and then defines some graph-based features that we use in this thesis.

2.5.1 Overview

A graph, G, consists of a set of nodes, N, and a set of arcs, A, whose elements are ordered pairs of distinct nodes (Ahuja et al. 2013). Figure 2.5 represents a graph where the nodes are numbered 1 through 6 and arcs connect the individual nodes.

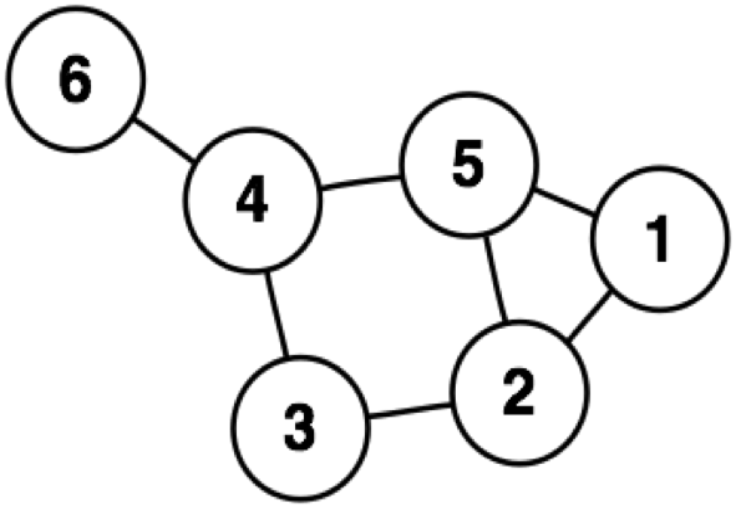


Figure 2.5. Example of a graph. Source: Remahl (2005).

Graphs have a wide range of applications within spatial analysis. In our case, the use of graphs is natural, as the towns in Iraq and Syria can be represented as nodes, and the road network in the two countries can be represented as arcs connecting the nodes.

2.5.2 Graph Features

The utility of a graph, in the context of our research, lies in the characterizations that can be made about the structure of a graph (Kolaczyk 2014). The characterizations, or features, can then be used to answer questions about the components of a graph. The features we focus on in this thesis are centrality and first order neighbors.

Centrality

The concept of centrality is designed to quantify the importance of a node in a graph (Kolaczyk 2009). In the case of predicting the spread of ISIS, we have a definitive interest in the importance of a node in the graph and how that contributes to our prediction ability. There are many centrality measures that can be computed to assess different kinds of importance. The most common measure of centrality is degree centrality, which is defined as the number of connections a node has. In a geographic setting, nodes with high degree centrality are those that are incident to many roads and may act as a hub in a geographic area of interest (Hammond 2018). Kolaczyk breaks down the other various centrality measures into three groups; closeness, betweenness, and eigenvector centralities (Kolaczyk 2009). These measures of centrality are explained in more detail in Section 3.1.2.

First Order Neighbors

A node is a first order neighbor to another node if they are connected by the same arc (Csardi and Nepusz 2006). That is, it takes one step to get from a node to a first order neighbor node. Thus, a node's first order neighbors are all of the nodes that are within one step from the node of interest. If we look back at Figure 2.5, we see that the first order neighbors of node 1 are nodes 2 and 5. In the context of this thesis, we hypothesize that the social, demographic, and geographic features of a node's neighbors are likely to influence, or be related to, the individual node's features. Furthermore, if a node's first order neighbors are inside of ISIS territory, it is likely that the node will lie in ISIS territory and vice versa.

2.6 Machine Learning

This section provides an overview of machine learning with a focus on supervised learning and classification problems. We then briefly describe the machine learning algorithms we use in this research as well as our model design approach. This section concludes with a description of some measures of performance associated with machine learning models.

2.6.1 Overview

Machine learning is the use of algorithms that learn from provided data to make predictions on new or future instances of data (Kohavi and Provost 1998). Machine learning algorithms

are broken down into three classes: supervised learning, unsupervised learning, and reinforcement learning (Huddleston and Brown 2018). This thesis uses supervised learning, which applies when the data has a definitive response variable. A response variable represents the outcome that we seek to successfully predict. The goal of supervised learning is to use data to predict the response variable of new data that may be encountered.

Furthermore, supervised learning is generally broken down into regression problems and classification problems. This thesis solves a classification problem as we are trying to classify the territorial location of each node in the network. The response variable is whether or not the node is in ISIS territory. Since there are a finite number of levels in the response variable, our response variable is categorical. The goal of a classification problem is to predict the class of a new, or future, observation.

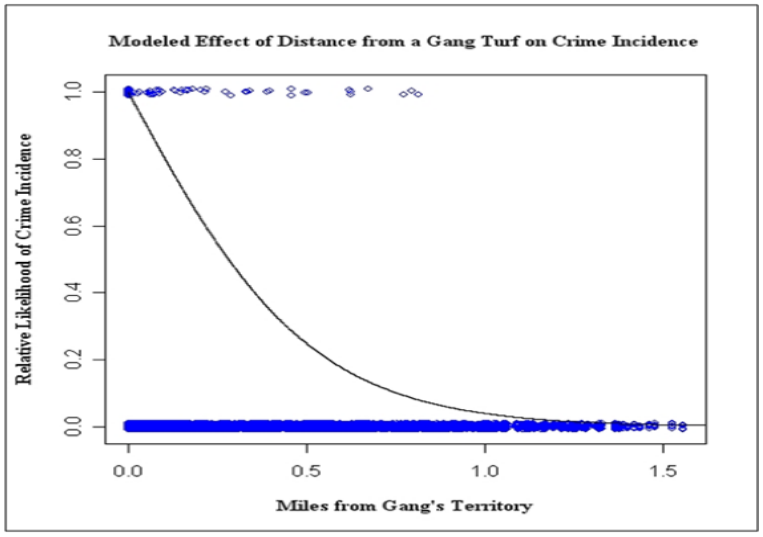
Machine learning models are trained from a set of training data, and the success of a model is generally determined by its accuracy predicting on a test data set. We employ a variety of algorithms and methods to predict the territorial gains and losses of ISIS. The next section provides a brief explanation of these methods.

2.6.2 Machine Learning Algorithms

This section briefly explains the fundamental concepts associated with logistic regression, classification trees and various ensemble methods we use to build machine learning models in this thesis.

Logistic Regression

Logistic regression is a type of Generalized Linear Model (GLM). These models can handle both numeric and non-numeric response variables such as categorical and binary response variables (Faraway 2016). Figure 2.6 provides an example of a relationship described with logistic regression model that shows how the probability of a crime decreases as you move farther from a gang's territory. Of note, as seen in the Figure 2.6, logistic regression can portray non-linear relationships between features and the response variables.



This figure shows the relationship between miles from a gang territory and the probability of a crime occurring that can be derived from Logistic Regression. The circles on the top of the figure are where crimes occurred and the circles on the bottom are where crimes did not occur. This figure shows how the probability of a crime decreases as you move farther from a gang's territory.

Figure 2.6. An example relationship derived from logistic regression. Source: Huddleston et al. (2012).

Classification Trees

Classification trees are a sub-class of Classification and Regression Tree (CART) models. These models provide high interpretability (Huddleston and Brown 2018). Classification trees partition the data into various subgroups where the resulting partitions, or splits, will be more homogeneous in the classes that are being predicted. Figure 2.7 shows an example of a classification tree predicting whether a passenger on the Titanic would survive. From this, we see that if the sex of an individual is not male, the individual is classified as survived. Additionally, if an individual is a male and older than 9.5 they are classified as dead. The probabilities associated with the classifications are also shown in Figure 2.7.

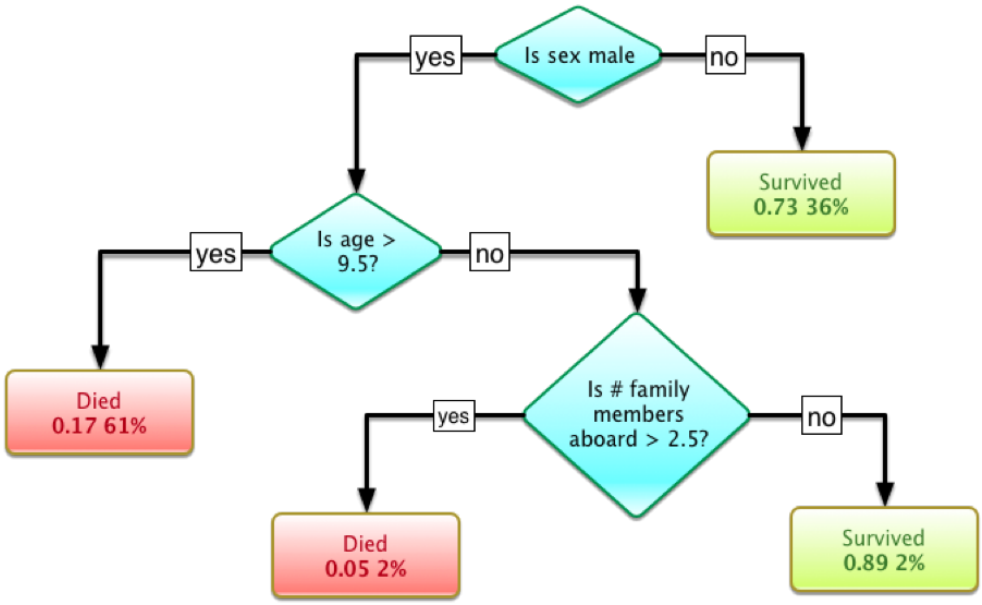


Figure 2.7. Example classification tree predicting the survival of a Titanic passenger. Source: Diary (2017).

Ensemble Methods

Ensemble methods combine, or average, predictions made from several machine learning algorithms. These models will generally have improved predictive performance. However, these models are more difficult to interpret (Huddleston and Brown 2018). In this thesis, we use bootstrap aggregating, or bagging, and boosted models.

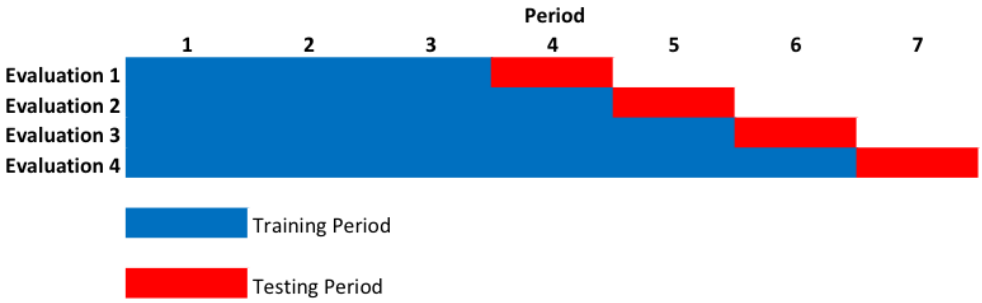
Bagging models repeatedly sample from a training data set to create an ensemble of models trained on different subsets of the training data. Each iteration, a new model is trained on a subset of the data and the classifications from that iteration each get one vote towards the overall classification of each observation in the overall data set. At the conclusion of algorithm and all iterations, each observation is classified based on the response variable with the most votes. The random forest algorithm is a common bagging method that uses many classification trees. It performs the bagging procedure as explained previously. Furthermore, the random forest algorithm, in each iteration, samples from the features that will be used to classify each observation. Thus, in each iteration of this algorithm, a different subset of features is used to make the predictions. (Huddleston and Brown 2018).

Boosted models are designed to build a set of models that sequentially improve performance

from the previous model in the set. These models reduce bias and lead to a better performing and stronger model at the end of the set. (Zhou 2012). At the end of the sequence of models, the predictions are aggregated to form a final prediction (Huddleston and Brown 2018). Adaboost is a common implementation of boosting that we use in this thesis.

2.6.3 Rolling Horizon Design

A rolling horizon design can be used to fit machine learning models when there is a time series component to the data being analyzed. In this design, rather than splitting a data set between training and test sets, observations from previous time periods are used to predict the outcome of the next time period. This process is repeated for each time period in the data set (Huddleston and Brown 2018). Figure 2.8 provides an illustration of a rolling horizon design. In this case, the performance of the model is estimated by taking the average of the performance of periods 4 through 7. Our machine learning approach predicting the territorial gains and losses of ISIS between 2014 and 2017 is implemented using a rolling horizon design.



In this rolling horizon design, evaluation 1 is the first iteration that uses periods 1-3 to make a prediction about period 4. In the next iteration, period 4 becomes a part of the training set. Thus, in evaluation 2, periods 1-4 are used to predict period 5. This kind of design is applied until the last time period in the data.

Figure 2.8. Illustration of a rolling horizon design. Source: Huddleston and Brown (2018).

2.6.4 Measures of Performance

This section briefly explains the primary measures of performance used to assess machine learning algorithms in this thesis.

Classification Accuracy

Performance for classification problems is generally assessed using classification accuracy, or its opposite, misclassification rate. Classification accuracy is calculated by:

$$Accuracy = \frac{\#\ of\ Correct\ Classifications}{Total\ \#\ of\ observations}. \tag{2.1}$$

A model performs well if its classification accuracy is high.

True Positive Rate

Another way to assess a machine learning model is by computing its true positive rate (TPR) for binary classification problems. This value tells you how well the model does in predicting a positive occurrence of the binary response variable. A model performs well if the TPR is close to 1 and is calculated by:

$$\text{TPR} = \frac{True\ Positives}{True\ Positives + False\ Negatives}. \tag{2.2}$$

False Positive Rate

The false positive rate (FPR) tells you the rate of false positives that are classified in your model. This represents another way to measure the performance of a machine learning model. A FPR close to 0 is desired for a good model. FPR is calculated by:

$$\text{FPR} = \frac{False\ Positives}{False\ Positives + True\ Negatives}. \tag{2.3}$$

Variable Importance

While not necessarily a measure of a model's performance, variable importance provides valuable insight into the reasons why a model will classify observations a certain way.

Machine learning algorithms have methods to assess and quantify what variables, or features, are most important to the classification of observations. That is, which variables have the largest impact on the classification of an observation. This information is valuable because it provides a way to relate the outputs of model to a military decision maker. For example, an analyst can describe which features have, in our case, the largest effect on whether or not a populated place will be in or out of ISIS territory. Furthermore, by determining what variables are not important, you can remove them from your model, if desired, which can potentially improve the predictive performance.

Each machine learning algorithm computes variable importance slightly differently. In tree-based machine learning algorithms such as classification trees, random forests, and boosted models, the importance can be computed two ways. The first method computes the mean decrease in accuracy by randomly permuting the variables that are being used to make predictions. The second method computes the decrease in impurity for every split of each variable for each tree. These values are then averaged over the number of trees created in the algorithm (Liaw and Wiener 2002).

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Data and Methodology

The purpose of this chapter is to describe the data and methodology used in this thesis. We first describe the variables we use in our analysis and the data source for these variables. Then, the overall methodology is explained.

3.1 Data and Variables

This section describes the variables and data sources we use to build our machine learning models. We first describe the variable, what it represents, and then discuss the source of data used for that variable.

We break these variables down into non-graph and graph features. The non-graph features consist of geographic, demographic, and social features, while the graph features are computed after building a network. The variables are all at the node-level, and are representative of each populated place in Iraq and Syria.

The data for each variable is compiled from various sources that will be explained in more detail in this section:

National Geospatial Intelligence Agency (NGA) Geodatabase

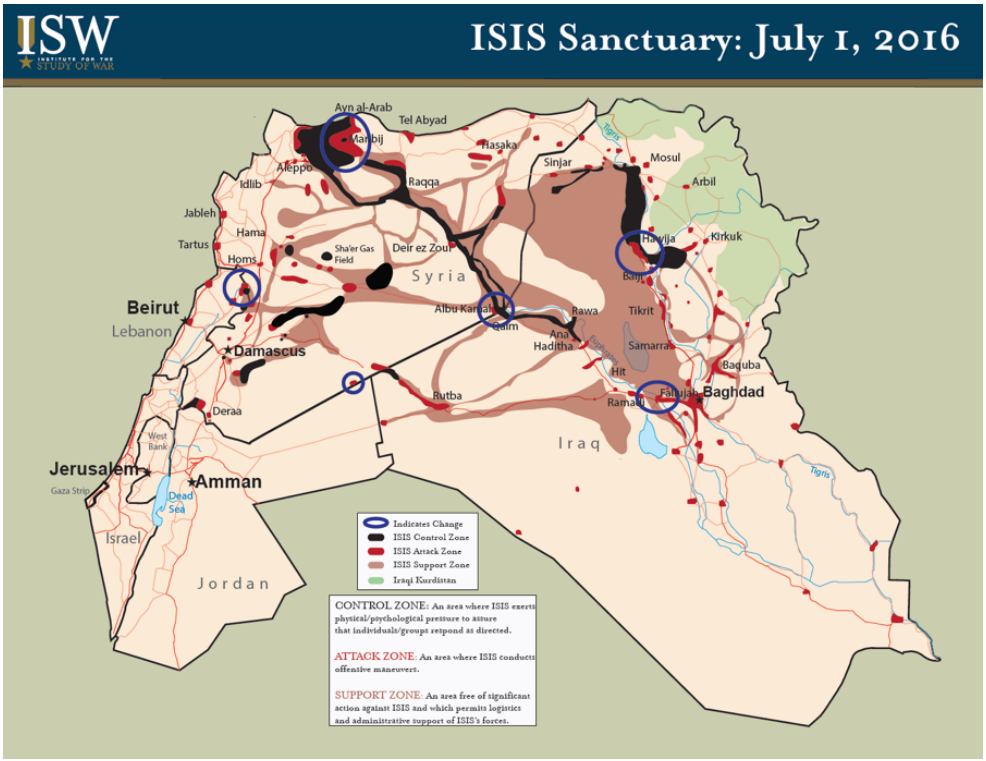
This geodatabase, provided by the NGA, contains shapefiles for the locations of all populated places, with associated demographic data, in Iraq and Syria, and the road network for both countries.

OpenStreetMap (OSM)

OSM is an open-source geospatial data repository that contains data broken down by every country in the world (OpenStreetMap Contributors 2017). We use their data to develop additional variables associated with each populated place in Iraq and Syria.

Institute for the Study of War (ISW)

The ISW developed an ISIS sanctuary map project that tracked ISIS territory in Iraq and Syria from 2014 to 2017 (Forrest 2016). These maps provide a visual representation of ISIS territory in Iraq and Syria at various points between 2014 and 2017. Figure 3.1 shows an example of a sanctuary map produced by ISW. It displays ISIS territory in three different levels: control, attack and support. The control zone is defined as an area where ISIS exerts physical control of an area and is depicted in black. The attack zone shows where ISIS forces conduct offensive operations and is depicted in red. Lastly, the support zone, depicted in brown, represents an area that allows freedom of movement, logistical, and administrative support for ISIS forces.



The ISIS Sanctuary maps developed by ISW show various levels of ISIS territory at time points between 2014 and 2017. This example map shows ISIS territory broken down into a control zone in black, attack zone in red and support zone in brown.

Figure 3.1. ISIS sanctuary map from July 1, 2016. Source: Forrest (2016).

University of Maryland Global Terrorism Database (GTD)
The GTD is an open-source repository of terrorist events around the world from 1970-2016 (National Consortium for the Study of Terrorism and Responses to Terrorism (START) 2016). We use their data to find ISIS events in Iraq and Syria between 2014 and 2016. One drawback of this database is that the data is only available through 2016. This is a noted limitation that we are willing to accept in the context of this thesis as this database contains specific ISIS coded events.

3.1.1 Non-Graph Features

This section describes the non-graph features we use in our analysis.

Feature Designation Code
A two to five-character code identifying the type of feature that is represented (National Geospatial Intelligence Agency GEOnet Names Server (GNS) 2017). The NGA geodatabase contains the feature designation codes for each populated place in Iraq and Syria. This variable describes the type of populated place such as a locality, administrative section, capital, and even a section of a populated place.

