1 Introduction

The article uses 205 national cyberdefense strategies, internet and communication technology strategies, and e-governance strategies from 110 countries from the years 2000 to 2020 as its dataset. *Each strategy was compiled into a txt file and was translated into English. We downloaded the English version of each strategy into a txt file.* To determine the contents of each strategy and whether it called for an open or closed internet we created a training set using sentence tokenization and two human reviewers. We then used this training set on a machine learning algorithm in python called sci-kit learn Random Forest Classifier or Support Vector Machine. The options for categorization were open, closed, and irrelevant.

2 Training Set

To produce our training set, we ran every country's respective ICT strategy policy through a sentence tokenizer and pulled 5 percent of the sentences. These 5558 lines were individually coded by two human reviewers using the key; 1 for open, 2 for closed, and 3 for irrelevant. Examples of sentences that were coded as 1, 2, and 3 by both coders can be seen below.

2.1 Open, Code 1

- **Japan, "Japan National Cybersecurity Strategy", 2015**

  Japan will further cooperate with domestic and foreign stakeholders in these discussions and actively promote the development of international rules and norms with a view that ensuring openness, interoperability, autonomy and the free flow of information in cyberspace will make a significant contribution to the development of society, economy, and culture.

- **Sweden, "Sweden National Cybersecurity Strategy", 2017**

  Access to an open, free and secure internet constitutes an important instrument for the global enhancement of human rights, democracy, the rule of law and development.

- **Norway, "Norway National Cybersecurity Strategy", 2017**

  Give greater emphasis to robust infrastructure and security and openness in cyberspace as part of the sustainable development agenda, in the context of the Government's global efforts to combat poverty and promote trade, social and economic development, education, health care, human rights and democracy.

- **Norway, "Norway Digital Agenda", 2013**

  Internet users are entitled to an Internet connection that is free of discrimination with regard to type of application, service or content or based on sender or receiver address.

- **Japan,"Japan National Cybersecurity Strategy", 2013**

  Activity regions for policies indicated as priority policies requiring domestic and international as well as private and public sector cooperation include (1) economics, (2) protecting our networks, (3) law enforcement, (4) military, (5) internet cyberspace which supports international trade, strengthens international security, promotes freedom of expression and innovation, is open and interoperable, as well as secure and reliable; the strategy for protecting economic and social values without impeding innovation in the internet business field;43 the "Department of Defense Strategy for Operating in Cyberspace"44 which adds cyberspace to the preexisting land, sea, air and outer space regions; and "The Cybersecurity Strategy for the Homeland Security Enterprise",45 which aims for a cyberspace which supports secure and resilient infrastructure, brings about innovation and prosperity and protects citizen's freedoms such as privacy from the initial design stages, and also plans for the establishment of protections for critical information infrastructure and cyber-ecosystems.

- **Afghanistan, "Afghanistan EGovernment Strategy", 2011**

  These goals of Universal and Open Access can be achieved by implementing the following steps:

2.2 Closed, Code 2

- **Latvia, "Latvia ICT Policy", 2013**

  According to the European Public Sector Information Platform data, nearly no measures to promote the use of open data (information on available data sets, educational and solution-creation facilitating measures and initiatives) have been introduced in Latvia.

- **Montenegro, "Montenegro National Cybersecurity Strategy", 2013**

  Law enforcement authorities and prosecution should be able to conduct investigations and prosecute offences against computer data and systems, offences committed over computers, as well as electronic evidence material relating to any criminal offence.

- **Afghanistan, "Afghanistan ICT Policy", 2015**

  Appropriate ICT laws and regulations would be developed and deployed to monitor and regulate the ICT sector consistent with the need to respect the human rights of Afghanistan citizens.

- **Sweden, "Sweden National Cybersecurity Strategy", 2017**

  The proposal includes a new crime making it punishable to disseminate certain images and data that are sensitive in terms of privacy.

