Evolutionary Computation and Machine Learning for Fake News Detection

by

Marcellus Smith

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
May 7, 2022

Keywords: Fake News, Feature Selection, GEFeS, GAT, Auto-Labeler, EMO-GAT

Approved by

Gerry Dozier, Chair, Charles D. McCrary Eminent Chair Professor of Computer Science and Software Engineering
Cheryl Seals, Charles W. Barkley Endowed Professor of Computer Science and Software Engineering
Jakita Thomas, Philpott Westpoint Stevens Associate Professor of Computer Science and Software Engineering
Dean Hendrix, Associate Professor of Computer Science and Software Engineering
Michael King, Associate Professor of Computer Engineering and Sciences

Abstract

Misinformation or fake news has a history of being weaponized in order to deceive or mislead a target audience. Presently, the ease of generation, dissemination, and effects that are achievable by leveraging fake news makes fake news detection a critical issue that needs to be addressed. Effectively detecting fake news is faced with many challenges. Some of these challenges include the countless features that are able to be extracted from a text corpus, adversarial effects on fake news detection systems, and wide scale propagation of information. This document outlines research aimed at tackling and overcoming these challenges. Genetic and evolutionary feature selection (GEFeS) is combined with fake news detection to improve detection accuracy and identify essential features; this work was published in IEEE Symposium Series on Computational Intelligence [1]. A novel method is proposed for selecting subsets adversarial examples for adversarial training in order to strengthen the defensive posture of a machine learning system; this work was published in IEEE Congress on Evolutionary Computation [2]. Determining the propagation of information types during an infodemic and building an auto-labeler; this work was published in IEEE SoutheastCon. The novel method, genetic adversarial training, is extended to overcome learning constraints and to incorporate multi-objective optimization; this work was published in IEEE Symposium Series on Computational Intelligence [3]. The work concerning the propagation of information types is extended to examine a broader scope of information and leverages a more advanced auto-labeling method; this work is pending publication.

Acknowledgments

I wish to express my sincere gratitude to my dissertation committee members for their guidance, support, and commitment in my academic journey. Specifically, I want to give the utmost thanks to my committee chair and advisor, Dr. Dozier. Without Dr. Dozier's support and dedication, my academic journey would have been fraught with mistakes and aimless efforts. I would also like to thank Dr. King, Dr. Michel, and PhD candidate Brandon Brown for their expertise and efforts in the included research.

This research was partially supported by the National Science Foundation under Grant No. NSF-DGE-1663616. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

This research was completed in part with resources provided by the Auburn University Easley Cluster. Auburn has named this cluster in honor of Annie J. Easley. Easley helped usher science and technology into a new era while overcoming both racial and gender discrimination. Her contributions include advancements in computer science, mathematics, storage batteries, and aerospace engineering.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AE	Adversarial Examples
AI	Artificial Intelligence
AT	Adversarial Training
CNN	convolutional neural network
CS	Candidate Solutions
EMO	Evolutionary Multi-objective Optimization
EMO-GAT	Evolutionary Multi-objective Genetic Adversarial Training
FE	Function Evaluation
FN	False Negative
FP	False Positive
GAT	Genetic Adversarial Training
GEFeS	Genetic and Evolutionary Feature Selection
KNN	k-Nearest Neighbors
LIWC	Linguistic Inquiry and Word Count
LPC	LIWC Psychometric Category
ML	Machine Learning

NB Naive Bayes

NSGA-II Non-dominated Sorting Genetic Algorithm II

PDF Probability Density Function

PE Probability Estimates

RF Random Forest

SSGA Steady State Genetic Algorithm

SVM Support Vector Machine

XGB XGBoost

Chapter 1

Introduction

1.1 Motivation

Fake news can be defined as fabricated information that mimics news media content in form but not in organizational process or intent [4]. One of the first recorded uses of fake news was in 13th century BC [5] when Rameses the Great spread misinformation claiming victory over the Egyptians in the Battle of Kadesh that actually resulted in a stalemate. This falsely claimed victory and disinformation allowed Rameses the Great to achieve peace within his home front. Since then, the characteristics of fake news have evolved as society has progressed. Modern society now enables any individual the ability to disseminate information broadly and at scale. The consequence of this transition is the more frequent weaponization of information [6]. In order to effectively respond to the growing threat of fake news, robust and accurate fake news detection systems are essential.

Fake news detection typically falls into two categories [7]: linguistic cue and network analysis methods. Linguistic cue methods tackle fake news detection from a natural language processing perspective. Linguistic cue methods employ different tools and techniques to provide in-depth analysis of text in order to lead to classification. While network analysis methods rely on external sources of information in order to *fact check* target content. Ultimately, each category attempts to detect fake news from different perspectives and have varying amounts of success. Fake news detection is encumbered by many different challenges. Some of these include: countless features that are able to be extracted from a text corpus, adversarial effects on classification systems and wide scale propagation of information. This document presents

research conducted using a hybrid of both fake news detection methods while leveraging evolutionary computation to address the issue of fake news detection.

1.2 Dissertation Overview

This document includes five previous research efforts involving evolutionary computation in support of fake news detection. Chapter 4 presents a study of the impact of evolutionary based feature selection. This work was published in IEEE Symposium Series on Computational Intelligence [1]. Chapter 5 presents a novel method for selecting optimal subsets of adversarial examples for use in adversarial training that leads to an increased defensive posture. This work was published in IEEE Congress on Evolutionary Computation. Chapter 6 presents a study of the Twittersphere in order to identify information types and attribute what leads to information propagation across a social network. This work was published in IEEE SoutheastCon [8]. Chapter 7 presents an extension to our novel method, genetic adversarial training (GAT), that utilizes multi-objective optimization to incorporate multiple types of adversarial examples and substantially improve model robustness. This work was published in IEEE Symposium Series on Computational Intelligence [3]. Chapter 8 presents an extension to the work presented in Chapter 4, that incorporates multi-objective optimizations within the auto-labeler construction and does not constrain the diversity of the sampled Tweets. Finally, in Chapter 9 the conclusion is presented.

Chapter 2

Literature Review

2.1 Fake News Detection

Currently, we are witnessing an increasing amount of electronic and data dependence [9]. This dependency has created a shift in how news information is both created and consumed [10]. One consequence of this transition is the emergence of fake news [11]. Fake news can be described as fabricated information that mimics news media content in form but not in organizational process or intent [4]. There are many factors that have influenced the proliferation of fake news; however, social media is the leading contributor [12]. With social media being an open forum, it has created an ideal vector for the propagation fake news [13].

When approaching the issue of fake news detection there is much to consider. Shu et al. [11] conduct a comprehensive survey in the field of fake news detection. The authors present an in depth review on detecting fake news on social media and include fake news categorization on psychology and social theories. The authors outline an interesting delineation between the different types of fake news by grouping them into two categories based on their source: traditional media and social media. Analyzing these types of fake news independently, allow for further introspection into their characteristics and how to facilitate detection.

Reis et al. [14] explore fake news detection on social media. Specifically, the authors focus on evaluating how commonly used features in fake news detection perform and propose several additional features. Reis et al. measure the effectiveness of the features by evaluating their performance using several different machine learning classifiers. Their results reveal interesting observations on the usefulness and importance of specific features for fake news detection.

Ahmed et al. [15], examine fake news detection on traditional news media. They investigate and compare two different feature extraction techniques and six different machine classification techniques. The authors evaluate the performance of their extracted features on an aggregation of news articles from multiple sources. Ahmed et al. were able to achieve an accuracy of 92% using uni-gram features and a linear SVM classifier.

Wang et al. [16] present LIAR, a new dataset consisting of a 12.8k manually labeled short statements gathered from politicalfact.com. The authors introduce a novel hybrid convolutional neural network (CNN) and compare its performance against other models on the LIAR dataset. The author's hybrid CNN out performs all other models with a 7% increase in accuracy.

2.2 Adversarial Training & Adversarial Examples

Adversarial examples [17, 18] can be defined as malicious inputs that are designed to fool machine learning systems. These malicious inputs can be generated numerous ways, but they all share the intent to deceive the target system. The methods for generating adversarial examples can generally be grouped into 3 categories: threat modeling, perturbation, and benchmark [19]. Each category involves varying amounts of difficulty and computational costs in order to generate the adversarial examples. Threat modeling involves taking an introspective view of the machine learning systems in order to better discover points of weakness. Adversarial methods in this category include false positive and false negative attacks [20]. Perturbation is one of the most fundamental methods for generating adversarial examples. Perturbation involves adding a mild amount noise to a sample in order to influence a model's output [21]. While straightforward in application, perturbation attacks have been demonstrated to generate adversarial examples that are indistinguishable from clean examples but lead to misclassification [22]. Benchmarking involves evaluating model performance under varying conditions. This method involves analyzing how the model responds to different types of input and measure the changes in the response. This method requires the least amount of information on the target system to effectively generate adversarial examples.

The primary defensive strategy to adversarial examples is referred to as adversarial training (AT) [23, 24]. AT is the process of explicitly training a model on adversarial examples in

order to make it more robust against adversarial attacks. The overall goal of AT is to (1) mitigate the effects of adversarial attacks without (2) compromising the overall performance (i.e. classification accuracy).

Brown et al. [20] demonstrate two novel adversarial machine learning attacks, the Universal False Positive and the Universal False Negative attacks, to target a collection of machine learning classifiers. The research illustrates potential weaknesses inherent in the classifiers by exploiting ”holes” that allow generated feature vectors to pass through undetected (UFN) and by reducing confidence in overall performance (UFP). Both attacks occur at the feature vector level. Each attack operates by iteratively evaluating the performance of each model within the target collection. This accomplished by repeatedly training and generating predictions using different training and testing splits of the dataset. This process is repeated for each model over a predefined number of runs. This repetitive process enables constructing a more comprehensive picture of each model's performance. The generated predictions are then analyzed to determine what instances were misclassified universally between all the classifiers. Using k-means clustering, the UFPs and UFNs are clustered using a predefined number of clusters, and a quasi-probability distribution function (qPDF) is generated for each cluster. The qPDFs are sampled to generate adversarial UFP and UFN instances. The UFPs and UFNs are labeled accordingly and tested against the originally trained model to determine the misclassification rate.

Liu et al. [25] conduct a comprehensive survey on adversarial attacks and defenses with a focus on the perspective of machine learning interpretation. They present a plethora of different attacks and defenses and outline their respective nuances. Those most relevant to this paper include: Perturbation attack and model robustification. The authors define a perturbation attack as introducing specific noises to the input so that it is misclassified by the model. The solution presented for this attack is referred to as module robustification. The authors describe model robustification as a refinement and process for preparing a model against potential threats. This defensive posturing is accomplished by changing either the training objective or modifying the model structure. Changing the training objective includes methods such as adversarial training while modifying the structure can include model distillation.

Goodfellow et al. [26] present an extensive study into the inner working of adversarial examples and the concept of adversarial training in relation to deep learning. The authors suggest that there is no need to consider the non-linearity of neural networks and that adversarial examples can be created by exploiting the linear behavior in high dimensional spaces. Additionally, the authors introduce a novel method to generate adversarial examples, Fast Gradient Sign. Fast Gradient Sign can efficiently generate reliable adversarial examples that cause a variety of models to misclassify their input. Goodfellow et al. utilize examples generated using the Fast Gradient Sign method to adversarially train a model and demonstrate that adversarial training can be used as a regularization technique.

Wang et al. [27] presented an novel study where they discuss different nuances surrounding AT. Specifically, the authors focus on the performance of AT when the class distribution of either the initial training data or adversarial examples is imbalanced. The authors observe that when AT is conducted, irregardless of the class imbalance being within the initial training set or adversarial examples, the adversarially trained model presents a degradation in performance on the under-represented classes when compared to the naturally trained model. The authors conclude that AT is more sensitive to an imbalanced data distribution than natural training. The authors suggest a novel method, Separable Reweighted Adversarial Training (SRAT), to enable a reweighting strategy when conducting AT with an imbalanced class distribution present.

Zhang et al. [28] conducted an expansive study focusing on the trade-off between robustness and accuracy when crafting defenses against adversarial examples. The authors decompose the prediction error for adversarial examples as the sum of the natural classification error and boundary error in an effort to theorize a differentiable upper bound. Zhang et al. use a theoretical analysis to craft a new defense method, TRADES, that acts to optimize the regularized surrogate loss. By optimizing the regularized surrogate loss, the decision boundary is pushed further away from the data and thereby improves adversarial robustness.

Terzi et al. [29] investigated the application of a directional based adversarial training method. The authors propose a novel method, called Wasserstein Projected Gradient Descent (WPGD), to facilitate mapping out the output space of a network to determine the directions

that robustness is required. By focusing robustification efforts only in the needed directions of the network, a directional trade-off between accuracy and robustness can be derived.

2.3 Genetic & Evolutionary Feature Selection

Genetic and evolutionary feature selection (GEFeS) [30] [31] [32] uses a steady state genetic algorithm or other evolutionary computation technique to evolve a population of feature masks. GEFeS operates as follows. First, a population of feature masks are randomly generated where each mask is a binary string. Each feature mask is of length N, where N represents the size of the feature set. For each feature mask, a 1 represents the value will be used while a 0 means the feature will be discarded. These feature masks will be referred to as candidate solutions (CS). Next, each CS is assigned a fitness value as a measure of quality. The evaluation of a CS to determine the quality will be referred to as a function evaluation (FE). Two parents are then selected from the population based on a selection strategy [33]. The two parents are used to create a single offspring through a crossover [34] and mutation [35] operation. Next, the created offspring are assigned a measure of fitness and inserted back into the population replacing the CS with the worst fitness. This evolutionary process continues until reaching the predefined stopping condition. Following GEFeS completion, a population of evolved CSs will remain. Fig. 2.1 provides a psuedocode version of an SSGA algorithm.

