VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

J Component Report

Russia Ukraine War Analysis

Team members:

Manvik Sreedath 20BCE1479

Aryan Vigyat 20BCE1452

Kartik Deepu 20BCE1441

Submitted to:

Dr. Parvathi Pattabiraman

Abstract

The ongoing confrontation between Russia and Ukraine has killed a significant number of people and seriously damaged both military and civilian infrastructure. This analytical endeavor attempts to give a thorough and objective evaluation of the equipment and people loss brought on by the conflict.An extensive evaluation of the harm done to military hardware, including tanks, artillery, aircraft, and naval vessels, will be done as part of the analysis. It will also examine how civil infrastructure, including buildings, roads, bridges, and power plants, has been destroyed. The project will also keep track of how many people have died during the conflict—including military personnel and civilians—on both sides.We believe that our analysis will help to clarify the effects of the conflict on the area and its inhabitants. The project's findings can help guide attempts to resolve the conflict and lessen its consequences through policy and decision-making. It can also help in identifying places where rebuilding efforts need support, such as in restoring vital infrastructure and giving aid to impacted communities.

In order to examine the loss of soldiers and equipment as a result of the current conflict between Russia and Ukraine, this methodology makes use of a variety of data science techniques and technologies.

At the beginning, the XGBoost and AdaBoost algorithms will be used to predict the amount of harm done to military and civilian infrastructure as well as the number of casualties on both sides of the war. These machine learning approaches have been demonstrated to be extremely efficient in managing huge and complicated datasets, and they can assist in identifying patterns and trends that might not be immediately obvious.

Second, Tableau will be used to visualize the data, offering a user-friendly, interactive interface for examining the data and locating important insights. Charts, graphs, and maps will be used in the visualizations so that it will be possible to spot patterns and trends in the data as well as places that could need more research.

Lastly, data aggregation and numerous graphs that show various kinds of losses in terms of machinery, manpower, and other pertinent metrics will be created using the R programming language. R is a very versatile and strong programme that can be used for data cleansing, aggregation, and visualization. It may also assist in identifying important trends and insights that can guide decision-making.

In order to enable a quicker and more effective examination of the data, AutoML will also be used to automate the model selection and hyperparameter tuning processes. By automating these operations, we can make sure that our results are accurate and trustworthy by more simply identifying the best algorithms and methods for data analysis.

The scope of the project is to provide an in-depth analysis of the destruction of equipment and personnel resulting from the ongoing conflict between Ukraine and RussiaA significant amount of data will need to be gathered and analyzed for the project, including details on casualties, infrastructure damage to both military and civilian targets, and other pertinent metrics.The project will emphasize on several data science methods and tools, including XGBoost and AdaBoost for forecasting and prediction, Tableau for data visualization, R for data aggregation, and numerous plots that show various forms of losses in terms of equipment and personnel. Additionally, the process of choosing a model and fine-tuning its hyperparameters will be automated using AutoML. The project's overall goal is to give a full and accurate study of the equipment and people losses brought on by the Ukraine-Russian conflict, using a variety of data science approaches and technologies to offer insights, guide decision-making, and aid in recovery and rebuilding efforts.

The project's overall goal is to give a full and accurate study of the equipment and people losses brought on by the Ukraine-Russian conflict, using a variety of data science approaches and technologies to offer insights, guide decision-making, and aid in recovery and rebuilding efforts.

1. Introduction

When Russia seized Crimea from Ukraine in 2014, the confrontation between the two countries erupted. The international community broadly denounced this action, and tensions between Russia and Ukraine have since remained high. Since then, the conflict in eastern Ukraine has grown more intense, with both sides resorting to the use of force to further their goals.Both the military and civilian populations in the area have been devastated by the battle. There have been fatalities and thousands of people have been displaced as a result of the destruction of military and civilian infrastructure. Moreover, the conflict has had a substantial economic impact, negatively affecting regional trade and investment.

There have been numerous reports of torture, forced disappearances, and other atrocities committed by both sides during the battle. The involvement of pro-Russian separatist organizations, who are alleged to have violated Ukrainian residents' human rights, has made the situation even more difficult.The international community has played a significant role in attempting to end the conflict, which has required multiple rounds of discussions between Russia and Ukraine. A long-term resolution has not yet been reached, and the violence in the area is still raging.There is an urgent need for an accurate and thorough examination of the loss of soldiers and equipment as a result of the battle, given the continued effects it has on the military and civilian populations in the area. Such an analysis can give decision-makers the crucial data they need to decide how best to end the violence and support the region's recovery and reconstruction efforts. We have conducted an analysis endeavor to provide such information and assist regional decision-making.

For instance, knowing the amount of the harm done to crucial infrastructure, such as power plants and water treatment facilities, can help decision-makers choose how to best provide aid and support to enable the quickest possible restoration of these essential services. Similar to this, decision-makers can focus resources and efforts on meeting the urgent needs of affected populations by identifying the regions with the highest number of casualties.

Our analysis study intends to give decision-makers a thorough evaluation of the damage that has been done to troops and equipment as a result of the ongoing conflict. We will anticipate and predict the amount of harm done to military and civilian infrastructure, as well as the number of casualties on both sides of the fight, using machine learning methods like XGBoost and AdaBoost. Our research will be based

on the collection and examination of a sizable amount of data, including details on casualties, infrastructure damage to both military and civilian targets, and other pertinent metrics. With the help of several data visualization tools, including Tableau, the insights and conclusions from our analysis will be presented, allowing decision-makers to rapidly spot important trends and patterns in the data. Our ultimate objective is to assist decision-makers in making wise choices regarding the conflict's resolution and the support of the area's recovery and reconstruction efforts. We can ensure that resources and efforts are directed where they are most needed and that decision-makers have the knowledge they need to take decisive action by providing an accurate and thorough evaluation of the damage of equipment and personnel arising from the Ukraine-Russia conflict.

2. Review of literature

Research conducted by Filippos Giannakas, et al., in [1], examines and contrasts the accuracy of predictions made by XGBoost and deep neural networks with regard to the performance of teams.

This section of the study begins by offering an introduction to XGBoost and deep neural networks, including an overview of their structures as well as their advantages and disadvantages. The experiment that was conducted to assess the performance of the two models in predicting team performance is then described by the author. The author uses a variety of performance metrics to evaluate the models' effectiveness, including accuracy, precision, recall, the F1-score, and the area under the receiver operating characteristic (ROC) curve.

According to the findings of the experiment, XGBoost performed better than deep neural networks when it came to forecasting the performance of teams. It was able to achieve greater levels of accuracy, precision, recall, F1-score, and AUC-ROC. In this research, the various reasons for XGBoost's improved performance over deep neural networks are discussed. These reasons include the interpretability of the model and how it deals with missing data.

Finally, the conclusion of the paper highlights the practical significance of the findings for a variety of industries, including healthcare, marketing, and finance, and offers future research directions for the application of machine learning in these areas.

In general, the research presented in this study makes an important contribution to the growing subject of machine learning in predictive analytics.

The piece of academic work authored by Devon K. Barrow, et al., in [2] examine and contrasts various AdaBoost algorithms that are used for the combining of time series forecasts.

In the beginning of the study, the authors discuss the significance of forecast combining in time series analysis and the difficulties that are involved with this topic. After that, the author offers a summary of the AdaBoost algorithms and their various iterations, such as AdaBoost-SAMME, AdaBoost-SAMME.R, and AdaBoost.R2.

Following this, the author of the study outlines the experiment that was carried out in order to examine the effectiveness of the various AdaBoost algorithms when used to combine time series forecasts. The performance of the algorithms was evaluated by the author using a variety of measures, including mean squared error, mean absolute error, and symmetric mean absolute percentage error.

The findings of the experiment indicate that AdaBoost-SAMME.R fared better than the other algorithms in terms of the accuracy of its predictions and the efficiency with which it completed computations. In the study, the reasons for AdaBoost-SAMME.R's higher performance are discussed. These reasons include the programme's capacity to handle continuous and discrete data, as well as its regularization method.

Last but not least, the conclusion of the paper highlights the practical significance of the findings for time series prediction combinations and offers future research topics for increasing the performance of AdaBoost algorithms. This brings the total number of words in the paper to 10. The research makes a significant contribution to the study of time series analysis and the combination of forecasts in general, and this contribution is beneficial.

Ling Xiao's, et al, research article, [3], makes a proposal for an improved combination strategy that is based on the Adaboost algorithm. This approach is intended to be used for wind speed time series forecasting.

The first part of the paper is an overview that discusses the significance of predicting wind speeds and the difficulties that are connected with doing so. After that, the author goes on to discuss the Adaboost algorithm and its many iterations, such as the classic Adaboost method, AdaBoost.M1, and AdaBoost.R2.

Next, the author of the research offers an improved combination strategy that is based on the Adaboost algorithm. This approach, which is named IWAC (Improved Weighted

Average Combination), assigns different weights to the basic models depending on how well they perform. The performance of IWAC was compared to that of other combination methods using a variety of assessment measures, such as mean squared error, mean absolute error, and root mean squared error.

The findings of the experiment reveal that IWAC performed better than the other combination methods in terms of prediction accuracy, which demonstrates the usefulness of using it for forecasting wind speed time series. In the paper, the reasons behind IWAC's improved performance are discussed. These reasons include its capacity to lessen the influence of data points that are considered to be outliers and its adaptability to a variety of data distributions.

In conclusion, this study highlights the practical implications of the suggested approach for wind power generation and recommends future research directions for the application of the Adaboost algorithm in time series forecasting. Finally, the work comes to a close by highlighting these practical implications. The research makes a significant contribution to the study of time series analysis and the combination of forecasts in general, and this contribution is beneficial. Research that was carried out by Filippos Giannakas and titled "XGBoost and Deep Neural Network Comparison: The Case of Teams' Performance" compares and contrasts the accuracy of predictions provided by XGBoost and deep neural networks in relation to the performance of teams. The research was carried out by Filippos Giannakas.

This part of the research begins with an introduction to XGBoost and deep neural networks, which includes an overview of the topologies of both types of networks as well as an examination of both types' benefits and drawbacks. The author then moves on to detail the experiment that was carried out to evaluate how well the two models predicted the performance of the teams. When assessing the efficacy of the models, the author makes use of a range of performance criteria, such as accuracy, precision, recall, the F1-score, and the area under the receiver operating characteristic (ROC) curve.

When it came to anticipating the performance of teams, the results of the experiment showed that XGBoost performed significantly better than deep neural networks did. The levels of accuracy, precision, recall, F1-score, and AUC-ROC that it was able to achieve were all improved. This research delves into the myriad of factors that contribute to XGBoost's superior performance when compared to that of deep neural networks. The interpretability of the model is one of these reasons, as is the manner in which it handles missing data.

Last but not least, the conclusion of the paper underlines the practical implications of the findings for a range of industries, including healthcare, marketing, and finance, and it provides future research directions for the application of machine learning in these different fields. In general, the findings of the research that were provided in this paper constitute a significant contribution to the rapidly developing field of machine learning in predictive analytics.

Yan Wang's research work, [4] provides a mixed model for forecasting stock market volatility in time series data that is based on ARIMA and XGBoost. This model is used to predict stock market volatility.

This section of the article will begin by providing an outline of the significance of accurately predicting the volatility of the stock market as well as the difficulties associated with doing so. The author continues by elaborating on the ARIMA and XGBoost models, discussing their respective benefits and drawbacks.

Next, the author(s) of the study propose a mixed model that improves upon both the ARIMA and XGBoost models in terms of their ability to accurately predict the volatility of the stock market. When comparing the performance of the mixed model to that of the ARIMA and XGBoost models, the author utilised a variety of evaluation criteria, such as mean absolute error, mean squared error, and root mean squared error.

The findings of the experiment indicate that the mixed model fared better than the ARIMA and XGBoost models in terms of prediction accuracy, which demonstrates the mixed model's capability of accurately predicting the volatility of the stock market. In the study, the reasons for the greater performance of the mixed model are discussed. Some of these reasons include the ability of the mixed model to capture both linear and non-linear relationships in the data, as well as the ability of the mixed model to accommodate missing data points.

In conclusion, the study highlights the practical implications of the suggested technique for financial risk management and offers future research topics for the application of mixed models in time series forecasting. This brings the total number of key takeaways from the paper to five. In general, the research study makes an important contribution to the field of time series analysis as well as the prediction of stock market volatility.

[5] authored by Polikar, et al., offers an introduction to the idea of "ensemble learning," which refers to the process of merging several models in order to increase the accuracy and robustness of predictions. The study also provides an overview of the concept.

The first part of the paper provides a definition of ensemble learning as well as several subtypes, including bagging, boosting, and stacking. Following this, the author discusses the benefits of ensemble learning, which include a reduction in overfitting and an improvement in generalisation, as well as its applications in a variety of domains, including classification, regression, and time series forecasting.

Following this, the author(s) of the paper present an in-depth overview of the various kinds of ensemble learning and their respective algorithms. These forms of learning include decision trees, neural networks, and support vector machines. The author also covers the difficulties and restrictions that are associated with ensemble learning, such as the requirement for a variety of models and the possibility of overfitting.

In the end, the conclusion of the paper focuses on the practical implications of ensemble learning and makes some suggestions for future research areas to improve the effectiveness and efficiency of the method. The study, as a whole, offers a comprehensive and interesting explanation of the idea of ensemble learning as well as its applications in a variety of subfields that fall under the umbrella of machine learning.

Xibin Dong's research [6], offers a detailed description of the approach known as "ensemble learning," which is the process of merging numerous models in order to increase the accuracy and consistency of predictions.

This is followed by a discussion of the many approaches to ensemble learning, such as bagging, boosting, and stacking, along with a definition of the term "ensemble learning." The author then moves on to highlight the benefits of ensemble learning, which include enhanced prediction accuracy and improved robustness, as well as its applicability in numerous disciplines, such as classification, regression, and anomaly detection.

Following this, the author(s) of the study offer a comprehensive analysis of the many methods and algorithms for ensemble learning. These include decision trees, random forests, gradient boosting, and neural networks, among others. The author also covers the difficulties and restrictions that are associated with ensemble learning. Some examples of these include the requirement for varied models, the possibility of overfitting, and the increasing complexity of computing work.

In the end, the conclusion of the paper focuses on the practical implications of ensemble learning and makes some suggestions for future research areas to improve the effectiveness and efficiency of the method. In general, the paper is a useful resource for

scholars and practitioners who are interested in comprehending and putting ensemble learning techniques into practise.

[7] authored by Gulden Kaya Uyank, et al. offers an introduction to the idea of multiple linear regression analysis as well as its applications in a variety of different areas of study.

The multiple linear regression analysis and its assumptions are first defined in this section of the text. Some of these assumptions include linearity, independence, and homoscedasticity. The author continues by elaborating on the methodology behind multiple linear regression analysis, including the phases of data preparation, model fitting, and model evaluation.

Following this, the article offers a comprehensive analysis of the various forms that multiple linear regression models can take, such as basic linear regression, polynomial regression, and multiple regression with interaction effects. The author also examines the benefits and drawbacks of multiple linear regression analysis, in addition to its applicability in a variety of industries, including the business world, the marketing industry, and the medical area.