Primary Country Code
A two alphabetic character code identifying the country that feature belongs to. The data for this variable is sourced from the NGA geodatabase.

Administration Division
A two alphabetic character code identifying the primary administrative division of the populated place. For example, this code represents a state in the United States. This data is sourced from the NGA geodatabase.

Ethnicity
A factor representing the ethnicity associated with each populated place. The ethnicity for each populated place is sourced from polygon shapefiles in the NGA geodatabase. The levels of this factor are:

- Arab
- Assyrian
- Druze
- Kurd
- Shabak
- Turkomen
- Yezidi

Sect

A factor representing the religious sect associated with each populated place. The religious sect for each populated place is sourced from polygon shapefiles in the NGA geodatabase. The levels of this factor are:

- Alawite
- Assyrian
- Druze
- Kakai
- Shabak
- Shia
- Sunni
- Yezidi

Type of Place

A factor representing the kind of populated place. The possible levels are city, town, village, hamlet, national capital, suburb, farm, island, region, and locality. Each level is defined by the population of the populated place (Ramm 2017), and sourced from OSM.

Population

A numeric variable representing the population of a populated place. A value of zero is used to represent places that do not have population numbers. The data for this variable is sourced from OSM.

Distance to Border
A numeric variable representing the great circle distance from each populated place to the nearest country border in kilometers. This variable is computed using the raster and geosphere packages in R (Hijmans 2017, 2016).

Distance to Capital
A numeric variable representing the great circle distance from each populated place to their national capital in kilometers. The data for this variable is computed using the sp package in R (Bivand et al. 2013).

Number of ISIS Events Within Five Kilometers
A numeric variable representing the count of ISIS events that occur within five kilometers of a populated place. This variable is broken down by the time periods we use in our rolling horizon machine learning design which is discussed in more detail in Section 3.2. We obtain the ISIS event data from the University of Maryland GTD.

Territorial Zone
A factor variable representing the territorial zone that each populated place falls in. The level for this variable are:

- Not in ISIS territory
- Control
- Attack
- Support

This variable is also broken down by time period according the dates of the ISW maps, resulting in 28 features for each populated place representing the zone of that place in each respective time period.

3.1.2 Graph Features

The graph features we develop in this analysis are all computed using the igraph package in R (Csardi and Nepusz 2006) after converting the populated places and road network

into a graph as described in Section 3.2.2. These features will also be referred to as graph statistics.

Kleinburg's Authority Centrality Score
A numeric variable defined as the principal eigenvector of $A^\intercal A$ where A is the adjacency matrix of the graph (Kleinberg 1999).

Betweenness Centrality
A numeric variable that quantifies where a node is located in relation to paths in the network (Kolaczyk 2009). In other words, when traveling around the network, a node with high betweenness centrality will have many paths that travel through the node. In a geographic setting, a node with high betweenness centrality controls access between different areas, because you have to go through this node to get from one place to another (Hammond 2018). For each node, i, betweenness centrality is computed by finding the number of shortest paths, $g_{jk}(i)$, between nodes j and k that travel through node i and dividing by the total number of shortest paths from j to k, g_{jk} :

$$betweenness(i) = \sum_{j \neq i \neq k} \frac{g_{jk}(i)}{g_{jk}}. \tag{3.1}$$

Closeness Centrality
A numeric variable that measures how close a node is to other nodes in the network. It is computed by first calculating, for each node i, how many arcs are needed to reach every other node j and is represented by d_{ij}. Then, by taking the inverse of the average number of steps from each node, i, to all other nodes, j, in N we compute closeness centrality:

$$closeness(i) = \frac{1}{\sum\limits_{j \neq i \in N} d_{ij}}. \tag{3.2}$$

Burt's Constraint
A numeric variable that measures the degree in which a node's neighbors are connected to other nodes that are not connected to one another (Valente and Fujimoto 2010). Burt's constraint is higher if the node has less, or more redundant, neighbors (Burt 2004).

Coreness
A numeric variable that measures the coreness of a node. The coreness of a node is computed by determining what k-core of the graph that each node belongs to. The k-core of a graph is a subgraph where every node has at least degree k (Csardi and Nepusz 2006).

Count of Triangles
A numeric variable that determines how many triangles each node belongs to. A triangle graph, in the most basic case, is where 3 nodes are connected by arcs in the shape of a triangle (Miklavic and Milanic 2011).

Degree Centrality
A numeric variable that is determined by the number of connections a node has. A node has large degree centrality if it is connected to many other nodes.

Node Eccentricity
A numeric variable that represents the shortest path distance from the farthest node in the graph (Csardi and Nepusz 2006). The eccentricity of a node is calculated by determining the shortest path from a node to every other node that it can reach, and taking the maximum, or longest, path (Harary 1994).

Eigenvector Centrality
A numeric variable that measures the influence a node has on a network. Nodes are ranked under the idea that the more connections a node has to other high ranking nodes, the more influence that node has. This is generally expressed as the eigenvector solutions to linear systems of equations (Kolaczyk 2009). To compute these solutions, the graph's adjacency

matrix $A = (a_{ij})$ is used where $a_{ij} = 1$ if there is a connection between node i and j. With λ as a constant, the eigenvector centrality of node i (c_i) can be computed as:

$$c_i = \frac{1}{\lambda} \sum_{j \in N} a_{ij} c_j. \tag{3.3}$$

Average Nearest Neighbor Degree

A numeric variable that is the average degree of each node's set of neighbors. We compute this variable for each node by taking the average degree centrality value for all other nodes that are a part of a node's set of neighbors.

Neighbor Ethnicity

A set of numeric variables of the percentage of a node's neighbors that are coded as a certain ethnicity. We compute these variables for each node by determining the ethnicity of each node's neighbors and then calculate the percentage of neighbors that are of each ethnicity.

Neighbor Religious Sect

A set of numeric variables of the percentage of a nodes neighbors that are coded as being of a certain religious sect. We compute these variables for each node by determining the religious sect of each node's neighbors and then calculate the percentage of neighbors that are of each religious sect.

Neighbor Territorial Zone

A set of numeric variables that provides the percentage of a node's neighbors that fall in a specified ISIS territorial zone. We compute these variables for each node by determining the territorial zone of a node's neighbors for each ISW map. Then we calculate the percentage of neighbors that fall in each of the 4 levels of ISIS territory. Additionally, these variables are computed for every time step we use in our rolling horizon design.

3.2 Methodology

This section describes the methodology we use in this thesis. It discusses the aggregation of data sources, conversion to a graph, computation of graph statistics, and the employment of machine learning algorithms. Figure 3.2 shows a flowchart that visually explains our methodology. The components of the flowchart that lie inside the dashed orange box are our contributions that add to existing approaches.

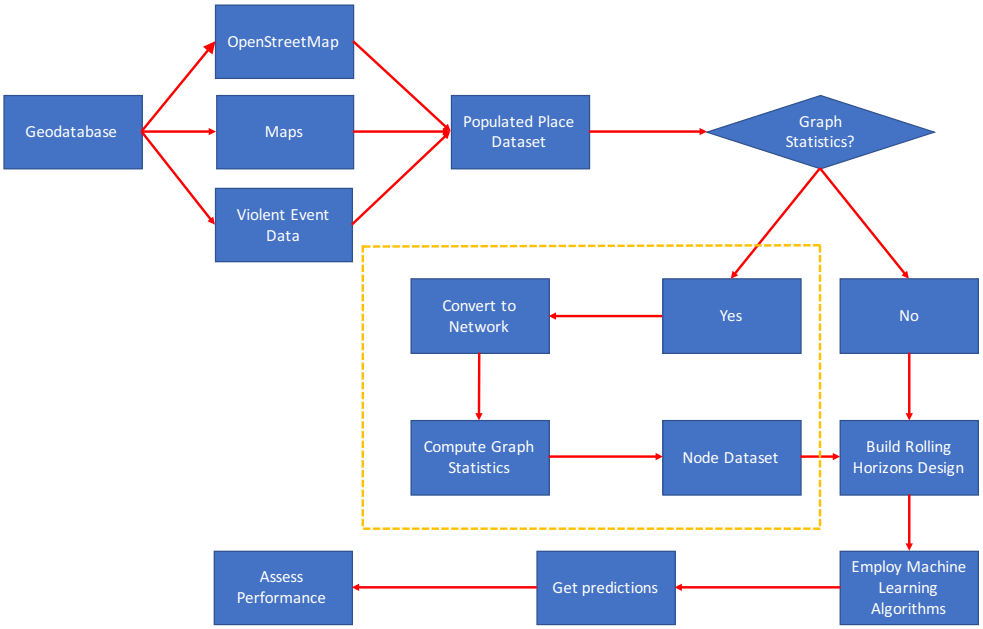
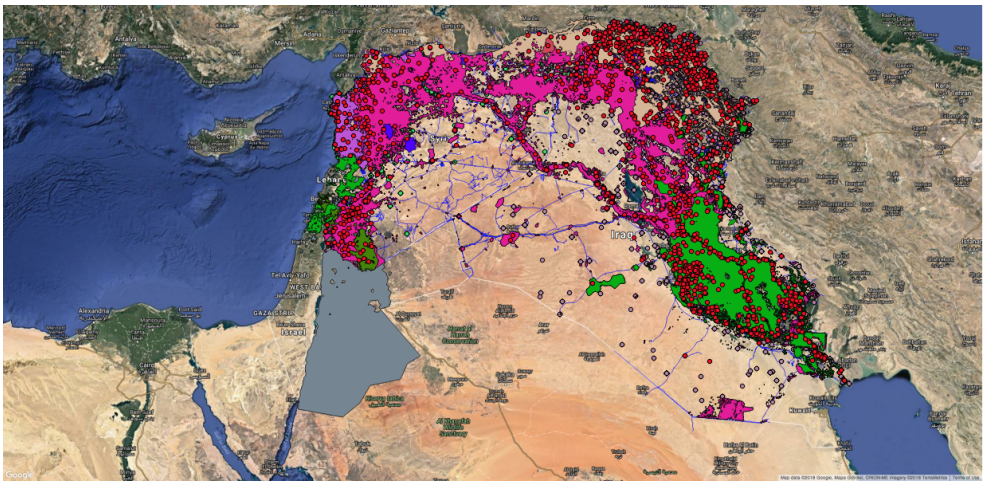


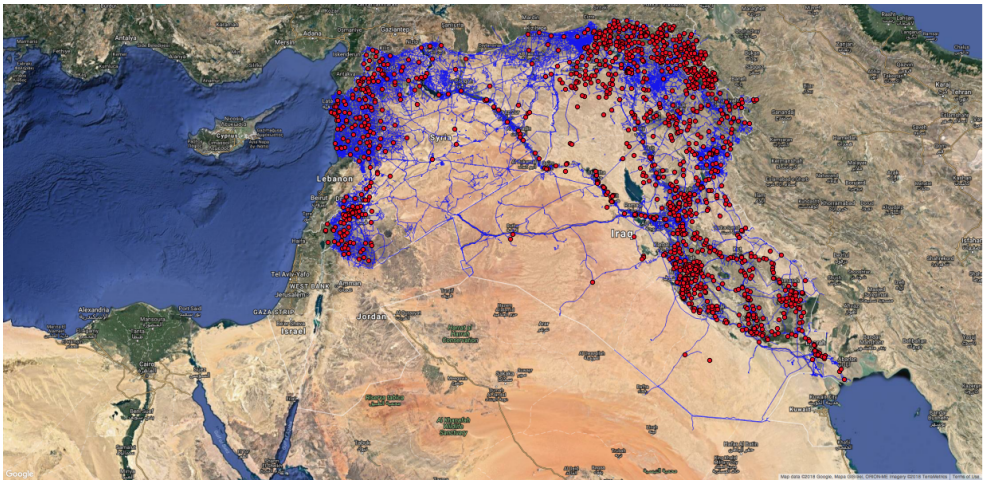
Figure 3.2. Flowchart of methodology.

3.2.1 Data Aggregation

This section describes the steps we take to aggregate the data from our various sources as described previously. The focal point of our analysis is on the populated places within Iraq and Syria. Thus, the geodatabase, which contains point shapefiles of the populated places, is the base data set. We take the religious sect and ethnicity polygon data and, within a GIS, build spatial overlays determining the religion and ethnicity associated with each populated place. Figure 3.3 shows the outcome of conducting spatial overlays, and how this process cleans up our visualization of the data and region.



Before spatial overlay.



After spatial overlay.

These figures show the outcome of a spatial overlay. The top figure shows the religion and ethnicity data polygon data overlayed with the populated places in red. The bottom figure shows the outcome of the spatial overlay where each populated place now maintains the data about what religion and ethnicity it represents.

Figure 3.3. Spatial overlay process

We then use our violent event data from the GTD to compute how many ISIS events occur within five kilometers from each populated place. We also determine the number of casualties associated with each event. This ties the data from the GTD to the populated places in Iraq and Syria.

Next, we use the ISW maps to determine what level of ISIS territory each populated place

is in. To do this, we first georeference the image of a map to Iraq and Syria in the same projected coordinate system. Figure 3.4 shows one of our georeferenced maps.

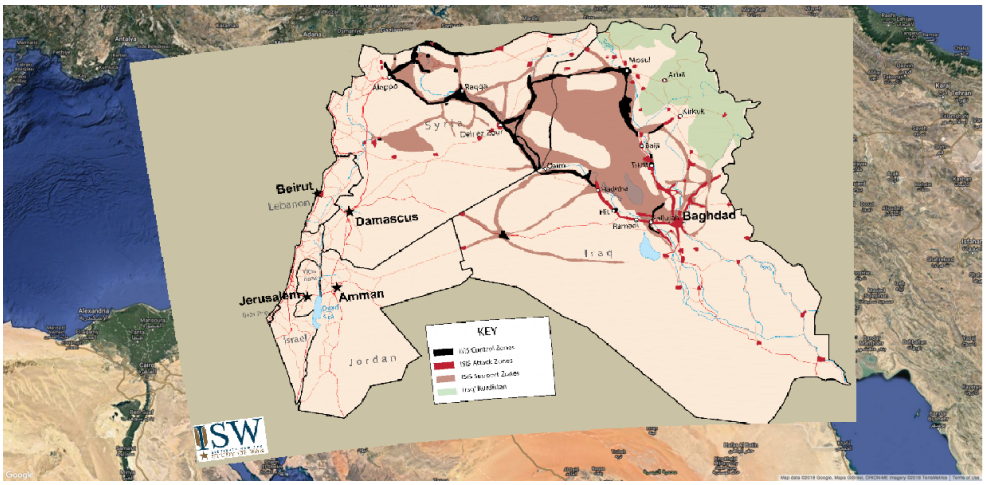


Figure 3.4. Georeferenced ISW map.

After georeferencing a map, we overlay the populated places on the georeferenced map. We then use the pixel color values associated with the levels of ISIS territory to determine what zone each populated place falls inside of. So, if a populated place lies on the pixel color values of a territorial zone, we say that place falls in that territorial zone at that time. Figure 3.5 shows each of the populated places overlayed on a georeferenced map. We conduct this process for each of the ISW maps to develop a territorial history for each populated place in Iraq and Syria.

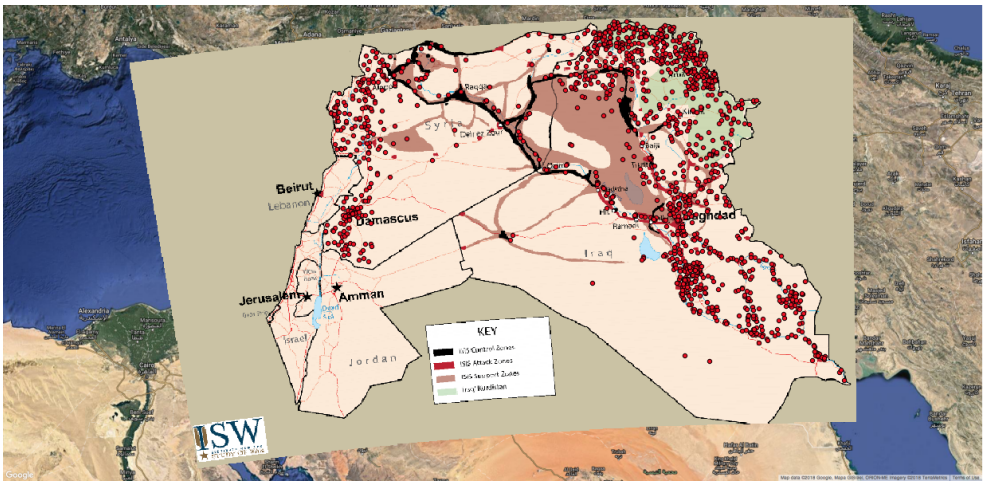


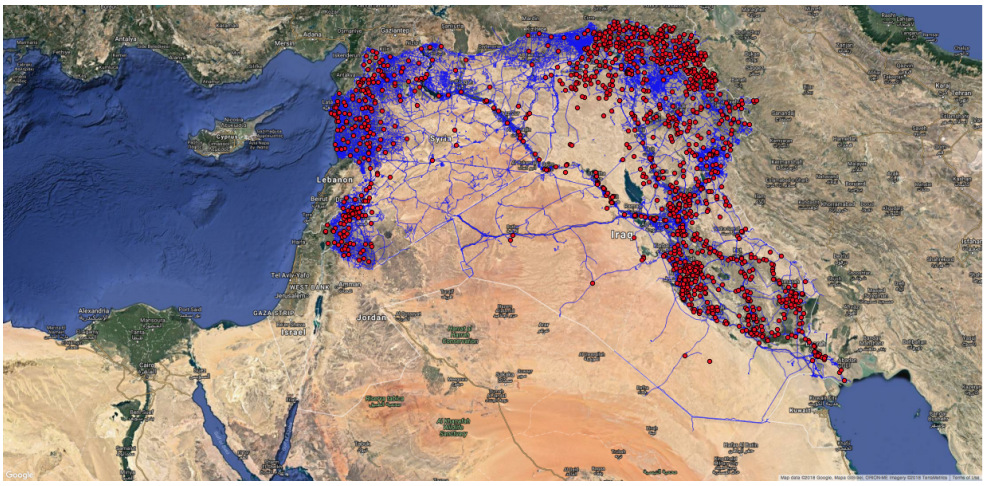
Figure 3.5. Georeferenced ISW map overlayed with the populated places.

Lastly, we compute the distance from each populated place to the nearest border and nearest capital as these distance metrics were shown to influence violence in much of the previous literature. At the conclusion of the data aggregation, each populated place contains the following data about itself:

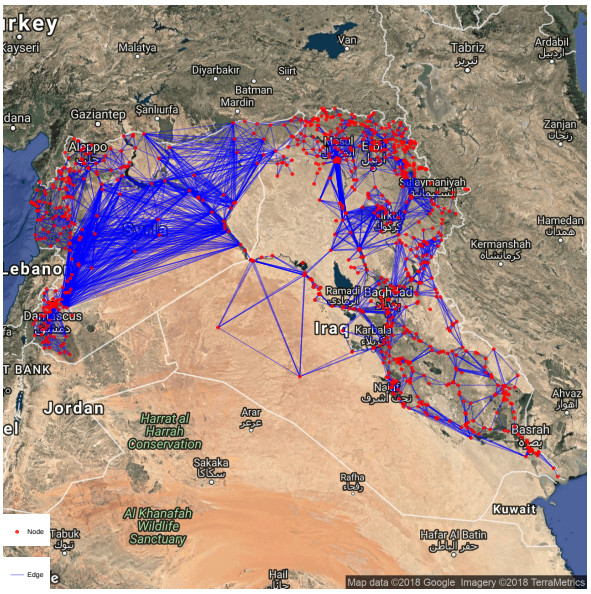
1. Demographic Features
    - Population
    - Ethnicity
    - Religious Sect
2. ISIS Event Data
    - Number of ISIS Events
    - Number of Casualties
3. Proximity Features
    - Distance to nearest border
    - Distance to capital
4. Territorial Zone

3.2.2 Conversion to a Network

In our next step we convert the geographic region of Iraq and Syria into a network of populated places and roads. The populated places are the nodes in the network, and the road network is converted into a series of arcs that connect the nodes. In our network, an arc represents that there is a way to get from one place to another, on the road network, without going through any other populated place. This network allows for the application of a variety of graph and statistical techniques to improve our understanding of the system. Furthermore, each node in the network maintains all of the data about itself as described in the previous section. Figure 3.6 depicts the beginning and end of this conversion. The tutorial located in Appendix A provides a detailed step by step approach for the conversion process (Vanderzee et al. 2017).