2.4 Non-dominated Sorting Generic Algorithm II

Non-dominated Sorting Genetic Algorithm II [36] (NSGA-II) is a multi-objective evolutionary algorithm that operates by approximating and creating pareto-optimal fronts that construct trade-offs between the different objectives in the problem. A front represents a collection of individuals that are only dominated by the proceeding fronts. For example, the first front represents all of the non-dominated individuals in the population. Meaning that all other individuals in the population are no better in any objective and are worse at least one objective when compared to individuals in the first front. This relationship is continued through the following fronts with respect to the preceding front.

```
Procedure SSGA{
   t = 0;
   Initialize P(t);
   Evaluate P(t);
   While (Not Done)
   {
        Parents(t) = Select_Parents(P(t));
        Offspring(t) = Procreate(Parents(t));
        Evaluate(Offspring(t));
        P(t+1)= Replace(Worst(P(t)),Offspring(t));
        t = t + 1;
    }
}
```

Figure 2.1: SSGA Pseudocode

Chapter 3

Datasets

3.1 BuzzFace

BuzzFace is a fake news detection dataset [14,37] composed of news stories posted to Facebook during September 2016 that are related to the 2016 United States presidential election. These articles were labeled by journalists and expanded with relating metadata such as comments, shares, and reactions. BuzzFace includes 2282 articles and each article is represented using 179 features. The features can broadly be grouped into 3 different categories: textual features, news source features, and environmental features. The textual features are extracted from the article text using methods including Linguistic Inquiry Word Count [38], bag-of-words, lexical and semantic analysis [39,40]. The news source features relate to information about the news article publisher. These features are generated by inspecting information surrounding the publisher such as political bias, credibility, and domain location. The environmental features consist of various metrics such as user statistics and engagement patterns for each article in relation to social media. These features include comment count, share count, like count, and various temporal patterns.

Additionally, each article is assigned a label as a measure of factual content. These labels include: non factual content, mostly false, mostly true, and a mixture of true and false. The articles labeled as “non factual content” are removed and those labeled as ”mostly false” and ”mixture of true and false” are combined into a single class (1.0). The remaining articles are labeled as true news (-1.0). In Fig. 5.1, the class distribution for the dataset is shown.

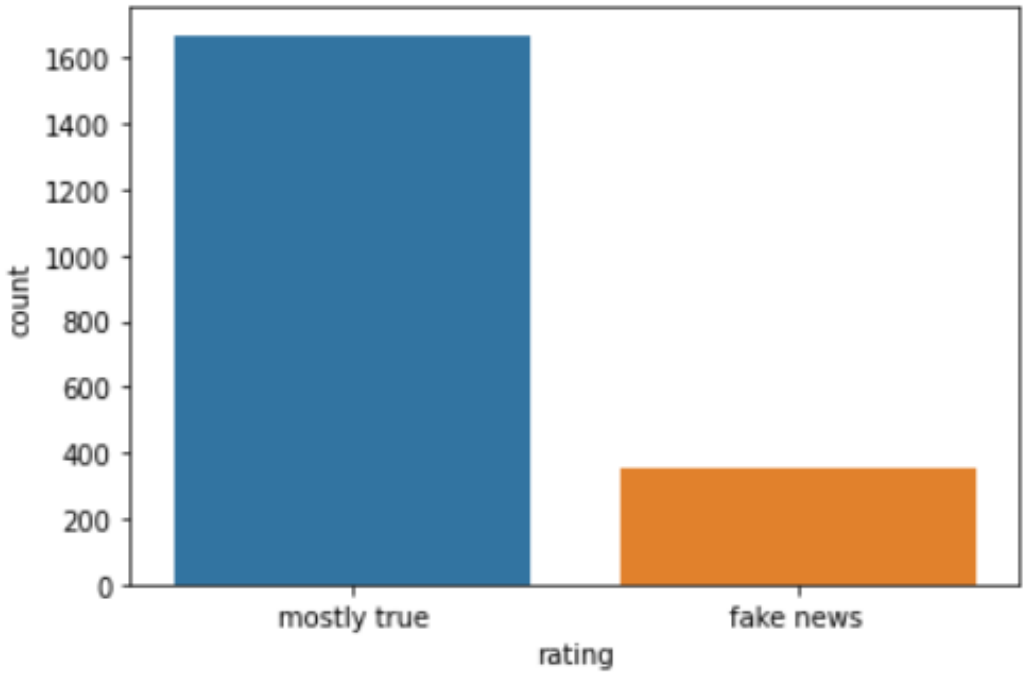

Figure 3.1: Class Distribution

3.2 Twitter Dataset

The Twitter Dataset consists of COVID-19 related Tweets posted to Twitter during the month of March 2020. These tweets are collected using their Tweet ID. The Tweet IDs originate from a GitHub repository [41, 42]. This repository maintains an active and ongoing collection of Tweet IDs associated with COVID-19. The repository uses Twitter's Search API to gather historical Tweets from the proceeding 7 days, leading back to the start date of 21 January 2020. Additionally, Twitter's Streaming API is also leveraged to collect Tweet IDs from a subset of accounts and Tweets that mention specific keywords. Following data collection, the Twitter dataset consists of approximately 37M Tweets.

3.3 FX & Pert Adversarial

The FX & Pert Adversarial datasets are based on a modified version of the BuzzFace dataset [14, 37]. The original Buzzface dataset is balanced by randomly removing instances of the true news until the classes are equal. The final distribution for the classes is shown in Fig. 5.1.

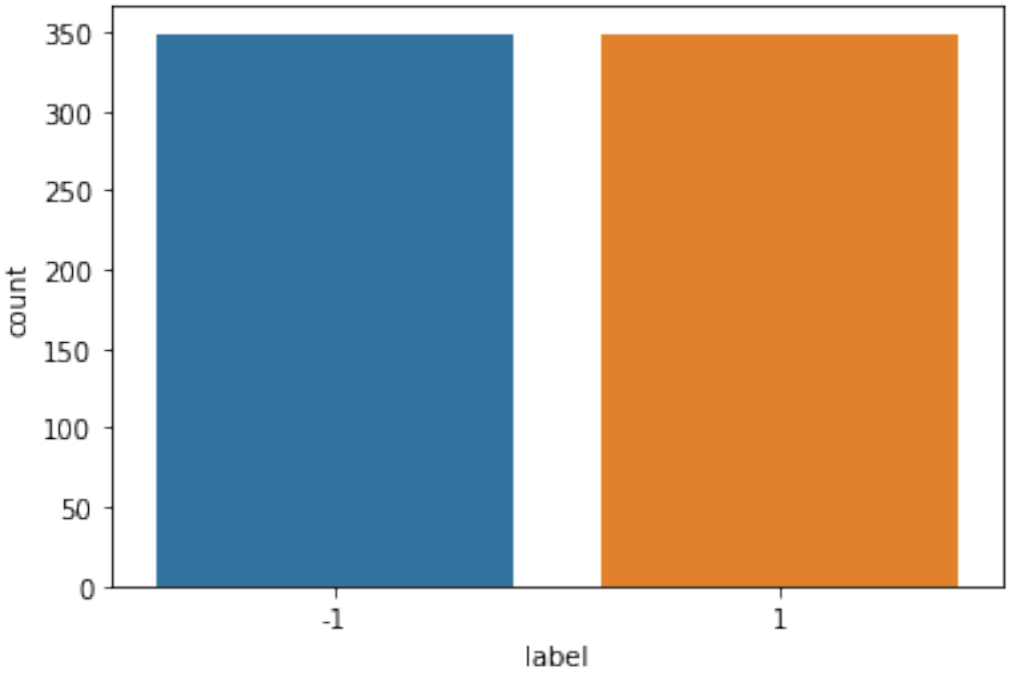

Figure 3.2: Class Distribution

The FX & Pert Adversarial datasets are then created by generating adversarial examples using the FX and Perturbation attack methods. These adversarial datasets will be referred to as $\text{Adversarial}_{\text{FX}}$ and $\text{Adversarial}_{\text{Pert}}$ respectively. Both $\text{Adversarial}_{\text{FX}}$ and $\text{Adversarial}_{\text{Pert}}$ contain 400 adversarial examples. Each classifier (RF, XGB, kNN, SVM) has it own corresponding set of adversarial examples. Each adversarial dataset is split into a training and test set where the distribution is 40/60.

Chapter 4

A Study of the Impact of Evolutionary-Based Feature Selection for Fake News Detection

4.1 Introduction

Fake news is becoming an increasingly invasive problem within our society. As our society becomes more ingrained in technology, news has become more susceptible to technological predation. Fake news is a challenging problem due to two inherent properties [43]. First, it is difficult to adequately quantify a flexible definition for classifying fake news [4]. When dealing with human expression, it is often influenced by a variety of factors such as biases, beliefs, and socioeconomic backgrounds [11]. The manifestation of these factors culminate into how news information is perceived [44]. Thereby, any news information could be described as fake news depending on the objectivity. Research has been conducted exploring the different conditions that affect the consumption of news and social media in an effort to craft a succinct definition for classifying fake news [45].

The second issue presents itself in the intent of fake news [4, 46]. For example, The Onion, is considered the most popular satirical news site on the internet [47]. However, when The Onion articles are shared without context, they are often difficult to discern from factual news [48]. An example of this occurrence took place on September 2011 when United States Capitol Police investigated a series of tweets coming from The Onion's Twitter account. The tweets described a hostage situation inside the Capitol Building where members of Congress held children captive [49]. These tweets circulated widely and garnered national attention. Thus, with the results from even satirical fake news, it is of grave concern when considering the intent behind fake news.

Table 4.1: Baseline Comparison

	Reis et al. [23] (5 Fold 10 Runs)		5 Fold 10 Runs		Hold Out - 30 Runs	
Classifier	AUC	F1	AUC	F1	AUC	F1
KNN	0.800 ± 0.009	0.750 ± 0.008	0.849 ± 0.017	0.904 ± 0.013	0.840 ± 0.022	0.904 ± 0.008
NB	0.720 ± 0.009	0.750 ± 0.001	0.763 ± 0.024	0.881 ± 0.017	0.763 ± 0.026	0.871 ± 0.027
RF	0.850 ± 0.007	0.810 ± 0.008	0.871 ± 0.023	0.907 ± 0.018	0.874 ± 0.015	0.907 ± 0.011
SVM	0.790 ± 0.030	0.760 ± 0.019	0.705 ± 0.049	0.906 ± 0.014	0.695 ± 0.060	0.903 ± 0.011
XGB	0.860 ± 0.006	0.810 ± 0.011	0.869 ± 0.017	0.902 ± 0.017	0.871 ± 0.018	0.904 ± 0.011

In this chapter, we focus on building upon and improving a surveyed approach [14] to fake news detection. We attempt to improve upon Reis et al. results through the use of evolutionary-based feature selection [50]. Through evolutionary-based feature selection, we can optimize classifier performance while reducing the feature space.

4.2 Experiment

In this experiment we compare the baseline performances of 5 classifiers: k-Nearest Neighbors (KNN), Naive Bayes (NB), Random Forests (RF), Support Vector Machine with RBF kernel (SVM), and XGBoost (XGB), with the performances of six GEFeS$_{\text{ML}}$ hybrids where $ML \in \{KNN, NB, RF, SVM, XGB\}$. The dataset used for this work is BuzzFace [51].

4.2.1 Baseline

To effectively evaluate the performance of GEFeS, a Baseline was created. The Baseline was created to be consistent with the results outlined by Reis et al. [14]. This is accomplished by first processing the data in a method consistent with the previous authors. Next, the 5 classifiers are implemented and fine-tuned until the results are similar or surpass those outlined in the previous authors paper [14]. However, it is worth noting that the methods used to split the data, train the classifiers, and the number of total runs are slightly different. Reis at al. use 5-fold cross validation for 10 runs. While, the results generated in this paper will use the hold-out method where the dataset is split to 80% training and 20% testing. The hold-out method is applied for 30 runs in all the executions. A comparison of the results with respect to the Baseline are shown in Table 4.1.