The conclusion of the paper focuses on the practical consequences of multiple linear regression analysis and makes some suggestions for future research paths aimed at increasing the analysis's effectiveness and efficiency. In general, the paper is a useful resource for researchers and practitioners who are interested in learning and utilizing approaches for multiple linear regression analysis.

The objective of [8] is to investigate the effect that the conflict between Russia and Ukraine has had on stock markets around the world.

The background of the conflict between Russia and Ukraine, as well as its possible effects on the economy of the entire world, is discussed in the first section of this study. The author then goes on to detail the technique that was employed in the study, which consisted of gathering daily stock market data from 16 different countries and conducting statistical analysis in order to determine the influence that the conflict had on the stock markets.

Following this, the findings of the analysis are presented in the paper. These data imply that the war between Russia and Ukraine had a major negative influence on stock markets around the world, notably in Russia and Ukraine. In addition, the author notes a number of factors that contributed to the impact, including economic sanctions, political unpredictability, and fluctuating oil prices.

In the end, the discussion of the practical consequences of the study and the suggestions for future research avenues for looking into the connection between geopolitical events and financial markets are included in the conclusion section of the paper. The research study, taken as a whole, offers insightful information regarding the effect that the conflict between Russia and Ukraine has had on stock markets around the world and emphasises the significance of geopolitical risk analysis in the process of making financial decisions. Research that was carried out by Filippos Giannakas and titled "XGBoost and Deep Neural Network Comparison: The Case of Teams' Performance" compares and contrasts the accuracy of predictions provided by XGBoost and deep neural networks in relation to the performance of teams. The research was carried out by Filippos Giannakas.

This part of the research begins with an introduction to XGBoost and deep neural networks, which includes an overview of the topologies of both types of networks as well as an examination of both types' benefits and drawbacks. The author then moves on to detail the experiment that was carried out to evaluate how well the two models predicted the performance of the teams. When assessing the efficacy of the models, the author makes use of a range of performance criteria, such as accuracy, precision, recall, the F1-score, and the area under the receiver operating characteristic (ROC) curve.

When it came to anticipating the performance of teams, the results of the experiment showed that XGBoost performed significantly better than deep neural networks did. The levels of accuracy, precision, recall, F1-score, and AUC-ROC that it was able to achieve were all improved. This research delves into the myriad of factors that contribute to XGBoost's superior performance when compared to that of deep neural networks. The interpretability of the model is one of these reasons, as is the manner in which it handles missing data.

Last but not least, the conclusion of the paper underlines the practical implications of the findings for a range of industries, including healthcare, marketing, and finance, and it provides future research directions for the application of machine learning in these different fields. In general, the findings of the research that were provided in this paper constitute a significant contribution to the rapidly developing field of machine learning in predictive analytics.

Bojan Obrenovic, et al. in [9] intend to provide an overview of the economic consequences and implications of the Ukraine-Russia war that began in 2014, is to provide an overview of those consequences and implications.

The paper begins with a description of the conflict's historical backdrop and the geopolitical setting in which it took place. The author then moves on to provide an examination of the economic impact that the war has had, not just on Russia but also on Ukraine, including the influence that it has had on economic growth, trade, and foreign investment.

The effects of the conflict on the larger international community are the next topic to be covered in this section of the article. These effects include the disruption of energy markets around the world and the revival of the Cold War. The author also addresses the geopolitical ramifications of the conflict for the region, such as the possibility of additional instability and conflict breaking out as a result.

In its final section, the article draws to a close by analysing the extent to which the aftermath of the conflict presents opportunities for economic recovery and reconciliation. The author indicates that a peaceful resolution to the conflict is vital for economic recovery, but at the same time, they recognise the considerable hurdles that are involved in attaining this goal.

Overall, the article provides a complete summary of the economic effects and implications of the war between Ukraine and Russia. Additionally, the report stresses the necessity for continued research and analysis of the impact that the conflict has had on the region as well as the world.

The objective of Whelsy Boungou's research paper [10] is to analyse the effect that the conflict between Ukraine and Russia has had on the returns of stock markets around the world. The paper begins with a description of the conflict's historical backdrop and the geopolitical setting in which it took place. After that, the author provides a study of how the war affected the returns on the stock markets of five different countries, namely the United States of America, the United Kingdom of Great Britain and Northern Ireland, Germany, France, and Japan. The analysis was carried out using daily data pertaining to the stock market for the years 2014 to 2018. The findings indicate that the conflict between Ukraine and Russia had a considerable adverse effect on the returns of the stock markets in all five nations that were investigated, with the United States and the United Kingdom seeing the greatest degree of this effect. The author also comes to the conclusion that the impact of the conflict on the returns of the stock market changed over time and was most obvious during times of high tension and conflict. Overall, the article demonstrates the enormous negative impact that the Ukraine-Russia war is having on the returns of global stock markets and emphasises the necessity for ongoing research and analysis of the impact that the conflict is having on the global economy.

3. Materials and methods

3.1 Models used and algorithms

Multiple linear regression:

The statistical method known as multiple linear regression is employed in the process of modeling the connection that exists between two or more independent variables and a dependent variable. The following is an example of a straightforward procedure for multiple linear regression:

- Collect data and clean, collect information on the independent variables as well as the variable that will be the focus of your analysis.To clean the data, remove any information that is either absent or superfluous from the dataset.
- Determine which variables in the dataset will be used as independent variables and which will be used to represent the dependent variable. Identify the variables that will be used as independent variables and the variables that will be used as dependent variables.
- Check for linearity by analyzing the relationship between the independent variable and the dependent variable to determine whether or not there is a linear connection between the two. Creating a scatterplot from the data will allow you to accomplish this goal.
- Conduct an analysis on the model, by computing the value of the R squared statistic, you may evaluate how well the model matches the data. A strong fit is indicated by an R-squared number that is near 1, whereas a bad fit is indicated by a value that is close to 0.
- The model is put to the test by making a prediction about the value of the dependent variable based on the values of the independent variables using the model. Evaluate the accuracy of the model by contrasting the expected values with the values that actually occurred.
- The findings of the regression analysis need to be interpreted, and then any inferences that may be drawn about the relationship between the independent variables and the dependent variables should be based on those interpretations.
- Refine the model: Refine the model by either adding or eliminating independent variables and then re-testing the model to see if the changes result in an improvement in the model's ability to accurately predict outcomes.
- Utilize the model to create predictions about future data based on the relationships between the variables that are independent and those that are dependent on one another using the model.

Regression Coefficients

$$b_1 = \frac{\left(\sum x_2^2\right)\left(\sum x_1 y\right) - \left(\sum x_1 x_2\right)\left(\sum x_2 y\right)}{\left(\sum x_1^2\right)\left(\sum x_2^2\right) - \left(\sum x_1 x_2\right)}$$

$$b_2 = \frac{\left(\sum x_1^2\right)\left(\sum x_2 y\right) - \left(\sum x_1 x_2\right)\left(\sum x_1 y\right)}{\left(\sum x_1^2\right)\left(\sum x_2^2\right) - \left(\sum x_1 x_2\right)}$$

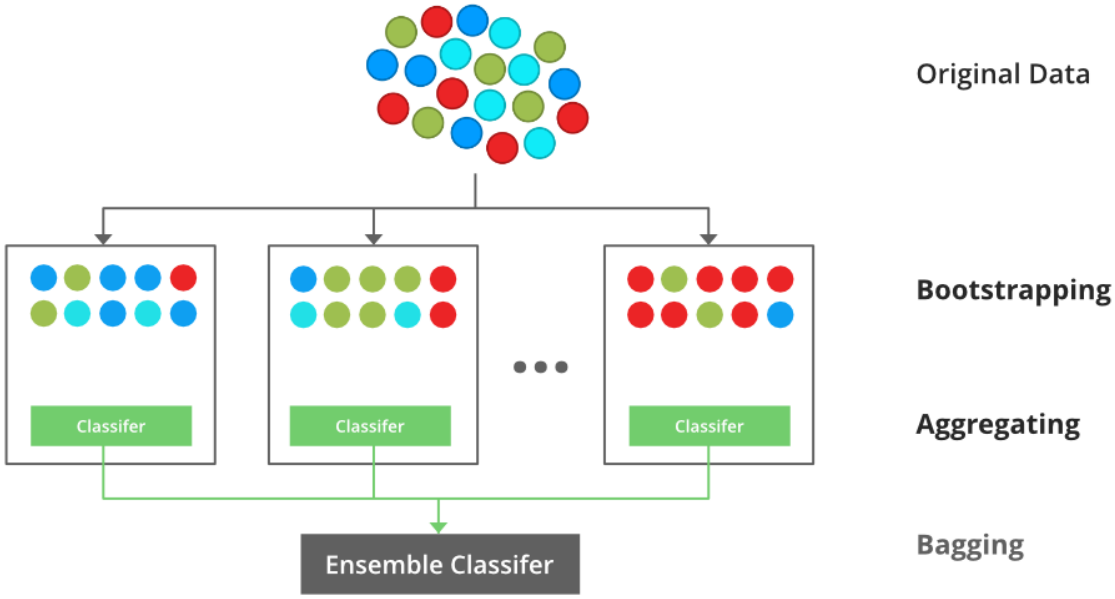
$$a = b_0 = Y - b_1 X_1 - b_2 X_2$$

XG boost

Extreme gradient boosting, more commonly known as XGBoost, is a robust machine learning technique that may be applied to regression as well as classification problems. The fundamental algorithm for XGBoost is as follows:

- Collect and clean data, collect information on the independent variables as well as the variable that will be the focus of your analysis. To clean the data, remove any information that is either absent or superfluous from the dataset. The data should be separated, with the dataset being split into a training set and a testing set.
- The model is constructed with the help of the training set, while its performance is assessed with the help of the testing set.
- Specify the model as follows: Find out the values for the XGBoost model's hyperparameters, which include the learning rate, the number of trees, the depth of each tree, and the objective function.
- Exercise the model by adjusting the XGBoost model so that it best fits the training data. Performing this action requires iteratively building a series of trees, with each tree built to repair the flaws of the tree that came before it in the chain.
- Conduct an analysis of the model: When conducting the analysis, use the testing set to determine how well the XGBoost model performed. Perform the calculations necessary to determine the model's accuracy, precision, recall, and F1 score.

- Tune the hyperparameters: To enhance the overall performance of the model, you should try out a variety of various hyperparameters. In order to determine the ideal combination of hyperparameters, this can be done using either a grid search or a random search.
- The findings of the XGBoost model need to be interpreted, and then the inferences you draw about the relationship between the independent factors and the dependent variables need to be based on those interpretations. Use the XGBoost model to produce predictions about fresh data based on the relationships between the independent and dependent variables.
- These predictions may then be tested against the actual data. Keeping an eye on performance It is important to keep an eye on the performance of the XGBoost model over time and to retrain it as necessary to ensure that it continues to perform well on new data.



AutoML

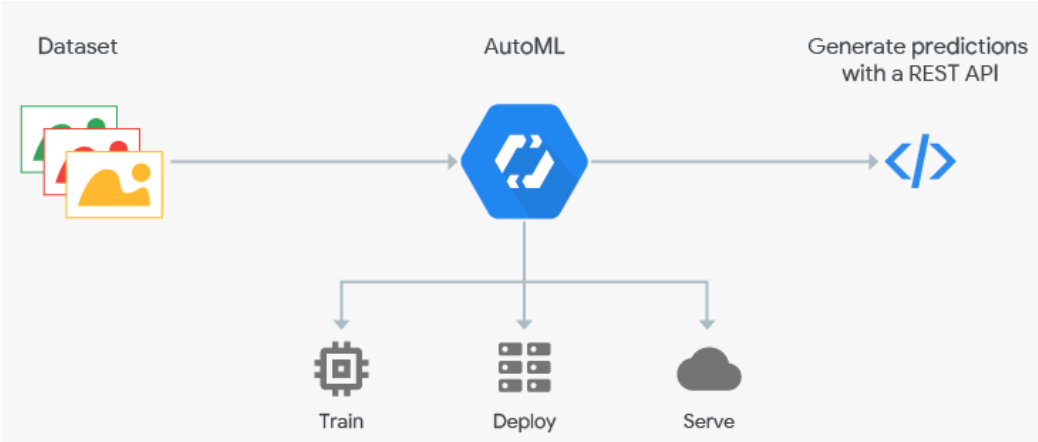
Automated Machine Learning, also known as AutoML, is a subfield of machine learning that entails the process of automating the development and optimisation of machine learning models. A machine learning model that has been constructed and optimized using automated processes is referred to as an AutoML model. This type of model does not require a significant amount of manual interaction from data scientists or specialists in machine learning.

For the purpose of automating the process of developing and optimizing machine learning models, AutoML models make use of a number of different algorithms and statistical methods. These techniques include:

The process of automatically producing new features from previously collected data in order to enhance the overall performance of a machine learning model is referred to as "feature engineering."

Model selection refers to the process of automatically choosing the most effective machine learning algorithm for a specific job.The process of automatically constructing a machine learning model by selecting the optimal algorithm and hyperparameters is referred to as "automated model creation."

The use of AutoML models can be advantageous since they can shorten the amount of time needed to construct and optimize machine learning models, as well as lower the associated costs.



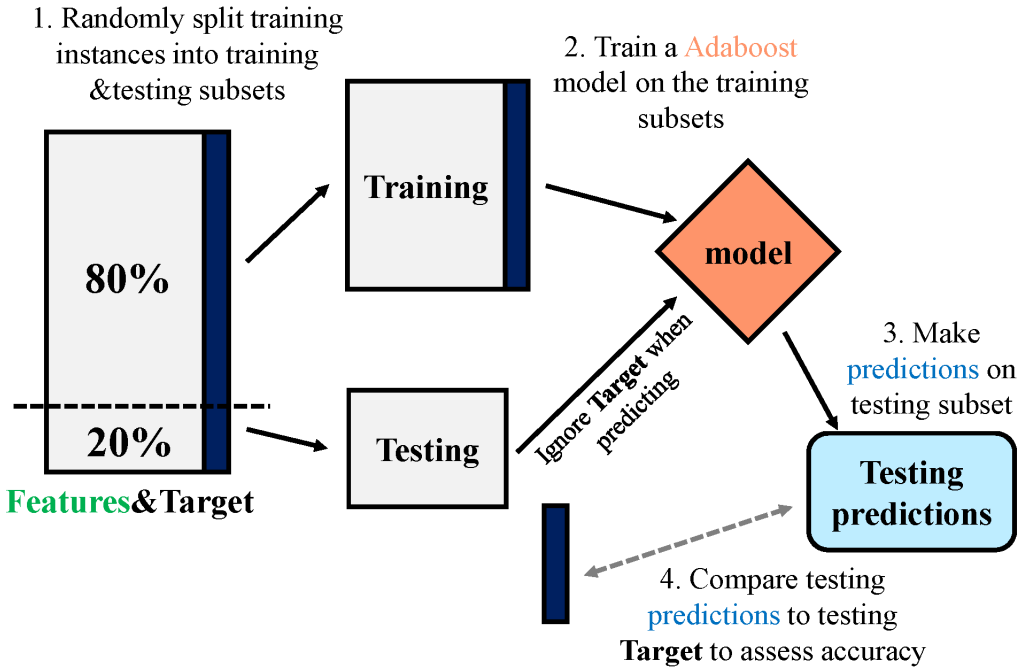
ADA Boost

In the field of machine learning, AdaBoost, also known as adaptive boosting, is a prominent ensemble learning method that is used for classification and regression applications. The AdaBoost method achieves its results by fusing together a number of less accurate classifiers into a single, more accurate one. The accuracy of the weak classifier is used to determine the weights that are assigned to each training example during each iteration of the algorithm. These weights are assigned to each training

example based on the example being evaluated. The subsequent weak classifier is then trained on the weighted dataset, with more weight being given to the cases that were initially misclassified. The final classifier is derived via a combination that takes into account the relative importance of each of the weaker classifiers. AdaBoost's capacity to handle high-dimensional data while avoiding overfitting is one of the advantages offered by this algorithm. However, it is susceptible to being affected by noisy data as well as outliers. The following are the primary operations involved in the AdaBoost algorithm:

- Set the weights to their default values for each training example.
- Train a simple classifier using the data that has been weighted.
- Determine how well the weak classifier performs on the data that has been weighted.
- Adjust the weights of the training samples in accordance with the degree to which the weak classifier has been successful. It is necessary to repeat steps 2-4 for a predetermined amount of times.
- Create a final strong classifier by combining the weak classifiers using their weights in the combination.