Populated places on the road network of Iraq and Syria.



Network of Iraq and Syria.

These figures show the conversion of the populated places and road network of Iraq and Syria into a graph. The top figure shows Iraq and Syria before the conversion process. The bottom figure shows the outcome of the conversion and displays Iraq and Syria as a network of the populated places and roads in the region.

Figure 3.6. Graph conversion

3.2.3 Calculation of Graph Statistics

After building the network of Iraq and Syria, we compute node-level graph statistics as described in Section 3.1.2. These node-level statistics provide additional data about each of the nodes in the network. After these computations, we build our final data set based on the nodes in the network.

3.2.4 Employment of Machine Learning Algorithms

Our machine learning approach consists of three main steps. First, we create a rolling horizon design based upon the dates of the ISW maps. Then, we apply multiple machine learning algorithms to our data and rolling horizon design to make predictions about ISIS territory over time. Lastly, we compute performance metrics, determine the important variables, and compare the overall performances across all algorithms.

We break the classification task into five different cases:

1. **Binary Classification: Base Case**
   In our base case, a node is classified as either being in or out of ISIS territory at each time period in our design. Control, attack, and support zones are aggregated in this case to represent ISIS territory.

2. **Binary Classification: Control**
   In this case, a node is classified as either being in or out of ISIS control territory.

3. **Binary Classification: Attack**
   In this case, a node is classified as either being in or out of ISIS attack territory.

4. **Binary Classification: Support**
   In this case, a node is classified as either being in or out of ISIS support territory.

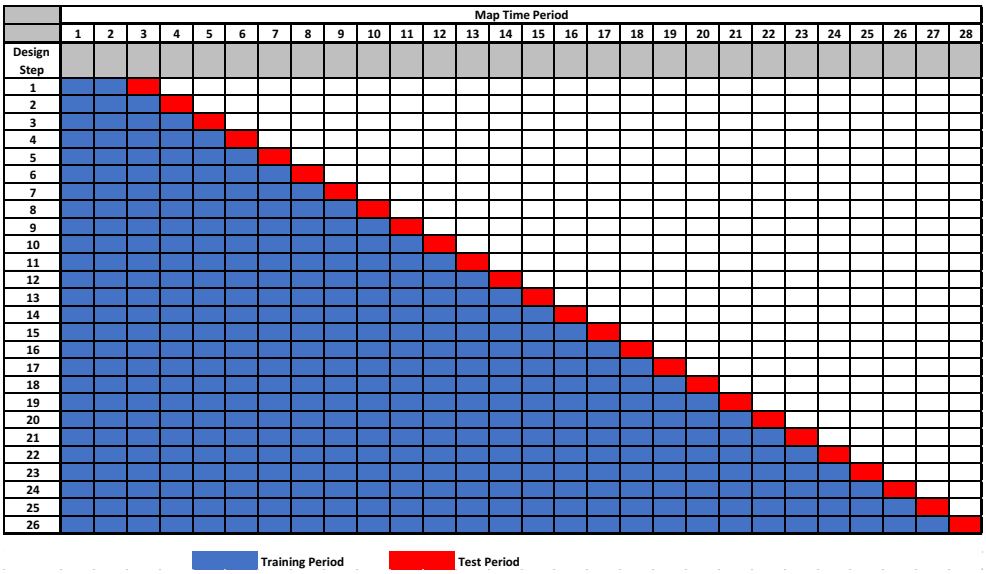
5. **Multinomial Classification**
   Our last case consists of classifying each node at one of four levels of ISIS territory:
   - Not in ISIS territory
   - Control

- Attack
- Support

We also use each these cases both with and without graph statistics. For the remainder of this thesis, these five cases are referred to respectively as base case, control, attack, support, and multinomial case.

Rolling Horizon Design

We build our rolling horizon design using the publishing dates of the ISW maps. Since there are 28 published maps, we break our data into 28 time periods, one for each map. Our rolling horizon design is pictured in Figure 3.7.



This figure depicts the rolling horizon design used for this thesis. The left column (Design Step) represents the step in the rolling horizon design. The top row (Map Time Period) represents the 28 time intervals we have data for from the ISW maps. A blue square represents that time period is part of the training period, and a red square represents that time period is being tested during that iteration. After a time period is tested, it becomes a part of the next training period. This design is performed sequentially until the last time period is tested.

Figure 3.7. Full rolling horizon design.

The first step of our design starts with a training period on time periods one and two with a test on time period three. In the next step, time period three become a part of the training

period and we test on time period four. So, once a time period is tested, it becomes a part of the training period the next step. We perform this process to test on every time period and conclude our design with 26 steps. This design allows our models to make predictions over time while gaining more information with each step in the design.

Machine Learning Algorithms

During every step of our rolling horizon design, a machine learning model is trained and subsequently tested on the next time period. That is, each node is classified by the trained model, according to our five cases, during each step. We use multiple machine learning algorithms for each of our five cases. Furthermore, each case is tested both with and without graph statistics. As a result, each algorithm is used 10 times. The algorithms we elect to use are:

- Logistic Regression
- Classification Trees
- Random Forest
- Adaptive Boosting (AdaBoost)

Measuring Performance

After completing all 26 steps of our rolling horizon design, we assess the performance of the machine learning model in two ways. We first compute the classification accuracy, both by individual iteration and overall. This is determined by comparing the classifications from our machine learning models with the ISW maps. We then compute a change accuracy for each individual iteration and overall. After assessing performance, we determine which variables are the most influential throughout the design.

1. **Classification Accuracy**

   We first compute the classification accuracy for each model and case as shown in Equation 2.1. This is computed by step to see how the accuracy changes in different time periods. We also compute a cumulative classification accuracy to see how the model performs overall throughout the 26 step design. We say that a model performs well if its classification accuracy is high.

2. **Territorial Change Accuracy**

   In a military context, territorial change is important to planning and executing operations. Thus, the subset of nodes where the models predicted a change are of vital importance to those trying to identify changes in the OE. With this in mind, we compute our models' accuracy at predicting these changes as shown in equation 3.4:

$$ChangeAccuracy = \frac{\#\ of\ Correctly\ predicted\ changes}{Total\ \#\ of\ nodes\ predicted\ to\ changed}. \tag{3.4}$$

   We also compute the cumulative change accuracy to see how the models perform at predicting change throughout the 26 step design.

3. **Variable Importance**

   To explain the predictions of the models, we compute the variable importance for our machine learning models at the beginning, middle, and end of our design. We do this to determine what variables, or features, have influence on the territory associated with each populated place. Furthermore, by computing the variable importance at various times in the design, we can see how the important variables change over time.

CHAPTER 4:
Analysis and Results

In this chapter, we examine the results of applying our machine learning approach to the data set we built. We first determine which machine learning model performs the best by comparing performance of the different models across all five cases. Then, using the best performing machine learning model, we determine if graph features, or statistics, improve the performance of the model by comparing the best model, both with and without graph statistics. We then evaluate the model's ability to predict territorial changes that occur throughout our design, examine what variables had the most influence on our results, and develop a way to convert our results into a map depicting predicted ISIS territory.

4.1 Best Performing Machine Learning Algorithm

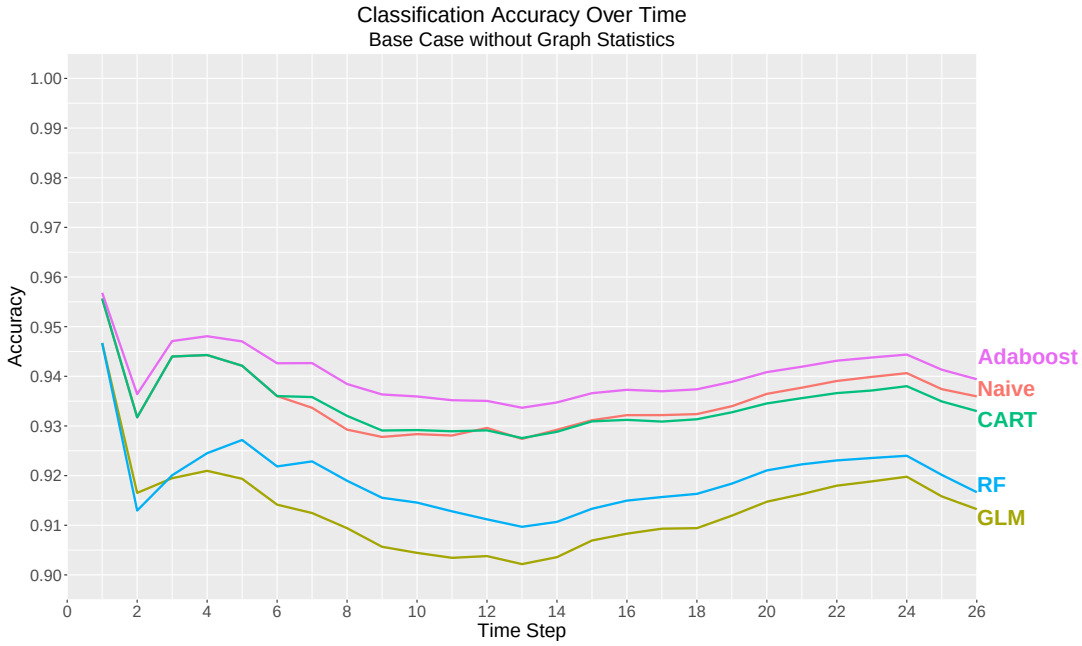
In this section, we examine the performance of four different machine learning algorithms. We do this by comparing the cumulative classification accuracy throughout our 26 step design for each machine learning model. We also compare the model performance across each of our five cases to see if models perform differently across the cases.

Furthermore, for reference, we compare each model to the naïve case. The naïve case is where we say everything stays the same. That is, each populated place will be in the same territory that they were in the prior time period. We use this case as the benchmark that a model would need to "beat" to be considered effective. The naïve case will always be incorrect if a change in territory occurs between time periods, so we use the naïve case as the benchmark because, if a model "beats" the naïve case then we know that model can successfully predict a change in territory between time steps. In a military context, a change in territory is arguably more important to predict than predicting that everything stays the same.

Case 1: Base Case

Our first case, or base case, is where each node is classified as either being in or out of ISIS territory at each step in our design. Control, attack, and support zones are aggregated to

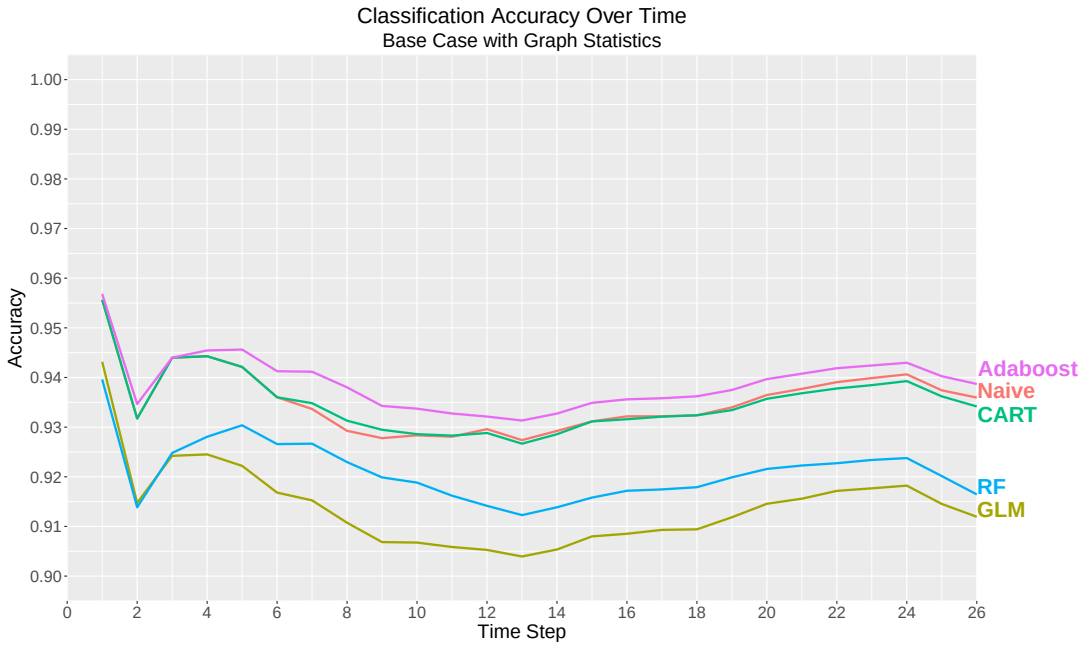
represent ISIS territory. Figure 4.1 shows the performance of our models over time in the base case without graph statistics. We see that only the Adaboost model is superior to the naïve case, depicted in orange. We also see that naïve case is a little over 93% accurate over the course of our design, which is a high bar to beat for a machine learning model. This shows us that not a lot of the places change territory each time period, and confirms the difficulty of this classification problem. A model that beats this naïve case has some ability to successfully predict the changes that occur in Iraq and Syria.



This graph shows the cumulative classification accuracy over time for the base case without graph statistics. The only model that beat the naïve approach is the Adaboost model.

Figure 4.1. Base case performance without graph statistics

Figure 4.2 shows the performance of our models with graph statistics. Here, we again find that the Adaboost model, with an overall accuracy of approximately 94% is the only model that beats the naïve case.

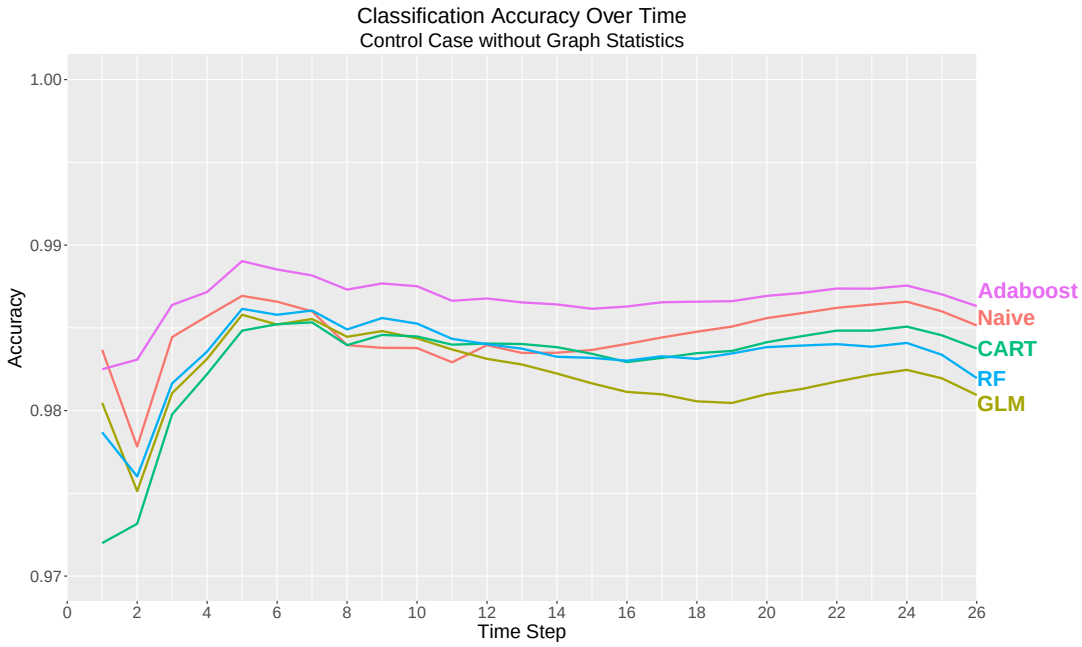


This graph shows the cumulative classification accuracy over time for the base case with graph statistics. The Adaboost Model is the only model to beat the naïve approach.

Figure 4.2. Base case performance with graph statistics

Case 2: Control

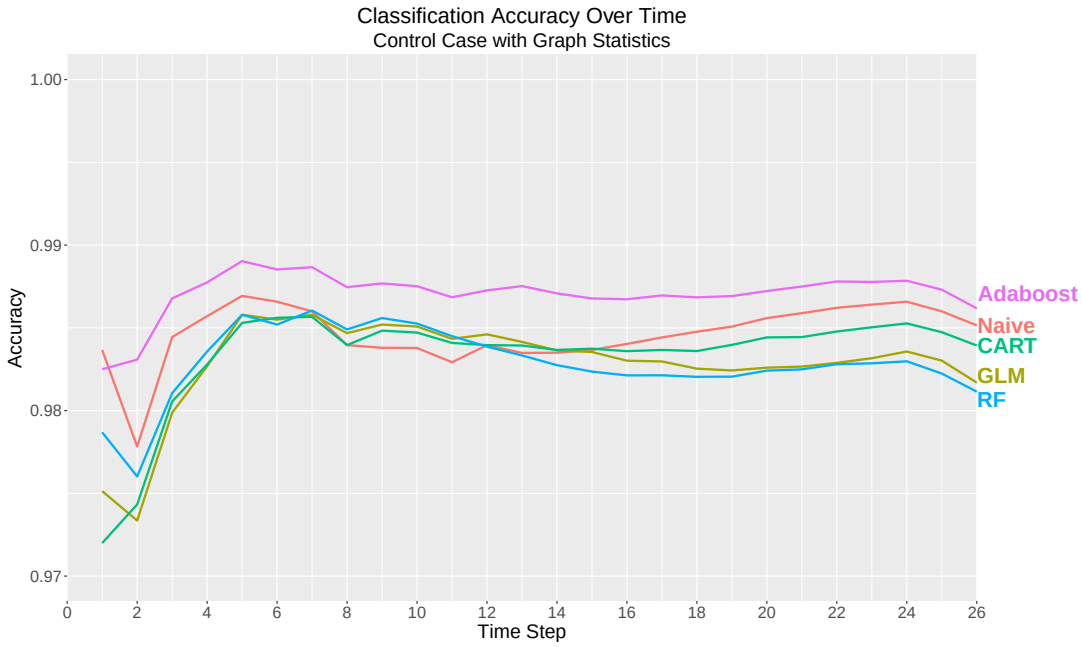
In this case, each populated place is classified as either being in or out of ISIS control territory at each step in our design. In Figure 4.3 we again see that only the Adaboost model, with a classification accuracy of almost 99%, performs better than the naïve case. However, all of these models perform rather similarly with an overall classification accuracy greater than 98%.



This graph shows the cumulative classification accuracy over time for the control case without graph statistics. The Adaboost model is the only model that beats the naïve approach in this case.

Figure 4.3. Control case performance without graph statistics

Figure 4.4 shows the performance of our models in the control case with graph statistics. Here, we find that only the Adaboost model performs better than the naïve case. Additionally, we see that the linear regression, classification tree, and random forest models all slightly pass the naïve approach between steps 8 and 12 in our rolling horizon design. However, these models do not perform as well as the naïve case towards the end of the design, and are ultimately beat by the naïve case.