Table 4.2: GEFeS Control Parameters

Representation	Binary string
Recombination	Uniform crossover
Recombination Probability	100%
Mutation	Inverted with independent probability p_m
Mutation Probability P_m	1%
Parent Selection	Random 2
Survival Selection	Replace worst
Population Size	20
Number of Offspring	1
Initialization	Random
Termination Condition	100, 1K, 2K, 4K, 8K, 16K function evaluations

Table 4.3: Experiment Results - Accuracy

Baseline/FEs	$GEFeS_{KNN}$	$GEFeS_{NB}$	$GEFeS_{RF}$	$GEFeS_{SVM}$	$GEFeS_{XGB}$
Baseline	0.881 (0.831 $\pm$ 0.018)	0.831 (0.782 $\pm$ 0.024)	0.858 (0.833 $\pm$ 0.012)	0.876 (0.828 $\pm$ 0.016)	0.876 (0.838 $\pm$ 0.018)
100	0.876 (0.852 $\pm$ 0.010)	0.854 (0.819 $\pm$ 0.017)	0.886 (0.854 $\pm$ 0.016)	0.869 (0.830 $\pm$ 0.016)	0.884 (0.859 $\pm$ 0.011)
1000	0.881 (0.857 $\pm$ 0.012)	0.851 (0.825 $\pm$ 0.014)	0.886 (0.863 $\pm$ 0.014)	0.859 (0.836 $\pm$ 0.014)	0.894 (0.873 $\pm$ 0.012)
2000	0.884 (0.864 $\pm$ 0.012)	0.851 (0.820 $\pm$ 0.014)	0.889 (0.863 $\pm$ 0.015)	0.856 (0.831 $\pm$ 0.013)	0.894 (0.870 $\pm$ 0.010)
4000	0.901 (0.864 $\pm$ 0.014)	0.869 (0.825 $\pm$ 0.020)	0.896 (0.864 $\pm$ 0.014)	0.861 (0.828 $\pm$ 0.017)	0.913 (0.876 $\pm$ 0.014)
8000	0.889 (0.864 $\pm$ 0.014)	0.861 (0.824 $\pm$ 0.019)	0.899 (0.866 $\pm$ 0.014)	0.861 (0.828 $\pm$ 0.016)	0.906 (0.875 $\pm$ 0.013)
16000	0.891 (0.862 $\pm$ 0.015)	0.864 (0.830 $\pm$ 0.018)	0.879 (0.866 $\pm$ 0.011)	0.869 (0.834 $\pm$ 0.016)	0.901 (0.875 $\pm$ 0.013)

Table 4.4: Experiment Results - Features

Baseline/FEs	$GEFeS_{KNN}$	$GEFeS_{NB}$	$GEFeS_{RF}$	$GEFeS_{SVM}$	$GEFeS_{XGB}$
Baseline	179 (179.0 $\pm$ 0.00)	179 (179.0 $\pm$ 0.00)	179 (179.0 $\pm$ 0.00)	179 (179.0 $\pm$ 0.00)	179 (179.0 $\pm$ 0.00)
100	75 (89.0 $\pm$ 6.33)	72 (88.9 $\pm$ 7.21)	73 (89.6 $\pm$ 8.51)	68 (87.8 $\pm$ 7.88)	76 (89.1 $\pm$ 5.75)
1000	73 (90.8 $\pm$ 6.94)	71 (86.8 $\pm$ 8.37)	73 (88.3 $\pm$ 5.90)	78 (88.2 $\pm$ 4.58)	71 (89.3 $\pm$ 7.51)
2000	76 (87.9 $\pm$ 5.56)	71 (86.6 $\pm$ 7.67)	75 (91.1 $\pm$ 6.16)	74 (89.8 $\pm$ 6.55)	73 (90.3 $\pm$ 7.46)
4000	76 (88.6 $\pm$ 6.16)	70 (86.9 $\pm$ 7.26)	75 (87.8 $\pm$ 7.14)	68 (87.7 $\pm$ 7.95)	75 (89.0 $\pm$ 5.61)
8000	74 (87.5 $\pm$ 6.25)	71 (83.8 $\pm$ 7.06)	75 (89.4 $\pm$ 7.17)	67 (88.5 $\pm$ 7.26)	71 (89.8 $\pm$ 6.65)
16000	72 (90.5 $\pm$ 8.09)	73 (84.9 $\pm$ 6.42)	75 (88.5 $\pm$ 6.00)	73 (86.2 $\pm$ 6.92)	71 (91.1 $\pm$ 6.97)

4.3 Results

The results presented in this section were generated as follows. First, a population of 20 CSs are generated. These solutions are randomly generated binary strings of length N, where N, represents the size of the feature set. Next, an evaluation function is used to determine the quality of each solution. Each hybrid uses a different classifier in the evaluation function. The measurement of fitness used is how well the population's feature mask performs in correctly classifying fake news. Next, two parents are chosen from the population using random selection. One child is then created from the two parents using uniform crossover [52]. The newly created child is mutated by stochastically flipping the child's genes. This child is inserted into the population replacing the worst performing solution. This cycle is repeated until the target function evaluation threshold is met. In Table 4.2 the GEFeS control parameters are presented.

The performance of the GEFeS hybrids in comparison to the Baseline is presented in Table 4.3 and Table 4.4. The GEFeS_{ML} hybrids being compared will be denoted as $\text{GEFeS}_{\text{ML-n}}$ where $ML \in \{KNN, NB, RF, SVM, XGB\}$ and $n \in \{100, 1000, 2000, 4000, 8000, 16000\}$. Therefore, ML is taken from the set of classifiers and n represents the number of function evaluations (FEs) used by a particular hybrid. In Table 4.3, the first column presents the baseline and the number of FEs. The following columns show the accuracies for the Baseline and each hybrid with respect to the number of FEs. In these columns, the first value denotes the highest accuracy achieved. The values in parenthesis are the accuracy, averaged over 30 runs, and the standard deviation. Table 4.4 shows the number of features from the Baseline and the evolved solutions following each execution. The first column represents the Baseline and the number of FEs . The following columns show the number of features. In these columns, the first value denotes the lowest number of features used among the evolved solutions. The values in parenthesis present the number of features in the evolved solutions, averaged over 30 runs, and the standard deviation.

From Table 4.3, one can note that within the Baseline performances, the highest average accuracy achieved is 83.8% in $\text{Baseline}_{\text{XGB}}$, while the lowest average accuracy is 78.2% in $\text{Baseline}_{\text{NB}}$. Apart from $\text{Baseline}_{\text{NB}}$, the average accuracy among the Baseline performances

remain consistent around 83%. With the initial GEFeS hybrids at 100 FEs, there is an increase in the average accuracy across all of the hybrids. The most prominent increase in average accuracy is $\text{GEFeS}_{\text{NB-100}}$ with an increase of 3.7%. The highest average accuracy achieved is $\text{GEFeS}_{\text{XGB-100}}$ at 85.9%. The average accuracy continues to generally increase with the GEFeS hybrids as the number of FEs grow. This upward trend in accuracy can be attributed to better coverage of the search space. With the current feature set consisting of 179 features, there exists 2^{179} potential solutions. By increasing the number of FEs, the GEFeS hybrids can further iterate through the search space searching for solutions of higher quality.

In Table 4.4 the number of features from each of the GEFeS hybrids are presented. The Baseline does not incorporate feature selection and thereby uses the entire feature set. However, the following rows outline the number of features from the evolved solutions from each GEFeS hybrid. With GEFeS, there is dramatic decrease in the number of features used. One point that is of significant interest is that the average number of features for each hybrid remain nearly consistent throughout the increasing number of FEs. From this, one can conclude that there is a significant amount of optimization being achieved with only 100 FEs.

In summary, one can see that the GEFeS hybrids continuously outperform the Baseline performances in terms of accuracy. Moreover, GEFeS also dramatically decreases the number of features used by nearly 50%. Thus, GEFeS not only optimizes classifier performance, but it also removes the non-essential features. The essential features can be visualized by inspecting the Feature Consistency. Feature Consistency can be defined as the representation of how reliable a feature is. This is calculated by counting the number of instances a feature occurs in the evolved solutions from each GEFeS hybrid and dividing that by the number of runs. Fig. 4.1 and Table 4.5 show the feature consistency results for $\text{GEFeS}_{\text{XGB-8000}}$. $\text{GEFeS}_{\text{XGB-8000}}$ is presented due to it resulting in one of the lowest number of features while achieving one of the highest accuracies. In Fig. 4.1, the x-axis is the numerical index of the feature within the dataset. The y-axis is the consistency the feature appears in an evolved solution over the 30 runs. In Table 4.5, the columns represent the index of the feature in the dataset, the name of the feature, and the consistency the feature appears in an evolved solution.

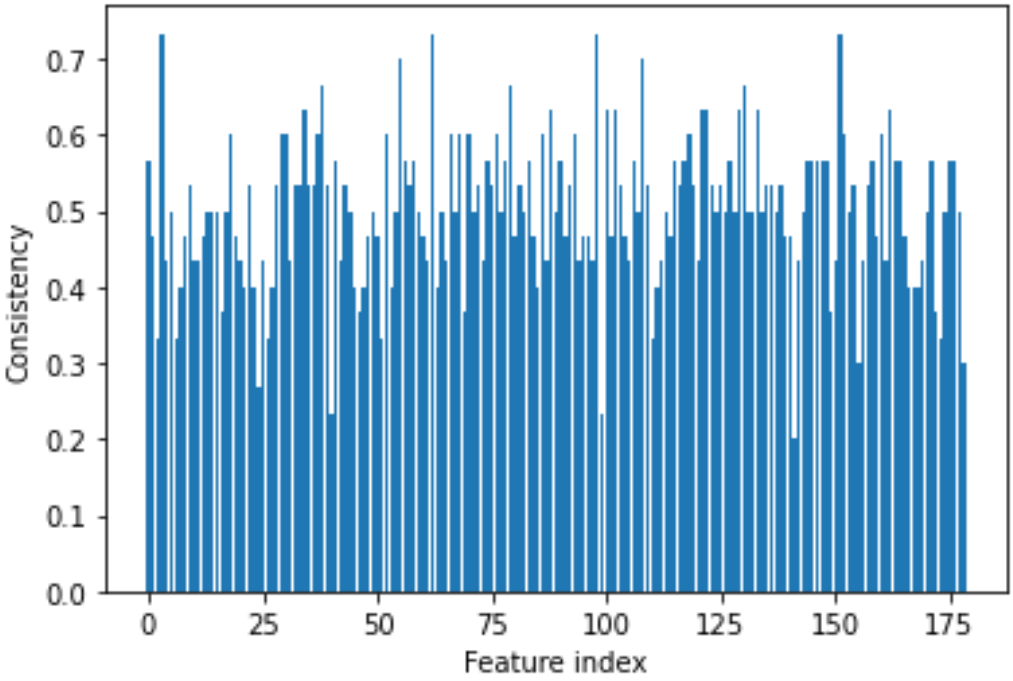

Figure 4.1: $\text{GEFeS}_{\text{XBG-8000}}$ Feature Consistency

Table 4.5: $\text{GEFeS}_{\text{XGB-8000}}$ Top 10 Consistent Features

Feature Index	Feature Name	Consistency
3	share_count	73%
62	affiliation	73%
98	toxicity	73%
151	political_bias	73%
55	feel	70%
108	readability_flesch_reading_ease_index	70%
38	anger	67%
79	death	67%
130	word_usage_nominalization	67%
34	affect	63%

When considering that the types of features used in the BuzzFace dataset are consistent with other fake news detection implementations, examining the feature consistency can provide a deeper understanding of which features are more influential in determining whether an article is classified as fake news. In Table 4.5, the top 10 features with the highest feature consistency from $\text{GEFeS}_{\text{XGB-8000}}$ are presented. These feature resided in at least 63% of the evolved solutions over 30 runs, meaning that they consistently contribute to correct classification. The features were extracted via a variety of sources including: Linguistic Inquiry and Word Count [38], PerspectiveAPI [39], and SentiStrength [40].

Table 4.6 show the results of dividing the performances into equivalence classes. Each equivalence class represents the grouping of statically similar results. The equivalence classes are created by conducting a series of ANOVA tests independently on the results of each of the algorithms. The results of an ANOVA test determine if there exist statistical significance among the results. By applying the ANOVA test iteratively, a ranking between the results is established. In Table 4.6, the equivalences classes are presented.

By examining the equivalence classes, two observations can be made. First, the equivalence classes enable direct comparison between the performances of the algorithms. In all cases, the Baseline performances with respect to a specific classifier exist in a lower class than the respective GEFeS hybrid performances. For example, when examining the equivalence class ranking of $\text{Baseline}_{\text{NB}}$, one can note that it resides in a equivalence class below all other GEFeS_{NB} performances. This illustrates that the Baseline performances perform statically worse when compared to the GEFeS hybrids. Second, by using the established equivalence classes, a trade-off between performance and computational costs can be determined. When examining the first equivalence class, one can note that it consists entirely of $\text{GEFeS}_{\text{XGB}}$ performances. However, the number of FEs vary significantly. Indicating that, while increasing the number of FEs tend to lead to a higher average accuracy, statically similar results can be achieved with a lower number of FEs. For example, when comparing $\text{GEFeS}_{\text{XGB-16000}}$ and $\text{GEFeS}_{\text{XGB-1000}}$, they will both generate statistically similar results. However, they dramatically differ in execution time. By using $\text{GEFeS}_{\text{XGB-1000}}$ one can achieve similar results at a fraction of the execution time.