In general, AdaBoost is a robust method that has the potential to increase the performance of a wide variety of learning algorithms, particularly when working with data that is both complicated and high-dimensional.



3.2 Dataset Description:

The dataset contains the following attributes that will be used for visualization and forecasting.

The dataset included cumulative values which were later changed to discrete values for better understanding and visualization

- Personnel
- Prisoner of War (POW) - - has not been tracked since 2022-04-28
- Aircraft
- Helicopter
- Tank
- Armored Personnel Carrier (APC)
- Multiple Rocket Launcher (MRL)
- Field Artillery
- Military Auto - has not been tracked since 2022-05-01; joined with Fuel Tank into Vehicles and Fuel Tanks
- Fuel Tank - has not been tracked since 2022-05-01; joined with Military Auto into Vehicles and Fuel Tanks
- Anti-aircraft warfare
- Drone - UAV+RPA
- Naval Ship - Warships, Boats
- Anti-aircraft Warfare
- Mobile SRBM System - has not been tracked since 2022-05-01; joined into Cruise Missiles
- Vehicles and Fuel Tanks - appear since 2022-05-01 as a sum of Fuel Tank and Military Auto
- Cruise Missiles - appear since 2022-05-01
- Direction of Greatest Losses - appear since 2022-04-25

3.3 Architecture and explanation

The two softwares used in the project are RStudio and Tableau. Rstudio has been used to visualize basic details of the war using bar graphs, line charts, etc. and also to determine the best fit while forecasting and using other predictive algorithms.

Tableau has been used to visualize more complex data such as data on the world map and visualizing the predictions made by using the various previously mentioned algorithms.

Multiple Linear Regression:

The following elements are included in the multiple linear regression model architecture:

- Dependent variable: The variable that the model is attempting to predict or explain is known as the dependent variable. It is also called the response variable or the outcome variable.
- Independent variables: The variables that are used to predict the dependent variable are referred to as independent variables, predictor variables, or explanatory variables. Multiple independent variables can be handled via multiple linear regression.
- Model parameters: The coefficients or weights used in the multiple linear regression model to characterise the relationship between the independent variables and the dependent variable are known as the parameters.
- Error term: The error term is the difference between the predicted value of the dependent variable and the actual value. The model aims to minimize the sum of the squared errors, which is also known as the residual sum of squares.

The mathematical formula for multiple linear regression can be expressed as follows:

$Y = \beta0 + \beta1X1 + \beta2X2 + ... + \betapXp + \varepsilon$

where Y is the dependent variable, X1, X2, ..., Xp are the independent variables, β0 is the intercept, β1, β2, ..., βp are the coefficients, and ε is the error term.

XGBoost:

The following elements make up the XGBoost algorithm:

- Decision Trees: Decision trees are binary classifiers that divide the data into two sections based on a threshold value. They are the base learners in the XGBoost model. The XGBoost decision tree is a regression tree, which means that each node's output is a real-valued score that represents the target variable's predicted value.
-

- Ensemble Methods: To enhance the performance of the model, XGBoost combines the results of various decision trees using ensemble methods. Several low-bias models can be combined using ensemble methods to create a higher-bias, lower-variance model.
- Gradient Boosting: XGBoost employs gradient boosting to enhance model performance and optimise model parameters. Gradient boosting is a method for updating model parameters by incrementally reducing the loss function. The loss function's negative gradient is followed to update the parameters using the gradient descent approach.
- Regularization: To avoid overfitting and enhance the model's generalisation capabilities, XGBoost employs regularisation approaches. By including penalty terms to the loss function, regularisation techniques like L1 and L2 regularisation can lessen the model's complexity.
- Objective Function: XGBoost uses a specific objective function that is optimized during the training process. The objective function is a measure of the error between the predicted values and the actual values of the target variable. The XGBoost algorithm can optimize different objective functions depending on the type of problem being solved, such as binary classification, multiclass classification, or regression.
- Learning Rate: In XGBoost, the learning rate regulates the gradient descent algorithm's training step size. A slower model convergence is caused by a smaller learning rate, whereas a slower convergence is caused by a bigger learning rate.

AdaBoost:

The following elements make up the AdaBoost algorithm:

- Weak Classifiers: In the AdaBoost model, the basic learners are weak classifiers, which are straightforward decision rules that outperform random guessing just marginally. Any classification algorithm, including decision trees, SVMs, and logistic regression, can be a weak classifier.
- Training Data: Using a training set of labelled data, AdaBoost trains the weak classifiers. The labelled data is composed of pairs of input features and binary (i.e., either 0 or 1) output labels.
- Weights: Each example in the training set is assigned a weight during training, reflecting its significance in the classification task. Each weak classifier is

trained on a weighted subset of the training data with all weights initially set to the same value.

- Adapting Weights: After each iteration of training, the weights of the misclassified examples are increased, while the weights of the correctly classified examples are decreased. This way, the subsequent weak classifiers will focus more on the misclassified examples in the previous iterations.
- Boosting: Based on how accurately each weak classifier classified the training data, AdaBoost combines the results of the weak classifiers. Each weak classifier's error rate, or the percentage of incorrectly classified examples, determines how much weight it has.
- Final Classifier: In AdaBoost, the final classifier is created by collecting all weak classifiers' outputs and weighting them based on how accurate they were. The accuracy of each weak classifier determines its weight in the final classifier, which is a linear combination of the weak classifiers.

AutoML:

The following elements make up an AutoML system's architecture:

- Data preprocessing: The preprocessing of the data is the first stage in an AutoML pipeline. Data cleansing, normalization, feature selection, and transformation are a few of the tasks involved in this. This stage is essential for getting the data ready for the machine learning algorithms and enhancing the models' performance.
- Feature Engineering: Feature engineering is the process of creating new features or transforming existing features to improve the performance of the models. AutoML systems use techniques such as principal component analysis (PCA), feature scaling, feature selection, and feature extraction to automate this process.
- Model Selection: The next step in an AutoML pipeline is to select the best model for the task. This involves testing a variety of machine learning algorithms, such as decision trees, random forests, neural networks, and support vector machines (SVMs), and selecting the one that performs the best on the data.
- Hyperparameter Optimization: After the best machine learning algorithm has been chosen, its hyperparameters must be optimised. Hyperparameters, which include the learning rate, regularisation parameter, and number of hidden layers, are settings that regulate how the algorithm behaves. This procedure is

automated by autoML systems using methods like grid search, random search, and Bayesian optimisation.

- Ensembling: Ensembling is the process of combining multiple models to improve their performance. AutoML systems use techniques such as bagging, boosting, and stacking to combine the outputs of multiple models and create a more accurate ensemble model.
- Model Evaluation: Finally, AutoML systems evaluate the performance of the models using a variety of metrics, such as accuracy, precision, recall, and F1 score. They also perform cross-validation and test the models on a holdout set of data to ensure that they generalize well to new data.

4. Proposed works

4.1 Novelty

The war in Russia has affected many people across the world. Giving numbers out in the air without any sort of visualization fails to create a deep impact of the negative effects the war has on the world and on Russia and Ukraine itself.

By visualizing data, which is updated on a daily basis, we are able to see the impact the war has on the world and on the two fighting nations. While also being able to forecast and predict future losses that the two countries might face which can help them take necessary measures to avoid it from taking place in the first place.

Also, since multiple models have been used to predict the outcome of certain parameters, we can gain an insight as to which machine learning mode was the best and can be used in the future for similar situations

4.2 Project contributions

Manvik Sreedath - Data preprocessing, AdaBoost and AutoML model implementation, Results

Aryan Vigyat - Abstract,Review of Literature, Data Discovery,Visualization in R,XGBoost Model,Explanations

Kartik Deepu - Introduction, Tableau Visualizations, Forecasting, Technical Report, Multiple Linear Regression Model

5. Results and discussion

5.1 Results

We have implemented multiple machine learning models like Multiple Linear Regression, XGBoost, ADABoost and AutoML to the current dataset and compared each of the models. All models were used to predict the number of helicopters lost based on the loss of aircrafts, tanks and fuel tanks.

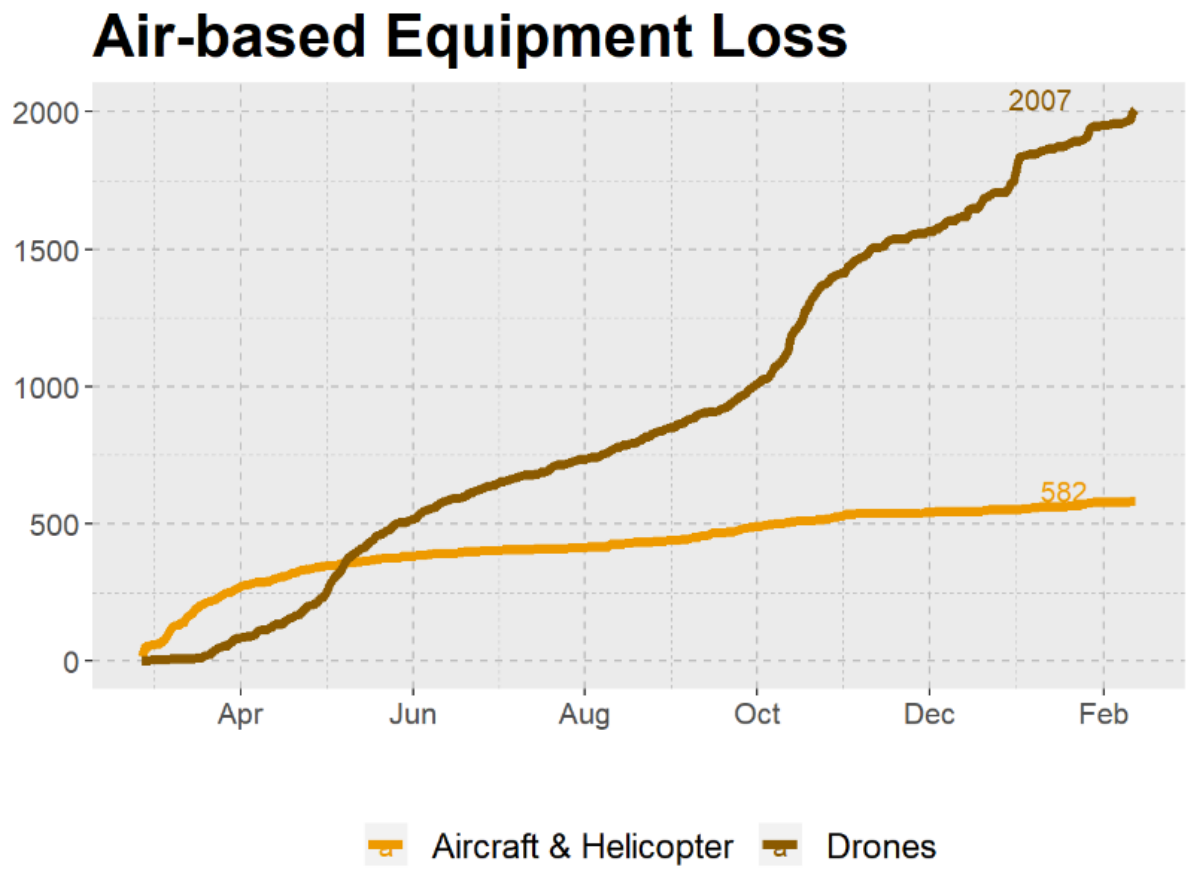
1. Multiple Linear Regression
    - Adjusted R Squared = 0.9187
    - Residual standard error = 11.9
    - F-statistics = 242.1

2. XGBoost
    - Root Mean Squared Error = 1.052

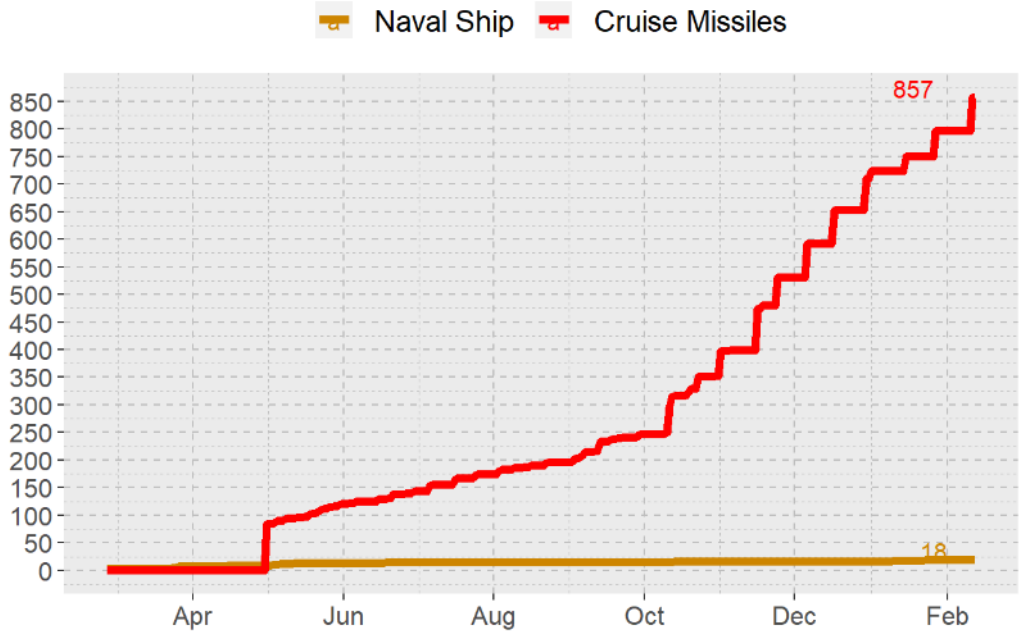
3. ADABoost
    - Mean Squared Error = 118.43
    - Root Mean Squared Error = 10.88