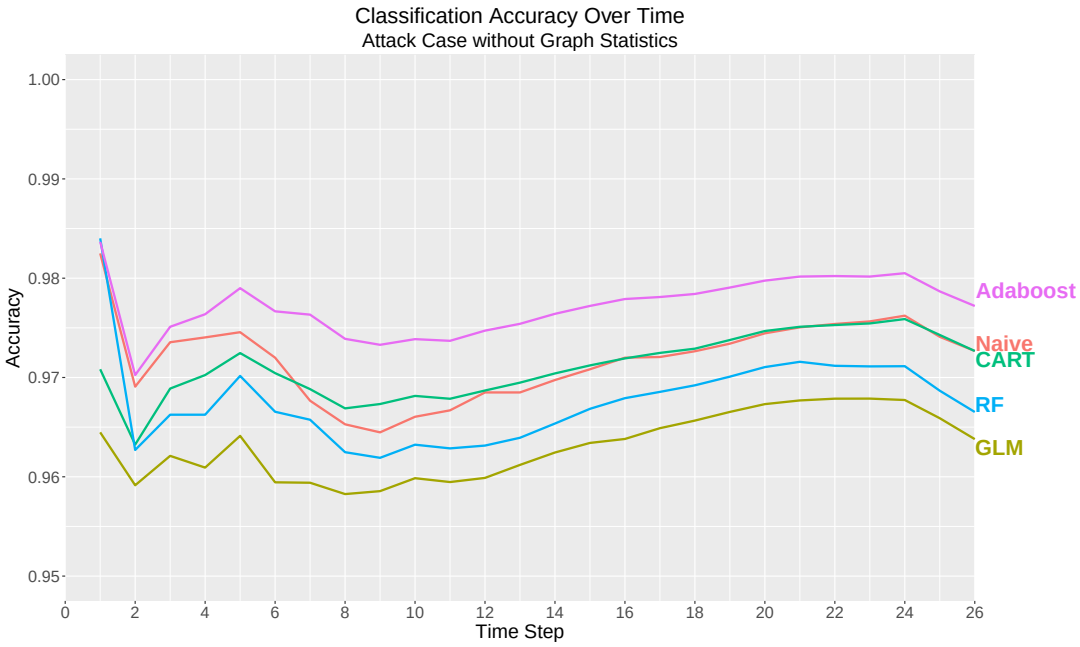


This graph shows the cumulative classification accuracy over time for the control with graph statistics. Only the Adaboost model beats the naïve case in the control case with graph statistics.

Figure 4.4. Control case performance with graph statistics

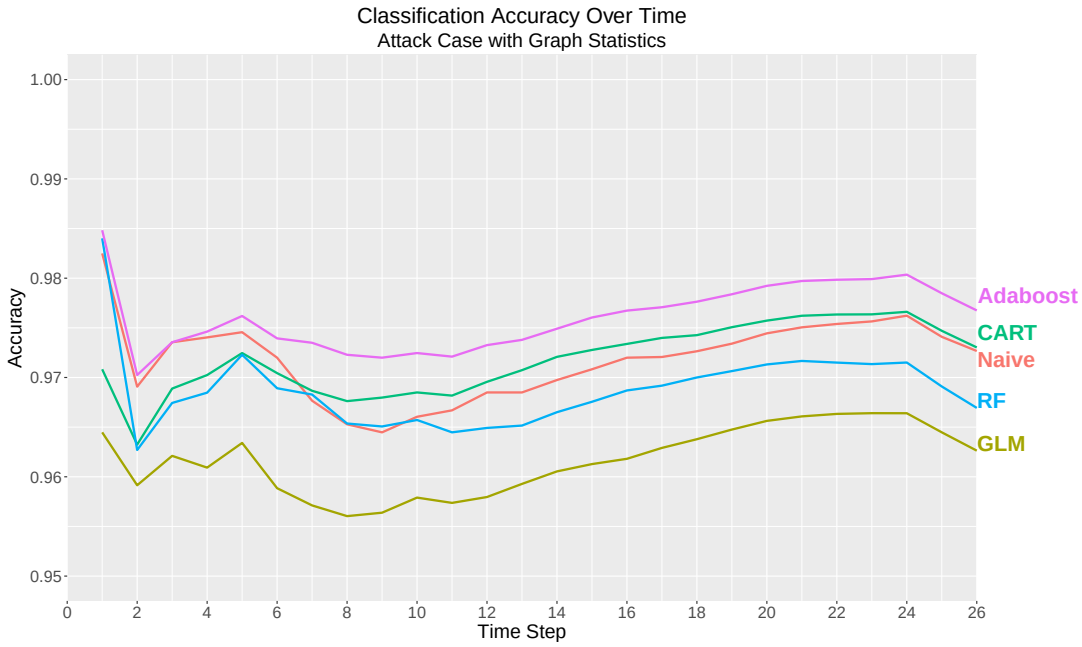
Case 3: Attack

In the attack case, each populated place is classified as either being in or out of ISIS attack territory. Without graph statistics (Figure 4.5), only the Adaboost model beats the naïve case. However, we also see that the classification tree model performs just as well as the naïve case with a classification accuracy of over 97% . We see the same results with graph statistics in Figure 4.6. That is, both the Adaboost and classification tree models are as good or better than the naïve case. The Adaboost model is the highest performing model both with and without graph statistics.



This graph shows the cumulative classification accuracy over time for the attack case without graph statistics. The Adaboost model is the only model that clearly beats the naïve approach in this case.

Figure 4.5. Attack case performance without graph statistics

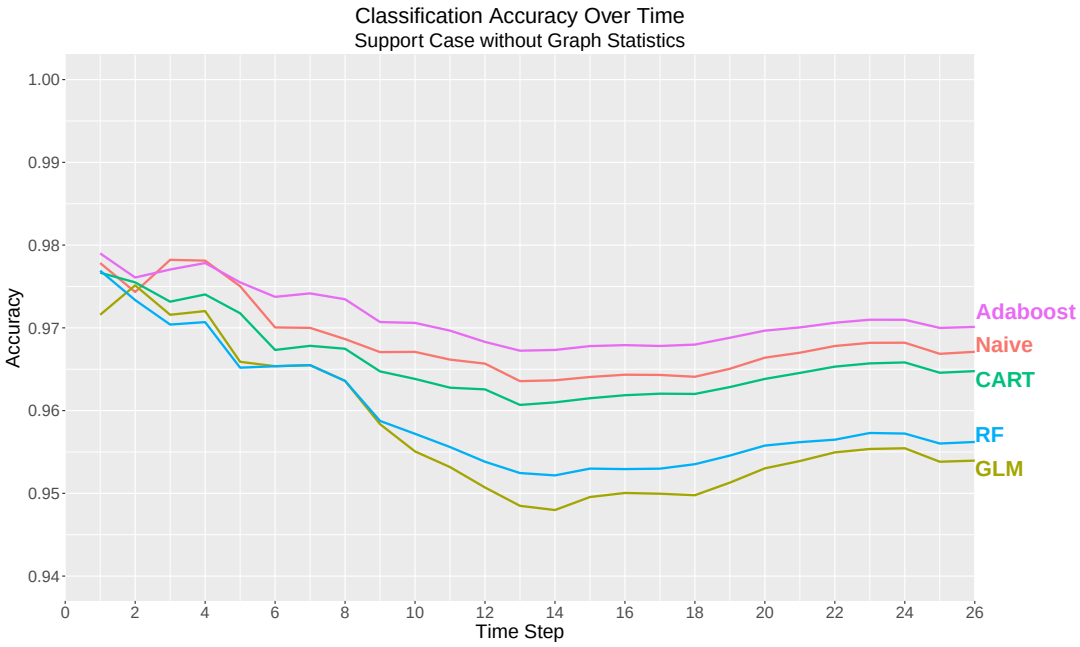


This graph shows the cumulative classification accuracy over time for the attack case with graph statistics. The Adaboost model and the classification tree perform better than the naïve case.

Figure 4.6. Attack case performance with graph statistics

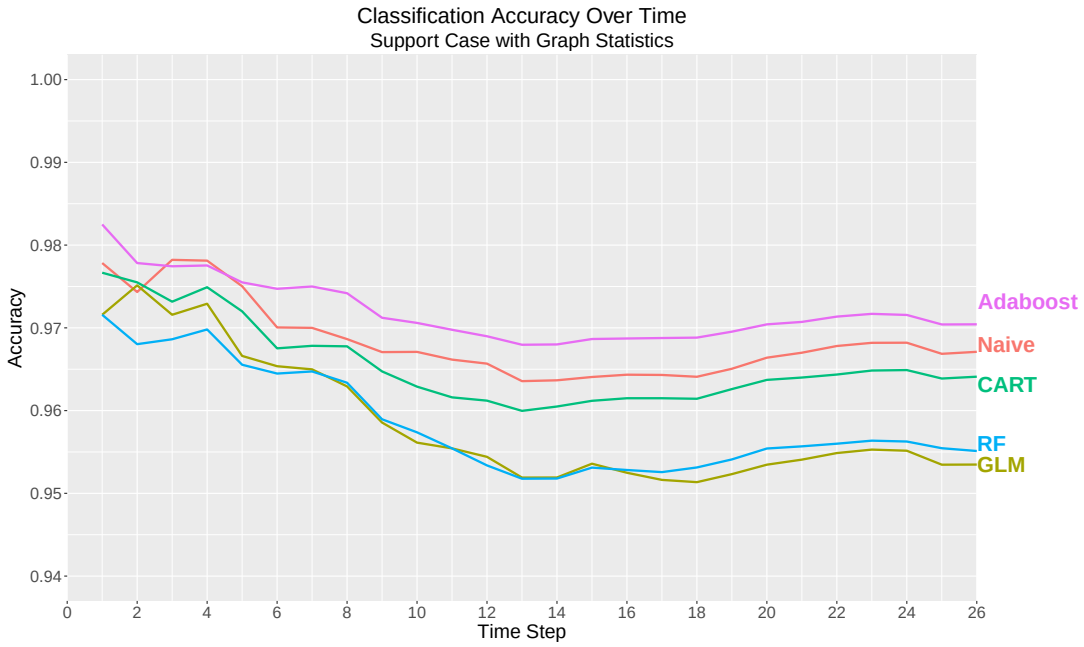
Case 4: Support

In this case, every populated place, at each step in our design, is classified as being in our out of ISIS support territory. Again, we see similar performance in Figures 4.7 and 4.8 as in the previous cases. However, this time, we saw that there was a more noticeable decline in performance over time. While this decrease was not large in magnitude it suggests that the models may require more steps in the design to learn what constitutes a support zone.



This graph shows the cumulative classification accuracy over time for the support case without graph statistics. Only the Adaboost model is superior to the naïve approach in this case.

Figure 4.7. Support case performance without graph statistics

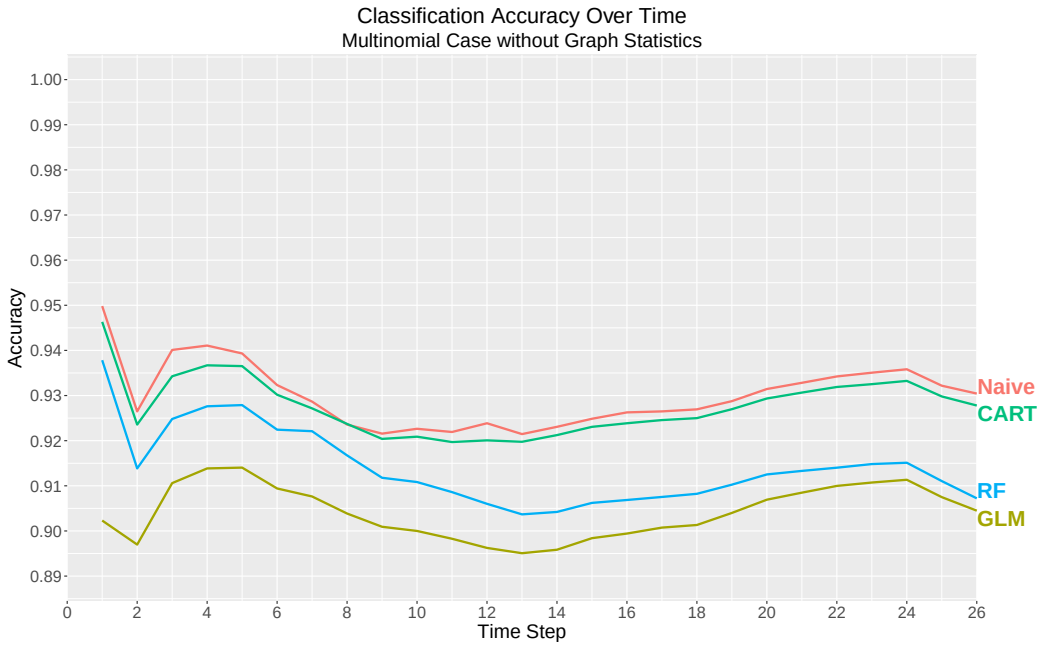


This graph shows the cumulative classification accuracy over time for the support case with graph statistics. The Adaboost model is the only model superior to the naïve case in the support case.

Figure 4.8. Support case performance with graph statistics

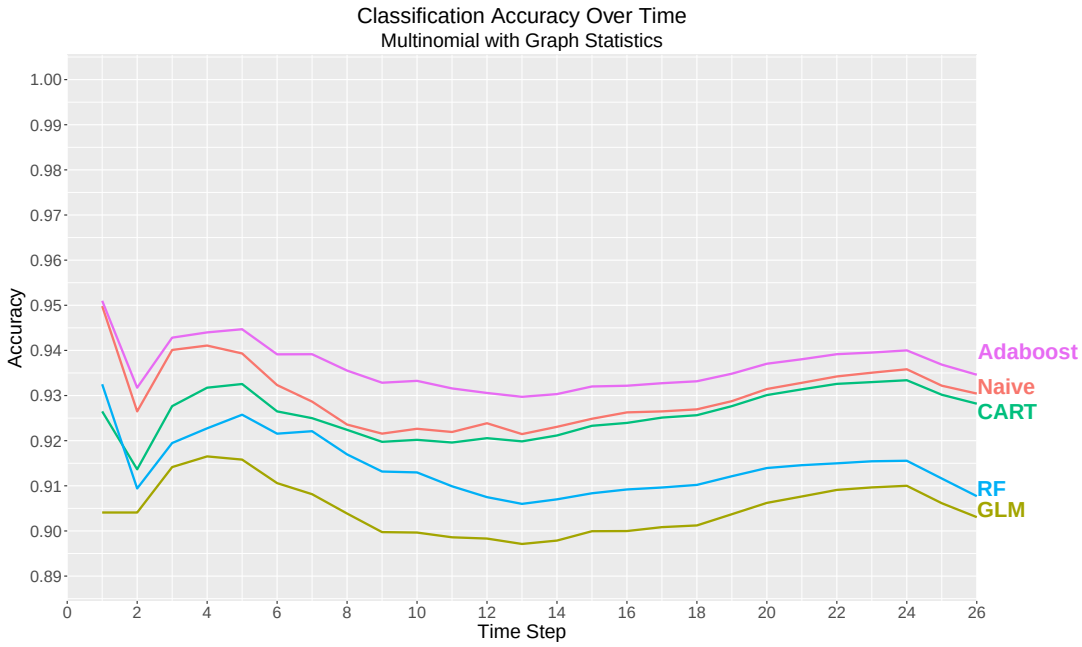
Case 5: Multinomial

In our last case, each node is classified as one of the four levels of ISIS territory; not in ISIS territory, control, attack or support. Model performance as shown in Figures 4.9 and 4.10 relative to the naïve case is consistent with what we saw in our previous cases. However, in this multinomial case, we see a lower overall classification accuracy for all models both with and without graph statistics. This makes sense as we expect a lower accuracy with four levels of territory compared with two. In the multinomial case, making classifications is more complicated as the model has to distinguish between four classes as opposed to two in the binomial case. Of note, the Adaboost model without graph statistics in this case did not converge when we ran the model through our rolling horizon design. As a result, we dropped this model from our consideration and it is not displayed in Figure 4.9. In this case, only the Adaboost model with graph statistics beats the naïve case. This result demonstrates that by adding graph features, not only will the model converge, we can improve our ability to predict ISIS territorial gains and losses.



This graph shows the cumulative classification accuracy over time for the multinomial case without graph statistics. The Adaboost model did not converge and was dropped from consideration. The naïve model performs the best in this case

Figure 4.9. Multinomial case performance without graph statistics



This graph shows the cumulative classification accuracy over time for the multinomial case with graph statistics. Only the Adaboost model beats the naïve approach in this multinomial case.

Figure 4.10. Multinomial case performance with graph statistics

Key Takeaways

From these results, we see that in every case, both with and without graph statistics, the Adaboost model performs better than the other machine learning models and the naïve case. Thus, we determine that the Adaboost model is the best performing model. Additionally, in the binomial cases, we cannot yet see any clear distinction between models with or without graph statistics. This is further analyzed in the next section to determine if there is a difference. However, in the multinomial case, we see that the Adaboost model was able to converge when graph statistics were added and thus outperforms the model without graph statistics. This shows that in a multinomial setting, graph statistics do improve our ability to predict. The performance of all models in all cases with associated confidence intervals is provided in Appendix B.

Our last key takeaway from this part of the analysis focuses on the naïve case. In each case, we see that the naïve accuracy is greater than 90%. To us, this means any improvement from the naïve case demonstrates a models ability to correctly predict the changes that occur in

ISIS territory over time. Thus, the greater the difference between a model's classification accuracy and the naïve accuracy tells us how that model performs at predicting the changes in territory. In a military context, predicting the changes is extremely important in preparing an understanding of the OE and will be further analyzed in Section 4.3. Furthermore, since the naïve accuracy is greater than 90 percent, that tells us that many of the populated places are not changing territory each time period. This makes the need to successfully predict the changes, of the small subset of places that actually do change, even more important.

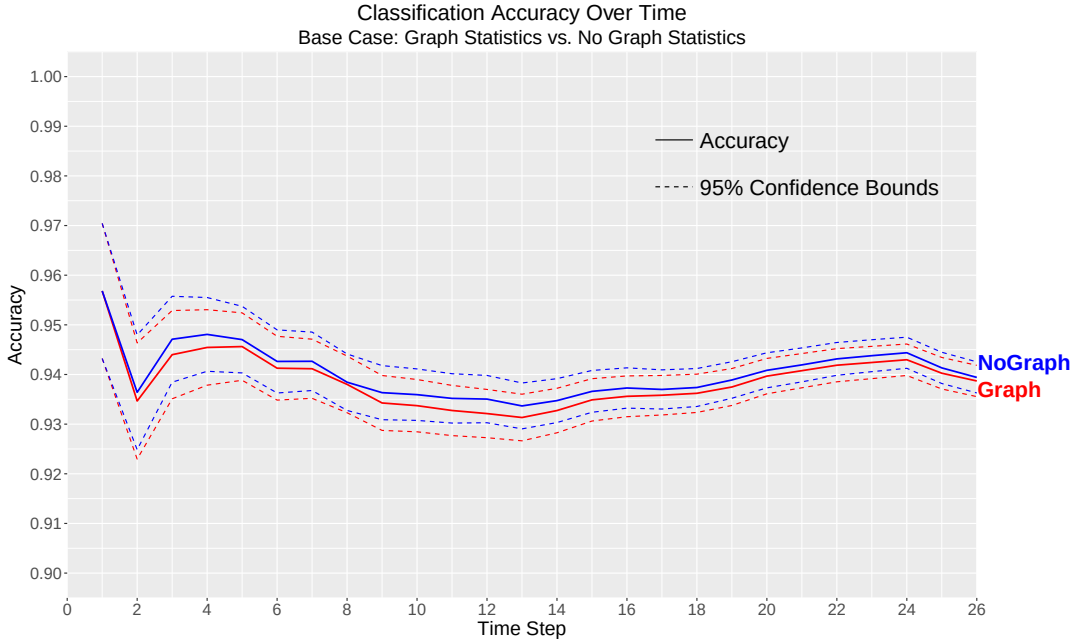
4.2 Graph Statistics Versus No Graph Statistics

In this section we compare the performance of the Adaboost model both with and without graph statistics. In the previous section, we saw that, for all binomial cases, there was no clear distinction between the models with graph statistics and the models without graph statistics. To develop a clear answer to the question of whether graph statistics help the model perform better in the binomial case, we directly compare the Adaboost model both with and without graph statistics. We use the Adaboost models for this comparison as we determined they were the best performing models in Section 4.1.