Table 4.6: Equivalence Classes

Class	Group
1	$\text{GEFeS}_{\text{XGB-16000}}$, $\text{GEFeS}_{\text{XGB-1000}}$, $\text{GEFeS}_{\text{XGB-2000}}$, $\text{GEFeS}_{\text{XGB-4000}}$, $\text{GEFeS}_{\text{XGB-8000}}$
2	$\text{GEFeS}_{\text{KNN-1000}}$, $\text{GEFeS}_{\text{XGB-100}}$, $\text{GEFeS}_{\text{KNN-16000}}$, $\text{GEFeS}_{\text{RF-1000}}$, $\text{GEFeS}_{\text{RF-2000}}$, $\text{GEFeS}_{\text{RF-4000}}$, $\text{GEFeS}_{\text{KNN-2000}}$, $\text{GEFeS}_{\text{KNN-8000}}$, $\text{GEFeS}_{\text{KNN-4000}}$, $\text{GEFeS}_{\text{RF-16000}}$, $\text{GEFeS}_{\text{RF-8000}}$
3	$\text{GEFeS}_{\text{KNN-100}}$, $\text{GEFeS}_{\text{RF-100}}$
4	$\text{GEFeS}_{\text{NB-4000}}$, $\text{Baseline}_{\text{KNN}}$, $\text{GEFeS}_{\text{SVM-8000}}$, $\text{GEFeS}_{\text{SVM-4000}}$, $\text{GEFeS}_{\text{SVM-100}}$, $\text{GEFeS}_{\text{NB-16000}}$, $\text{GEFeS}_{\text{SVM-2000}}$, $\text{GEFeS}_{\text{SVM-16000}}$, $\text{GEFeS}_{\text{SVM-1000}}$, $\text{Baseline}_{\text{XGB}}$, $\text{Baseline}_{\text{RF}}$
5	$\text{GEFeS}_{\text{NB-100}}$, $\text{GEFeS}_{\text{NB-2000}}$, $\text{Baseline}_{\text{SVM}}$, $\text{GEFeS}_{\text{NB-8000}}$, $\text{GEFeS}_{\text{NB-1000}}$
6	$\text{Baseline}_{\text{NB}}$

Table 4.7: First Equivalence Class

Execution	AUC	F1
$\text{GEFeS}_{\text{XGB-1000}}$	0.914 (0.872 ± 0.016)	0.922 (0.905 ± 0.009)
$\text{GEFeS}_{\text{XGB-2000}}$	0.915 (0.876 ± 0.015)	0.933 (0.908 ± 0.011)
$\text{GEFeS}_{\text{XGB-4000}}$	0.914 (0.872 ± 0.016)	0.926 (0.907 ± 0.009)
$\text{GEFeS}_{\text{XGB-8000}}$	0.905 (0.872 ± 0.013)	0.921 (0.906 ± 0.008)
$\text{GEFeS}_{\text{XGB-16000}}$	0.902 (0.872 ± 0.015)	0.933 (0.906 ± 0.011)

In Table 4.7 the performance of the first equivalence class is presented. The first equivalence class consists of statistically significant and top preforming hybrids. The first column represents the hybrid and the following columns are the accuracy under the curve and F1 score. The results were generated as follows. First, the features from the evolved solution that achieved the highest accuracy are selected. Next, the evolved features were evaluated over 30 runs and their results averaged. One can see that in all the performances of the hybrids, the Baseline is out performed.

4.4 Summary

In this chapter, we explored the use of evolutionary-based feature selection on a fake news detection implementation involving several different machine learning classifiers. Our results suggest that evolutionary-based feature selection increases fake news detection accuracy while

dramatically reducing the number of features needed. Additionally, evolutionary-based feature selection out performs the baseline in all instances of execution.

Chapter 5

Mitigating Attacks on Fake News Detection Systems using Genetic-Based Adversarial Training

5.1 Introduction

The study of adversarial effects on AI systems is not a new concept, but much of the research has been devoted to deep learning. In this chapter we explore the effects of adversarial examples on 4 machine learning classifiers and measure the effectiveness of adversarial training. Additionally, we present a novel method for selecting adversarial training examples that lead to a more robust machine learning system. Our results suggest that adversarial examples can significantly hinder the classification performance and that adversarial training (AT) is an effective defensive counter-measure.

In this chapter, we study the effects of AT on a number of machine learning classifiers and measure the effectiveness of adversarial training. We accomplish this by generating adversarial examples targeting each machine learning system and evaluate the performance of adversarial training. Additionally, we propose a novel method for selecting sets of adversarial training examples that promote a more robust machine learning system.

5.2 Adversarial Examples

In order to preform adversarial training, adversarial examples must be acquired. To better emulate an adversarial perspective when generating these examples, two grey box [53] attacks are performed. The first attack is a targeted false positive and false negative attack and will be defined as an FX attack [54]. The second attack is a Perturbation attack [22, 25, 55]. Both attacks

are considered grey box because access to each trained model is available and the dataset is known. However, the specific training set is unknown. Without the knowledge of the training set, the search space for discovering examples that cause adversarial effects increases dramatically. These two attacks are used to target a collection of machine learning classifiers independently. This collection includes: Random Forests (RF) [56], XGBoost (XGB) [57], k-Nearest Neighbors (KNN) [58], and Support Vector Machine with RBF kernel (SVM) [59]. Each classifier is used in the aforementioned attacks to create the adversarial examples. For each attack, 200 false positives (FP) and 200 false negatives (FN) examples are created. Thus, a total of 800 adversarial examples are created targeting each classifier.

5.2.1 FX Attack

The FX attack follows the methodology used by Brown et. al [20]. However, instead of analyzing a collection of machine learning classifiers together, the FX attack examines each classifier independently. The intent of this attack is to generate adversarial examples that will either be classified as a FP or a FN. The FX Attack operates as follows:

First, the target model is used to generate predictions on the entire dataset. The predictions are analyzed to identify instances that are classified incorrectly. Next, the misclassified instances are separated by FP and FN. The FP and FN instances are then clustered and a probability density function (PDF) is created for each cluster. Each cluster is randomly sampled to generate the adversarial examples.

5.2.2 Perturbation Attack

The Perturbation attack operates by perturbing a target instance in the dataset until the label is flipped. This is accomplished by using a steady state genetic algorithm (SSGA) [60]. A total of 40 true positive and 40 true negative instances are randomly selected. From each selected instance 5 adversarial examples will be created. The perturbation attack operates as follows.

First, an instance is randomly selected from the dataset. This instance is used to create the initial population. The population is created by generating slight variations of the original selected instance. This is accomplished by introducing gaussian noise to slightly perturb the

features of the target instance. Thereby, a population is created consisting of individuals that are slight variations from the selected instance. Next, the population is evaluated using the target model from the collection {RF, XGB, KNN, SVM}. The target model is used to generate probability estimates (PE) for each individual. The probability estimates represent the probabilities for the individual to belong to either class one (-1.0) or class two (1.0). The fitness for each individual is calculated using the following expression:

$$fitness = -PE_0 + PE_1$$

This fitness measure serves as the measurement of quality for each individual within the objective function. Depending on the starting label, the objective function will either minimize or maximize the fitness. If the starting label is (1.0), the fitness will be minimized to generate a FN adversarial example (-1.0). If the starting label is (-1.0), the fitness will be maximized to create a FP adversarial example (1.0). Following the evaluation of the population, 2 parents are selected using binary tournament selection. Next, a single offspring is created using flat crossover with the selected parents. The offspring is then mutated using gaussian mutation. Finally, the offspring is assigned a fitness value and replaces the worst performing individual in the population. This process continues until reaching the stopping condition.

As previously discussed, the methods presented to generate adversarial examples operate at the feature level and consist of feature vectors. Because of this, there potentially exists a semantic gap between the generated feature vectors and the reconstructed text corpus. To address this issue, successful adversarial examples can be reverse engineered to generate and validate the corresponding text.

5.3 Experiments

5.3.1 Dataset

The experiments presented in this section are based on a modified version of the BuzzFace dataset [14, 37]. The original Buzzface dataset is balanced by randomly removing instances of

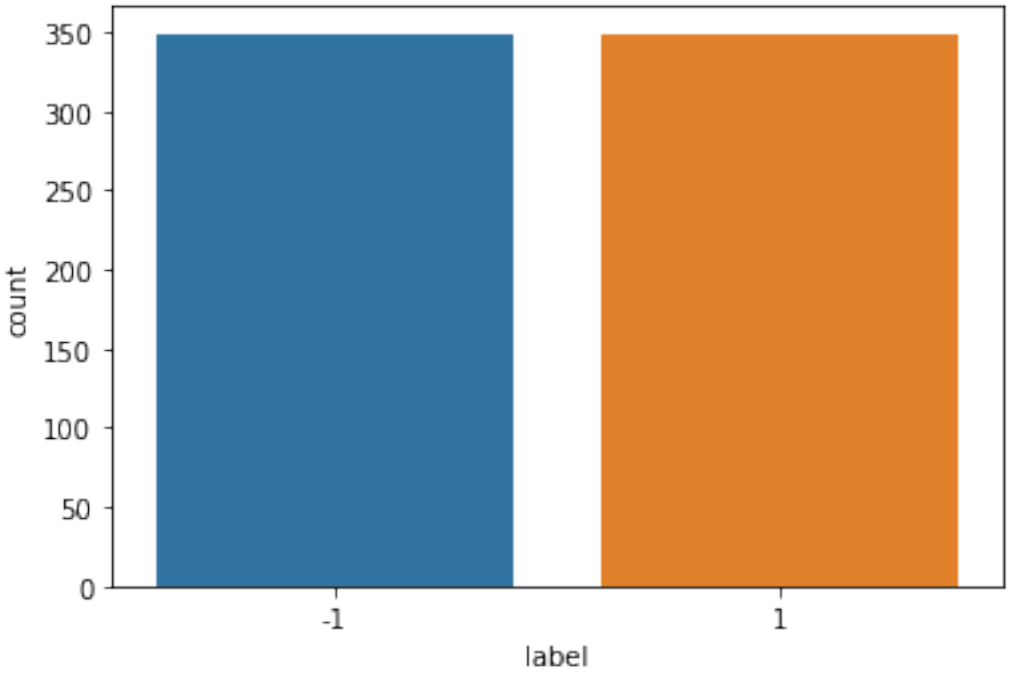

Figure 5.1: Class Distribution

the true news until the classes are equal. The final distribution for the classes is shown in Fig. 5.1.

The adversarial datasets are created by generating adversarial examples using the FX and Perturbation attack methods. These adversarial datasets will be referred to as $\text{Adversarial}_{\text{FX}}$ and $\text{Adversarial}_{\text{Pert}}$ respectively. Both $\text{Adversarial}_{\text{FX}}$ and $\text{Adversarial}_{\text{Pert}}$ contain 400 adversarial examples. Each classifier (RF, XGB, kNN, SVM) has it own corresponding set of adversarial examples. For the following experiments, each adversarial dataset is split into a training and test set where the distribution is 40/60. The adversarial datasets are split to enable objective evaluations of performance in the following experiments.

5.3.2 Baseline

There are 4 distinct models used in the following experiments. Each model was trained using an 80/20 split and the corresponding data retained for further evaluation. This retained data will be used in conjunction with each classifiers' generated adversarial datasets. This data will be referred to as $\text{Baseline}_{\text{train}}$ and $\text{Baseline}_{\text{test}}$.

5.3.3 Experiment I: Adversarial Example Performance

The purpose of the first experiment is to evaluate how well each of the classifiers perform when generating predictions on adversarial examples. Each classifier is used to generate predictions on the $\text{Adversarial}_{\text{FX-test}}$ and $\text{Adversarial}_{\text{Pert-test}}$ datasets in order to effectively measure the adversarial effects of the examples. These results will serve as a foundation to compare the results of the following experiments.

5.3.4 Experiment II: Adversarial Training

The primary focus of the second experiment is to measure the effectiveness of adversarial training. This is accomplished by augmenting the $\text{Baseline}_{\text{train}}$ with 30 random examples and evaluating the performance when generating predictions on the adversarial test sets. The adversarial training is constrained to 30 examples due to the size of the $\text{Baseline}_{\text{train}}$ set. Additionally, 30 examples represent approximately 5% of the $\text{Baseline}_{\text{train}}$ set. The secondary focus is to examine the effect that adversarial training has on the baseline performance when generating predictions on the $\text{Baseline}_{\text{test}}$ set.

5.3.5 Experiment III: Genetic Adversarial Training

When considering that there are countless ways to generate adversarial examples, the question of quality should be established. The purpose of this experiment is to discover a metric or methodology to evaluate the adversarial examples. The method presented, genetic adversarial training (GAT), presents a methodology to measure the quality of the adversarial examples and discover subsets of different adversarial examples that result in an increased defensive posture.

GAT uses a SSGA to evolve a population of training masks. The training masks consist of randomly generated binary strings of length N, where N, represents the size of the adversarial training set. Within the training masks, a 1 means the adversarial example will be used in the training while a 0 means it will be discarded. Thus, each training mask represents a subset of adversarial examples that will be used in the adversarial training. Each classifier uses its

Table 5.1: GAT Control Parameters

Representation	Binary string
Recombination	Uniform crossover
Recombination Probability	100%
Mutation	Invert with constrained probability Pm
Mutation Probability Pm	5%
Parent Selection	Binary tournament
Survival Selection	Replace worst
Populations size	100
Number of Offspring	1
Initialization	Random
Termination Condition	4K function evaluations

respective adversarial examples, $\text{Adversarial}_{\text{FX}}$ and $\text{Adversarial}_{\text{Pert}}$, and the $\text{Baseline}_{\text{train}}$ set in the GAT procedure. The procedure of GAT is as follows:

Step 1: Generate a random population of training masks (individuals).

Step 2: Evaluate each individual in the population by augmenting the individual's selected adversarial examples and the target classifiers $\text{Baseline}_{\text{train}}$ set to create the adversarial training set. The classifier is then trained with this adversarial training set and is used to generate predictions on the adversarial test set. The resulting accuracy is used as the measure of fitness.

Step 3: Select two parents using binary tournament selection and create an offspring using uniform crossover.

Step 4: Evaluate the created offspring using the procedure outlined in Step 2.

Step 5: Replace the worst performing individual in the population with the offspring.

Step 6: Repeat until reaching the stopping condition.

As discussed in Experiment II, GAT is constrained to select no more than 30 examples. The GAT control parameters are presented in Table 5.1.

Table 5.2: Baseline Performance

Classifier	Baseline	FX Examples	Perturbed Examples
RF	0.864	0.562	0.000
XGB	0.843	0.237	0.000
KNN	0.836	0.388	0.000
SVM	0.764	0.017	0.000

5.4 Results

The results in this section were generated by conducting the experiments outlined in Section 5.3. Each experiment evaluates each classifier with their respective adversarial datasets and $\text{Baseline}_{\text{train}}$ and $\text{Baseline}_{\text{test}}$ sets.