- **United States,**

  The United States believes deterrence in cyberspace is best accomplished through a combination of "deterrence by denial" – reducing the incentive of potential adversaries to use cyber capabilities against the United States by persuading them that the United States can deny their objectives – and "deterrence through cost imposition" – threatening or carrying out actions to inflict penalties and costs against adversaries that conduct malicious cyber activity against the United States.

2.3 Irrelevant, Code 3

- **Saudi Arabia, "Saudi Arabia ICT Policy", 2013**

  This strategy recommends updating and hardening the ICT infrastructure of the Kingdom to a state of acceptable resilience.

- **Saudi Arabia, "Saudi Arabia ICT Policy", 2013**

  Finally, a comprehensive program beginning in primary school identifies and encourages bright and interested children to acquire computer, analytic and IS skills.

- **Montenegro, "Montenegro National Cybersecurity Strategy", 2013**

  There have been a series of introductory e-Government workshops that have taken place in recent times; however, they have only been effective in increasing e-Government awareness and familiarity for staff.

- **Colombia, "Colombia National Cybersecurity Strategy", 2011**

  Furthermore, Colombian government agencies have been raising awareness about the importance of developing a cybersecurity and cyberdefense policy since 2007.

- **Swaziland, "Swaziland ICT Policy", 2012**

  The country generally lags behind in terms of developing and/or adopting new ICT systems and technologies aimed at improving the delivery of financial products to the public.

3 ML Coding of Labels

3.1 Random Forest Models

A Random Forest is an ensemble learning technique for classification, regression, and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. The key concepts include:

- **Ensemble Learning:** Random Forest leverages the power of multiple decision trees to improve prediction accuracy. Each tree in the forest is built from a random sample of the data, leading to diversity among the trees. This diversity is crucial for reducing overfitting, as errors from individual trees are likely to be uncorrelated.
- **Bagging (Bootstrap Aggregating):** Each tree is trained on a different bootstrap sample of the data, with replacement. This process introduces variability among the trees, enhancing the ensemble's overall predictive power.
- **Feature Randomness:** When splitting a node, the model considers a random subset of features, rather than all features. This strategy contributes to diversity among trees, reducing the risk of overfitting on the training data.

Hyperparameter Adjustments and Their Effects
The adjustments made to the 'n_estimators' and 'max_depth' parameters directly influence the model's complexity and its ability to generalize:

- **Increasing 'max_depth' from 10 to 15:** This increases the number of trees in the forest. More trees can lead to better model performance because the ensemble can capture more complex patterns and reduce variance through averaging predictions across more trees. However, the benefits tend to diminish after a certain point, and computation time increases linearly with the number of trees.
- **Increasing 'n_estimators' from 200 to 250:** This allows the trees to grow deeper, enabling them to model more complex relationships by creating more decision boundaries. However, if the depth is too high, there's a risk of overfitting, as the model may start to learn noise from the training data rather than the actual signal.

Impact of Sample Size Reduction
Reducing the sample size, as Adaiba did, leads to a decrease in model performance (from 89% to 81% accuracy) for several reasons:

- **Reduced Generalization:** A smaller training set provides less information about the underlying distribution of the data, making it more challenging for the model to generalize well to unseen data.
- **Increased Variance:** With fewer data points, the model's predictions become more sensitive to the specific samples included in the training set, leading to higher variance and, consequently, lower accuracy on the test set.
- **Less Robust Feature Selection:** The model has fewer opportunities to identify and leverage the most informative features, potentially leading to suboptimal splitting decisions in the trees.

The adjustments made to the Random Forest hyperparameters—specifically increasing the number of estimators and the maximum depth of the trees—initially led to an improvement in accuracy by allowing the model to capture more complex patterns and reduce variance. However, the subsequent reduction in sample size highlighted a critical trade-off in machine learning: while a model can be tuned to perform exceptionally well on a large dataset, its performance can significantly degrade if the amount of training data is reduced, underscoring the importance of having a sufficiently large and representative dataset for training.