We use the base case to assess the performance differences, because all binomial cases performed relatively similar. We also calculate a 95% binomial confidence interval at each step in the design using equation 4.1:

$$\hat{p} \pm z\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}. \tag{4.1}$$

Figure 4.11 shows the differences in performance between the two types of models for the base case. We see that there is no real difference between the two models as their cumulative accuracy lines lie essentially on top of one another.



This graph shows the cumulative classification accuracy over time for the Adaboost model in the base case, both with and without graph statistics. The solid lines represent the observed accuracy and the dashed lines represent the 95% confidence intervals for the observed accuracy. There is no statistical difference, at the 95% confidence level, between the two Adaboost Models.

Figure 4.11. Adaboost classification accuracy base case: graph statistics vs. no graph statistics

This shows that adding graph statistics as features does not necessarily improve the overall predictive accuracy of a model employed in our rolling horizons design in a binomial classification case. However, as noted in Section 4.1, in the multinomial case, models with graph statistics perform better than models without graph statistics.

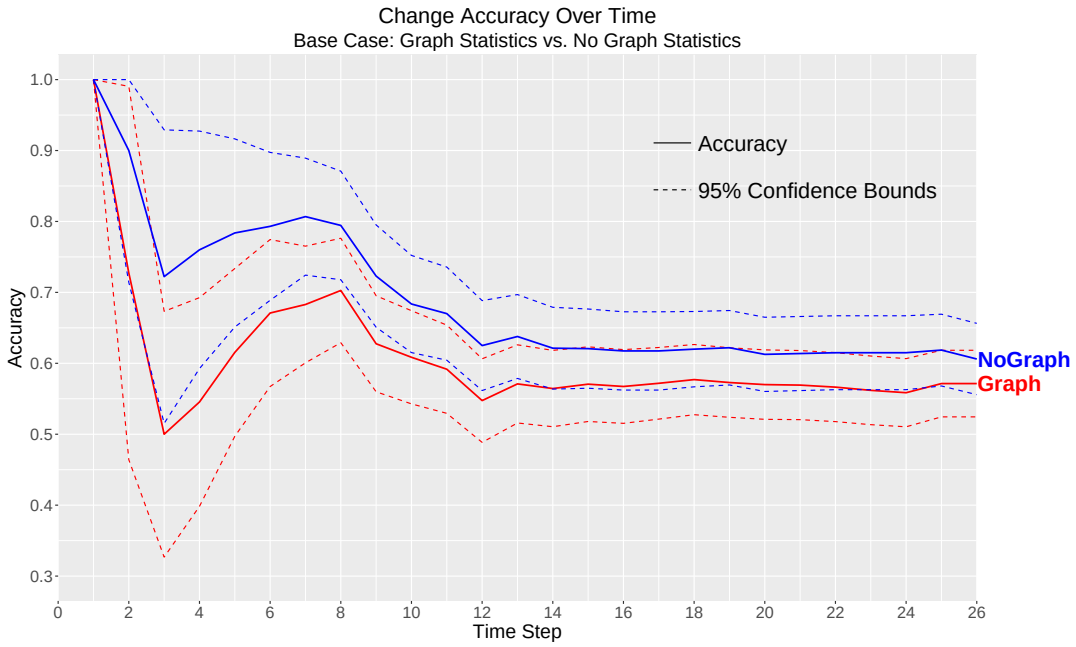
4.3 Predicting Changes

In a military context, it is important to be able to classify the territory of each node accurately throughout our design. However, what may be more important is evaluating the model's ability to successfully predict the changes that occur at each time period. These changes are likely to have a greater influence on the battlespace and OE than a node that remains in the same territory as in the previous state. With this in mind, we assess how a model with graph statistics can improve upon the ability to predict changes.

To evaluate the ability for a model with graph statistics to predict change, we first take the subset of nodes where the model predicted a change in territory to occur. We then compute the classification accuracy on this subset to determine how accurate a model is when predicting changes. We explore this using the base case and multinomial case. The cumulative change accuracy for all other cases and machine learning models is located in Appendix B.

Base Binomial Case

In the base case, we define a change as the situation where a populated place was in ISIS territory in one time period and not in ISIS territory the next time period and vice versa. In Figure 4.12, we compare the change accuracy over time of models both with and without graph statistics for the base case. We see that the model without graph statistics actually has better accuracy when predicting a change compared to the model with graph statistics. The model without graph statistics is approximately 60% accurate when predicting a change, and the model with graph statistics is approximately 57% accurate. However, when we look at the 95% confidence intervals we see that, at the 95% confidence level, there is no statistical difference between the two models ability to predict change as their confidence intervals clearly overlap throughout our design.



This graph shows the cumulative classification accuracy for nodes that are predicted to change territory with the Adaboost model in the base case. The solid lines represent the observed accuracy and the dashed lines represent the 95% confidence intervals for the observed accuracy. The model without graph statistics performs better than the model without graph statistics. However, at the 95% confidence level, there is not a statistical difference between the two models as their confidence intervals overlap throughout the design.

Figure 4.12. Change accuracy: Adaboost base case

These results show that in the binomial case, a model with graph statistics as added features, does not necessarily improve the accuracy of predicting changes.

Multinomial Case

In the multinomial case, we define a change as the situation where a populated place was in one level of ISIS territory in one time period and in a different level of ISIS territory the next time period. Since the Adaboost model without graph statistics did not converge in the multinomial case, we cannot directly compare the models as in the previous section. However, the Adaboost model with graph statistics had a cumulative change accuracy of 54.85% throughout the course of our design. If we consider the situation where you randomly guess the new territory of a node that changes, over time we would expect that you

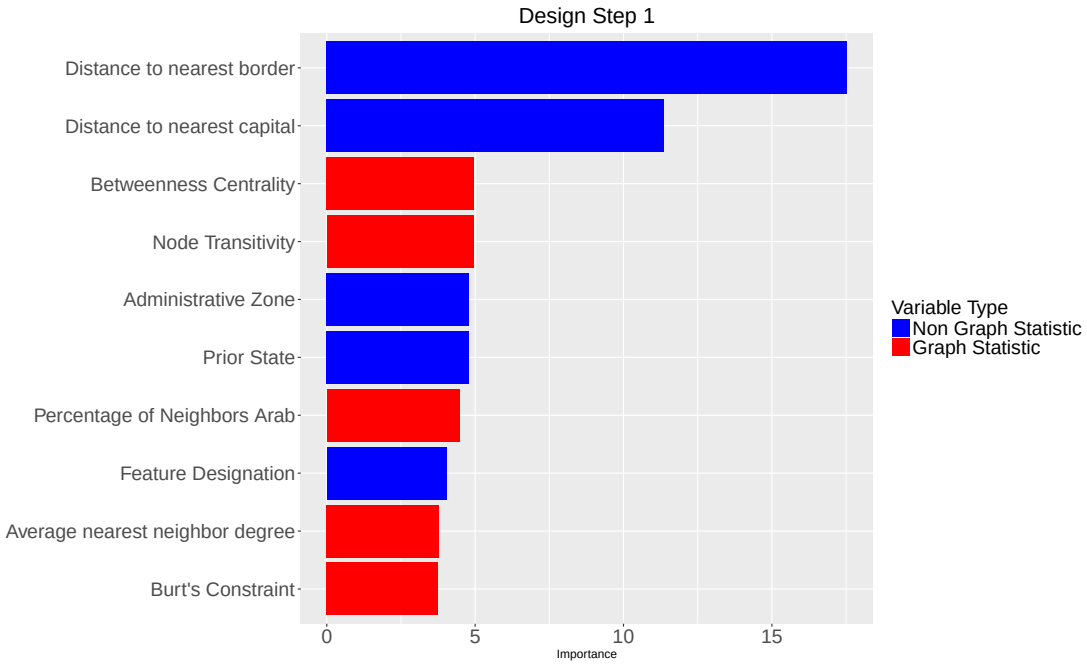
to be correct 1/3 of the time. With this situation in mind, our change accuracy performance of 54.85% with the Adaboost model with graph statistics is very good. This means that, in the multinomial case, when our model asserts a change, more than half of the time that assertion is correct. This finding again demonstrates that, in the multinomial case, graph statistics can improve the performance of predicting ISIS territorial gains and losses.

4.4 Variable Importance

This section describes what variables are most influential to the predictions made throughout our design. Since we determined that our Adaboost model was the best performing model, we use it to compute the variable importance. We will focus on only the Adaboost Model with graph statistics in the binomial base case so we can see which variables, with all included, are the most important to making predictions. The Adaboost algorithm computes variable importance by determining the average gain in Gini index for each variable across all trees build within the algorithm (Alfaro et al. 2013).

We use the "adabag" package in R to compute the variable importance at three different steps in our design; step 1, step 13, and step 26 (Alfaro et al. 2013). By computing the importance at the beginning, middle, and end of the design, we can see how variable importance changes throughout the design.

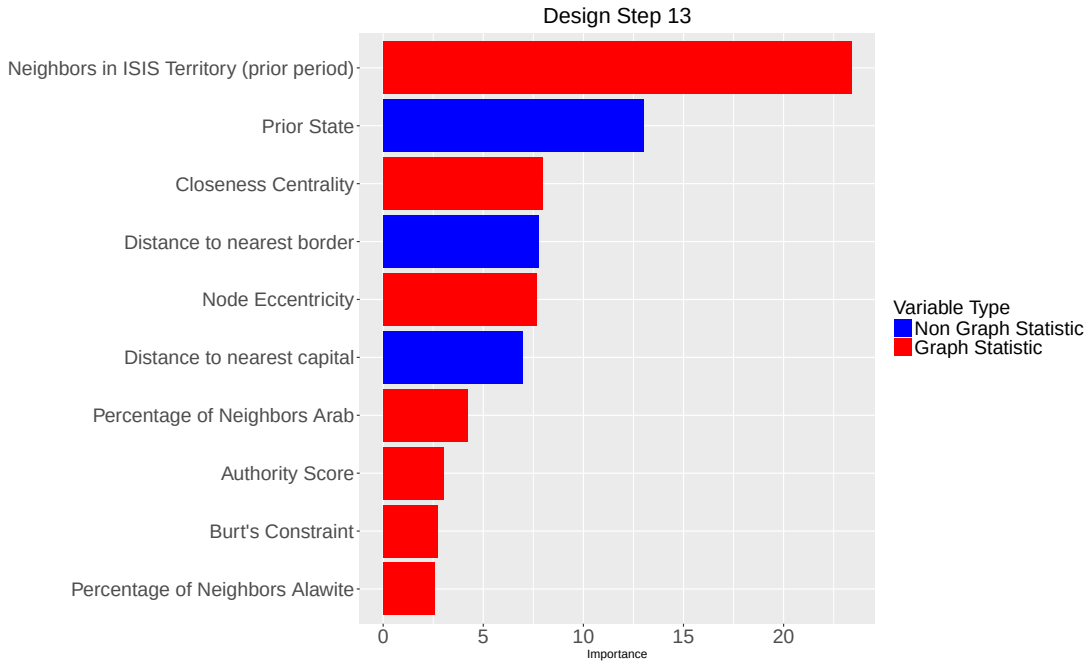
From the variable importance plots, we see in the first step of our design (Figure 4.13) the distance to the nearest border and distance to the nearest capital are the two most important variables. A red bar means the variable is a graph statistic, whereas a blue bar means the variable is not a graph statistic. We also see 5 of the top 10 most important variables are graph statistics in the initial step of our design.



This graph shows the variables that are most important to making the predictions in first step of our design. The blue bars represent variables that are not graph statistics. A red bar represents a variable that is a graph statistic. The distance to the nearest border is the most influential variable in design step 1.

Figure 4.13. Adaboost step 1 variable importance

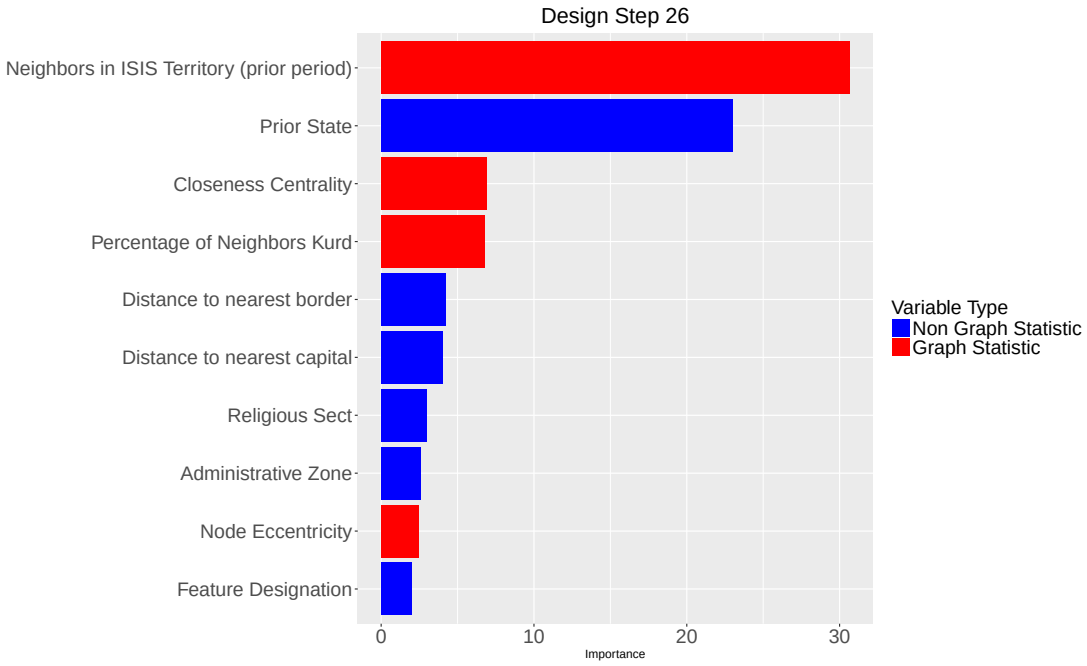
As we move into the middle of our design (Figure 4.14), we see that the percentage of neighbors in ISIS territory becomes the most important variable. We also see that the territory in the prior state develops more importance as it passes the two distance metrics to the nearest border and nearest capital. Here we see 7 of the 10 most important variables in this design step are graph statistics.



This graph shows the variables that were most important to making the predictions in design step 13. The blue bars represent variables that are not graph statistics. A red bar represents a variables that is a graph statistic. The percentage of neighbors in ISIS territory is the most influential variable in design step 13.

Figure 4.14. Adaboost step 13 variable importance

In the last step of our design (Figure 4.15), we see that the top two important variables remain the same as in design step 13. The graph also shows that these variables are the most important variables by a wide margin. Each of the most important variables makes sense in the context of what we have learned about our data. For the neighbor statistic, it is clear that if a high percentage of a populated places neighbors were in ISIS territory in the previous period, the model asserts that it is more likely for the place itself to be in ISIS territory in the next period and vice versa. For the territory in the prior time period, it makes sense when we consider the performance of the naïve case. Since the naïve accuracy in the base case is 94%, we know that not a lot of places change territory each period. Thus, it is very likely for a place to remain in the same territory as the previous time period which results in this variable heavily influencing the model's predictions.



This graph shows the variables that were most important to making the predictions in design step 26. The blue bars represent variables that are not graph statistics. A red bar represents a variables that is a graph statistic. The percentage of neighbors in ISIS territory is the most influential variable in step 26.

Figure 4.15. Adaboost Step 26 Variable Importance

These plots also allow us to see that the importance of variables changes over time which may be indicative of a fluid battlefield dynamic where ISIS decision making may have evolved over time. Furthermore, we see that the important variables in this problem trend towards the graph statistics. This shows us that our methodology is providing value to this classification problem, and that by adding graph statistics as features we can learn more about what drives ISIS territorial gains and losses.

After examining the variable importance associated with our Adaboost model, we say that the five most important variables in this classification problem are:

1. Percentage of neighbors in ISIS territory in the prior time period
2. Territory in the prior state
3. Closeness centrality
4. Percentage of neighbors that are Kurdish

5. Distance to the nearest border

4.5 Machine Learning Takeaways

Binomial Classification

After analyzing the results of our machine learning algorithms we are surprised that, in the binomial base case, the model with graph statistics did not have a higher classification accuracy than the model without graph statistics. We expected to see performance improvements in models that had added graph statistics and were proven wrong. This finding is even more interesting when we consider that the graph statistics seem to be more important in the presence of the non-graph statistics as shown in Section 4.4, which would suggest that a model with graph statistics could perform better.

When we look at Table 4.1 we see the confusion matrices for the Adaboost models for the entire rolling horizon design in the base case. We look at these cumulative confusion matrices to see if they provide reasoning to suggest why the models with graph statistics do not perform better than those without graph statistics.

Table 4.1. Confusion matrices for Adaboost models in the base case.

Base case: Adaboost with graph statistics

		Actual	
		Yes	No
Predicted	Yes	1260	501
	No	865	19656

Base case: Adaboost without graph statistics

		Actual	
		Yes	No
Predicted	Yes	1287	512
	No	838	19645

These tables show the confusion matrices for the Adaboost models in the base case for the entire rolling horizon design. The top table shows the confusion matrix for the model with graph statistics and the bottom table shows the matrix for the model without graph statistics.

The first thing we notice is that there is not any glaring differences in the performance of the models as shown in Section 4.2. We do see a larger number of true positives without graph statistics, but a larger number of true negatives with graph statistics. Furthermore, the model with graph statistics has a smaller number of false positives, but has a larger number of false negatives. These differences as whole, when we consider the entire design, are so small that they do not provide definitive information about why the model with graph statistics did not perform better.

One reason we may see no difference between the two models in this case, is the large imbalance in the quantity of places that are in or out of ISIS territory at each time period. On average, 90.54% of the nodes are not in ISIS territory in each time period. Having this large of disparity in classes, considering the base binomial case, can present difficulties for any machine learning algorithm. There are simply not enough observations of nodes that are in ISIS territory for the algorithm to truly learn what data constitutes ISIS territory. We see the exploration of this imbalance and adjustment of the machine learning models to better handle the imbalance as a great source of future research to improve our ability to solve this problem.

Another possibility could be that with the increased number of variables in a model with graph statistics, some of these variables could be adding a lot of noise which makes it harder for the model to classify correctly on a test set of data. With this in mind, we believe there is value in future research that conducts a variable selection procedure to limit the amount of graph statistics added to the model. This could potentially improve performance in the model with graph statistics by eliminating variables that detract from the model's ability to predict.

Multinomial Classification

In the multinomial case, we see that graph statistics do improve the predictive performance of these machine learning models. The Adaboost model with graph statistics is the only model to beat the naïve case in the multinomial classification setting. Furthermore, as described in Section 4.3 we see the model with graph statistics predict change with very good accuracy. In these results, we see that in a multinomial case, our method of incorporating graph statistics as features, increases our ability to accurately predict the territorial gains and losses of ISIS thus establishing validity in our overall methodology.