5.4.1 Experiment I: Results

The results of Experiment I are shown in Table 5.2. The first column represents the specific classifier used to generate the results. The second column represents the classifiers accuracy when generating predictions on the $\text{Baseline}_{\text{test}}$ set. The third and fourth columns represent the accuracy when generating predictions using the $\text{Adversarial}_{\text{FX-test}}$ and $\text{Adversarial}_{\text{Pert-test}}$ sets. From Table 5.2, one can see that all the classifiers experience a significant decrease in performance when generating predictions on each type of adversarial example. The RF classifier shows the most resiliency to the adversarial examples while the SVM performs the worst. However, notice that only examples from the $\text{Adversarial}_{\text{FX-test}}$ set are able to be correctly classified. No classifiers were able to correctly classify the examples within the $\text{Adversarial}_{\text{pert-test}}$ dataset. From this, one can conclude that different types of adversarial examples are more effective than others.

5.4.2 Experiment II: Results

For Experiment II, the results were generated using 30 runs where each run augments 30 random examples from the respective adversarial training sets to the $\text{Baseline}_{\text{train}}$ set. The respective classifier is then trained with the augmented training set. The adversarially trained classifier

is then used to generate predictions on the $\text{Baseline}_{\text{test}}$, $\text{Adversarial}_{\text{FX-test}}$ and $\text{Adversarial}_{\text{Pert-test}}$ sets. The results are shown in Table 5.3 and Table 5.4.

In Table 5.3, the first column represents the specific classifier used to generate the results. The second column presents the classifiers' average accuracy and standard deviation on the $\text{Baseline}_{\text{test}}$ set when adversarially trained with the $\text{Baseline}_{\text{train}}$ set and 30 random adversarial examples from the $\text{Adversarial}_{\text{FX-train}}$ set. The third and fourth columns show the classifier's average accuracy and standard deviation when generating predictions on the $\text{Adversarial}_{\text{FX-test}}$ and $\text{Adversarial}_{\text{Pert-test}}$ sets respectively. Similarly, Table 5.4 presents the same metrics but when the specific classifier is adversarially trained with the $\text{Baseline}_{\text{train}}$ set and 30 random adversarial examples from the $\text{Adversarial}_{\text{Pert-train}}$ set. In both Tables, a blue value denotes an average that is greater than the corresponding value in the baseline.

When examining Table 5.3 and Table 5.8, one can observe how adversarial training affects the baseline performance. One can see that the average baseline accuracy changes slightly across most of the classifiers when trained with adversarial examples. Because of this observation, one can conclude that adversarial training does affect the baseline performance on non-adversarial examples. A second observation of interest is how adversarial training affects the performance when generating predictions on adversarial examples. Notice that when the classifiers are adversarially trained with FX examples, AT_{FX}, there is an average increase of 33.5% accuracy and at most 63.7% accuracy when generating predictions on the $\text{Adversarial}_{\text{FX-test}}$ set as compared to the baseline results in Table 5.2. This behavior is more apparent as one examines the adversarial training results when using the $\text{Adversarial}_{\text{Pert}}$ set, AT_{Pert}. Following AT_{Pert}, the accuracy on the $\text{Adversarial}_{\text{Pert-test}}$ set increases, on average, from 0% to 42.8% and at most 77.3%.

One can also see that adversarial training using one example type enables correct classification for examples of the other type. This is critical because it demonstrates that there exists information that can be gained using adversarial training that is shared across the different types of adversarial examples. When considering that generating adversarial examples is often computationally prohibitive, it is of value to identify what samples can be generated at a low cost but offer an increased defensive posture.

Table 5.3: FX Adversarial Training Performance

Classifier	AT_{FX}	FX Examples	Perturbed Examples
RF	0.872 ± 0.008	0.985 ± 0.016	0.628 ± 0.070
XGB	0.831 ± 0.011	0.874 ± 0.061	0.189 ± 0.072
KNN	0.833 ± 0.006	0.486 ± 0.031	0.002 ± 0.008
SVM	0.764 ± 0.002	0.202 ± 0.054	0.037 ± 0.035

Table 5.4: Perturbed Adversarial Training Performance

Classifier	AT_{Pert}	FX Examples	Perturbed Examples
RF	0.869 ± 0.009	0.462 ± 0.103	0.773 ± 0.025
XGB	0.839 ± 0.011	0.154 ± 0.069	0.716 ± 0.038
KNN	0.832 ± 0.013	0.418 ± 0.086	0.213 ± 0.045
SVM	0.764 ± 0.002	0.083 ± 0.113	0.010 ± 0.016

5.4.3 Experiment III: Results

The results of Experiment III are shown in Table 5.7 and Table 5.8. These results were generated using 30 runs of the GAT procedure described in Experiment III. Specifically, Table 5.7 presents the results generated by GAT while using the $Adversarial_{FX\text{-}train}$ set within the adversarial training procedure and optimizing the resulting $Adversarial_{FX\text{-}test}$ set accuracy. While Table 5.8 presents the results generated by GAT while using the $Adversarial_{Pert\text{-}train}$ set within the adversarial training procedure and optimizing the resulting $Adversarial_{Pert\text{-}test}$ accuracy. In both tables, the first column represents the specific classifier used to generate the results. The second column presents the average classification accuracy for GAT trained model on the $Baseline_{test}$ set and standard deviation. The third column shows the average classification accuracy and standard deviation on the $Adversarial_{FX\text{-}test}$ set. The forth column shows the average classification accuracy and standard deviation on the $Adversarial_{Pert\text{-}test}$ set. In both Tables, a red value denotes an average that is greater than the corresponding values in the AT_{FX} and AT_{Pert}.

In Table 5.7, one can see that GAT_{FX}, when compared with AT_{FX} in Table 5.3 for each of the four classifiers, is similar in performance. This shows that the baseline performance is only slightly changed using GAT. On the FX examples, GAT_{FX} outperforms AT_{FX} for all four classifiers. On the Perturbed examples, GAT_{FX} outperforms AT_{FX} on each of the classifiers except for RF.

Table 5.5: GAT_{FX} Performance

Classifier	GAT_{FX}	FX Examples	Perturbed Examples
RF	0.869 ± 0.008	1.000 ± 0.000	0.625 ± 0.044
XGB	0.833 ± 0.010	0.987 ± 0.009	0.202 ± 0.040
KNN	0.823 ± 0.007	0.567 ± 0.025	0.169 ± 0.019
SVM	0.756 ± 0.004	0.281 ± 0.032	0.052 ± 0.009

Table 5.6: GAT_{Pert} Performance

Classifier	GAT_{Pert}	FX Examples	Perturbed Examples
RF	0.863 ± 0.010	0.551 ± 0.096	0.867 ± 0.018
XGB	0.841 ± 0.011	0.159 ± 0.037	0.794 ± 0.022
KNN	0.841 ± 0.008	0.429 ± 0.024	0.395 ± 0.034
SVM	0.763 ± 0.003	0.181 ± 0.058	0.038 ± 0.008

In Table 5.8, GAT_{Pert} outperforms AT_{Pert} on XGB and KNN and is outperformed by AT_{Pert} on RF and SVM. Notice, however, that GAT_{Pert} outperforms AT_{Pert} on all four classifiers on both the FX and Perturbed Examples.

These results show that GAT is an effective strategy for selecting which adversarial examples to augment to the baseline training set. In all models, GAT resulted in an increase in classification accuracy when trained with the corresponding $Adversarial_{train}$ set. An additional observation that one can see in Tables 5.7 and 5.8 is that once again, similar to what is seen in Tables 5.3 and 5.4 is that GAT_{FX} has better performance on the FX examples and GAT_{Pert} has better performance on the Pert examples. These results suggest that a hybrid adversarial training strategy can be formed by combining FX and Pert adversarial examples. This hybrid training strategy will be a direction for future work.

The results of Experiment III are shown in Table 5.7 and Table 5.8. These results were generated using 30 runs of the GAT procedure described in Experiment III. Specifically, Table 5.7 presents the results generated by GAT while using the $Adversarial_{FX\text{-}train}$ set within the adversarial training procedure and optimizing the resulting $Adversarial_{FX\text{-}test}$ set accuracy. While Table 5.8 presents the results generated by GAT while using the $Adversarial_{Pert\text{-}train}$ set within the adversarial training procedure and optimizing the resulting $Adversarial_{Pert\text{-}test}$ accuracy. In both tables, the first column represents the specific classifier used to generate the results. The second column presents the average classification accuracy for GAT trained model on the $Baseline_{test}$

Table 5.7: GAT_{FX} Performance

Classifier	GAT_{FX}	FX Examples	Perturbed Examples
RF	0.869 ± 0.008	1.000 ± 0.000	0.625 ± 0.044
XGB	0.833 ± 0.010	0.987 ± 0.009	0.202 ± 0.040
KNN	0.823 ± 0.007	0.567 ± 0.025	0.169 ± 0.019
SVM	0.756 ± 0.004	0.281 ± 0.032	0.052 ± 0.009

set and standard deviation. The third column shows the average classification accuracy and standard deviation on the $Adversarial_{FX\text{-}test}$ set. The forth column shows the average classification accuracy and standard deviation on the $Adversarial_{Pert\text{-}test}$ set. In both Tables, a red value denotes an average that is greater than the corresponding values in the AT_{FX} and AT_{Pert}.

In Table 5.7, one can see that GAT_{FX}, when compared with AT_{FX} in Table 5.3 for each of the four classifiers, is similar in performance. This shows that the baseline performance is only slightly changed using GAT. On the FX examples, GAT_{FX} outperforms AT_{FX} for all four classifiers. On the Perturbed examples, GAT_{FX} outperforms AT_{FX} on each of the classifiers except for RF.

In Table 5.8, GAT_{Pert} outperforms AT_{Pert} on XGB and KNN and is outperformed by AT_{Pert} on RF and SVM. Notice, however, that GAT_{Pert} outperforms AT_{Pert} on all four classifiers on both the FX and Perturbed Examples.

These results show that GAT is an effective strategy for selecting which adversarial examples to augment to the baseline training set. In all models, GAT resulted in an increase in classification accuracy when trained with the corresponding $Adversarial_{train}$ set. An additional observation that one can see in Tables 5.7 and 5.8 is that once again, similar to what is seen in Tables 5.3 and 5.4 is that GAT_{FX} has better performance on the FX examples and GAT_{Pert} has better performance on the Pert examples. These results suggest that a hybrid adversarial training strategy can be formed by combining FX and Pert adversarial examples. This hybrid training strategy will be a direction for future work.

Table 5.8: GAT_{Pert} Performance

Classifier	GAT_{Pert}	FX Examples	Perturbed Examples
RF	0.863 ± 0.010	0.551 ± 0.096	0.867 ± 0.018
XGB	0.841 ± 0.011	0.159 ± 0.037	0.794 ± 0.022
KNN	0.841 ± 0.008	0.429 ± 0.024	0.395 ± 0.034
SVM	0.763 ± 0.003	0.181 ± 0.058	0.038 ± 0.008

5.5 Discussion

In this section, we provide a further analysis (discussion) with respect to the types of adversarial examples that our research shows to be most successfully used for adversarial training as well as a closer look into the features of these examples that lead to their success.

For Experiment III, GAT was run 30 times where each run selects the best performing individual from the population. Thus, following GAT execution, there are 30 evolved individuals where each individual is a training mask that identifies an optimal subset of adversarial examples. To measure the success of each adversarial example, a measurement of quality is established. The quality of each adversarial example is calculated by consolidating the 30 evolved training masks into a single dataset. Thus, given a dataset of GAT evolved training masks M such that:

$$M_{\mathrm{i}} = \{m_{\mathrm{ij}}\},$$

where i is from 1 to N, j is from 1 to K , where N represents the number masks, and K represents the number adversarial examples.

The value of m_{ij} is defined as:

$$m_{\mathrm{ij}} = \begin{cases} 0 & \text{if } j \text{ example is not selected in } M_{\mathrm{i}} \\ 1 & \text{if } j \text{ example is selected in } M_{\mathrm{i}} \end{cases}$$

For each evolved training mask M_{i}, the corresponding GAT calculated fitness is denoted as f_{i}. Thus, the quality of each adversarial example q_{j} can be represented by:

$$\begin{bmatrix} m_{11} & \dots & m_{1\mathrm{K}} \\ \vdots & \ddots & \vdots \\ m_{\mathrm{N1}} & & m_{\mathrm{NK}} \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_{\mathrm{N}} \end{bmatrix}$$

$$q_{\mathrm{j}} = \frac{\sum_{i=1}^{n} m_{\mathrm{ij}} f_{\mathrm{i}}}{N}$$

Given the calculated quality for each adversarial example, linear regression is used to determine what features within the adversarial examples are statistically significant in relation to the quality measure and thereby lead to their success. These statistically significant features are identified using their p-value such that p-value < 0.05.

Both Fig. 5.2 and Fig. 5.3 present a visual overview of the statistically significant features in relation to each classifier and the adversarial example types. In each figure, the circles represent the group of statistically significant features for the respective classifier. The numerical quantities within the circled areas present the number of statistically significant features in relation to the classifiers. By inspecting the relationships between the significant features and classifiers, one can identify which features contribute to an increase in the quality of an adversarial example.

Within Fig. 5.2, each classifier is shown with their associated significant feature relationships for the $\mathrm{GAT_{FX}}$ selected examples. The results show that the SVM classifier has the greatest number of features that are significant in relation to the quality measure while the XGB has the least. This is of interest because previously, the SVM performed worst when faced with adversarial examples and subsequent adversarial training performance. This indicates that the SVM has the largest attack surface among all the classifiers. Table 5.9 presents the shared statistically significant features between the $\mathrm{GAT_{FX}}$ selected adversarial examples. Notice that most of the shared significant features are textual based. This points to potential issues when extracting features from text. Since textual based features are easier to fabricate than environmental or news source features, there exists a larger vulnerable attack space.