4. AutoML
    - Accuracy = 96.34%
    - Root Mean Squared Error = 0.259
    - Kappa = 0.94

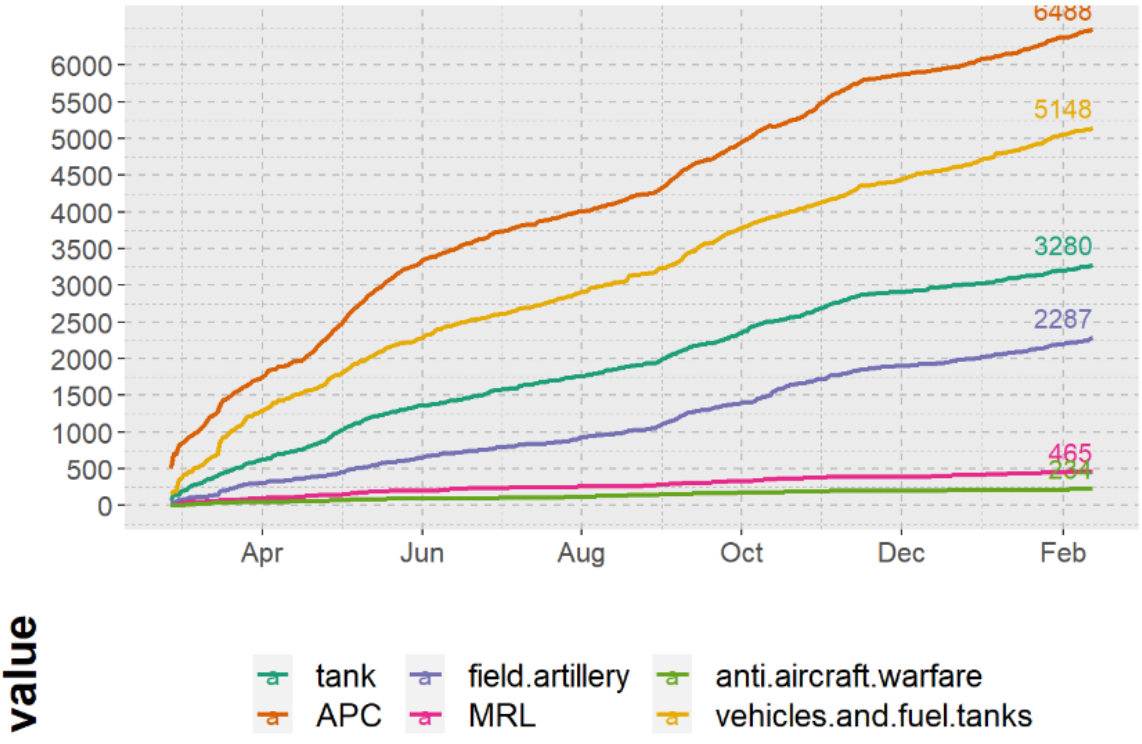
5.2 Figures and Comparisons Tables



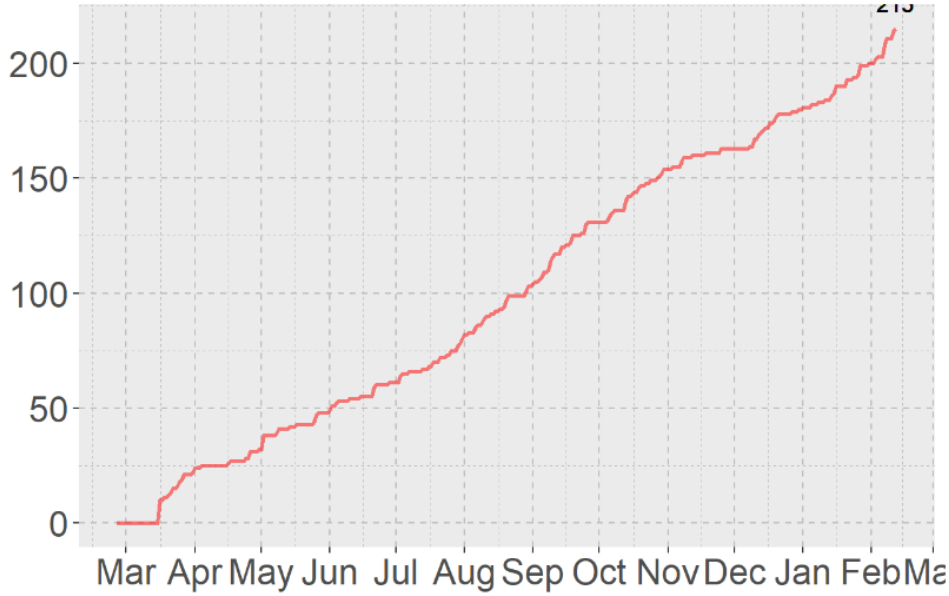
Water-based Equipment Loss



Field-based Equipment Loss

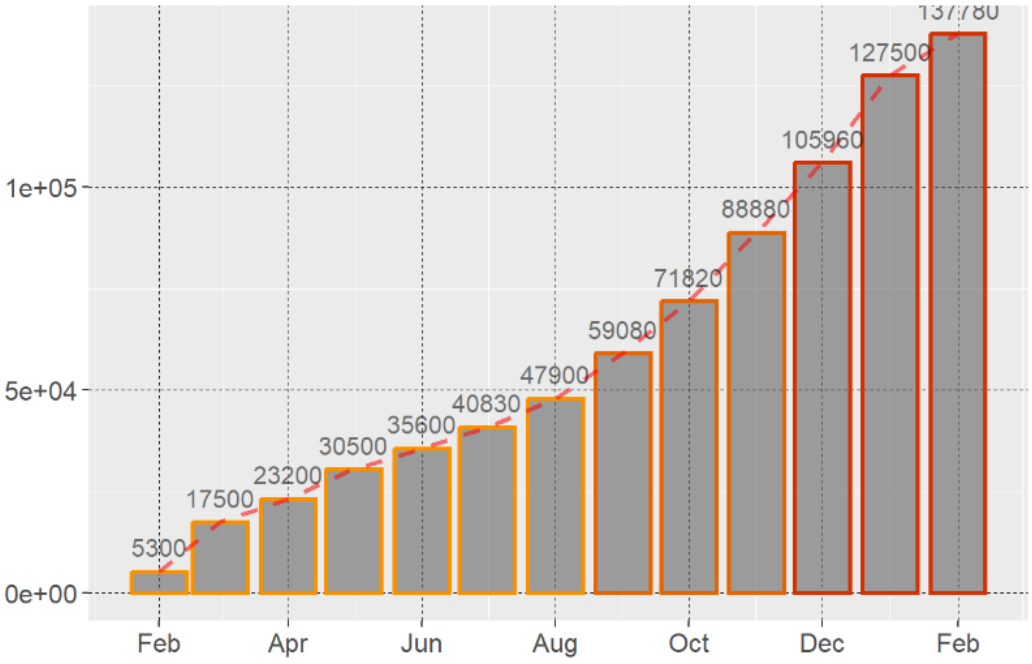


Special Equipment Loss

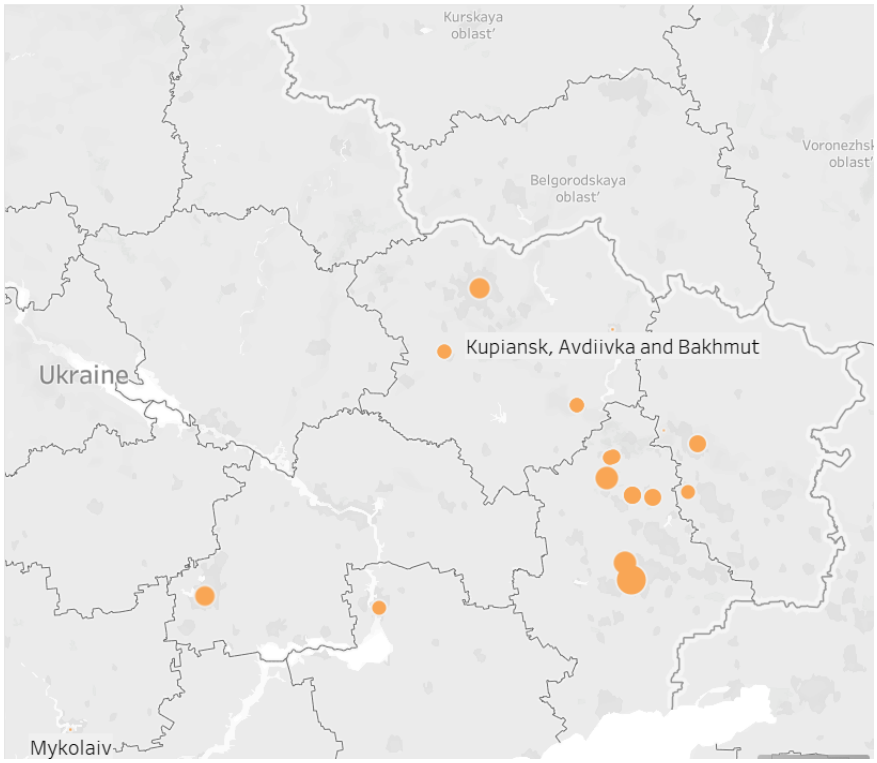


Death Toll and Russian War

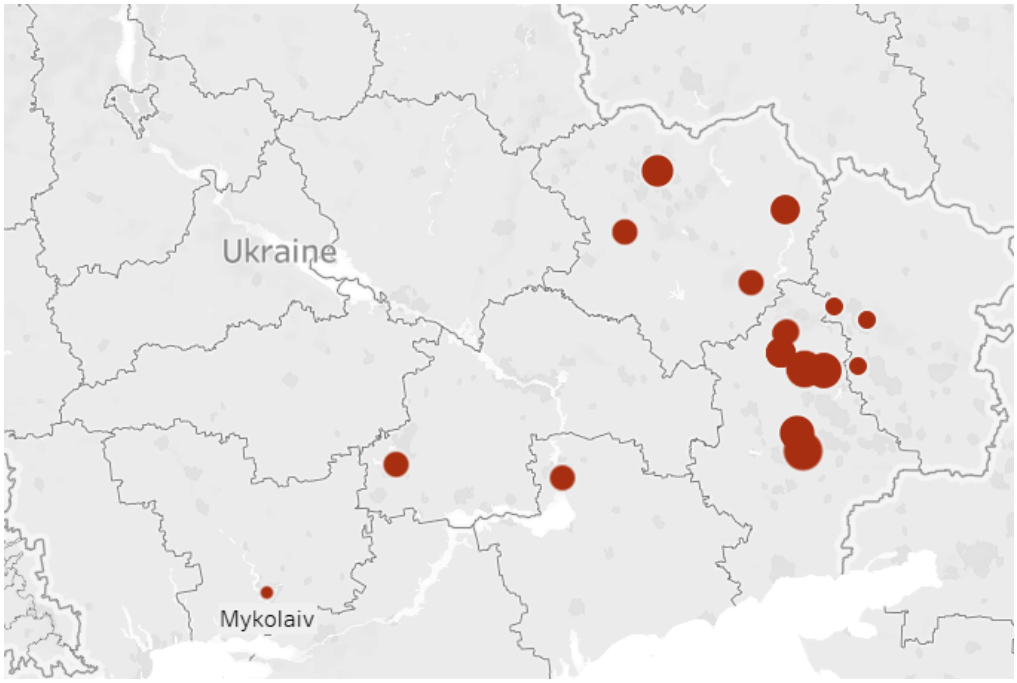
Death Toll



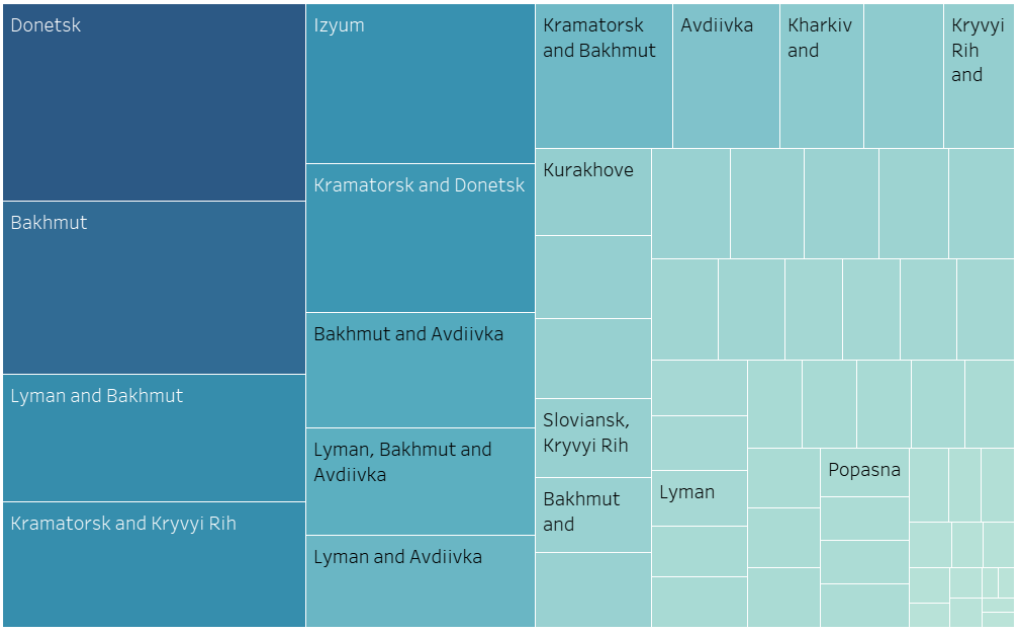
Air vehicles loss



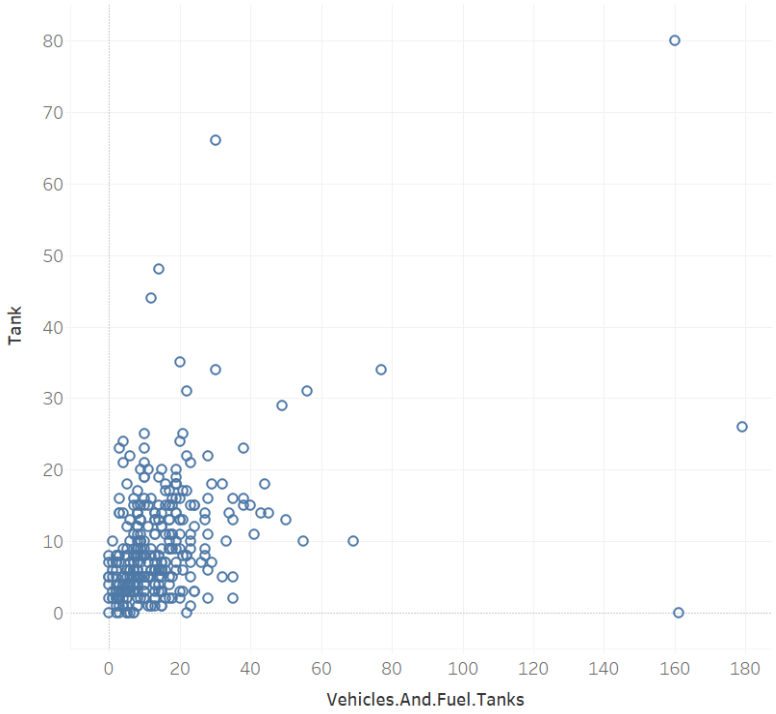
Personnel loss



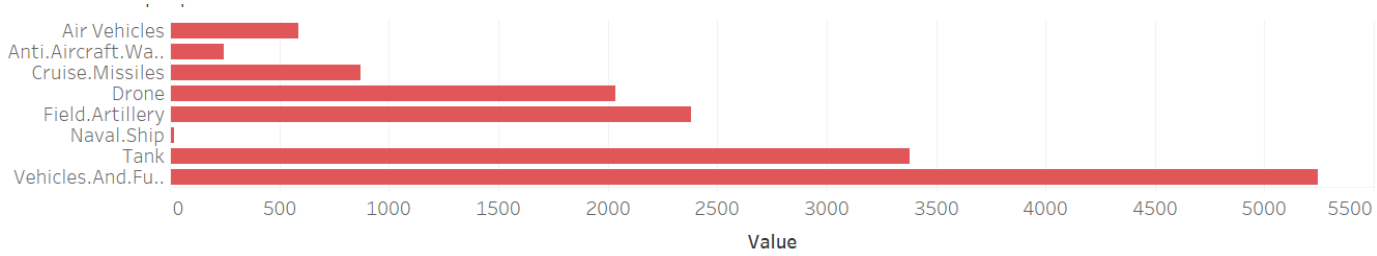
Drone loss heat map



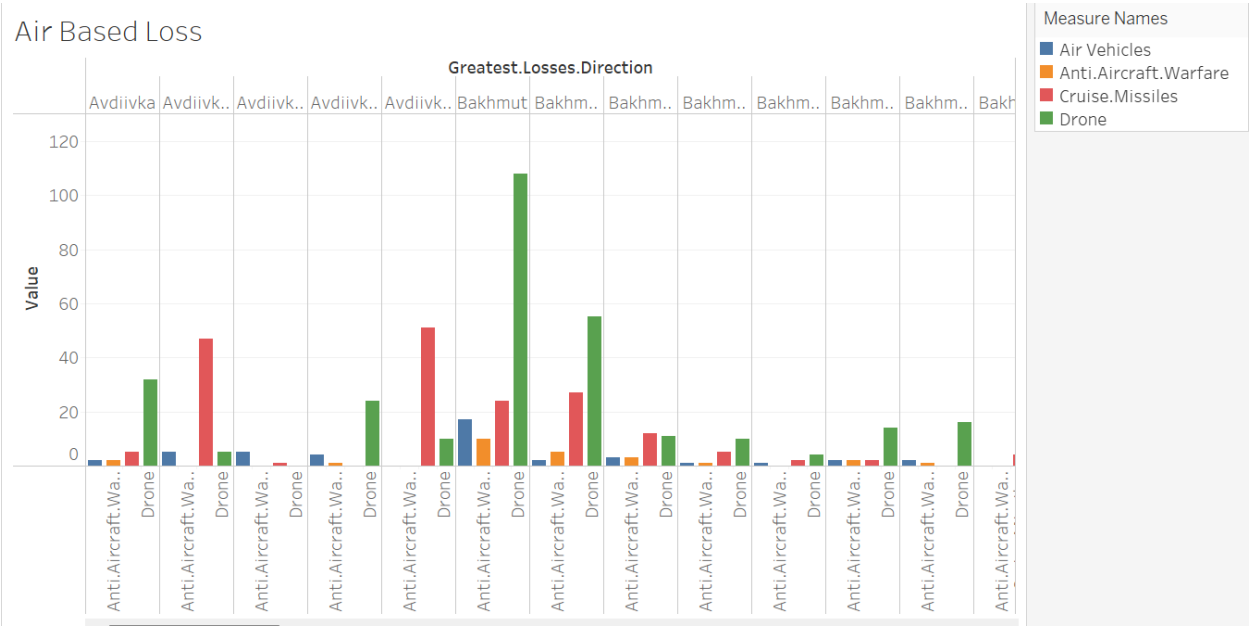
Scatterplot of Vehicles and Fuel Tanks vs Tanks



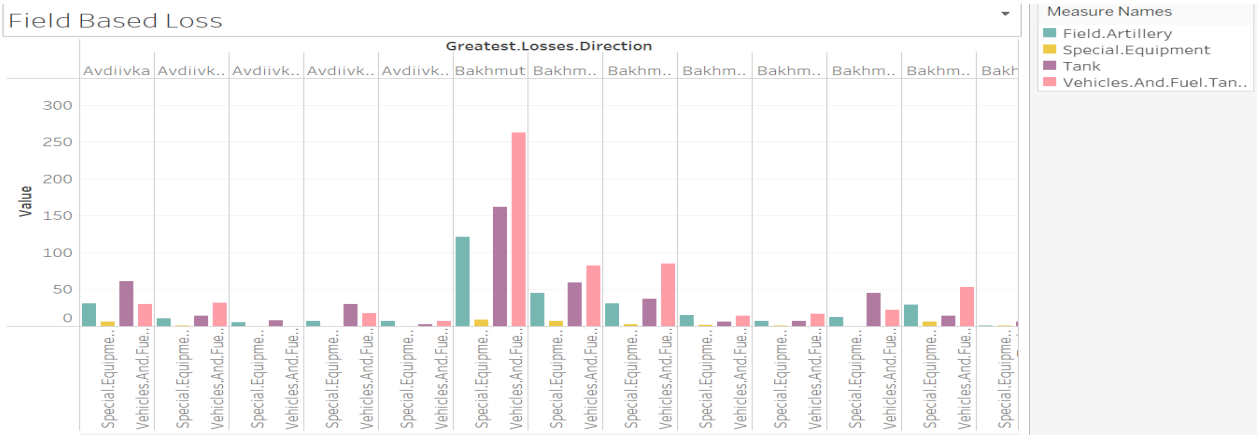
Equipment Loss



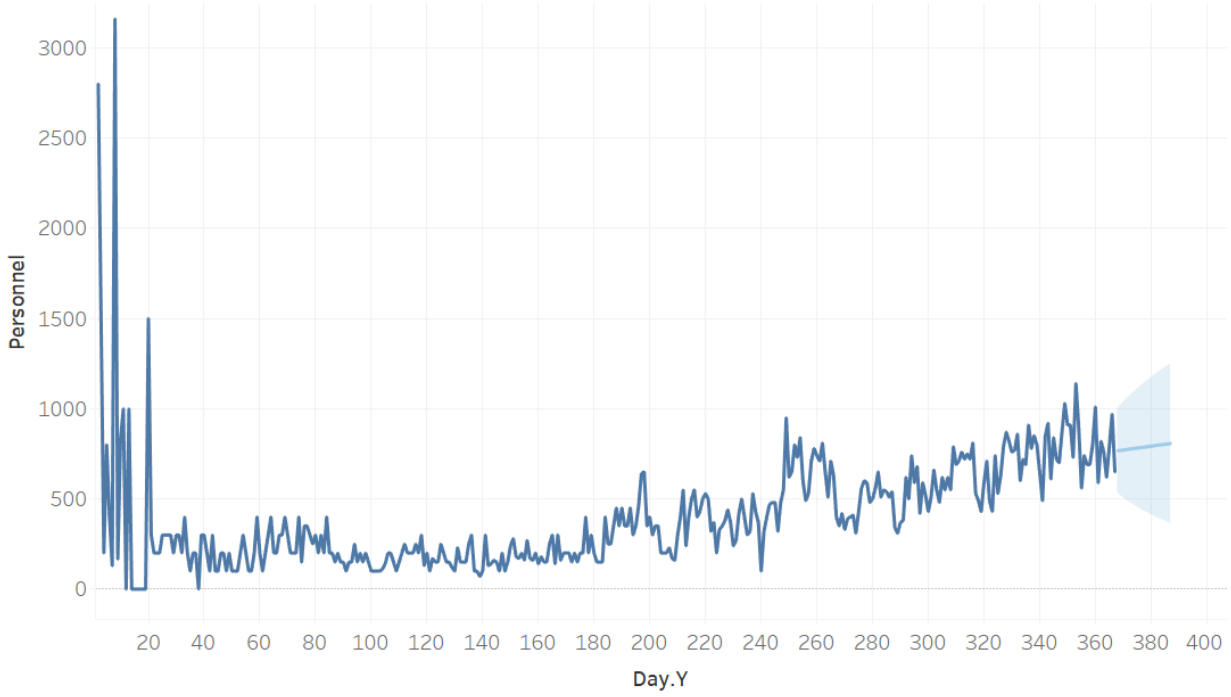
Air Based Loss



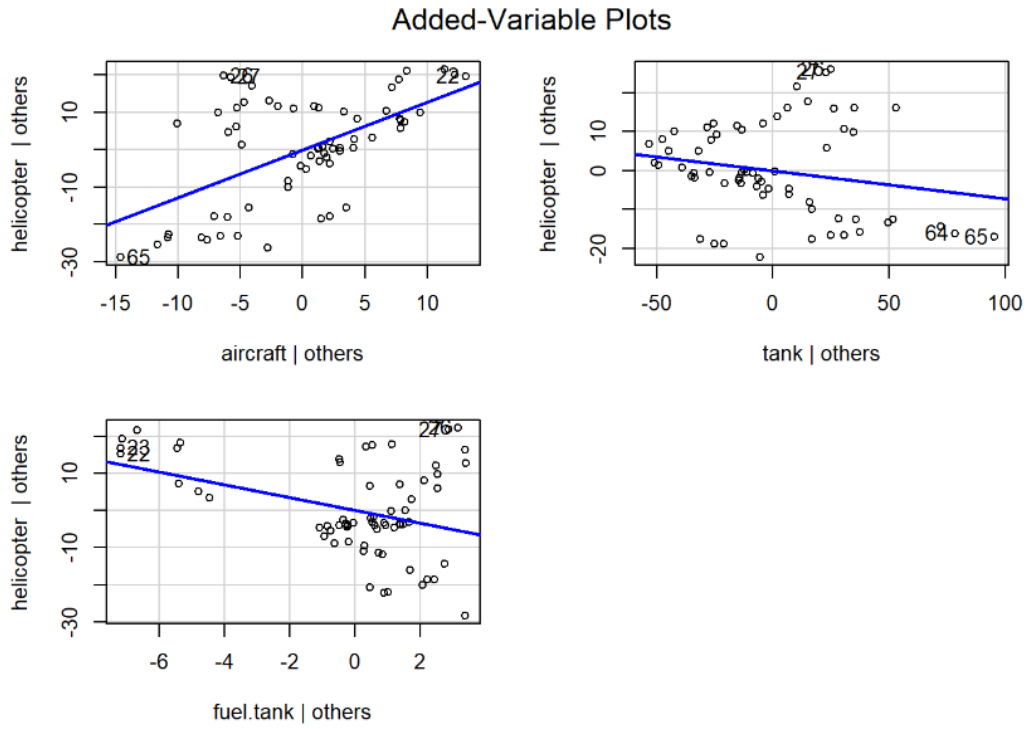
Field Based Loss



Personnel Loss Forecast



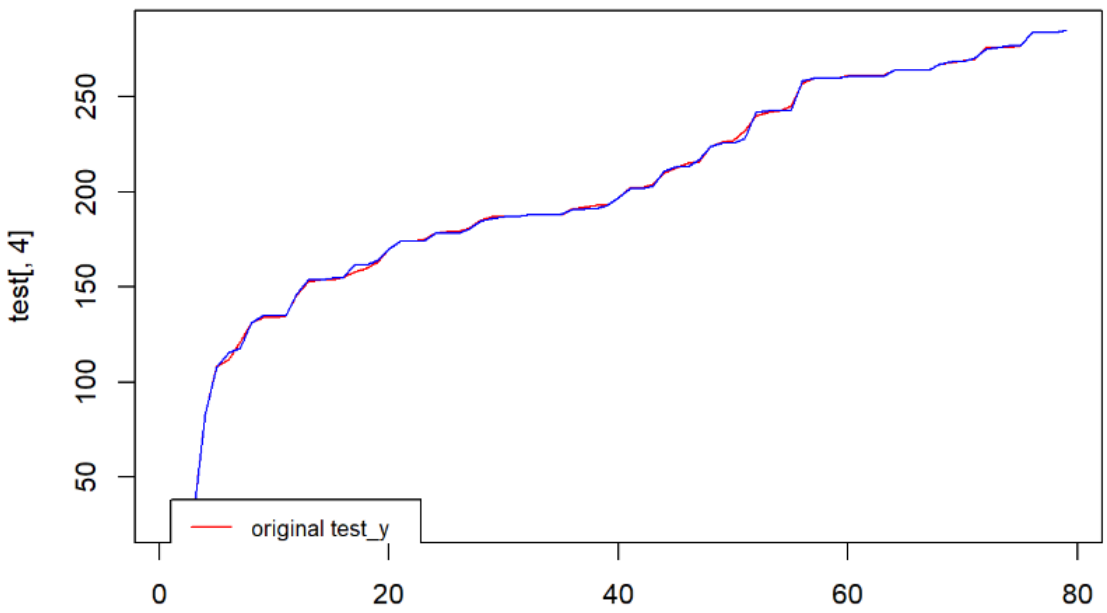
1. Multiple Linear Regression



Summary of the Model

```
##
## Call:
## lm(formula = helicopter ~ aircraft + tank + fuel.tank, data = df1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.657  -6.857  -2.242   7.466  27.787
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 126.24891   31.64741   3.989  0.00018 ***
## aircraft      1.27752    0.23989   5.325 1.53e-06 ***
## tank         -0.07197    0.04552  -1.581  0.11903
## fuel.tank    -1.72683    0.55422  -3.116  0.00280 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.9 on 61 degrees of freedom
##   (288 observations deleted due to missingness)
## Multiple R-squared:  0.9225, Adjusted R-squared:  0.9187
## F-statistic: 242.1 on 3 and 61 DF,  p-value: < 2.2e-16
```

2. XGBoost



Summary of the Model

```
##                Length Class              Mode
## handle              1 xgb.Booster.handle externalptr
## raw            185812 -none-             raw
## niter               1 -none-             numeric
## evaluation_log      2 data.table         list
## call               14 -none-             call
## params              2 -none-             list
## callbacks           1 -none-             list
## feature_names       3 -none-             character
## nfeatures           1 -none-             numeric
```