4.6 Predicted Territory

This section discusses how we take the results of our machine learning models and develop a map that depicts the predicted ISIS territory. For our methodology to be useful to intelligence analysts charged with developing an understanding of the battlespace, we need to convert the results from our machine learning models into a map that shows what territory ISIS is predicted to hold in a future time period. To demonstrate this, we first divide the region of Iraq and Syria into individual polygons based upon the populated places in the two countries. Then, we use the results from our base case Adaboost model with graph statistics to build maps of the predicted ISIS territory at each step in our design.

4.6.1 Dividing the region

To determine a sub-region associated with each populated place in Iraq and Syria we use a Voronoi diagram. This diagram partitions the region of Iraq and Syria into an area for each populated place. All locations within the sub-region computed for each place are closer to that specific populated place than any of the other populated places that we use in our analysis (ESRI Technical Support 2018). Figure 4.16 shows the diagram for Iraq and Syria. This diagram provides us with a sub-region for each populated place that we can use to develop a map based on our predictions.

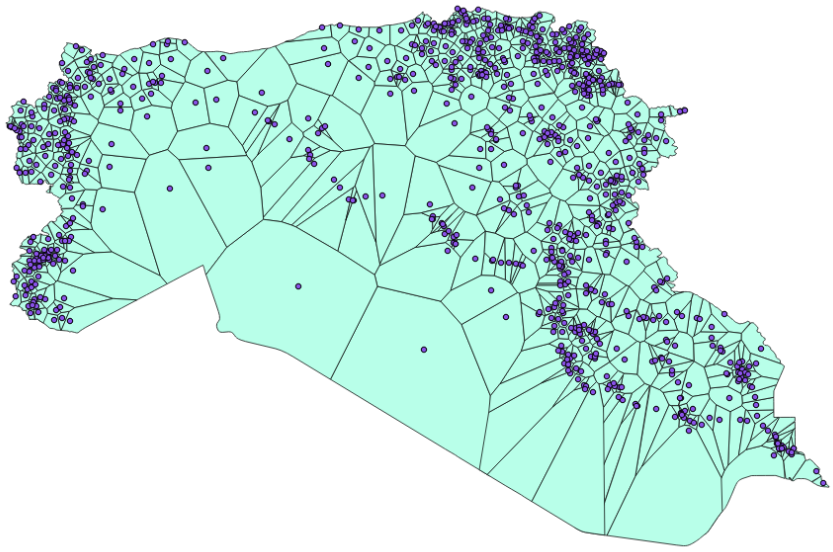


Figure 4.16. Voronoi diagram of Iraq and Syria

4.6.2 Map Product

Once we develop our Voronoi diagram, we use the predictions at each step in our design to build a map of the territory that is predicted to be in ISIS territory. So, if a populated place was predicted to be in ISIS territory at a certain step in our design, we color that area in the Voronoi diagram red. Figure 4.17 shows the map we develop for our predictions in design step 26. This map is dated May 10, 2017, as that is the date of the last ISW map that we use in our analysis.

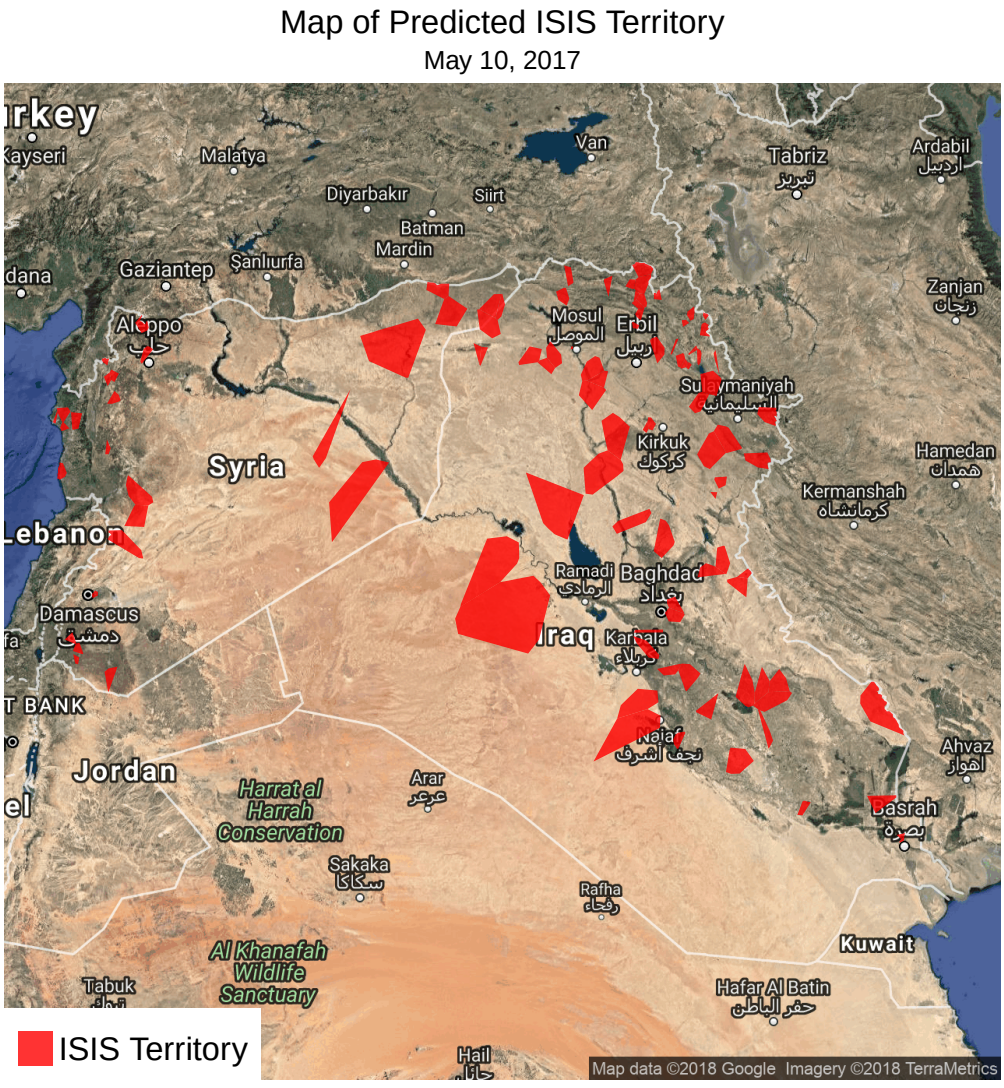


Figure 4.17. Map of predicted ISIS territory

By developing this map, an intelligence analyst can use it to focus their study of Iraq and Syria on the specific regions we have highlighted in red. This utlimately allows the intelligence analyst to provide a better description of the OE when developing their JIPOE. Furthermore, with an improved JIPOE, the JFC can make better decisions about how to employ forces in the battlespace.

CHAPTER 5:
Conclusions and Recommendations

This chapter summarizes how our findings answer our research questions and provides recommendations for areas of future research that could improve upon our methodology.

5.1 Conclusions

This thesis presents a new methodology that predicts the territorial gains and losses of ISIS between 2014 and 2017 while also providing an understanding into what features may drive the observed gains and losses. To summarize this, we go back to our research questions and provide a brief answer to the question based on our findings:

Can we use machine learning methods to predict the territorial gains and losses of ISIS during the period 2014-2017?

The rolling horizon design study demonstrates that we can use a variety of machine learning algorithms to make predictions about the territorial gains and losses of ISIS over time. From this design, we developed models that can be used to make accurate predictions about ISIS territory over time and provide a means for intelligence analysts to easily understand the outputs of our models. Our Adaboost model beat the naïve benchmark case which establishes the validity of this methodology.

Do graph statistics, derived from representing Iraq and Syria as a graph, improve our ability to predict ISIS territorial gains and losses?

Our analysis shows that by adding graph statistics computed from a graph of Iraq and Syria, we can improve the accuracy of predicting ISIS territory compared to the naïve case. While we do not see a significant improvement in predictive accuracy over models without graph statistics in the binomial case, we do see performance improvements in the multinomial classification setting. This shows that graph statistics can help us make better predictions, and when combined with the fact that one of our models beat the naïve case, it shows us that our methodology can improve the capabilities of intelligence analysts in the DoD and IC. These models can allow intelligence analysts to focus their research on specific areas of the

battlespace. Additionally, this can also empower the individual intelligence analysts to use the machine learning results, combined with their expertise, to improve how the battlespace is analyzed and understood.

What predictors, or features, are most important for predicting the territorial gains and losses of ISIS during the period 2014-2017?

Our models show that two variables were the most important features for predicting the territorial gains and losses of ISIS: the percentage of neighbors that are in ISIS territory in the previous time period and the populated places' territory in the previous time period. Our models also show that graph features, as a whole, were more important than the non-graph features. This shows that our methodology and inclusion of graph features can provide a benefit to solving these kinds of problems. It also shows that graph statistics can help describe the factors that drive territorial change. Furthermore, our models generally show that the important variables can change over time which is indicative of a fluid battlefield where our enemies change how they operate over time. Our use of updated machine learning models, in lieu of static models, allows us to capture these changes over time, and build insights into what variables dirve the territorial gains and losses of terrorist groups.

5.2 Recommendations

This section provides areas of research that could improve upon the methodology described in this thesis.

Application to Other Conflicts

Applying this methodology to other ongoing conflicts and terrorist organizations across the world will allow researchers to determine the robustness of the results found in this thesis. This application can highlight ways to improve upon the methodology and how we can alter the use of machine learning algorithms to solve these kinds of problems. Additional insight into other features that affect the territorial spread of a terrorist organization may also be gained by applying this design in other regions of the world.

Use of Classified Data Sets

In this thesis, we used all open-source data sets in our analysis. With the success of our methodology and its application to ISIS in Iraq and Syria, the IC could benefit from applying this methodology using their own classified data sets and potentially see better results.

Analysis of Road Network Impact

Our research was focused on the populated places and how various features associated with a populated place would affect the likelihood of being in or out of ISIS territory. By developing additional features associated with the road network, researchers can determine how the road network, and infrastructure at large, affects the spread of a terrorist organization.

Adjusting Machine Learning Approach

In Section 4.5 we point out different ways that future researchers could improve upon our machine learning approach to determine if a model with graph statistics can statistically improve upon a model without graph statistics. One possible approach is to build models that can better handle the class imbalance in this data set. The other possible approach is to conduct variable selection to limit the variables that contribute noise to this classification problem. Both of these areas of future research could allow new machine learning models to have improved predictive accuracy.

Adjusting Rolling Horizon Design

In our rolling horizon design, each design step used all previous information for each prediction. That is, each step used data from the previous time period all the way back to our very first time period. If we consider the fact that organizations like ISIS can change their tactics throughout time, it may be worth exploring an adjustment to this design and adjust how far back in time each iteration looks. For example, instead of using all previous data to make a prediction, future researchers could use the previous 6 months worth of data to make predictions at each step in the design. Exploring and adjusting the amount of data we use to make predictions could lead to performance improvements. Furthermore, this exploration could allow for a better understanding of a time period that better explains future territorial gains and losses.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A:
Graph Conversion Tutorial

This tutorial was developed as a class project for OA 3802, Computational Methods for Data Analytics at the Naval Postgraduate School. Source: Vanderzee et al. (2017)

Geographic Database to Network Graph

Christopher Fletcher, Coleman Strickland, Tony Vanderzee

11/14/2017

Situation

As OR analysts we have been tasked by our organization to develop a way to predict the spread of a terrorist group. The only data we have available is a geographic database containing shapefiles of populated places, road networks, ethnicity, significant events, etc. Our organization has determined that if the geographic database can be converted to a network graph with the populated places as the nodes or vertices, and the roads as the arcs or edges, then some interesting predictions about nodes in future states could be of significant value in predicting the spread of a terrorist organization.

Thus, we have been tasked to convert the geographic database into a network graph. To demonstrate our methodology of this conversion, we will use Iraq and Syria as the geographic region. Ultimately, the goal is to produce a network graph to help predict the spread of ISIS in the region. The tutorial below explains our methodology to accomplish this goal.

Sytems, Packages, and Concepts Used

To convert the geographic database to a network graph we use a variety of systems, packages, and concepts to build the network. Below is a list of the main systems, packages, and concepts utilized during this tutorial.

- Geographic Databases
- Geographic Information Systems (GIS) - QGIS,GRASS
- R
  - Main Packages: igraph, sp, rgdal, maptools, rgeos and more!
- Map Projections
- String Distance
- Interactive Visualizations
- Leaflet
- Network Flows and Graphs
- OpenStreetMap

Geographic Database

We receive the geographic database from the National Geospatial Intelligence Agency (NGA) which contains the below information.

- Populated Places (Point Shapefiles)
  - 47,430 in Iraq
  - 21,093 in Syria
- Roads (Line Shapefiles)
  - 131,452 in Iraq
  - 105,358 in Syria
- Sects (Polygon Shapefiles)
  - Alawite, Assyrian, Druze, Kakai, Shabak, Shia, Sunni, and Yezidi
- Ethnicities (Polygon Shapefiles)
  - Arab, Assyrian, Druze, Kurd, Shabak, Turkomen, Yezidi
- Significant Events (Point Shapefiles)

- Global Database of Events, Language, and Tone (GDELT)
- 1,029,548 Events in Iraq
- 294,676 Events in Syria

For our conversion process, we use the populated place shapefiles, the road shapefiles, and will develop a method to use the Sect and Ethnicity shapefiles as attributes of the population centers. For this tutorial, we will not be using the significant events data. However, this will certainly be valuable to the organization once the network is built.

Method

We will now highlight the steps we take to go from the geographic database to a network graph representation of Iraq and Syria.

Visually Inspect the Data

We first pull the geographic database into a GIS system to see what kind of data we possess. We use QGIS, a free open-source GIS, to render the shapefiles from the geographic database. To do this in QGIS, follow the below steps.

1. Navigate to 'Add Vector Layer' from the 'Layer' menu.
2. Navigate to the 'directory' button, select 'OpenFileGDB' as type, and browse for location of your geographic database. Then select 'Open'.
3. Select the layers from the geographic database that you want to load.
   When the entire database is loaded into QGIS, we get the below image.

We see all of the points, lines, and polygon shapefiles loaded on top of one another. We will use these layers to build the network. To save individual shapefiles, we can right-click on the desired layers, select 'Save As' and then place them in the desired location by clicking 'Browse'.

Data Preparation

Task 1 - Assign "attributes" to populated places

Sect and Ethnicity

We first assign each of the populated places their respective sect or ethnicity. To do this, we use R to conduct a spatial overlay of the provided sects and ethnicities on each of the populated places. A spatial overlay is simply laying two spatial layers (in this case a points layer and a polygon layer) to find what points fall inside the region of each polygon. The overlay allows us to determine the sect and ethnicity for each populated center A simplified example of a spatial overlay is provided below followed by the R code we use to build the overlay. The R code below shows the spatial overlay process for Iraq with each of the provided sects. The same methodology is applied to the ethnicities and to the Syria shapefiles.

```
library(rgdal)
library(rgeos)
library(sp)
library(foreign)

my_directoryplaces = 'insert directory where population shapefiles are located'
my_directorysects = 'insert directory where sect shapefiles are located'

# Read in Shapefiles
```

```
Iraq.pop <- readOGR(my_directoryplaces,layer='Iraq_Full')
alawite <- readOGR(my_directorysects,layer = 'Sectarian_Alawite')
assyrian <- readOGR(my_directorysects,layer = 'Sectarian_Assyrian')
druze <- readOGR(my_directorysects,layer = 'Sectarian_Druze')
kakai <- readOGR(my_directorysects,layer = 'Sectarian_Kakai')
shabak <- readOGR(my_directorysects,layer = 'Sectarian_Shabak')
shia <- readOGR(my_directorysects,layer = 'Sectarian_Shia')
sunni <- readOGR(my_directorysects,layer = 'Sectarian_Sunni')
yezidi <- readOGR(my_directorysects,layer = 'Sectarian_Yezidi')

#Assign Id's to shapefiles that don't have them
assyrian@data$Id <- 'Assyrian'
kakai@data$Id <- 'Kakai'
shabak@data$Id <- 'Shabak'
shia@data$Id <- 'Shia'
sunni@data$Id <- 'Sunni'
yezidi@data$Id <- 'Yezidi'

#broke this into 2 parts as two of the shapefiles had more columns of data then the others
#section 1
sect1 <- rbind(assyrian,kakai,shabak,shia,sunni,yezidi)
#SPATIAL OVERLAY
sect.points <- over(Iraq.pop,sect1)
#section 2
sect2 <- rbind(alawite,druze)
#SPATIAL OVERLAY
sect.points2 <- over(Iraq.pop,sect2)

# Assign Sects from section 1
Iraq.pop@data$Sect <- sect.points$Id
#find which points did not have a sect from section 1
other.list <- c(which(!(is.na(sect.points2$Ethnicity))))
#assign sects from section 2 to remaining points
Iraq.pop@data$Sect[other.list] <- as.character(sect.points2$Ethnicity[other.list])

#write out new shapefile
writeOGR(Iraq.pop,my_directoryplaces,
         layer = 'Iraq_Full',driver = "ESRI Shapefile",overwrite_layer = T)
#write out a new database file. This allows for proper encription of Arabic location names
write.dbf(Iraq.pop@data,my_directoryplaces+'/Iraq_Full.dbf')
```

Population and Type of Place

The data used for the populated places does not have information such as population or the type of population settlement (capital, city, hamlet, etc.). We feel the information, if available, would be vital attributes of the populated places. To retrieve this critical information we utilize OpenStreetMap (http://www.openstreetmap.org/). OpenStreetMap, amongst the data available, maintains shapefiles on every country in the world. The shapefiles of populated places for Iraq and Syria contain the population for some of the locations (not all) as well as the type of populated place that it is.

Our first thought is to match the data with the OpenStreetMap data by Lat/Long coordinates of the locations; however, this does not work as the exact coordinates differ from one another. Thus, we decide to use a String distance calculation to match each town in our data to the closest match in their data using the **Jaro-Winkler** string distance metric. The equations for Jaro-Winkler are shown below.

Jaro-Winkler String Distance

The code below demonstrates how we completed this task in R. We only show the code we used for Iraq, but the same code is applied to the Syria data as well.

```
library(rgdal)
library(rgeos)
library(sp)
library(stringdist)

wd = 'insert working directory with both populated places shapefiles'
#read in shapefiles
osm.pop <- readOGR(wd,layer='places1')
Iraq.pop <- readOGR(wd,layer='Iraq_Full')
#pull names out of shapefiles
osm.names = osm.pop@data$name
name = Iraq.pop@data$FULL_NAME1
#find closest match for each location using Jaro-Winkler String Distance calculation
matches = amatch(name,osm.names,method = 'jw',p=.1, maxDist=.3)
#Assign the type and population to closes matches
Iraq.pop@data$Type <- osm.pop@data$fclass[matches]
Iraq.pop@data$Population <- osm.pop@data$population[matches]

#write out new shapefiles and database file
writeOGR(Iraq.pop,wd,layer = 'Iraq_Full',driver = "ESRI Shapefile",overwrite_layer = T)
write.dbf(Iraq.pop@data, wd+ '/Iraq_Full.dbf')
```

Task 2 - Merge Shapefiles

With each of our populated places containing the attributes we desire, we merge the new shapefiles for Iraq and Syria together. Additionally, we perform the same task with the shapefiles for the roads in both countries. With the shapefiles combined, we are left with one road and one population shapefile to be used in the network build.

To do this in QGIS, we followed the steps below:

1. Navigate to the 'Merge Vector Layer' feature from the 'Vector' menu.
2. Select the ellipsis next to the text box that says 'Layers to Merge'
3. Select the shapefiles to merge. **They must be of the same type**.
4. Name the output (or have the output be a temporary layer that can be saved as a shapefile if desired). Then hit run and the output shapefile will populate in the QGIS window.