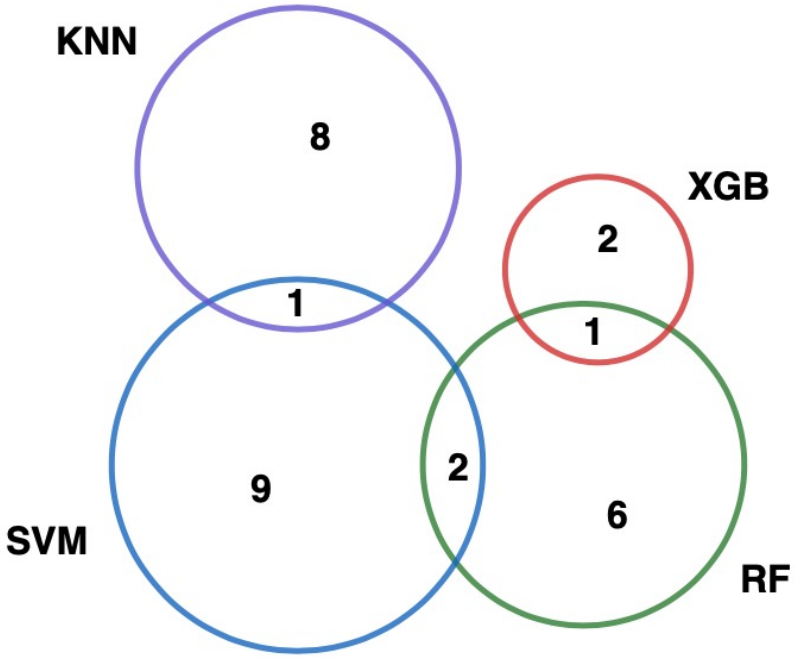

Figure 5.2: GAT_{FX} Statistically Significant Feature Relationships

Within Fig. 5.3, each classifier is also shown with their associated significant feature relationships for the GAT_{Pert} selected examples. Again, the results show that the SVM classifier has the most significant features in relation to the quality measure while the KNN has the least. Once again, one can see that there are shared significant features between the classifiers. Table 5.10 presents the shared statistically significant features between the GAT_{Pert} selected adversarial examples. Again, notice that all the shared significant features are textual based which can lead to a larger space for potential attack.

Notice, in Fig. 5.2 and Fig. 5.3, that there exists an overlap between most of the classifiers with respect to which features are statistically significant in relation to the quality of the adversarial example. This shared relationship among the features potentially illustrates points of vulnerability. By identifying what features are essential to the defensive posturing created by GAT, the attack space is reduced for adversaries. Controversially, this identification facilitates the process of model robustification by prioritizing and focusing defensive efforts.

Table 5.9: GAT_{FX} Selected Adversarial Examples Shared Statistically Significant Features

Feature	Category	RF	XGB	KNN	SVM
sentence_begin_pronoun	Textual			X	X
sentence_begin_article	Textual	X			X
word_usage_pronoun	Textual	X			X
Sixltr	Textual	X	X		

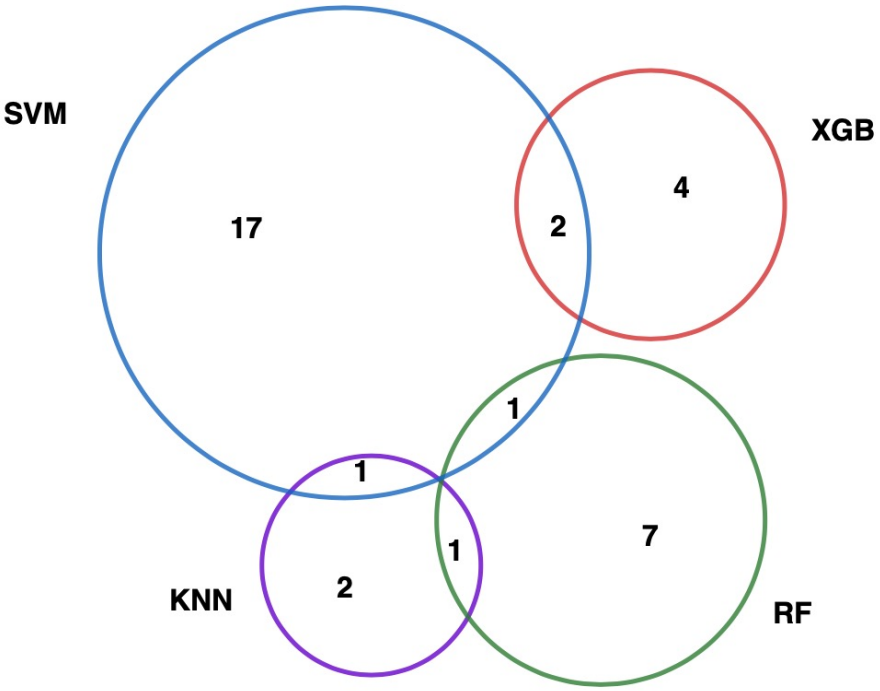

Figure 5.3: GAT_{Pert} Statistically Significant Feature Relationships

Table 5.10: GAT_{Pert} Selected Adversarial Examples Shared Statistically Significant Features

Feature	Category	RF	XGB	KNN	SVM
900 (domain)	News Source			X	X
filler	Textual	X		X	
auxverb	Textual		X		X
assent	Textual		X		X
word_usage_auxverb	Textual	X			X

5.6 Summary

In this chapter, we presented two methods for generating adversarial examples. We then use these examples to measure the affect that the adversarial examples have on the classification accuracy of four machine learners. Additionally, we introduced a novel method for discovering subsets of adversarial training examples that lead to a more robust machine learning system. Our result suggest that adversarial training can be used to create a stronger defensive posture against adversarial attacks and that there exists a transfer of knowledge gained from using different types adversarial examples. Our results also show that the performance of adversarial training can be increased by discovering optimal subsets of adversarial examples using GAT.

Chapter 6

A Study of Social Network Messages During the COVID-19 Infodemic: Salient Features and the Propagation of Information Types

6.1 Introduction

Amid a world of the COVID-19 pandemic, people often have turned to social media as a means of communicating and receiving information. Within existing social networks one can discover a wide range of messages that vary in sentiment, measure of helpfulness, and/or intended harm. In this chapter, we measure the utility of Tweets based on their ability to propagate throughout the Twittersphere as well as the salient features that contribute to their successful propagation. In this work, we consider five types of Tweets that can be classified as: (1) Caution & Advice, (2) Doubt & Criticism, (3) Rumor & Counter-Rumors, (4) Generic Harm, and (5) Non-Situational Information. Our results suggest that, on average, Caution & Advice messages propagate fastest while messages classified as Generic Harm have the slowest propagation.

For the greater part of 2020, we have been in a worldwide battle with the novel Coronavirus. This virus has not only caused a global pandemic, it has also caused a global infodemic [61, 62]. According to [63], the word, infodemic (coined by Washington Post columnist, David Rothkopf, in 2003), can be defined as "*... a blend of 'information' and 'epidemic' that typically refers to a rapid and far-reaching spread of both accurate and inaccurate information about something, such as a disease. As facts, rumors, and fears mix and disperse, it becomes difficult to learn essential information about an issue. Infodemic was coined in 2003, and has seen renewed usage in the time of COVID-19.*"

In this chapter, we study the utility of Tweets communicated during the early stages of the COVID-19 infodemic based on their ability to propagate throughout the Twittersphere [64, 65]

as well as the salient features that contribute to their successful propagation. In this study, we consider five types of Tweets that can be classified as: (1) Caution & Advice, (2) Doubt & Criticism, (3) Rumor & Counter-Rumors, (4) Generic Harm, and (5) Non-Situational Information. We seek to answer three questions:

Question #1: What are the most salient features for classifying social media messages?

Question #2: Which types of messages, on average, propagate the fastest?

Question #3: Given an information type can one determine the characteristics that will increase its propagation across a social network?

6.2 Our Process

6.2.1 Data Collection

News of the novel coronavirus, COVID-19, became established and began to widely circulate social media at the beginning of 2020 [66]. The effects and implications of COVID-19 for United States persons began emerging early February 2020. Due to the timeline of the COVID-19 response and the focus on the collection of English Tweets, Twitter Dataset is used as the source for COVID-19 related Tweets.

Within the Twitter Dataset, the Tweet IDs are hydrated using Twitter's Lookup API and the non-English Tweets are discarded. The hydration process fetches the requested data associated with the supplied Tweet IDs. This data includes both Tweet and user metadata. Following the hydration process, the final Tweet count is approximately 37M. From the 37M Tweets, the authors that have at least 100 Tweets are identified and their respective Tweets retained. These Tweets will be referred to as Tweets_{100} and will serve as the base dataset for this study. The Tweets_{100} dataset consists of 2464 authors and 2,308,924 Tweets. From the Tweets_{100} dataset, 3000 Tweets are randomly sampled for manual labeling. These 3000 Tweets will be refereed to as $\text{Tweets}_{\text{3K-labeled}}$. An additional dataset, is also sampled from Tweets_{100} by retrieving 15 additional Tweets for each author within the $\text{Tweets}_{\text{3K-labeled}}$ dataset. These Tweets will be referred to as $\text{Tweets}_{\text{unlabeled}}$. The Tweets in the $\text{Tweets}_{\text{3K-labeled}}$ dataset are then labeled with the following information types.

Type 1 - Caution and Advice

Includes facts and provides situational awareness of scientific observations, studies, statistics on number of cases, spread, and deaths, along with expert health-related guidelines to prevent harm from the virus. Precautionary aspects notify the public to help them protect themselves from the harm of the virus and include explanations of wearing masks, hand-washing, social distancing, and avoiding crowds and closed spaces for protection against virus transmission.

Type 2 - Doubt and Criticism

Involves sociopolitical questioning of government handling, responsibilities, or implications related to the pandemic crisis. Message context can contain uncertainty and validity of the COVID-19 pandemic reality and can range from legitimate concerns to mis/disinformation when understanding or sharing virus data.

Type 3 - Rumors and Counter-Rumors

Rumors are defined as talk, opinion, or statements widely disseminated with no discernible source or basis for truth, or a story or statement in general circulation without confirmation or certainty as to facts. Counter-Rumors may attempt to refute fake news or rumors by presenting credible evidence such as demonstrating how wearing masks helps, proving the virus is real with statistical hospitalization data feedback, and contact-trace data showing human spread paths. Typically has a temporal component.

Type 4 - Generic Harm

Intend to cause damage, injury or pain to physical, digital, mental, economic, psychological, reputational, social, political, success and societal aspects. Messaging can contain harm-related contextual and influential aspects for the purpose of gaining sensitive data, hiding data or information, causing physical damage to human populations/groups, inciting confusion or violence, and influencing election voting for specific candidates.

Type 5 - Non-Situational Information

This type includes all information that does not fit within any of the previously defined types.

6.2.2 Pre-Processing

The datasets used by the machine learning applications consist of two different feature sets: linguistic and psychometric features and those feature related to the Tweet metadata. The linguistic and psychometric features are extracted by processing the Tweet content using LIWC [38]. LIWC is a text analysis program that categorizes the words used in text into 93 different categories. The remaining features are extracted from the Tweet metadata surrounding the content, user, and date. These additional features include:

- timestamp
- user followers
- user following
- isVerified
- isRetweet
- retweet count
- tweet length
- hasUrl
- hasHashTag

Following feature extraction, both feature sets: LIWC and Tweet metadata, are processed independently. First, the following features are log transformed: timestamp, user followers, user following. Next, both feature sets are independently standardized by removing the mean and scaling to unit variance. Finally, both feature sets are combined to create the processed datasets [67].