3. ADABoost

```
#Make Predictions
pred_test=predict(model_adaboost,newdata=test[, c("aircraft", "tank", "fuel.tank")])
mse <- mean((as.numeric(pred_test$class) - test$helicopter)^2)
print(paste("Mean Squared Error: ", mse))
```

```
## [1] "Mean Squared Error:  118.430379746835"
```

Summary of the Model

```
##             Length Class   Mode
## formula          3 formula call
## trees          100 -none-  list
## weights        100 -none-  numeric
## votes        34250 -none-  numeric
## prob         34250 -none-  numeric
## class          274 -none-  character
## importance       3 -none-  numeric
## terms            3 terms   call
## call             5 -none-  call
```

4. AutoML

Leaderboard of 17 different Models trained by AutoML

```
##                                                model_id
## 1                          GBM_5_AutoML_14_20230403_104225
## 2                          XRT_1_AutoML_14_20230403_104225
## 3                          DRF_1_AutoML_14_20230403_104225
## 4              GBM_grid_1_AutoML_14_20230403_104225_model_2
## 5  StackedEnsemble_BestOfFamily_1_AutoML_14_20230403_104225
## 6     StackedEnsemble_AllModels_1_AutoML_14_20230403_104225
## 7                          GBM_4_AutoML_14_20230403_104225
## 8              GBM_grid_1_AutoML_14_20230403_104225_model_3
## 9                          GBM_2_AutoML_14_20230403_104225
## 10                         GBM_3_AutoML_14_20230403_104225
## 11             GBM_grid_1_AutoML_14_20230403_104225_model_1
## 12    DeepLearning_grid_1_AutoML_14_20230403_104225_model_1
## 13    DeepLearning_grid_3_AutoML_14_20230403_104225_model_1
## 14                DeepLearning_1_AutoML_14_20230403_104225
## 15                         GLM_1_AutoML_14_20230403_104225
## 16    DeepLearning_grid_2_AutoML_14_20230403_104225_model_1
## 17                         GBM_1_AutoML_14_20230403_104225
##    mean_per_class_error   logloss      rmse        mse
## 1             0.3142149 0.3149237 0.2686622 0.07217938
## 2             0.3314886 0.6661353 0.4149923 0.17221861
## 3             0.3845667 0.8321450 0.3152004 0.09935132
## 4             0.4190768 0.3513080 0.2888671 0.08344420
## 5             0.4321451 0.3755353 0.3034317 0.09207078
## 6             0.4324078 0.3510587 0.2949666 0.08700529
## 7             0.4339969 0.3589757 0.2945908 0.08678372
## 8             0.4353772 0.3424001 0.2877712 0.08281228
## 9             0.4361116 0.3701043 0.2949597 0.08700122
## 10            0.4440481 0.3622663 0.3054266 0.09328542
## 11            0.5538709 0.4935235 0.3728822 0.13904116
## 12            0.6349171 0.9437064 0.4036172 0.16290683
## 13            0.6370760 0.7059682 0.4041664 0.16335051
## 14            0.6519518 0.7709660 0.4335289 0.18794728
## 15            0.6634310 0.7696067 0.4379939 0.19183865
## 16            0.6713675 1.8539556 0.4451082 0.19812127
## 17            0.8222222 1.2066582 0.6687527 0.44723024
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  4  6  8
##          1 37  0  0  0  0  0
##          2  0 28  1  0  0  0
##          3  0  0  0  0  0  0
##          4  0  1  0  7  0  0
##          6  1  0  0  0  7  0
##          8  0  0  0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.9634
##                  95% CI : (0.8968, 0.9924)
##     No Information Rate : 0.4634
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9436
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 6 Class: 8
## Sensitivity            0.9737   0.9655   0.0000  1.00000  1.00000       NA
## Specificity            1.0000   0.9811   1.0000  0.98667  0.98667        1
## Pos Pred Value         1.0000   0.9655      NaN  0.87500  0.87500       NA
## Neg Pred Value         0.9778   0.9811   0.9878  1.00000  1.00000       NA
## Prevalence             0.4634   0.3537   0.0122  0.08537  0.08537        0
## Detection Rate         0.4512   0.3415   0.0000  0.08537  0.08537        0
## Detection Prevalence   0.4512   0.3537   0.0000  0.09756  0.09756        0
## Balanced Accuracy      0.9868   0.9733   0.5000  0.99333  0.99333       NA
```

5.3 Explanation

1. From the Air Based Equipment Loss graph we can see that the loss of drones is increasing exponentially and is much higher than the loss of aircrafts and helicopters. This is because drones are used for surveillance and for tasks that are risky. This saves cost and saves human life as drones can be remotely operated.

2. From the Water Based Equipment Loss we can see that Cruise Missiles loss increases exponentially. This is because hundreds of missiles would be used during the war, whereas very few naval ships would be destroyed.

3. From the Field Based Equipment Loss we can see that Armored Personnel Carriers faced the most losses when compared to loss of tanks, anti aircrafts, field artillery etc.

4. Special Equipment loss and Death Toll has been exponentially increasing since the start of the war.

5. Analyzing the Air Vehicle Loss, most of it is based on cities like Donetsk,Avdivka and Lyman with Donetsk (31) being the maximum and Mykolaiv(0) being the lowest.

6. Donetsk and Bakhmut region displays the biggest drone losses, while Sloviansk town shows the lowest losses, per the heat map research. Drone crashes are significantly concentrated in Donetsk and Bakhmut, according to the heat map data, which may be related to the continuing fighting in the area. North of Donetsk, in contrast, has reported significantly fewer drone losses. This discrepancy in drone losses may be caused by a variety of elements, including the topography, the weather, or even the presence of anti-aircraft defences. Overall, the heat map study sheds important light on how drone losses are distributed and identifies regions that need more research

7. Donetsk and Lyman have suffered the most casualties, according to a review of personnel losses in Russia's combat zones, while Mykolaiv has suffered the least. The continuous armed struggle in the area, which has led to fierce fighting and skirmishes between the opposing factions, is to blame for the high number of fatalities in Donetsk and Lyman. Both the military and the civilian population have suffered greatly as a result of the fighting. Mykolaiv, a city in southern Ukraine, has recorded significantly fewer casualties, maybe as a result of its distance from the fighting areas. The military's effective use of military methods and tactics to safeguard the lives of soldiers and civilians may also be responsible for the lesser number of casualties in Mykolaiv. Yet, every human life lost is tragic, and the loss of personnel in all combat zones is worrying and emphasizes the need for peace and security in the area..

8. Vehicles and gasoline tanks appear to be the most severely damaged according to the data that is currently available on the losses from the Ukrainian war, while naval ships sustain the fewest casualties. This might be explained by the fact that the war has primarily been fought on land with few naval encounters. Also, because they are essential for military operations and logistics, vehicles and gasoline tanks are a top target for opposing forces looking to limit their adversary's mobility and resupply. Naval ships, on the other hand, are often more protected and have greater mobility, allowing them to avoid battles or escape danger with greater ease. It's crucial to remember that as the dispute progresses, the knowledge may become limited or incomplete.

9. In the continuing battle in Ukraine, the cities of Donetsk and Bakhmut have sustained the greatest air-based losses, including air vehicles, anti-aircraft warfare drones, and cruise missiles. This is explained by the strategic placement of these cities, which are important transit hubs for both military and civilian purposes and are located close to the area of war. Both sides have therefore attempted to take control of these locations, which has resulted in heavy aerial bombardment and defensive actions.

10. The cities of Donetsk, Kramatorsk, and Kharkiv have suffered a substantial number of losses in terms of Russian field-based equipment, such as field artillery, tanks, special equipment, and vehicles and fuel tanks, according to the data available on the losses in the current conflict in Ukraine. This is due to the fact that these cities, which are close to the frontlines and have served as strategic flashpoints for both sides of the fight, are situated there. Heavy weaponry, like tanks and field artillery, has been frequently used in the fight since it is effective at supplying cover and assistance to ground forces. Due to the frequent skirmishes and fights that have taken place in these locations, both sides have suffered severe equipment losses.

- The Multiple Linear Regression model gave us an adjusted R squared value of 0.9187, which means the model is able to predict 91.87% of the variance in the data. The model was also used to predict the loss of helicopters if loss of aircrafts was 25, tanks were 150 and fuel tanks 60. The model gave an output of 43.7. The high F-statistics and very low p-value tells us that the model is very good.

- The XGBoost model was trained for 200 epochs and was found to overfit around 167th epoch. The model had a root mean squared error of 1.05 which means there is not much variance between the actual values and the predicted values by the model. The graph shows us the same.
- The ADABoost model was trained with 500 weak classifiers. The model gave a Mean Squared Error of 118.43 and Root Mean Squared Error of 10.88. This is quite a significant value and could lead to lots of difference in the predicted value vs actual values.
- The AutoML model was used to train 17 different machine learning models. A leaderboard of all the models' logloss, rmse and mse were found by AutoML. The best model was found to be Gradient Boosting Machine(GBM) 5 with root mean squared value of 0.268 and accuracy of 96.34%.

6. Conclusion

From the above graphs and models we can conclude that AutoML's GBM5 model was found to be the most accurate with an accuracy of 96.34% and root mean square error of 0.268.

The plots give us an insight into all the hotspot locations of the war and city wise losses during the war. The trends in loss of equipment and their forecasted values tell us that extending the war would only lead to more death and more economic issues for both countries.

Additionally, the comparison of other models reveals that AutoML's GBM5 model performed better than other models in terms of accuracy, demonstrating that it can be a useful tool for prognosticating outcomes and making educated judgements.

The locations with the greatest effects from the war are highlighted on the hotspot maps, which might be helpful in distributing funding for post-war rehabilitation efforts. The city-wise loss trends give a thorough overview of how the war has affected different areas, emphasizing the necessity for specific assistance and support for the affected areas.

Github Link:

aryan-vigyat2001/Russia-Ukraine-War-Analysis--DV-J-Component (github.com)

7. References

[1] Giannakas, Filippos, et al. "Xgboost and deep neural network comparison: The case of teams' performance." Intelligent Tutoring Systems: 17th International Conference, ITS 2021, Virtual Event, June 7–11, 2021, Proceedings 17. Springer International Publishing, 2021.

[2] Barrow, Devon K., and Sven F. Crone. "A comparison of AdaBoost algorithms for time series forecast combination." International Journal of Forecasting 32.4 (2016): 1103-1119.

[3] Xiao, Ling, Yunxuan Dong, and Yao Dong. "An improved combination approach based on Adaboost algorithm for wind speed time series forecasting." Energy Conversion and Management 160 (2018): 273-288.

[4] Wang, Yan, and Yuankai Guo. "Forecasting method of stock market volatility in time series data based on mixed model of ARIMA and XGBoost." China Communications 17.3 (2020): 205-221.

[5] Polikar, Robi. "Ensemble learning." Ensemble machine learning: Methods and applications (2012): 1-34.

[6] Dong, Xibin, et al. "A survey on ensemble learning." Frontiers of Computer Science 14 (2020): 241-258.

[7] Uyanık, Gülden Kaya, and Neşe Güler. "A study on multiple linear regression analysis." Procedia-Social and Behavioral Sciences 106 (2013): 234-240.

[8] Sun, Meihong, and Chao Zhang. "Comprehensive analysis of global stock market reactions to the Russia-Ukraine war." Applied economics letters (2022): 1-8.

[9] Khudaykulova, Madina, He Yuanqiong, and Akmal Khudaykulov. "Economic consequences and implications of the Ukraine-russia war." International Journal of Management Science and Business Administration 8.4 (2022): 44-52.

[10] Boungou, Whelsy, and Alhonita Yatié. "The impact of the Ukraine–Russia war on world stock market returns." Economics Letters 215 (2022): 110516.

Appendix:

Github Code:

[aryan-vigyat2001/Russia-Ukraine-War-Analysis--DV-J-Component (github.com)]

Dataset Link

[2022 Russia Ukraine War | Kaggle]

Importing Dataset

```
library(ggplot2)
library(reshape2)
df1=read.csv("russia_losses_equipment.csv")
df2=read.csv("russia_losses_equipment_correction.csv")
df3=read.csv("russia_losses_personnel.csv")
head(df1)
```

```
##         date day aircraft helicopter tank APC field.artillery MRL military.auto
## 1 2022-02-25   2       10          7   80 516              49   4           100
## 2 2022-02-26   3       27         26  146 706              49   4           130
## 3 2022-02-27   4       27         26  150 706              50   4           130
## 4 2022-02-28   5       29         29  150 816              74  21           291
## 5 2022-03-01   6       29         29  198 846              77  24           305
## 6 2022-03-02   7       30         31  211 862              85  40           355
##   fuel.tank drone naval.ship anti.aircraft.warfare special.equipment
## 1        60     0          2                    0                NA
## 2        60     2          2                    0                NA
## 3        60     2          2                    0                NA
## 4        60     3          2                    5                NA
## 5        60     3          2                    7                NA
## 6        60     3          2                    9                NA
##   mobile.SRBM.system greatest.losses.direction vehicles.and.fuel.tanks
## 1                 NA                                               NA
## 2                 NA                                               NA
## 3                 NA                                               NA
## 4                 NA                                               NA
## 5                 NA                                               NA
## 6                 NA                                               NA
##   cruise.missiles
## 1              NA
## 2              NA
## 3              NA
## 4              NA
## 5              NA
## 6              NA
```

```
summary(df1)
```

```
##      date               day         aircraft       helicopter
##  Length:353         Min.   :  2   Min.   : 10.0   Min.   :  7.0
##  Class :character   1st Qu.: 90   1st Qu.:205.0   1st Qu.:170.0
##  Mode  :character   Median :178   Median :234.0   Median :197.0
##                     Mean   :178   Mean   :224.7   Mean   :202.9
##                     3rd Qu.:266   3rd Qu.:278.0   3rd Qu.:261.0
##                     Max.   :354   Max.   :296.0   Max.   :286.0
##
##       tank           APC       field.artillery      MRL        military.auto
##  Min.   :  80   Min.   : 516   Min.   :  49    Min.   :  4.0   Min.   : 100
##  1st Qu.:1302   1st Qu.:3194   1st Qu.: 606    1st Qu.:201.0   1st Qu.: 600
##  Median :1907   Median :4212   Median :1018    Median :266.0   Median :1178
##  Mean   :1940   Mean   :4191   Mean   :1157    Mean   :279.6   Mean   :1048
##  3rd Qu.:2871   3rd Qu.:5797   3rd Qu.:1860    3rd Qu.:393.0   3rd Qu.:1437
##  Max.   :3280   Max.   :6488   Max.   :2287    Max.   :465.0   Max.   :1701
##                                                                 NA's   :288
##    fuel.tank         drone          naval.ship    anti.aircraft.warfare
##  Min.   :60.00   Min.   :   0.0   Min.   : 2.00   Min.   :  0.0
##  1st Qu.:60.00   1st Qu.: 480.0   1st Qu.:13.00   1st Qu.: 93.0
##  Median :73.00   Median : 803.0   Median :15.00   Median :141.0
##  Mean   :69.32   Mean   : 920.9   Mean   :13.35   Mean   :139.7
##  3rd Qu.:76.00   3rd Qu.:1525.0   3rd Qu.:16.00   3rd Qu.:209.0
##  Max.   :76.00   Max.   :2007.0   Max.   :18.00   Max.   :234.0
##  NA's   :288
##  special.equipment mobile.SRBM.system greatest.losses.direction
##  Min.   : 10.0     Min.   :2.000      Length:353
##  1st Qu.: 53.0     1st Qu.:4.000      Class :character
##  Median :102.0     Median :4.000      Mode  :character
##  Mean   :106.2     Mean   :3.944
##  3rd Qu.:161.0     3rd Qu.:4.000
##  Max.   :215.0     Max.   :4.000
##  NA's   :19        NA's   :317
##  vehicles.and.fuel.tanks cruise.missiles
##  Min.   :1796            Min.   : 84.0
##  1st Qu.:2698            1st Qu.:155.0
##  Median :3620            Median :239.5
##  Mean   :3578            Mean   :345.2
##  3rd Qu.:4466            3rd Qu.:531.0
```

```
summary(df2)
```

```
##      date               day           APC      field.artillery     drone
##  Length:1           Min.   :231   Min.   :-25   Min.   :32      Min.   :20
##  Class :character   1st Qu.:231   1st Qu.:-25   1st Qu.:32      1st Qu.:20
##  Mode  :character   Median :231   Median :-25   Median :32      Median :20
##                     Mean   :231   Mean   :-25   Mean   :32      Mean   :20
##                     3rd Qu.:231   3rd Qu.:-25   3rd Qu.:32      3rd Qu.:20
##                     Max.   :231   Max.   :-25   Max.   :32      Max.   :20
##    naval.ship
##  Min.   :1
##  1st Qu.:1
##  Median :1
##  Mean   :1
##  3rd Qu.:1
##  Max.   :1
```

```
head(df3)
```