3.2 Support Vector Classification Models

Support Vector Machines are supervised learning models used for classification and regression tasks. The core idea behind SVM is to find the hyperplane that best divides a dataset into classes. The primary concepts include:

- **Maximizing Margin:** SVM seeks to maximize the margin between the separating hyperplane and the nearest points from each class, known as support vectors. This maximization is crucial for improving the model's generalization ability.
- **Kernel Trick:** SVM uses a technique called the kernel trick to transform the input space into a higher-dimensional space where it is easier to find a linear separating hyperplane. This trick allows SVM to capture complex relationships without explicitly computing the dimensions in the higher-dimensional space.
- **Regularization:** SVM includes a regularization parameter (often denoted as C) that controls the trade-off between achieving a low training error and maintaining a low model complexity for better generalization.

Hyperparameter Adjustments and Their Effects

- **Changing Kernel from 'rbf' to 'poly':** The 'rbf' kernel is effective for non-linear problems, adapting to data of varying scales and making non-linear separations possible. Switching to a 'poly' kernel, which represents polynomial transformations, changes the way the SVM models the data. Polynomial kernels can capture relationships between features in datasets where the degree of connections is important. This change can lead to better modeling of complex patterns if the polynomial degree aligns well with the data's structure.

- **Adjusting class_weight to 'balanced':** This adjustment is crucial for datasets with imbalanced classes. By default, SVM treats all classes equally, which can lead to suboptimal performance on minority classes. Setting class_weight to 'balanced' automatically adjusts weights inversely proportional to class frequencies in the input data. This approach helps in giving more importance to minority classes, potentially improving overall accuracy by enhancing the model's ability to generalize across all classes.

Impact of Sample Size Reduction

Decreasing the sample size, as Adaiba experienced, led to a reduction in model accuracy (from 92% to 81%). This decline can be attributed to several factors:

- **Reduced Generalization Capability:** A smaller dataset provides less information for the model to learn from, potentially leading to a weaker generalization on unseen data. This reduction in data can particularly affect SVMs, as the determination of support vectors and the optimal separating hyperplane relies heavily on the representativeness of the training data.

- **Increased Sensitivity to Class Imbalance:** With fewer data points, the effects of class imbalance can become more pronounced, especially if the class_weight adjustment does not fully compensate for the reduced diversity and quantity of examples for each class.

- **Decreased Model Complexity Capacity:** With ample data, SVM can afford to construct more complex models (especially when using kernels like 'poly' with higher degrees) without overfitting. However, with less data, the model may not be able to leverage these complex transformations effectively without risking overfitting.

The adjustments made to the SVM model's hyperparameters—switching from an 'rbf' to a 'poly' kernel and adjusting the class weights to 'balanced'—initially led to an improvement in accuracy by enhancing the model's ability to capture complex patterns and address class imbalance. However, the reduction in sample size highlighted the critical role of having a sufficiently large and diverse dataset. Adequate data is essential for the SVM model to effectively learn the underlying patterns and maintain high performance, especially when dealing with complex models and imbalanced classes.

3.3 MLP Classifier Models

An MLPClassifier implements a multi-layer perceptron algorithm that trains using backpropagation. It is a class of feedforward artificial neural network (ANN) that consists of at least three layers of nodes: an input layer, a hidden layer(s), and an output layer. Each node, except for the input nodes, is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training.

- **Backpropagation:** It is the essence of neural net training. It consists of a forward pass where the input data is passed through the network to generate an output, and a backward pass where the error between the predicted output and the true output labels is propagated back through the network to adjust the weights.
- **Solver for Weight Optimization:** The solver 'adam' is an adaptive learning rate method, which means it adjusts the learning rate as it learns, making it suitable for larger datasets. 'Lbfgs' is an optimizer in the family of quasi-Newton methods, and it tends to perform better on smaller datasets due to its efficient handling of the curvature of the loss function, which can lead to faster convergence.