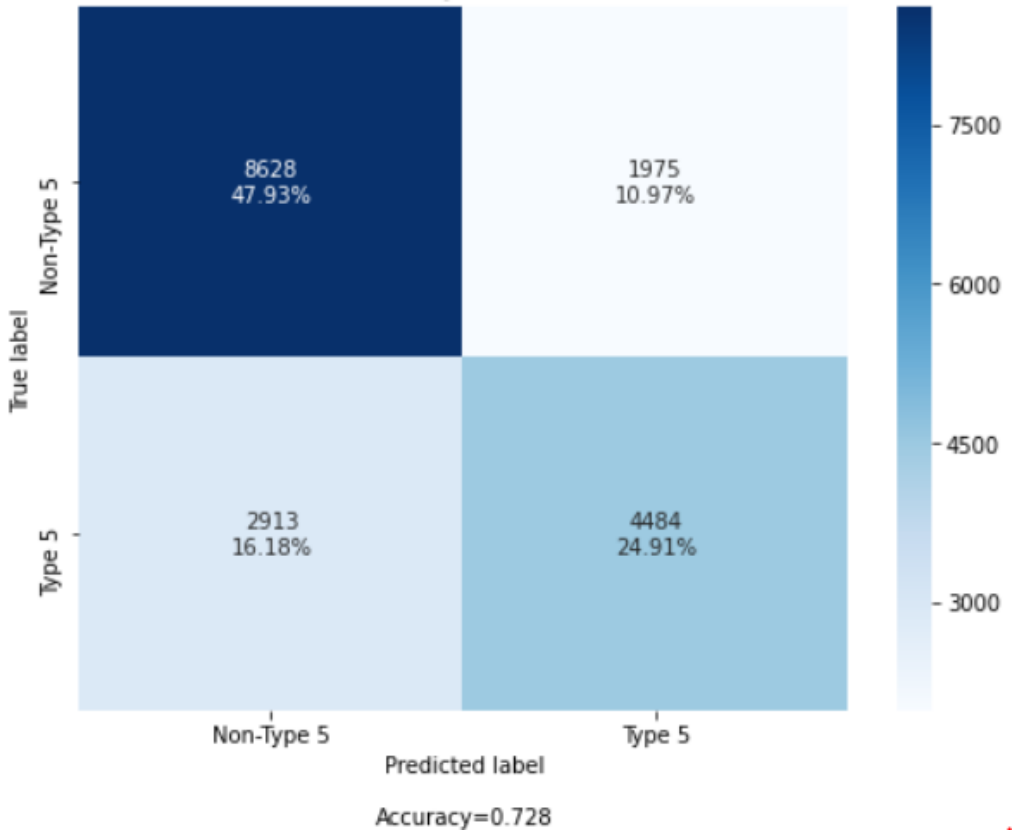

Figure 6.1: Type 5 Model Performance

Prior to starting the auto-label procedure, the issue of the Type 5 instances is addressed. The Type 5 class represents a *catch-all* for Tweets that did not belong to any of the other types. This class is also the largest in the $\text{Tweets}_{\text{3K-labeled}}$ dataset and thereby hinders classification performance. To address this issue, all instances of Type 5 Tweets are removed from both the $\text{Tweets}_{\text{3K-labeled}}$ and $\text{Tweets}_{\text{unlabeled}}$ datasets. This is accomplished by training a Type 5 vs non-Type 5 model using XGBoost (XGB) [57] to identify instances of Type 5 Tweets. The performance of this model is evaluated over 30 runs where each run has an 80/20 train/test split. The performance of this model is shown in Fig. 6.1. A total of 1,231 Type 5 Tweets are removed from the $\text{Tweets}_{\text{3K-labeled}}$ dataset and 9,775 from the $\text{Tweets}_{\text{unlabeled}}$ dataset.

6.2.3 Auto-Labeler

In order to label the $\text{Tweets}_{\text{unlabeled}}$ dataset, an auto-labeler is created using the $\text{Tweets}_{\text{3K-labeled}}$ dataset. The auto-labeler is iteratively evolved over 3 iterations to improve the classification accuracy and address the imbalance between the classes. For each iteration, the auto-labeler

uses an XGB model and a GEFeS evolved feature mask. Several additional classifiers were considered for auto-labeling use including: k-Nearest Neighbor [58], Naive Bayes [68], Random Forest [56], and Support Vector Machine with RBF kernel [59]. Ultimately, XGB was selected due to having a higher average accuracy and lower bias towards the Type 4 class. Fig. 6.2 presents the dataset distribution prior to the auto-labeling procedure. The steps of our auto-labeler procedure are as follows:

Step 1: Evolve a feature mask using $\text{GEFeS}_{\text{XGB}}$

Step 2: Build the auto-labeler with the $\text{GEFeS}_{\text{XGB}}$ feature mask applied

Step 3: Generate predictions on the $\text{Tweets}_{\text{unlabeled}}$ dataset

Step 4: Sample 95 Tweets of each information type from the generated predictions (380 total)

Step 5: Review the sampled predictions for correct labels and measure the performance

Step 6: Add the reviewed samples to the $\text{Tweets}_{\text{3K-labeled}}$ dataset and remove them from the $\text{Tweets}_{\text{unlabeled}}$ dataset

Step 7: Repeat 3 times

The procedure to build and evolve the auto-labeler is as follows. First, a feature mask is evolved using a $\text{GEFeS}_{\text{XGB}}$ model. This feature mask identifies and removes the non-essential and poorer performing features. Next, the $\text{GEFeS}_{\text{XGB}}$ model is used to generate predictions on the $\text{Tweets}_{\text{unlabeled}}$ dataset. For each class, 95 Tweets are sampled from the generated predictions. These sampled Tweets are then reviewed to evaluate and reinforce the auto-labeler's performance. This is accomplished by identifying the correct predictions from the sampled Tweets and calculating the Cohen's Kappa value [69]. Finally, the correct predictions are removed from the $\text{Tweets}_{\text{unlabeled}}$ dataset and added to the $\text{Tweets}_{\text{3K-labeled}}$ dataset. The expanded $\text{Tweets}_{\text{3K-labeled}}$ dataset is used to build the auto-labeler for the following iteration. This iterative process is conducted 3 times.

```
Type1: 93 (5.26%)
Type2: 191 (10.80%)
Type3: 464 (26.23%)
Type4: 1021 (57.72%)
Total: 1769
```

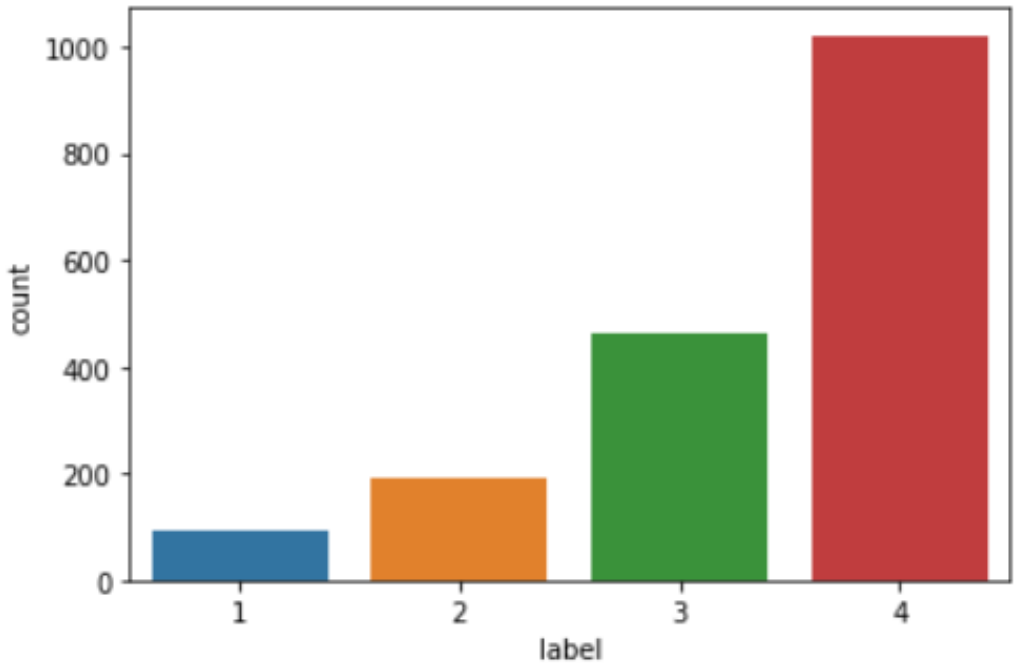

Figure 6.2: Iteration 1 $\text{Tweets}_{\text{3K-labeled}}$ Dataset

```
Type1: 631 (2.18%)
Type2: 1334 (4.61%)
Type3: 4539 (15.68%)
Type4: 22443 (77.53%)
Total: 28947
```

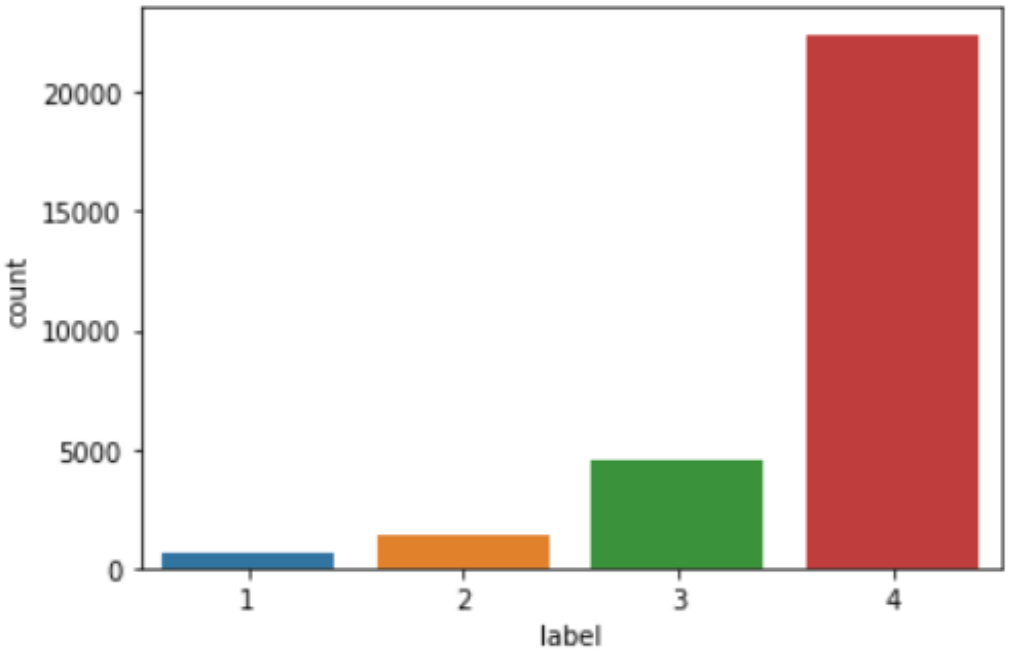

Figure 6.3: Final Dataset Distribution

Upon completion of the final iteration, the auto-labeler is used to generate the predictions for the remaining $\text{Tweets}_{\text{unlabeled}}$ dataset. Next, the generated predictions are combined with the $\text{Tweets}_{\text{3K-labeled}}$ dataset. Thus, the final dataset size is 28,948 Tweets. The distribution for the final dataset is shown in Fig. 6.3. The performance of the auto-labeler across each iteration is presented in Figs. 6.4 6.5 6.6. Notice that with each iteration the bias towards Type 4, the largest class, is reduced and the accuracy increased. Additionally, the inter-rater agreement increases with respect to the labels for each iteration. This is reflected by the increasing Cohen's Kappa value.