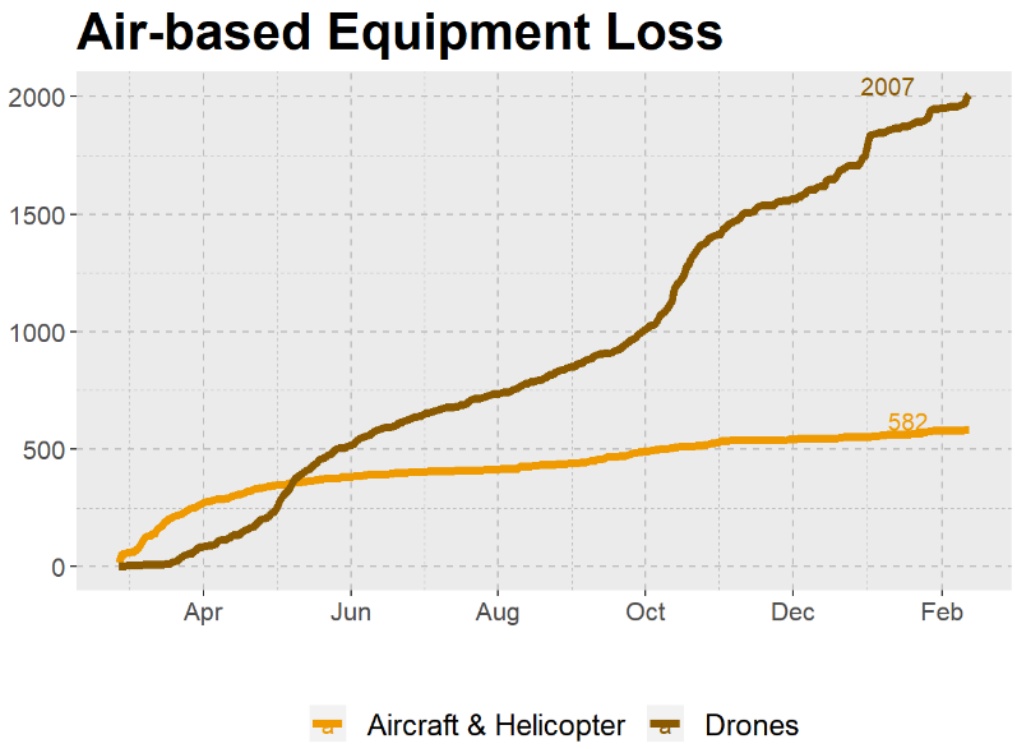
```
##         date day personnel personnel. POW
## 1 2022-02-25   2      2800      about   0
## 2 2022-02-26   3      4300      about   0
## 3 2022-02-27   4      4500      about   0
## 4 2022-02-28   5      5300      about   0
## 5 2022-03-01   6      5710      about 200
## 6 2022-03-02   7      5840      about 200
```

```
summary(df3)
```

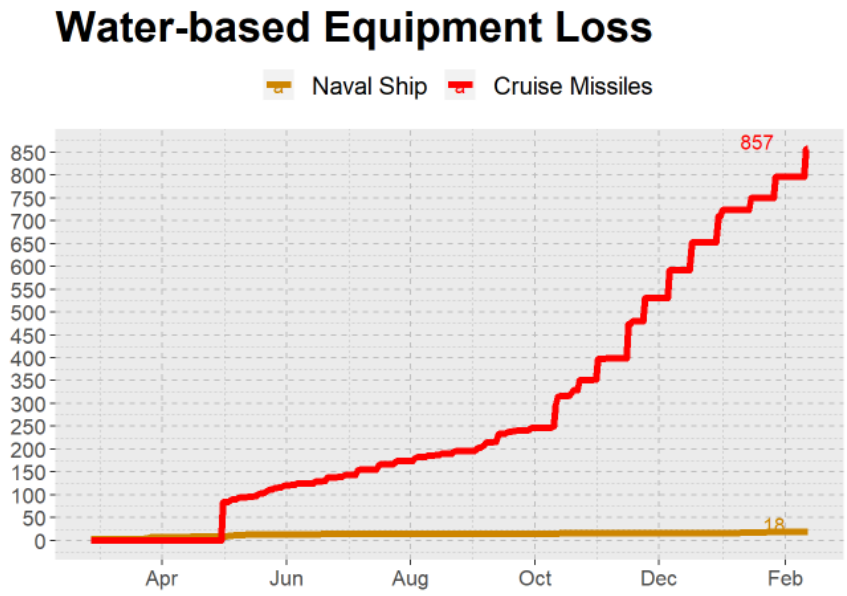
```
##      date               day        personnel       personnel.
##  Length:353         Min.   :  2   Min.   :  2800   Length:353
##  Class :character   1st Qu.: 90   1st Qu.: 29350   Class :character
##  Mode  :character   Median :178   Median : 44900   Mode  :character
##                     Mean   :178   Mean   : 55883
##                     3rd Qu.:266   3rd Qu.: 82710
##                     Max.   :354   Max.   :137780
##
##       POW
```

Data Cleaning

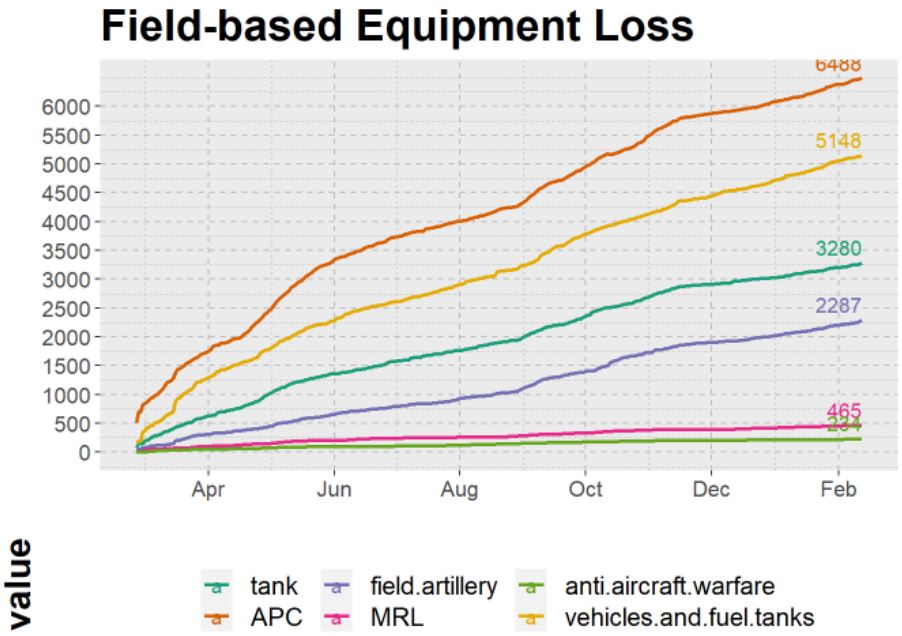
```
#Change date datatype
df1$date <- as.Date(df1$date, "%Y-%m-%d")
df3$date <- as.Date(df3$date, "%Y-%m-%d")

#Merge by inner join
dt <- merge(df1, df3, by = "date", all = TRUE)

#Combine columns

for (i in 1:nrow(dt))
  {
    if (is.na(dt$vehicles.and.fuel.tanks[i])==FALSE) next
      else {
         dt$vehicles.and.fuel.tanks[i] = dt$military.auto[i] + dt$fuel.tank[i]
          }
  }

for (i in 1:nrow(dt))
  {
   dt$air_vehicles[i] = dt$aircraft[i] + dt$helicopter[i]
  }

#Remove columns that are not useful
dt <- subset(dt, select= -c (personnel., military.auto, fuel.tank, helicopter,aircraft))

#Set NA to 0 in mobile SRBM system
dt$mobile.SRBM.system[is.na(dt$mobile.SRBM.system)] <- 0

#Replace NA with maxmimum in POW
dt$POW[is.na(dt$POW)] <- 496

# Replace any other NA present with 0
dt[is.na(dt)] <- 0
write.csv(dt,"newdf.csv")
```


Air-based Equipment Loss
2000
1500
1000
500
0
2007
582
Apr
Jun
Aug
Oct
Dec
Feb
Aircraft & Helicopter
Drones

```
# Water
melt_water <- dt %>% dplyr::select(c("naval.ship", "cruise.missiles", "date"))
melt_water <- melt(melt_water, id="date")

options(repr.plot.width=18, repr.plot.height=10)
ggplot(melt_water, aes(x=date,y=value, color=variable))+
  geom_line(size=1.8)+
  scale_y_continuous( limits = c(0, max(melt_water$value)), breaks= seq(0,max(melt_water$value),by=50))+
  scale_x_date(date_breaks='2 months', date_label = "%b")+
  geom_text(data = subset(melt_water, date==max(dt$date)),
            aes(label=value,hjust=2,vjust=0),
            size=4)+
  labs(title="Water-based Equipment Loss", color=NULL)+
  theme(title= element_text(face="bold", hjust=4, size=20),
        axis.text.x = element_text(size=12),
        axis.text.y = element_text(size=12),
        legend.position = "top",
        legend.text = element_text(size=14),
        panel.grid=element_line(size=0.4, color="gray", linetype=2))+
  scale_color_manual(values = c("orange3", "red"),
                     labels= c("Naval Ship", "Cruise Missiles"))
```



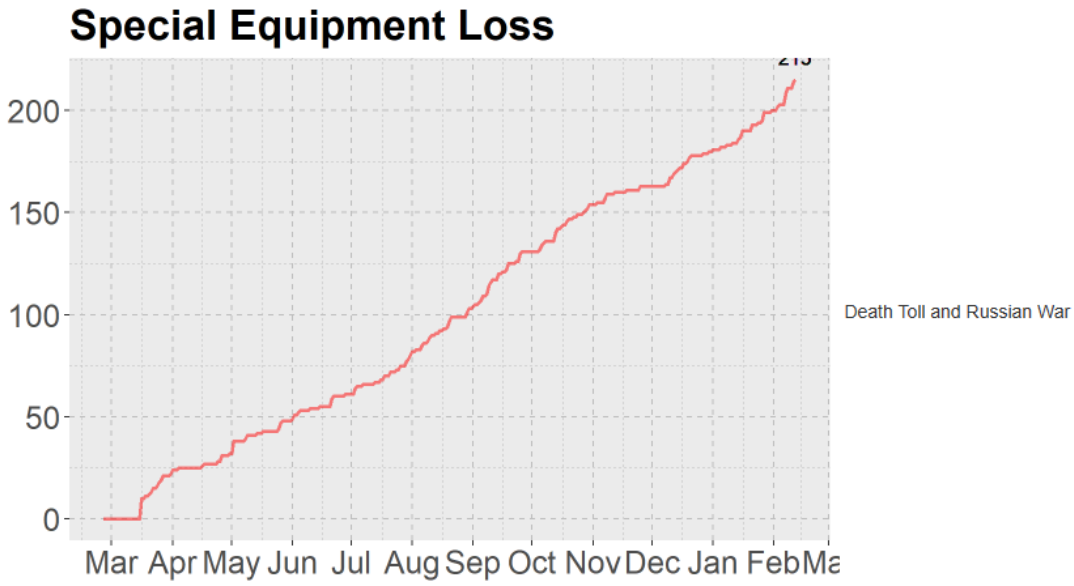
```
melt_field<- dt %>% dplyr::select(c("tank", "APC", "field.artillery","MRL","anti.aircraft.warfare","vehicles.and.fuel.tank
s","date"))
melt_field <- melt(melt_field, id="date")

options(repr.plot.width=18, repr.plot.height = 12)
ggplot(melt_field, aes(x=date,y=value, color=variable))+
  geom_line(size=1)+
  geom_text(data = subset(melt_field, date==max(dt$date)),
            aes(label=value,hjust=1,vjust=-0.5),
            size=4)+
  labs(title="Field-based Equipment Loss",color=NULL)+
  scale_y_continuous( limits = c(0, max(melt_field$value)), breaks= seq(0,max(melt_field$value),by=500))+
  scale_x_date(date_breaks= "2 months", date_label = "%b")+
  theme(title= element_text(face="bold", hjust=-0.5, size=20),
        axis.text.x = element_text(size=12),
        axis.text.y = element_text(size=12),
        legend.position = "bottom",
        legend.text = element_text(size=14),
        panel.grid=element_line(size=0.4, color="gray", linetype=2))+
        scale_color_brewer(type= "qual", palette = "Dark2")
```