Task 3 - Subset Shapefiles for data you want

For our graph build, we decide that we only want to use populated places coded as 'National Capital' or 'City'. Additionally, we choose to eliminate 'Residential' roads and all other small roads from our road network. The code we use to subset our shapefiles is below.

```
library(rgdal)
library(rgeos)
library(sp)
library(foreign)

pop <- readOGR('insert directory for population shapefiles',layer='Merged')

#select city and national capitals
```

```
pop.city <- pop[((pop@data$Type=='city') | (pop@data$Type=='national_capital')),]
pop.city <- pop.city[!(is.na(pop.city@data$Type)),]
#write out new shapefile and database file
writeOGR(pop.city,'directory you want new shapefile to be','pop_test',
         driver = 'ESRI Shapefile',overwrite_layer = T)
write.dbf(pop.city@data,'directory you want/pop_test.dbf')

#now for roads
roads <- readOGR('insert directory for roads shapefiles',layer = 'MergedRoads')
#select road types that you want
roads.new <- roads[((roads@data$TYPE == 'Motorway') | (roads@data$TYPE == 'Trunk') |
                    (roads@data$TYPE == 'Primary') | (roads@data$TYPE == 'Secondary') |
                    (roads@data$TYPE == 'Tertiary') | (roads@data$TYPE == 'Motorway Link') |
                    (roads@data$TYPE == 'Trunk Link') | (roads@data$TYPE == 'Primary Link') |
                    (roads@data$TYPE == 'Secondary Link')),]
#write new shapefile and database file
writeOGR(roads.new,'insert directory','Roads_Small',driver = 'ESRI Shapefile',
         overwrite_layer = T)
write.dbf(roads.test@data,'direcotry you want/Roads_Small.dbf')
```

Task 4 - Ensure proper projections

Now, we ensure the population and road shapefiles we will use to build our network are in the same Coordinate Reference System (CRS). Specifically, for our method to work, we need to ensure that each of the shapefiles is in a **PROJECTED** coordinate system. The graphic below explains how Geographic Coordinates can be projected into a given projected coordinate system. For our analysis, we used the Lambert Azimuthal Equal Area projection.

Re-projecting the shapefiles can be done in R or in any GIS software. We recommend conducting this process in a GIS as these systems maintain all of the necessary information on hundreds of different Coordinate Reference Systems. In R, this requires knowledge of the exact string of the CRS we desire to use.

To re-project a shapefile in QGIS, the below steps are followed:

1. After opening the 'Processing Toolbox' from the 'Processing' menu navigate to 'QGIS geoalgorithms', 'Vector general tools', and select 'Reproject Layer'.

2. Select the layer to re-project and then select the ellipsis next to the Target CRS text box.

3. Search for the desired CRS and select it.

4. Choose to save the re-projected layer to an output file or open a temporary layer and then run the algorithm. The output will display in the window.

Task 5 - Clean Road Network

Now that the shapefiles are in the desired projection and merged together, it is necessary to clean the files further by simplifying the road network structure through removing duplicates, simplifying curves, and snapping corners together. Simplifying the road structure makes the network build much more efficient.

Any GIS system can be used to conduct these steps, but the functions may vary. We choose to use GRASS due to the ease of interface with Python. GRASS is another free, open-source, GIS system. The Python script used to clean our road network as we desired and is depicted below. The script **requires** the use of a **Projected Coordinate Reference System** to work.

To run this script in GRASS, the below steps are followed:

1. Load the road shapefile we need to clean. **Ensure the GRASS session that is opened is in the same projection as the road shapefile**.
2. Open the GRASS python editor, open the below GRASS script and press the run button.

The run time of the script depends on the size of the road network. Using only the primary and secondary roads in Iraq and Syria, the script runs for approximately 9 minutes. The output of the script is a polygon of the road network, which we use for the network build. The full script is depicted below.

```
#!/usr/bin/env python
import grass.script as grass

def main():
    '''Change the 'stateslist' names to reflect the files you are WRITING TO DISK based '''
    stateslist = ['INSERT LAYER NAME(S)']
    #can use this script on a list of layers as well
    for state in stateslist:
            stateroads = 'LAYER NAME';
            state_out = 'INSERT DIRECTORY LOCATION'

            ## Command string 1: begin cleaning
          ''' breaks lines, removes duplicates, removes lines of zero length, removes dangles'''
            grass.run_command('v.clean',
                              input = stateroads,
                              output = 'temp', type = 'line',
                              tool = ['break', 'rmdupl', 'rmline',
                                      'rmdangle', 'rmdangle', 'rmdangle'],
                              thres = [0.00, 0.00, 0.00, 1500.00, 1500.00, 1500.00],
                              overwrite = True)

            ## Command string 2: simplification
            ###reduces vertices (breaks in lines by 20%)
        grass.run_command('v.generalize',
                          input = 'temp',
                          output = 'temp2',
                          method = 'douglas_reduction',
                          threshold = 5000,
                          reduction = 20,
                          overwrite = True)

            ## Command string 3: continue cleaning after simplification
        '''breaks lines, snaps the together, removes duplicates, removes lines of 0 length'''
            grass.run_command('v.clean',
                              input = 'temp2',
                              output = 'temp3',
                              type = 'line',
                              tool = ['break', 'snap', 'break', 'rmdupl', 'rmline', 'break'],
                              thres = [0.0, 50.0, 0.0, 0.0, 0.0, 0.0],
                              overwrite = True)

            ## Command string 4: remove <50m lines
            grass.run_command('v.edit',
                              map = 'temp3',
                              type = 'line',
                              tool = 'delete',
                              threshold = [-1, 0, -50],
```

```
                            query = 'length')

            ## Command string 5: finish cleaning
            grass.run_command('v.clean',
                            input = 'temp3',
                            output = 'temp4',
                            type = 'line',
                            tool = ['break', 'rmdupl', 'rmline', 'rmdangle'],
                            thres = [0.0, 0.0, 0.0, 1000.0],
                            overwrite = True)

            ## Command string 6: buffer around road network (one big polygon)
            grass.run_command('v.buffer',
                            input = 'temp4',
                            output = 'temp6',
                            distance = 51,
                            overwrite = True)

            ## Command string 7: write out
            grass.run_command('v.out.ogr',
                            input = 'temp6',
                            output = state_out ,
                            format = 'ESRI_Shapefile',
                            overwrite = True)

if __name__ == '__main__':
    main()
```

In the case of the Iraq and Syria road network, the output looks like the plot below.

Network Build

With the data prepared for processing, the construction of the network with the towns and roads is performed. The below steps are used to accomplish this task.

Task 1 - Load required packages and open parallel session

```
if (!require("pacman")) install.packages("pacman")
pacman::p_load(rgdal, maptools, raster, rgeos, sp
               , data.table, stringr, gsubfn, stringdist
               , foreach, doSNOW, igraph, Matrix, moments
               , geosphere, geojsonio, rmapshaper
               , countrycode)
# open parallel session if required
registerDoSNOW(cl <- makeCluster(6, type = 'SOCK'))
```

Task 2 - Initialize variables and set working directory

Below, we initialize some of the variables that will be used. The distance variables can be changed as required.

```
roadsloc <- 'INSERT DIRECTORY LOCATION FOR ROADS SHAPEFILE'
roadsfile <- 'NAME OF ROAD SHAPEFILE'
townsloc <- 'INSERT DIRECTORY LOCATION FOR TOWNS SHAPEFILE'
```

```
townsfile <- 'NAME OF TOWN SHAPEFILE'
roadsbufloc <- 'INSERT DIRECTORY LOCATION FOR GRASS CLEANING OUTPUT FROM ABOVE'
roadsbuffile <- 'NAME OF GRASS CLEANING OUTPUT'
#DESIRED PROJECTION
aeaproj <- CRS("+proj=laea +lat_0=90 +lon_0=0 +x_0=0 +y_0=0
               +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0")
roadsnapdist <- 2501 #DISTANCE USED FOR SNAPPING TOWNS TO ROADS
townthindist <- 2501 #DISTANCE USED FOR THNNING TOWNS
townbufferdist <- 5001 #DISTANCE USED FOR TOWN BUFFERS

setwd('INSERT DIRECTORY LOCATION FOR NETWORK OUTPUT')
```

Task 3 - Load road and town shapefiles

After loading the roads and towns we recommend plotting them to ensure that everything "makes sense" visually. The roads are loaded as a Spatial Lines Data Frame and the towns are loaded as a Spatial Points Data Frame.

```
roads <- readOGR(roadsloc, roadsfile)
plot(roads)

towns <- readOGR(townsloc, townsfile)
plot(towns,add=T,col='red')
```

Task 4 - Snap towns to nearest road

Since the lat/long coordinates for each town may not fall exactly on the road network, we need to "snap" each of the towns to the nearest road. This is done by building a buffer around the town and finding the nearest road in that buffer. After the buffer is built, the town is moved to the closest road inside of the buffer. This process breaks up the towns into groups to allow the snapping to be conducted in parallel, improving the speed of execution. If the towns are not within the desired distance (in this case 2.5km) the town is dropped. The output from this task is a new Spatial Points Data frame containing the towns and their "updated" coordinates.

```
#create indices for parallel processing
#break into groups of 1000
if(length(towns) > 1000){
  end_idx <- seq(1000,length(towns),by = 1000)
  start_idx <- c(seq(1, length(towns)-999, by = 1000), last(end_idx)+1)
  end_idx <- c(end_idx, length(towns))
} #break into groups of 100
else if(length(towns) <= 1000 & length(towns) > 100){
  end_idx <- seq(100,length(towns),by = 100)
  start_idx <- c(seq(1, length(towns)-99, by = 100), last(end_idx)+1)
  end_idx <- c(end_idx, length(towns))
} #break into groups of 10
else if(length(towns) <= 100){
  end_idx <- seq(10,length(towns),by = 10)
  start_idx <- c(seq(1, length(towns)-9, by = 10), last(end_idx)+1)
  end_idx <- c(end_idx, length(towns))
}

if(last(start_idx) > last(end_idx)){
  start_idx <- start_idx[-length(start_idx)]
```

```
  end_idx <- end_idx[-length(end_idx)]
}

#foreach is a looping construct that supports parallel execution
snaptowns <- foreach(i = start_idx,
                     j = end_idx,
                     .combine = rbind,
                     .packages = 'maptools') %dopar% {
                       #assigns nearest line
                       snapPointsToLines(points = towns[i:j, ],
                                         lines = roads,
                                         maxDist = roadsnapdist)}

snaptowns$nearest_line_id <- NULL
#set new coordinates for towns
snaptowns@data[, c('X', 'Y')] <- coordinates(snaptowns)
towns@data[, c('X', 'Y')] <- coordinates(towns)
#create data table with only towns that are within the
#maxdist of a road and their new coordinates
townsdata <- data.table(rbind(snaptowns@data,towns@data[!towns$id %in% snaptowns$id,]))
#convert to spatial points dataframe
towns <- SpatialPointsDataFrame(cbind(townsdata$X, townsdata$Y)
                                , data = townsdata, proj4string = aeaproj)
towns$id <- 1:nrow(towns)
#plot to make sure it worked
plot(roads)
plot(towns,add=T,col='red')
```

Now we see that each town is located on the road network!

Task 5 - Thin Towns

If towns are too close together after they have been snapped, it will clutter up the resulting network graph. To mitigate this we remove towns if they are within a specific distance of another town. The output of this step is an updated Spatial Points Data Frame including only the remaining towns after eliminations have been performed. The town thin distance that was set in task 1 can be adjusted here as required.

```
#data prep
locs.df <- towns
lat <- which(names(locs.df) == 'X')
long <- which(names(locs.df) == 'Y')
id <- which(names(locs.df) == 'id')
#crete datafame with only lat,long,id
locs.long.lat <- as.data.frame(cbind(locs.df[[long]], locs.df[[lat]], locs.df[[id]]))
#find distance between all towns (distance matrix)
rec.df <- locs.long.lat
DistMat <- spDists(x = rec.df, longlat = F)
diag(DistMat) <- NA

#loop while there are towns that are too close
while (min(DistMat, na.rm = TRUE) < townthindist & nrow(rec.df) > 1) {
  #which towns are within set distance
  CloseRecs <- which(DistMat < townthindist, arr.ind = TRUE)[, 1]
  # Remove the town that has the most occurences of a distance less than set distance
```

```
  RemoveRec <- as.numeric(names(which(table(CloseRecs) == max(table(CloseRecs)))))
  #if there is more than 1 that has the most occurences, pick one to remove
  if (length(RemoveRec) > 1) {
    RemoveRec <- sample(RemoveRec, 1)
  }
  #update dataframe and distance matrix
  rec.df <- rec.df[-RemoveRec, ]
  DistMat <- DistMat[-RemoveRec, -RemoveRec]
  if (length(DistMat) == 1) {
    break
  }
}

#update spatial points dataframe
colnames(rec.df) <- c("Longitude", "Latitude", 'id')
thinnedtowns <- rec.df
towns <- towns[towns$id %in% thinnedtowns$id, ]

plot(roads)
plot(towns,add=T,col='red')
```

Now that the towns are thinned, we are left with the towns located on the road network above.

Task 6 - Determine connections

A town is connected to another town if there is a direct route between the two towns. In other words, it can reach that town on the road network without going through any other town. To build a network graph, we needed to figure out which towns are connected to one another.

Town buffers

First we build buffers around each town and load the road polygon shapefile from our GRASS output.

```
#create buffers around town
townbuffers <- gBuffer(towns, byid = F, width = townbufferdist)
#load shapefile from GRASS oututput
roadbuffers <- readOGR(roadsbufloc, roadsbuffile)
#plot to make sure everything looks ok
plot(roadbuffers)
plot(townbuffers,col='red',add=T)
```

Create Road Chunks

The next step is to take the difference between the road network and the towns with buffers. This process will essentially "cookie cut" the town buffers out of the road network.

```
#take difference
roadbuffers <- gDifference(roadbuffers, townbuffers)
#plot to see output
plot(roadbuffers)
```

After taking the difference, our road network now looks like the below plot.

We can now see how the towns are "cut" out of the road network facilitating the next step which is to separate these "chunks" into individual road segments. We can accomplish this task by disaggregating the above polygon into several polygons (one for each "chunk").

```
#disag road chunks
roadbuffers <- sp::disaggregate(roadbuffers)
# convert to Spatial Polygons dataframe
roadbuffers <- SpatialPolygonsDataFrame(roadbuffers,
                                        data = data.frame('id' = 1:length(roadbuffers)))
```

Now we have separated the road network into a sequence of road "chunks" with a specific id number.

Find intersections

With the road chunks created in the previous step, we use town buffers to determine which road chunks the town buffers intersect. To do this, we need to expand the original town buffer distance to ensure that the buffers actually intersect parts of the road network. We adjust the distance, as required, to see the results For our example we added 1 km to the original town buffer distance.

```
#create new expanded town buffers
townbuffers <- gBuffer(towns, byid = T, width = townbufferdist+1000)
#determine which towns and roads intersect
ints <- gIntersects(roadbuffers, townbuffers, byid = T)
#build data table with info about intersections
townroads <- data.table(which(ints == T, arr.ind = T))
setnames(townroads, c('town', 'road'))
townroads[, town := townbuffers$id[town]]
townroads[, road := roadbuffers$id[road]]
setkeyv(townroads, c('town', 'road'))
#create matrix of the intersections
roadstownsmat <- Matrix(0, nrow = length(towns), ncol = length(roadbuffers), sparse = T)
roadstownsmat[as.matrix(townroads[, list(town, road)])] <- 1
```

The above code gives us a sparse town-road intersection matrix that outlines each intersection between a town and road chunk. From the data, we can now determine connectivity between towns.

Town connections

If two towns intersect the same road chunk, then we register the two towns as being connected. To retrieve this information from the town-road intersection matrix, we multiply the matrix by its transpose to get a matrix of connected towns.

```
townsmat <- roadstownsmat %*% t(roadstownsmat)
```

Task 7 - Build Adjacency Matrix

We want to build the output network graph from an adjacency matrix that tells us the connections of each town. We use the matrix built above to create our adjacency matrix by following the below steps. 1. Set the diagonal of our towns matrix to 0. We don't want to create a connection between our towns and themselves. 2. Normalize the matrix. If it connects with another town more than once (from more than one road chunk) we say it connects once. 3. Set upper triangular portion of the matrix to 0. All of the information we need is in the lower triangular portion of the matrix.

```
#set diagonal to zero
diag(townsmat) <- 0
#normalize
```

```
townsmat[which(townsmat > 1)] <- 1
#set upper triangular portion to 0.
townsmat[which(upper.tri(townsmat))] <- 0
```

Now we have an adjacency matrix to build the network graph!

Task 8 - Build Graph and add attributes

Using the adjacency matrix, we can build our network graph. Once the graph is built, we add attributes to both the nodes and edges. We use the information from our original data set to add attributes to the nodes (towns). We also calculate the great circle distance between the nodes and use the distance as a weight on the edges.

```
#build graph
townsnet <- graph.adjacency(townsmat, mode = 'undirected', diag = F)
#add attributes to nodes
townsnet <- igraph::set.vertex.attribute(townsnet, 'name',
                                         value = towns$name)
townsnet <- igraph::set.vertex.attribute(townsnet, 'x',
                                         value = coordinates(towns)[,1])
townsnet <- igraph::set.vertex.attribute(townsnet, 'y',
                                         value = coordinates(towns)[,2])
townsnet <- igraph::set.vertex.attribute(townsnet, 'LAT',
                                         value = towns$LAT)
townsnet <- igraph::set.vertex.attribute(townsnet, 'LON',
                                         value = towns$LONG)
townsnet <- igraph::set.vertex.attribute(townsnet, 'DSG',
                                         value = as.character(towns$DSG))
townsnet <- igraph::set.vertex.attribute(townsnet, 'CC1',
                                         value = as.character(towns$CC1))
townsnet <- igraph::set.vertex.attribute(townsnet, 'ADM1',
                                         value = towns$ADM1)
townsnet <- igraph::set.vertex.attribute(townsnet, 'FULL_NAME',
                                         value = as.character(towns$FULL_NAME_))
townsnet <- igraph::set.vertex.attribute(townsnet, 'Ethnicity',
                                         value = as.character(towns$Ethnicity))
townsnet <- igraph::set.vertex.attribute(townsnet, 'Sect',
                                         value = as.character(towns$Sect))
townsnet <- igraph::set.vertex.attribute(townsnet, 'Type',
                                         value = as.character(towns$Type))
townsnet <- igraph::set.vertex.attribute(townsnet, 'Population',
                                         value = as.numeric(as.character(towns$Population)))
# calc distance between nodes.
dist = spDists(cbind(vertex_attr(townsnet,'LAT'),vertex_attr(townsnet,'LON')),
               longlat = T)
#get edgelist
edgelist = as_edgelist(townsnet)
# find distances for each edge in our network
distance = rep(0,length(edgelist[,1]))
for (row in seq(1,length(edgelist[,1]),1)){
  #print(row)
  distance[row] = test[as.numeric(edgelist[row,1]),as.numeric(edgelist[row,2])]
}
edgelist = cbind(edgelist,distance)
```

```
#assign distance as weidght to edges
edge_attr(townsnet,'distance') <- distance
#plot to check out the network
#have to make the edge with a fraction of distance to see edges
plot(townsnet,edge.width=(E(townsnet)$distance/200))

## Close parallel session
stopCluster(cl)

#save network
save(townsnet, file = 'NAMEOFNETWORK.Rdata')
```

Now we have a graph that contains all of the information that we would like and it is in the form of an igraph object that we can use to conduct Network Analysis. Unfortunately, the basic plot above does not provide us with a great visualization of the network. The next section accomplishes this challenge.

Construct Interactive Plot

With the igraph object constructed, we build an interactive Leaflet plot using the relevant information contained in the network. The interactive plot offers a great visual display of all information pertinent to the user. Additionally, the plot provides a realistic visual layout of the network graph in its geographical setting.