Reasoning Behind Hyperparameter Selections and Adjustments

- **Initial Solver Choice ('adam'):** Given its efficiency on larger datasets in both training time and validation score, 'adam' was the default choice. It's well-suited for situations where you have thousands of training samples or more.
- **Switch to 'lbfgs' Solver:** For smaller datasets, 'lbfgs' can converge faster and perform better because it makes more informed update steps by approximating the second derivative of the loss function, leading to more precise weight updates.
- **Removing the max_iter Limit:** This allows the training process to continue until convergence (based on the tolerance levels) rather than stopping at a predetermined number of iterations. This can ensure the model is fully trained and can utilize the full potential of the dataset for learning, leading to improved accuracy.

Impact of Data Size Changes on Model Performance

- **Initial High Amount of Training Data (89% Accuracy):** With a substantial dataset, the MLPClassifier could learn the complex patterns and relationships within the data, leading to a high accuracy rate.
- **Accuracy Increase to 93%:** By optimizing the solver to 'lbfgs' and removing the max_iter limit, the model could better navigate the solution space and effectively adjust its weights to minimize the loss, thus improving accuracy on the test set.

- **Decrease in Sample Size (Dropped to 82%):** Reducing the sample size likely led to the model having less information to learn from, which can result in a poorer generalization to new data. This effect is exacerbated in neural networks, including MLPs, which thrive on large datasets to capture the underlying data distribution accurately.

The adjustments made to the MLPClassifier's hyperparameters were strategically chosen to optimize the model's learning from the available data, which initially led to an increase in test set accuracy. However, the reduction in the sample size highlights a critical challenge in machine learning: a model's performance is heavily dependent on the quantity and quality of the training data. Adequate data allows the model to learn the intricacies of the input space and generalize well to unseen data, whereas insufficient data can severely hamper this learning process.

4 Update to ML Training

4.1 Introduction

This report provides an analysis of the performance of three different machine learning models: Logistic Regression, Random Forest, and Support Vector Machine (SVM), applied to classify textual data into three categories. The goal is to evaluate each model's effectiveness based on precision, recall, and f1-score metrics.

4.2 Methodology

The dataset comprises text entries categorized into three classes. We split the data into training and test sets, with a test size of 5%. Each model was trained using a pipeline that includes TF-IDF vectorization of the text data. The performance was evaluated using the classification report from scikit-learn, focusing on precision, recall, f1-score, and accuracy.

4.3 Results and Discussion

4.3.1 Logistic Regression

The Logistic Regression model showed moderate effectiveness, with an overall accuracy of 84%. However, it struggled with categories 1 and 2, showing a significant imbalance in precision and recall. The model performed well for category 3, which is the majority class, indicating a bias towards more frequently occurring classes.

4.3.2 Random Forest

The Random Forest classifier performed poorly for the minority classes (1 and 2), with zero precision and recall, indicating it failed to identify any true pos-

Class	Precision	Recall	F1-Score
1	0.70	0.17	0.28
2	0.00	0.00	0.00
3	0.85	0.99	0.91

Table 1: Logistic Regression Classification Report

itives for these classes. Like Logistic Regression, it performed well for the majority class (category 3) but with an overall accuracy slightly lower than that of Logistic Regression.

Class	Precision	Recall	F1-Score
1	0.00	0.00	0.00
2	0.00	0.00	0.00
3	0.83	1.00	0.91

Table 2: Random Forest Classification Report

4.3.3 Support Vector Machine

The SVM classifier showed the best overall performance among the three models, with an accuracy of 85%. It demonstrated a better balance in handling the majority class and one of the minority classes (category 1). However, it still showed no improvement in classifying category 2.

Class	Precision	Recall	F1-Score
1	0.75	0.23	0.35
2	0.00	0.00	0.00
3	0.85	0.99	0.91

Table 3: Support Vector Machine Classification Report

4.4 Conclusion

The analysis highlights challenges in class imbalance and the models' ability to generalize across less frequent categories. While the SVM showed the most promise, further tuning and perhaps more sophisticated techniques for handling class imbalance are recommended for future work.