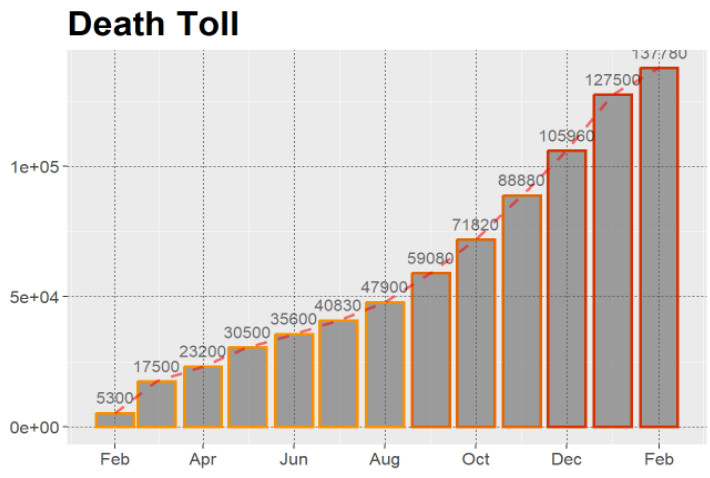
```
#Special Equipment Loss
ggplot(dt, aes(x= date,y= special.equipment)) +
geom_line(size=1, color="red", alpha=0.5)+
labs(title="Special Equipment Loss",x="",y="",color=NULL)+
geom_text(aes(x=max(dt$date), y=max(dt$special.equipment), label = max(dt$special.equipment)), size=4, vjust=-1)+
theme(title= element_text(face="bold", hjust=-0.5, size=20),
        axis.text.x = element_text(size=18),
        axis.text.y = element_text(size=18),
        legend.position = "bottom",
        legend.text = element_text(size=18),
        panel.grid=element_line(size=0.4, color="gray", linetype=2))+
        scale_color_brewer(type= "qual", palette = "Dark2")+
scale_x_date(date_breaks= "1 month", date_label = "%b")
```

```
## Warning: Use of `dt$date` is discouraged.
## i Use `date` instead.
```

```
## Warning: Use of `dt$special.equipment` is discouraged.
## i Use `special.equipment` instead.
## Use of `dt$special.equipment` is discouraged.
## i Use `special.equipment` instead.
```



```
options(repr.plot.width=20, repr.plot.height = 12)
 plot3 <- dt %>%
  group_by(month = lubridate::floor_date(date, "month")) %>%
  summarise(total=max(personnel)) %>%
  ggplot(., aes(x=month, y=total, alpha=0.1))+
  geom_col(aes(color=total), size=1)+
  geom_line(color="red", size = 1, linetype=2)+
  labs(title="Death Toll", x="", y="")+
  geom_text(aes(y=total,label=total), vjust=-0.8, size=4)+
  scale_x_date(date_breaks = '2 month', date_labels = "%b")+
  theme(panel.grid.major = element_line(size=0.2, color="gray3", linetype=2),
        axis.text.x = element_text(size=12),
        axis.text.y = element_text(size=12),
        title = element_text(size=20, face="bold"),
        legend.position="none")+
  scale_color_steps(low = "orange", high = "red3")



plot3
```



Splitting Data

```
library(caTools)
set.seed(123321)
df1[sapply(df1, is.character)] <- lapply(df1[sapply(df1, is.character)], as.factor)
split=sample.split(df1,SplitRatio = 0.8)
train=subset(df1,split=="TRUE")
test=subset(df1,split=="FALSE")
head(train)
```

```
##         date day aircraft helicopter tank APC field.artillery MRL military.auto
## 1 2022-02-25   2       10          7   80 516              49   4           100
## 2 2022-02-26   3       27         26  146 706              49   4           130
## 5 2022-03-01   6       29         29  198 846              77  24           305
## 6 2022-03-02   7       30         31  211 862              85  40           355
## 8 2022-03-04   9       33         37  251 939             105  50           404
## 9 2022-03-05  10       39         40  269 945             105  50           409
##   fuel.tank drone naval.ship anti.aircraft.warfare special.equipment
## 1        60     0          2                     0                NA
## 2        60     2          2                     0                NA
## 5        60     3          2                     7                NA
## 6        60     3          2                     9                NA
## 8        60     3          2                    18                NA
## 9        60     3          2                    19                NA
##   mobile.SRBM.system greatest.losses.direction vehicles.and.fuel.tanks
## 1                 NA                                               NA
## 2                 NA                                               NA
## 5                 NA                                               NA
## 6                 NA                                               NA
## 8                 NA                                               NA
## 9                 NA                                               NA
##   cruise.missiles
## 1              NA
## 2              NA
## 5              NA
## 6              NA
## 8              NA
## 9              NA
```

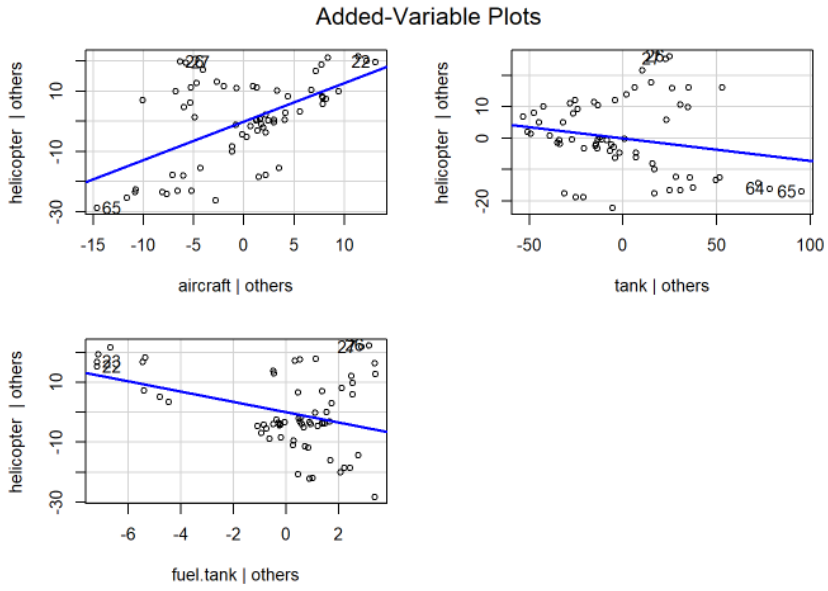
```
model=lm(helicopter~aircraft+tank+fuel.tank,df1)
summary(model) #p<0.05. there's a significant relationship between the predictor variables and the response variable
```

```
##
## Call:
## lm(formula = helicopter ~ aircraft + tank + fuel.tank, data = df1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.657  -6.857  -2.242   7.466  27.787
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 126.24891   31.64741   3.989  0.00018 ***
## aircraft      1.27752    0.23989   5.325 1.53e-06 ***
## tank         -0.07197    0.04552  -1.581  0.11903
## fuel.tank    -1.72683    0.55422  -3.116  0.00280 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.9 on 61 degrees of freedom
##   (288 observations deleted due to missingness)
## Multiple R-squared:  0.9225, Adjusted R-squared:  0.9187
## F-statistic: 242.1 on 3 and 61 DF,  p-value: < 2.2e-16
```

```
prediction=predict(model,newdata=data.frame(aircraft=25,tank=150,fuel.tank=60))
prediction
```

```
##        1
## 43.78178
```

```
avPlots(model)
```



```
train$helicopter=as.factor(train$helicopter)
model_adaboost <- boosting(helicopter~aircraft+tank+fuel.tank, data=train, nIter = 500, type = "real")
summary(model_adaboost)
```

```
##            Length Class   Mode
## formula         3  formula call
## trees         100  -none-  list
## weights       100  -none-  numeric
## votes       34250  -none-  numeric
## prob        34250  -none-  numeric
## class         274  -none-  character
## importance      3  -none-  numeric
## terms           3  terms   call
## call            5  -none-  call
```

```
#Make Predictions
pred_test=predict(model_adaboost,newdata=test[, c("aircraft", "tank", "fuel.tank")])
mse <- mean((as.numeric(pred_test$class) - test$helicopter)^2)
print(paste("Mean Squared Error: ", mse))
```

```
## [1] "Mean Squared Error:  118.430379746835"
```

```
library(caret)

set.seed(321123)
# Convert data to DMatrix format and using most important features
dtrain <- xgb.DMatrix(data = as.matrix(train[,c(3,5,10)]), label = train$helicopter)
dtest <- xgb.DMatrix(data = as.matrix(test[,c(3,5,10)]), label = test$helicopter)

#defining a watchlist
watchlist = list(train=dtrain, test=dtest)

# Train XGBoost model
model_xgb = xgb.train(data=dtrain, max.depth = 5, watchlist=watchlist, nrounds = 200)
```

```
## [1]  train-rmse:56.027804    test-rmse:187.890649
## [2]  train-rmse:39.644119    test-rmse:171.701243
## [3]  train-rmse:28.063240    test-rmse:160.474260
## [4]  train-rmse:19.873751    test-rmse:152.589847
## [5]  train-rmse:14.091442    test-rmse:147.016587
## [6]  train-rmse:10.009847    test-rmse:143.103477
## [7]  train-rmse:7.137581 test-rmse:140.317088
## [8]  train-rmse:5.108338 test-rmse:138.366889
## [9]  train-rmse:3.692073 test-rmse:136.981321
## [10] train-rmse:2.701361 test-rmse:135.998536
## [11] train-rmse:1.992717 test-rmse:135.288592
## [12] train-rmse:1.501754 test-rmse:134.775103
## [13] train-rmse:1.168276 test-rmse:134.416360
## [14] train-rmse:0.927716 test-rmse:134.167770
## [15] train-rmse:0.766808 test-rmse:133.969753
## [16] train-rmse:0.667391 test-rmse:133.839903
## [17] train-rmse:0.585397 test-rmse:133.747282
## [18] train-rmse:0.531792 test-rmse:133.680442
```

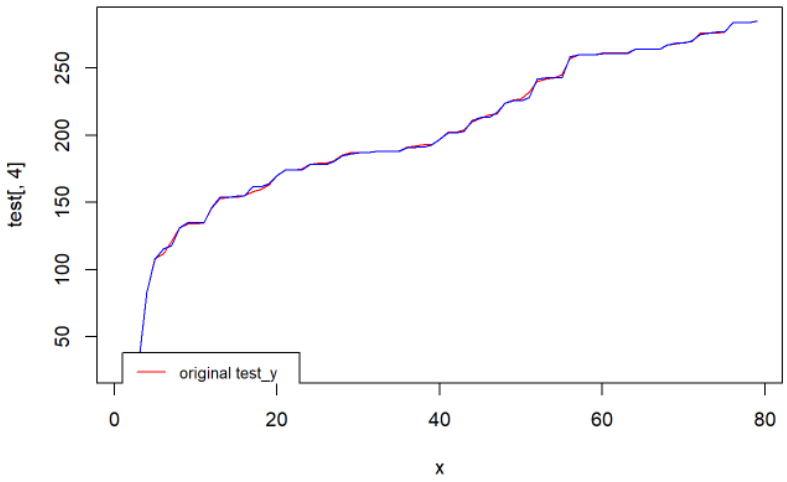
```
predictions <- predict(model_xgb, dtest)
#binary.predictions <- ifelse(predictions > 0.5, 1, 0)
#accuracy <- sum(binary.predictions == test$Attrition_Flag) /  length(test$Attrition_Flag)
#print(paste("Accuracy = ",accuracy))

print(paste("RMSE = ",RMSE(test[,4], predictions)))
```

```
## [1] "RMSE =  1.05197715130768"
```

```
tss =  sum((test[,4] - mean(test[,4]))^2 )
```

Visualize

```
x = 1:length(test[,4])                  # visualize the model, actual and predicted data
plot(x, test[,4], col = "red", type = "l")
lines(x, predictions, col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```



```
df_hf <- as.h2o(training)
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
y <- "outcome"
x <- names(training)[c(5,6,7,8,9,22)] #statename, side, starty,startm,startd,combatfat

aml <- h2o.automl(x = x, y = y,
                  training_frame = df_hf,
                  max_models = 15)
```

```
##
  |
  |                                                                      |   0%
  |
  |=                                                                     |   1%
## 16:51:13.188: AutoML: XGBoost is not available; skipping it.
## 16:51:13.254: _train param, Dropping bad and constant columns: [state_name]
  |
  |==                                                                    |   3%
## 16:51:15.506: _train param, Dropping bad and constant columns: [state_name]
## 16:51:16.561: _train param, Dropping bad and constant columns: [state_name]
  |
  |========                                                              |  11%
## 16:51:17.502: _train param, Dropping bad and constant columns: [state_name]
## 16:51:18.742: _train param, Dropping bad and constant columns: [state_name]
  |
  |============                                                          |  18%
## 16:51:19.663: _train param, Dropping bad and constant columns: [state_name]
## 16:51:20.378: _train param, Dropping bad and constant columns: [state_name]
## 16:51:21.179: _train param, Dropping bad and constant columns: [state_name]
  |
```

```
lb <- aml@leaderboard
print(lb, n = nrow(lb))
```

```
##                                                  model_id
## 1                          GBM_5_AutoML_14_20230403_104225
## 2                          XRT_1_AutoML_14_20230403_104225
## 3                          DRF_1_AutoML_14_20230403_104225
## 4              GBM_grid_1_AutoML_14_20230403_104225_model_2
## 5  StackedEnsemble_BestOfFamily_1_AutoML_14_20230403_104225
## 6     StackedEnsemble_AllModels_1_AutoML_14_20230403_104225
## 7                          GBM_4_AutoML_14_20230403_104225
## 8              GBM_grid_1_AutoML_14_20230403_104225_model_3
## 9                          GBM_2_AutoML_14_20230403_104225
## 10                         GBM_3_AutoML_14_20230403_104225
## 11             GBM_grid_1_AutoML_14_20230403_104225_model_1
## 12    DeepLearning_grid_1_AutoML_14_20230403_104225_model_1
## 13    DeepLearning_grid_3_AutoML_14_20230403_104225_model_1
## 14                DeepLearning_1_AutoML_14_20230403_104225
## 15                         GLM_1_AutoML_14_20230403_104225
## 16    DeepLearning_grid_2_AutoML_14_20230403_104225_model_1
## 17                         GBM_1_AutoML_14_20230403_104225
##    mean_per_class_error   logloss      rmse        mse
## 1             0.3142149 0.3149237 0.2686622 0.07217938
## 2             0.3314886 0.6661353 0.4149923 0.17221861
## 3             0.3845667 0.8321450 0.3152004 0.09935132
## 4             0.4190768 0.3513080 0.2888671 0.08344420
## 5             0.4321451 0.3755353 0.3034317 0.09207078
## 6             0.4324078 0.3510587 0.2949666 0.08700529
## 7             0.4339969 0.3589757 0.2945908 0.08678372
## 8             0.4353772 0.3424001 0.2877712 0.08281228
## 9             0.4361116 0.3701043 0.2949597 0.08700122
## 10            0.4440481 0.3622663 0.3054266 0.09328542
```

Testing

```
test <- as.h2o(testing)
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
model <- aml@leader
p1 = h2o.predict(model, newdata=test)
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
df2 <- as.data.frame(p1$predict)
df2$predict <- factor(df2$predict, levels = c(1,2,3,4,6,8))
mean(df2$predict==testing$outcome)
```

```
## [1] 0.9634146
```

```
confusionMatrix(df2$predict,testing$outcome)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  4  6  8
##          1 37  0  0  0  0  0
##          2  0 28  1  0  0  0
##          3  0  0  0  0  0  0
##          4  0  1  0  7  0  0
##          6  1  0  0  0  7  0
##          8  0  0  0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.9634
##                  95% CI : (0.8968, 0.9924)
##     No Information Rate : 0.4634
##     P-Value [Acc > NIR] : < 2.2e-16
##
```