- The important thing to keep in mind when constructing the plot is the use of the Spatial Lines object to build the arcs in the network. Of the following code, two data structures are necessary to display the information needed for the interactive plot:
  - "vert" data frame: contains all information from the igraph object pertaining to each node
  - "edges" Lines Object: contains all necessary information pertaining to the arcs in the network

```
# Step 1: Import the necessary packages
library(igraph)
library(leaflet)
library(sp)
library(htmlwidgets)

# Step 2: Bring in the igraph object from the working directory
load("insert directory where you have saved the igraph object")

# Step 3: Plot the network to ensure all network information
#was successfully constructed
plot(townsnet)

# Step 4: Construct "weight_list" in order to weight the
#arcs in the network based on distance
weight_list<-edge_attr(townsnet,'distance')/200

# Step 5: Build a dataframe containing all node information
#that the user wants displayed in the plot
gg <- get.data.frame(townsnet, "both")
lat_list<-gg$vertices$LAT
lon_list<-gg$vertices$LON
name_list<-gg$vertices$name
from_list<-as.character(gg$edges$from)
```

```
to_list<-as.character(gg$edges$to)
df<-data.frame("from"=from_list, "to"=to_list)

# Step 6: Create a dataframe containing the geographic
#coordinates of each node
meta<-data.frame("name"=name_list, "lon"=lon_list, "lat"=lat_list)

# Step 7: Create a dataframe containing the vertice
#information from the igraph object
vert <- gg$vertices

# Step 8: Create a SpatialPoints Dataframe using the
#meta dataframe from Step 6
coordinates(meta) <- ~lon+lat

# Step 9: Construct "edges" dataframe that contains the
#"from" and "to" nodes
edges <- data.frame('from'=as.character(from_list),
                    'to'=as.character(to_list), stringsAsFactors = FALSE)

# Step 10: Iterate through the "edges"" dataframe to contruct
#a SpatialLines object containing a list of lists of the coordinates of the "from" and "to" nodes
edges <- lapply(1:nrow(edges), function(i) {
  as(rbind(meta[meta$name == edges[i, "from"], ],
           meta[meta$name == edges[i, "to"], ]),
     "SpatialLines")
})

# Step 11: Iterate through the SpatialLines object and set
#the list "ID's" to the "from" node
for (i in seq_along(edges)) {
  edges[[i]] <- spChFIDs(edges[[i]], as.character(i))
}

# Step 12: Using rbind, convert the SpatialLines object to
#a Lines object for plotting
edges <- do.call(rbind, edges)

# Step 13: Execute the Leaflet plot code
x<-leaflet(vert) %>%
  addTiles() %>%
  addAwesomeMarkers(data = vert,
                    popup= paste("Name:", vert$FULL_NAME, "<br>",
                                 "Node #:", vert$name, "<br>",
                                 "Population:", vert$Population, "<br>",
                                 "Ethnicity:", vert$Ethnicity, "<br>",
                                 "Sect:", vert$Sect)) %>%
  addPolylines(data = edges, weight=weight_list, opacity = weight_list, color = "red",
               popup=paste("Distance (km):", round(weight_list*200,2))) %>%
  addProviderTiles(providers$OpenStreetMap)

# Step 14: Save the Leaflet map to an html file for sharing
saveWidget(x, "x.html", selfcontained = TRUE)
```

We now have an Interactive plot of the network that is much more helpful to the analyst. Each node contains specific information about itself and the edges display the distance between two nodes.

Simple Network Analysis

Now that we have a network graph structure with all of the attributes for the towns and roads, we can apply network and graph theory to our analysis. Using network and graph theory in this type of analysis serves to enhance, from a mathematical standpoint, the analysis from other subject matter experts, like political scientists or historians. The R package igraph, with hundreds of built-in network analysis functions, can be used to verify assumptions made about the network and aid in making further predictions.

From related works in this field of study, we know that "locations with high degree and betweenness centrality in the road network are significantly more likely to be fought over" (Hammond 2017). With this in mind, we know the centrality measures will be important in this network. Particularly, when we are trying to assess the strategic value of a node or arc. We will demonstrate this with degree and betweenness centrality.

Degree Centrality: The number of ties that a single node has **Betweenness Centrality**: The number of times a node acts as a bridge along the shortest path between two other nodes.

In the image above the two gray nodes (D,L) represent nodes with high degree centrality. The black node (H) represents a node with high betweenness centrality.

Using igraph, we can determine the centrality measures for our network, and then use the graphical representation to verify. The below code will calculate the node that has the highest measure of betweenness centrality in our network.

```
which.max(as.matrix(betweenness(out_network,directed = T
                                ,weights = edge_attr(out_network,'distance'))))
```

```
## [1] 833
```

We can also find the node with the highest degree centrality in the network

```
which.max(as.matrix(degree(out_network,normalized = T)))
```

```
## [1] 4
```

If we look at the interactive graph above, we can see that the nodes from our centrality calculations make perfect sense from the analyst's standpoint. As our network is defined, these nodes do appear to have strategic value, and would be important in our analysis of the terrorist network.

We can also use igraph to calculate the shortest path between two nodes. The below code shows the shortest path between the node with the highest degree centrality, and the node with the highest betweenness centrality;

```
shortest_paths(out_network,from = '833',to ='4',mode='out',
               weights = edge_attr(out_network,'distance'),output='vpath')$vpath[[1]]
```

```
## + 3/857 vertices, named, from 3ac3d75:
## [1] 833 196 4
```

These are only a few examples of the functions available in the igraph package. For more detailed information on igraph, visit the following website: http://igraph.org/r/doc/.

Conclusion

With this tutorial, we have shown how to take a geographic database and convert it into a network graph structure. We also demonstrated how to make an interactive display of the network graph once it has been

constructed. By using a network graph, analysts can use graph algorithms and techniques to find new information that may be more difficult using only the geographic database. Finally, we demonstrated some simple network analysis that can be done once you have a network graph. The kind of analysis presented in the tutorial has the potential to allow for different insights into existing problems or analysis.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B:
Overall Model Performance

The following table shows each one of our models cumulative classification accuracy, change accuracy, and associated 95% confidence intervals. Each algorithm is broken down by each of our five cases and whether or not graph features were used. The classification accuracy shown is the cumulative accuracy at the end of our rolling horizon design for each model with the 95% binomial confidence interval to the right. The change accuracy is the accuracy of the model when it predicted a change. Again, this is the cumulative change accuracy at the end of the rolling horizon design.

Our five cases are defined below:

1. **Binary Classification: Base Case**
   In our base case, a node is classified as either being in or out of ISIS territory at each time period in our design. Control, attack, and support zones are aggregated in this case to represent ISIS territory.

2. **Binary Classification: Control**
   In this case, a node is classified as either being in or out of ISIS control territory.

3. **Binary Classification: Attack**
   In this case, a node is classified as either being in or out of ISIS attack territory.

4. **Binary Classification: Support**
   In this case, a node is classified as either being in or out of ISIS support territory.

5. **Multinomial Classification**
   Our last case consists of classifying each node at one of four levels of ISIS territory:
   - Not in ISIS territory
   - Control
   - Attack
   - Support

Algorithm	Case	Graph Features	Classification Accuracy	Classification Accuracy 95% Confidence Interval		Change Accuracy	Change Accuracy 95% Confidence Interval	
Linear Regression	1	No	0.9132	0.8944	0.9321	0.4311	0.3785	0.4836
		Yes	0.9119	0.893	0.9309	0.4240	0.3775	0.4705
	2	No	0.9809	0.9718	0.9901	0.4046	0.3315	0.4778
		Yes	0.9817	0.9727	0.9907	0.4286	0.3504	0.5067
	3	No	0.9638	0.9513	0.9763	0.4964	0.4484	0.5444
		Yes	0.9626	0.9499	0.9753	0.4735	0.4232	0.5239
	4	No	0.9540	0.9399	0.968	0.4775	0.4256	0.5294
		Yes	0.9535	0.9394	0.9676	0.4670	0.4147	0.5194
	5	No	0.9045	0.8848	0.9242	0.4070	0.3695	0.4444
		Yes	0.9031	0.8833	0.9229	0.4003	0.3621	0.4385
Classification Trees	1	No	0.933	0.9163	0.9497	0.4113	0.3613	0.4613
		Yes	0.9342	0.9176	0.9508	0.4213	0.3605	0.482
	2	No	0.9838	0.9753	0.9922	0.4221	0.3535	0.4907
		Yes	0.9839	0.9755	0.9924	0.4308	0.3613	0.5003
	3	No	0.9727	0.9618	0.9836	0.5	0.4459	0.5541
		Yes	0.973	0.9622	0.9839	0.5126	0.4576	0.5675
	4	No	0.9648	0.9524	0.9771	0.3926	0.331	0.4541
		Yes	0.9641	0.9516	0.9766	0.3909	0.3363	0.4455
	5	No	0.9278	0.9105	0.9451	0.3947	0.3479	0.4416
		Yes	0.9282	0.9109	0.9455	0.4357	0.3914	0.48
Random Forest	1	No	0.9167	0.8981	0.9352	0.5028	0.4602	0.5455
		Yes	0.9165	0.8979	0.935	0.5	0.4629	0.5371
	2	No	0.982	0.9731	0.9909	0.4464	0.3713	0.5216
		Yes	0.9811	0.972	0.9903	0.4148	0.342	0.4876
	3	No	0.9665	0.9545	0.9786	0.5678	0.509	0.6265
		Yes	0.9669	0.955	0.9789	0.5719	0.5158	0.628
	4	No	0.9562	0.9425	0.9699	0.5238	0.472	0.5756
		Yes	0.9551	0.9413	0.969	0.5011	0.4543	0.548
	5	No	0.9072	0.8878	0.9267	0.4449	0.4078	0.482
		Yes	0.9077	0.8883	0.9271	0.4595	0.4242	0.4948
Adaboost	1	No	0.9394	0.9234	0.9554	0.6061	0.5558	0.6563
		Yes	0.9387	0.9226	0.9548	0.5714	0.5245	0.6184
	2	No	0.9863	0.9785	0.9941	0.5663	0.4969	0.6357
		Yes	0.9862	0.9784	0.994	0.5622	0.4907	0.6337
	3	No	0.9772	0.9672	0.9872	0.692	0.6362	0.7478
		Yes	0.9768	0.9667	0.9868	0.6631	0.6076	0.7185
	4	No	0.9701	0.9587	0.9815	0.6018	0.5489	0.6547
		Yes	0.9704	0.9591	0.9818	0.6121	0.5595	0.6647
	5	No	-	-	-	-	-	-
		Yes	0.9346	0.9181	0.9512	0.5484	0.5071	0.5897

List of References

Ahuja R, Magnanti T, Orlin J (2013) *Network Flows* (Pearson Education Ltd, United Kingdom).

Alfaro E, Gámez M, García N (2013) adabag: An R package for classification with boosting and bagging. *Journal of Statistical Software* 54(2):1–35, http://www.jstatsoft.org/v54/i02/.

Bivand R, Pebesma EJ, Virgilio GR (2013) *Applied Spatial Data Analysis with R* (Springer, New York, NY).

Brown D, Dalton J, Hoyle H (2004) Spatial forecast methods for terrorist events in urban environments. *Intelligence and Security Informatics* (Tucson, AZ), 426–435.

Buhaug H, Rod JK (2006) Local determinants of African civil wars, 1970-2001. *Political Geography* 25(3):315–335.

Burt RS (2004) Structural holes and good ideas. *American Journal of Sociology* 110(2):349–399, http://dx.doi.org/10.1086/421787.

Cronin AK (2015) ISIS is not a terrorist group; why counterterrorism won't stop the latest Jihadist threat. *Foreign Affairs* 94(2):87–98.

Csardi G, Nepusz T (2006) The igraph software package for complex network research. *InterJournal* Complex Systems:1695, http://igraph.org.

de Smith MJ, Goodchild MF, Longley PA (2009) *Geospatial Analysis: A Comprehensive Guide to Principles Techniques and Software Tools,* 3rd ed. (Matador, Leicester, UK).

Di Salvatore J (2016) Inherently vulnerable? Ethnic geography and the intensity of violence in the Bosnian civil war. *Political Geography* 51:1–14.

DIary (2017) Titanic survival decision tree. https://commons.wikimedia.org/wiki/File:Titanic_Survival_Decison_Tree_SVG.png.

Ding F, Ge Q, Jiang D, Fu J, Hao M (2017) Understanding the dynamics of terrorism events with multiple-discipline datasets and machine learning approach. *PLoS ONE* 12(6):1–11.

Djexplo (2018) Geographic coordinate system. https://commons.wikimedia.org/wiki/File%3ALatitude_and_Longitude_of_the_Earth.svg.

ESRI Technical Support (2018) Esri support GIS dictionary. https://support.esri.com/en/other-resources/gis-dictionary/.

Faraway JJ (2016) *Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models*. Texts in Statistical Science (CRC Press, Taylor & Francis Group), 2nd edition.

Ferenc J (2008) Predicting target selection by terrorists: a network analysis of the 2005 London underground attacks. *International Journal of Critical Infrastructures* 4(1/2):206–214.

Forrest C (2016) ISIS sanctuary map. http://www.understandingwar.org/project/isis-sanctuary-map.

Fotheringham AS, Rogerson PA (2009) *The SAGE Handbook of Spatial Analysis* (SAGE Publications Ltd, London, UK).

Frantzman SJ (2017) "By, with, and through": How the U.S. led coalition defeated the Islamic State in Iraq using tactics without coherent strategy for confronting Iranian influence. *Middle East Review of International Affairs* 21(3).

Glenn C (2017) Timeline: The rise, spread and fall of the Islamic State. https://www.wilsoncenter.org/article/timeline-the-rise-spread-and-fall-the-islamic-state.

GRASS Development Team (2017) Geographic resources analysis support system software, version 7.2. http://grass.osgeo.org.

Hammond J (2018) Maps of mayhem: Strategic location and deadly violence in civil war. *Journal of Peace Research* 55(1):32–46.

Harary F (1994) *Graph Theory* (Addison-Wesley, Reading, MA).

Hegre H, Otsby G, Raleigh C (2009) Poverty and civil war events, a disaggregated study of Liberia. *Journal of Conflict Resolution* 53(4):598–623.

Hijmans RJ (2016) geosphere: Spherical trigonometry. https://CRAN.R-project.org/package=geosphere, r package version 1.5-5.

Hijmans RJ (2017) raster: Geographic data analysis and modeling. https://CRAN.R-project.org/package=raster, r package version 2.6-7.

Hills A (1997) Warlords, militia and conflict in contemporary Africa: A re-examination of terms. *Small Wars & Insurgencies* 8(1):35–51.

Huddleston SH, Brown D (2009) A statistical threat assessment. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 39(6):1307–1315.

Huddleston SH, Brown GG (2018) Machine learning. *Analytics Body of Knowledge*. Forthcoming.

Huddleston SH, Fox J, Brown DE (2012) Mapping gang spheres of influence. *Crime Mapping: A Journal of Research and Practice* 4(2):39–67.

Joint Chiefs of Staff (JCS) (2014) Joint intelligence preparation of the operational environment. JP 2-01.3, Washington, DC. https://fas.org/irp/doddir/dod/jp2-01-3.pdf.

Joint Chiefs of Staff (JCS) (2017) Joint and national intelligence support to military operations. JP 2-01, Washington, DC. www.jcs.mil/Portals/36/Documents/Doctrine/pubs/jp2_01_20170705v2.pdf.

Kleinberg JM (1999) Authoritative sources in a hyperlinked environment. *Journal of The ACM - JACM* 46(5):604–632.

Kohavi R, Provost F (1998) Glossary of terms journal of machine learning. http://ai.stanford.edu/~ronnyk/glossary.html.

Kolaczyk ED (2009) *Statistical Analysis of Network Data*. Springer Series in Statistics (Springer Science+Business Media LLC).

Kolaczyk ED (2014) Statistical analysis of network data. Lecture, Workshop on Private Analysis of Social Networks, Boston University, Boston, MA. https://www.bu.edu/cs/files/2014/05/Kolaczyk.pdf.

Latora V, Marchiori M (2004) How the science of complex networks can help develop strategies against terrorism. *Chaos Solitions & Fractals* 20:69–75.

Liaw A, Wiener M (2002) Classification and regression by randomforest. *R News* 2(3):18–22, http://CRAN.R-project.org/doc/Rnews/.

Mattis J (2018) Summary of the 2018 National Defense Strategy of the United States of America. https://www.defense.gov/Portals/1/Documents/pubs/2018-National-Defense-Strategy-Summary.pdf.

McColl RW (1969) The insurgent state: Territorial bases of revolution. *Annals of the Association of American Geographers* 59(4):613–631.

Medina RM, Siebeneck LK, Hepner GF (2011) A geographic information systems (GIS) analysis of spatiotemporal patterns of terrorist incidents in iraq, 2004-2009. *Studies in Conflict & Terrorism* 34(11):862–882.

Miklavic S, Milanic M (2011) Equistable graphs, general partition graphs, triangle graphs, and graph products. *Discrete Applied Mathematics* 159(11):1148–1159.

Miller H (2015) Data-driven geography. *GeoJournal* 80:449–461.

National Consortium for the Study of Terrorism and Responses to Terrorism (START) (2016) Global terrorism database. https://www.start.umd.edu/gtd.

National Geospatial Intelligence Agency GEOnet Names Server (GNS) (2017) Description of names files for countries and territories format. http://omap.africanmarineatlas.org/BASE/data/gazetteer/geonet/nga_ngs_format.htm.

Nemeth SC, Mauslein JA, Stapley C (2014) The primacy of the local: Identifying terrorist spots using geographic information systems. *The Journal of Politics* 76(2):304–317.

Obama B (2014) Statement by the President on ISIL. https://obamawhitehouse.archives.gov/the-press-office/2014/09/10/statement-president-isil-1.

O'Loughlin J, Witmer FDW (2011) The localized geographies of violence in the North Caucasus of Russia, 1999-2007. *Annals of the Association of American Geographers* 101(1):178–201.

OpenStreetMap Contributors (2017) OpenStreetMap. https://www.openstreetmap.org/.

QGIS Development Team (2017) QGIS geographic information system. http://qgis.osgeo.org.

Raleigh C, Hegre H (2009) Population size, concentration, and civil war. A geographically disaggregated analysis. *Political Geography* 28:224–238.

Ramm F (2017) OpenStreetMap data in layered GIS-format. https://download.geofabrik.de/osm-data-in-gis-formats-free.pdf.

Reisio (2018) Map projection-Eckert IV. https://upload.wikimedia.org/wikipedia/commons/f/f1/Map_projection-Eckert_IV.png.

Remahl D (2005) Graph. https://commons.wikimedia.org/wiki/File:6n-graf.png.

Schutte S, Weidmann NB (2011) Diffusion patterns of violence in civil wars. *Political Geography* 30:143–152.

Sprusansky D (2014) Understanding ISIS: Frequently asked questions. *The Washington Report on Middle East Affairs* 33(7):19–20.

Valente T, Fujimoto K (2010) Bridging: Locating critical connectors in a network. *Social Networks* 32(3):212–220, http://dx.doi.org/10.1016/j.socnet.2010.03.003.

Vanderzee A, Fletcher C, Strickland C (2017) Network conversion tutorial. Unpublished tutorial, Naval Postgraduate School, Monterey, CA.

Zhou ZH (2012) *Ensemble Methods: Foundations and Algorithms*. Machine Learning & Pattern Recognition Series (CRC Press, Taylor & Francis Group).

Zhukov YM (2012) Roads and the diffusion of insurgent violence: The logistics of conflict in Russia's North Caucasus. *Political Geography* 31:144–156.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California