5 Grid Search

5.1 Introduction

Grid search is a widely used technique for hyperparameter tuning in machine learning models. The method involves specifying a grid of hyperparameter values and evaluating a model's performance for each combination of these parameters in order to find the optimal set that yields the best performance according to a predefined metric, typically accuracy, precision, recall, or a combination thereof. This brute-force approach is exhaustive and ensures that the best combination of parameters from the specified grid is found, but can be computationally expensive, especially with a large number of hyperparameters and/or wide ranges of values.

5.2 Methodology

In this project, we applied grid search to tune hyperparameters for three different models: Logistic Regression, Random Forest, and Support Vector Machine (SVM). We used the scikit-learn library's `GridSearchCV` function, which not only performs grid search across the specified parameter grid but also implements cross-validation to ensure that the model's performance is robust across different subsets of the training data.

5.3 Model Setup

For each model, we defined a range of hyperparameters to explore:

- **Logistic Regression**: Regularization strength (`C`) and solver algorithm.
- **Random Forest**: Number of trees (`n_estimators`) and maximum depth of the trees (`max_depth`).
- **Support Vector Machine**: Regularization parameter (`C`), kernel type, and gamma coefficient for the kernel.

5.4 Results and Discussion

5.4.1 Logistic Regression

- Best Parameters: The best performing Logistic Regression model used a regularization strength (C) of 10, with the liblinear solver, a maximum iteration count of 100, and l2 penalty.
- Cross-Validation Score: The best cross-validation accuracy achieved was approximately 84.62%, indicating a strong performance across different subsets of the training data.

- Classification Report: The model showed a high precision (86%) and recall (97%) for class 3, which seems to be the majority class. Performance on class 1 and class 2 was considerably lower, with class 2 having no correct predictions. Overall accuracy stood at 84%, with a weighted average precision of 80% and a similar recall rate.

5.4.2 Random Forest

- Best Parameters: For the Random Forest classifier, the optimal setup involved 1000 trees, a maximum depth of 30, no limitation on the maximum features considered for splitting, at least 2 samples required to be at a leaf node, and at least 5 samples required to split an internal node, with bootstrapping enabled.
- Cross-Validation Score: Achieved a cross-validation accuracy of 84.17%, slightly lower than the Logistic Regression model but still robust.
- Classification Report: The model had a relatively lower performance on class 1 with a precision of 36% and recall of 10%. Similar to Logistic Regression, class 2 saw no correct predictions. A high precision (83%) and recall (97%) for class 3 indicates good performance on the majority class but overall lower performance on minority classes compared to Logistic Regression.

5.4.3 Support Vector Machine

- Best Parameters: The SVM tuned to a linear kernel with C set to 1, degree at 2 (though not applicable for linear kernels), and default gamma ('scale').
- Cross-Validation Score: The SVM model matched the Logistic Regression in cross-validation accuracy at approximately 84.68%, indicating its efficacy.
- Classification Report: SVM showed improvement in handling class 1 with a higher precision (75%) but still a low recall (23%), suggesting some overfitting to the majority class. No improvement was seen in class 2 predictions. For class 3, similar high precision and recall rates as Logistic Regression, reinforcing its strong performance on this class.

5.5 Conclusion

All models demonstrated high accuracy and performance metrics for class 3, the majority class. This suggests that while the models are effective at capturing the patterns in the majority class, they struggle with the minority classes (1 and 2).

The relatively lower performance on minority classes across all models indicates a potential area for improvement, possibly through techniques such as

SMOTE (Synthetic Minority Over-sampling Technique) for balancing class distribution, or exploring model-specific adjustments to better capture the characteristics of these less represented classes.

Given the comparable performance between Logistic Regression and SVM, with slight variations in class-specific metrics, the choice between these models might come down to factors like training time, interpretability, and computational resources. Random Forest, while slightly behind in cross-validation accuracy, offers benefits in terms of model robustness and feature importance understanding.

6 Comparison of ML Models - 4/15

This report compares the performance of various classification models on a dataset. The models evaluated include Logistic Regression, Random Forest, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Multinomial Naive Bayes, and Multilayer Perceptron. Performance metrics such as precision, recall, and F1-score are reported for three classes.

The classification performance for each model is summarized in the tables below. Each table presents precision, recall, and F1-score for three classes identified by the models.

6.1 Logistic Regression

Class	Precision	Recall	F1-Score
1	0.11	0.02	0.03
2	0.80	0.27	0.40
3	0.79	0.96	0.87

Table 4: Logistic Regression Classification Report

6.2 Random Forest

Class	Precision	Recall	F1-Score
1	0.00	0.00	0.00
2	0.88	0.47	0.61
3	0.80	1.00	0.89

Table 5: Random Forest Classification Report

Class	Precision	Recall	F1-Score
1	0.00	0.00	0.00
2	0.75	0.20	0.32
3	0.80	1.00	0.89

Table 6: Support Vector Machine Classification Report

6.3 Support Vector Machine

6.4 K-Nearest Neighbors

Class	Precision	Recall	F1-Score
1	0.00	0.00	0.00
2	0.00	0.00	0.00
3	0.79	1.00	0.88

Table 7: K-Nearest Neighbors Classification Report

6.5 Multinomial Naive Bayes

Class	Precision	Recall	F1-Score
1	0.00	0.00	0.00
2	0.00	0.00	0.00
3	0.79	1.00	0.88

Table 8: Multinomial Naive Bayes Classification Report

6.6 Multilayer Perceptron

Class	Precision	Recall	F1-Score
1	0.26	0.16	0.20
2	0.50	0.27	0.35
3	0.80	0.89	0.84

Table 9: Multilayer Perceptron Classification Report

Across different models, Class 3 consistently shows high performance in terms of all three metrics, indicating the models' effectiveness in recognizing and predicting this class. Classes 1 and 2, however, exhibit poor performance with several models failing to correctly predict any instances of Class 1. This could be attributed to imbalanced class distribution, insufficient model training for these classes, or inherent difficulties in distinguishing these classes from others in the dataset.

6.7 Analysis

The performance of each model is outlined below, detailing the accuracy, macro average F1-score, and weighted average F1-score.

1. **Logistic Regression**
    - Accuracy: 0.77
    - Macro average F1-Score: 0.43
    - Weighted average F1-Score: 0.70
2. **Random Forest**
    - Accuracy: 0.80
    - Macro average F1-Score: 0.50
    - Weighted average F1-Score: 0.72
3. **Support Vector Machine (SVM)**
    - Accuracy: 0.79
    - Macro average F1-Score: 0.40
    - Weighted average F1-Score: 0.71
4. **K-Nearest Neighbors (KNN)**
    - Accuracy: 0.79
    - Macro average F1-Score: 0.29
    - Weighted average F1-Score: 0.70
5. **Multinomial Naive Bayes**
    - Accuracy: 0.79
    - Macro average F1-Score: 0.29
    - Weighted average F1-Score: 0.70
6. **Multilayer Perceptron**
    - Accuracy: 0.74
    - Macro average F1-Score: 0.46
    - Weighted average F1-Score: 0.71

From the summaries, the Random Forest model emerges as the best overall. It has the highest accuracy at 0.80 and the highest weighted average F1-Score at 0.72. Additionally, its macro average F1-Score of 0.50 is one of the highest, indicating a relatively balanced performance across all classes compared to the other models. The Random Forest model also shows particularly strong performance in predicting Class 3, with an F1-Score of 0.89, and the best performance for Class 2 among all models.

In summary, while the Random Forest model is strong in Class 3 predictions and has a respectable score for Class 2, its weakness lies in Class 1 predictions where it fails entirely, as do many other models. However, its overall robustness in accuracy and F1-Scores makes it the best choice among the evaluated models.