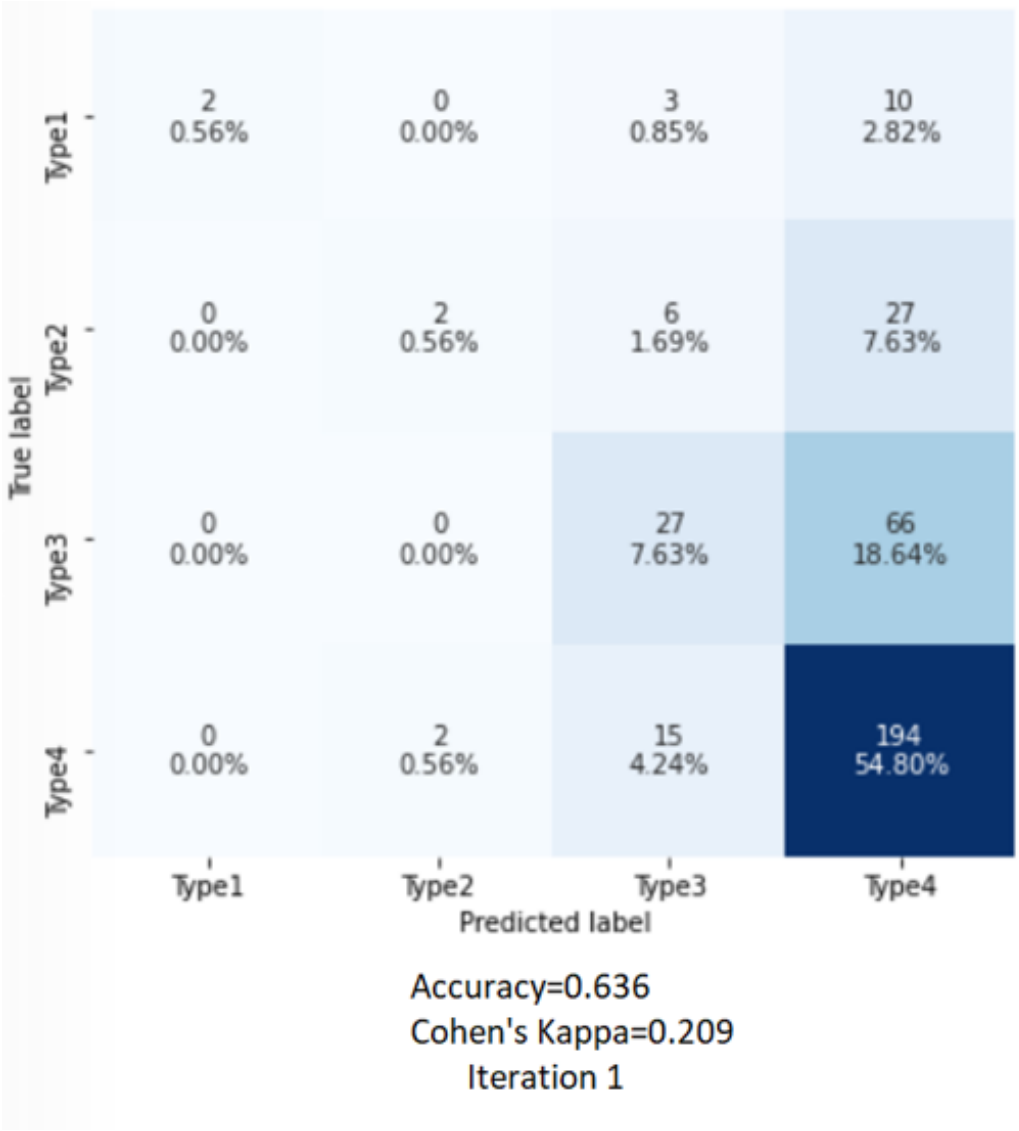

Figure 6.4: Auto-Labeler Performance Iteration 1

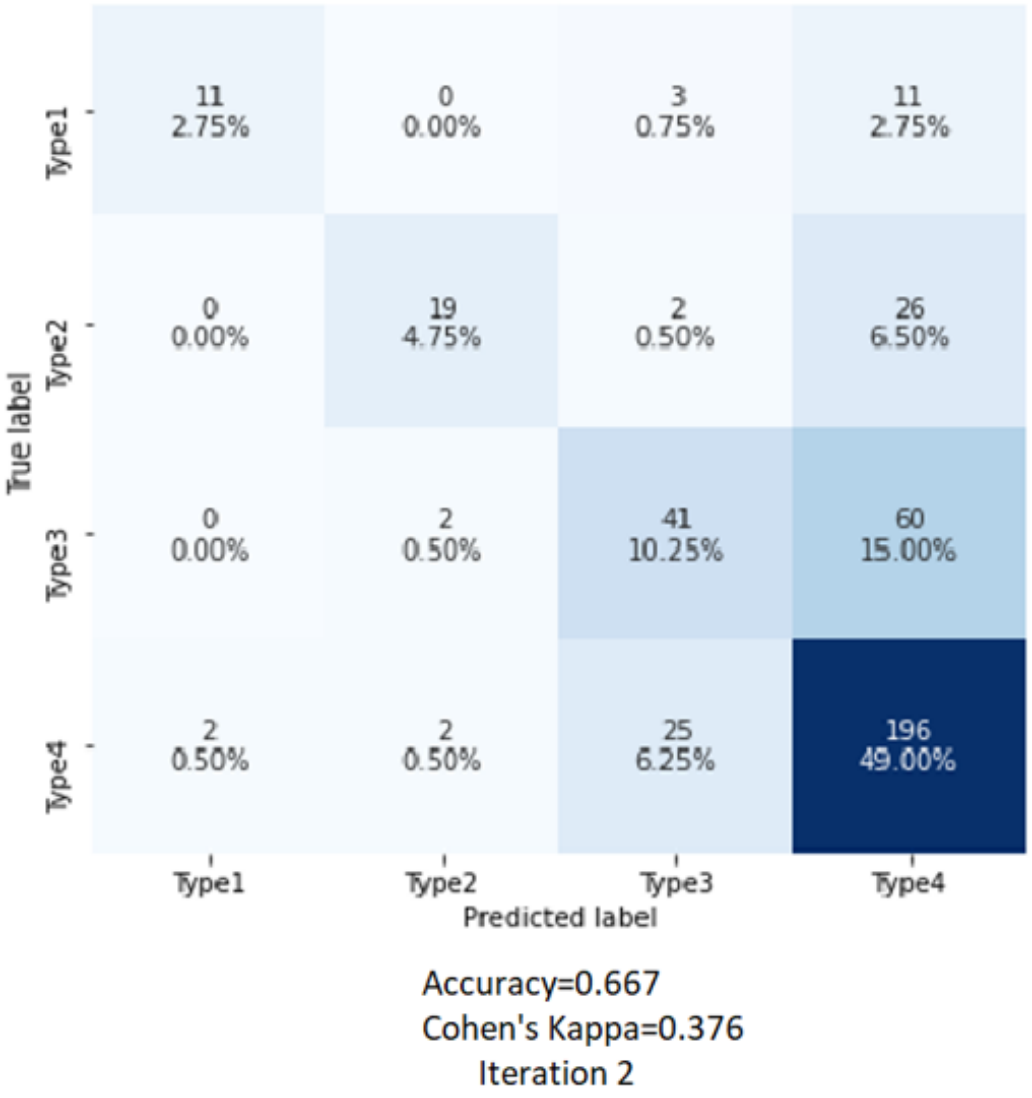

Figure 6.5: Auto-Labeler Performance Iteration 2

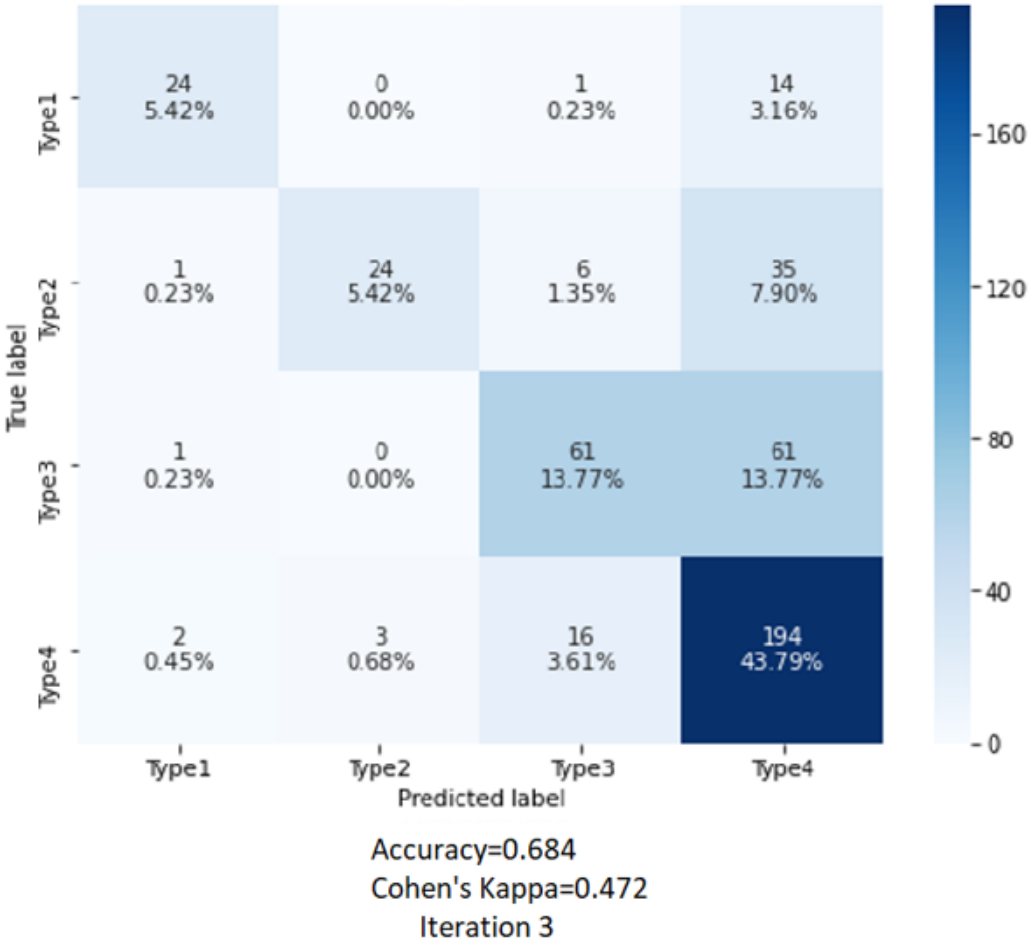

Figure 6.6: Auto-Labeler Performance Iteration 3

6.3 Results

6.3.1 Question #1: What are the most salient features for classifying social media messages?

To determine the most salient features for auto-labeling, one can examine the final feature masks that are evolved using GEFeS. During the final auto-labeler iteration, GEFeS is ran 10 times for 16000 FEs where on each run, the best performing CS is recorded. Examining the 10 recorded CSs enables identifying what features are the most salient and consistently contribute to a higher classification accuracy. This is accomplished by counting the occurrence of each feature in a CS and dividing each sum by the number of runs. Table 6.1 shows the top 10 salient features used to identify the types. Table 6.1 consists of four columns. The first column represents the feature index, the second column represents the feature associated with the feature index, and the third column represents the associated LIWC Psychometric Category (LPC). The fourth column represents the feature saliency (also known as Feature Consistency [1]).

When examining Table 6.1, one can note that the most salient features in determining the information types are the Linguistic/Psychometric features. This is of interest because it demonstrates that classifying the messages by type renders the non-linguistic features to be less essential. This indicates that users are propagating all information types without regard to their social media presence. The user's social media presence is reflected by the Log Follower Amount and Log Following Amount.

6.3.2 Question #2: Which types of messages, on average, tend to propagate the fastest?

Following our process, a large quantity of Tweets have been collected, processed, and labeled. The question of which types of messages, on average, propagate the fastest can now be answered. Several key features, delineated by type, are summarized in Table 6.2. The first column represents the information type. The second column represents the number of Tweets for the respective type. The third column represents the number of Tweets from a verified user. The fourth and fifth columns represent the summation of the log transformed user followers

Table 6.1: Final Auto-Labeler Top 10 Salient Features

Feature Index	Feature	LPC	Saliency
35	sad	Affective Processes	100%
59	achieve	Drives	100%
16	they	no-LPC	90%
93	OtherP	no-LPC	90%
6	WPS	no-LPC	80%
36	social	Social Processes	80%
43	cause	Cognitive Processes	80%
71	leisure	Personal Concerns	80%
85	colon	no-LPC	80%
100	TweetLength	no-LPC	80%

Table 6.2: Key Feature Summary

Types	Amount	Verified	Log Follower Amount	Log Following Amount	Retweet Count
Type 1 - Caution and Advice	631	43 (6.8%)	4,957 (7.86)	4,717 (7.48)	6,485,281 (10,277.8)
Type 2 - Doubt and Criticism	1,334	24 (1.8%)	10,096 (7.57)	9,997 (7.49)	4,399,517 (3298)
Type 3 - Rumors and Counter-Rumors	4,539	324 (7.1%)	36,496 (8.04)	34,731 (7.65)	16,658,787 (3,670)
Type 4 - Generic Harm	22,443	1,486 (6.6%)	174,638 (7.78)	166,704 (7.43)	57,392,850 (2,557.3)

amount and log transformed user following amount associated with the user propagating the Tweet. Finally, the sixth column represents the summation of the number of retweets. The values in parenthesis represent the average value for each metric. From Table 6.2, several observations can be made. First, one can note that messages that include Caution & Advice tend to propagate more on average than all other information types. However, messages that include Generic Harm are created significantly more than any other information type and thereby are able to reach a larger audience. Second, on average, Rumors and Counter-Rumors tend to be propagated by users with a larger social media presence.

Table 6.3: Salient Features Derived from Linear Regression of Type 1: Caution & Advice Tweets wrt Retweet Count (r-squared=0.434; adj r-squared=0.326)

Feature	Category	Coefficient	P-value
IsRetweet	Twitter Feature	0.2665	0.000
death	Personal Concern	-0.1785	0.000
HasHashTag	Twitter Feature	-0.1460	0.001
TweetLength	Twitter Feature	0.1651	0.002
auxverb	Linguistic Dimension	-0.2998	0.009
function	Linguistic Dimension	0.6126	0.020
posemo	Psychological Process	-0.6550	0.026
affect	Psychological Process	1.0097	0.029

6.3.3 Question #3: Given an information type can one determine the characteristics that will increase its propagation across a social network?

In Tables 6.3 – 6.6, one can see the results of using linear regression [70] in an effort show the effect that the features have with respect to retweets. Each table consists of 4 columns. The first column represents a feature. The second column represents the category of a particular feature. The third and fourth columns represent the linear regression coefficient associated with the feature and the corresponding P-value. The features are sorted by ascending P-value in each of the tables. The tables included only those features associated with P-values ≤ 0.05.

In Table 6.3, one can see the eight features that impact the retweet count for the Caution & Advice information type. Of the eight features, three are Twitter features (IsRetweet, HasHashTag, TweetLength), two are LIWC linguistic dimensions (auxverb, function), two are psychological processes (posemo, affect), and one is a personal concern psychometric (death). Only the IsRetweet and TweetLength features have a positive correlation with retweets. The results of Table 6.3 suggests that this information type has a greater chance of propagating if it is already a retweet, has some length, does not contain hashtags, and minimizes auxiliary verbs, positive emotion words, function words, and references to death.

In Table 6.4, for the Doubt & Criticism information type there are a total of 16 features. Of these features: 7 have a positive correlation (TweetLength, IsRetweet, number, focusfuture, anx, Analytic, health), with the first two being Twitter features (TweetLength, IsRetweet), number being a LIWC grammar metric, focusfuture being a time orientation psychometric, anx

Table 6.4: Salient Features Derived from Linear Regression of Type 2: Doubt & Criticism Tweets wrt Retweet Count (r-squared=0.244; adj r-squared=0.182)

Feature	Category	Coefficient	P-value
TweetLegnth	Twitter Feature	0.2283	0.000
IsRetweet	Twitter Feature	0.4338	0.000
LogUserFollowers	Twitter Feature	-0.1405	0.002
HasHashTag	Twitter Feature	-0.0717	0.005
prep	Linguistic Dimension	-0.1991	0.014
death	Personal Concern	-0.1013	0.017
money	Personal Concern	-0.0729	0.017
nonflu	Informal Language	-0.0817	0.019
number	Other Grammar	0.1275	0.023
achieve	Drive	-0.0718	0.024
focusfuture	Time Orientation	0.0580	0.028
WPS	Summary Language Variable	-0.0743	0.033
anx	Psychological Process	0.0624	0.039
female	Social Process	-0.0842	0.042
Analytic	Summary Language Variable	0.1868	0.045
health	Biological Process	0.3928	0.048

being a psychological process, Analytic being a LIWC summary language variable, and health being a biological process. Of the remaining ten that have a negative correlation, two are Twitter features (LogUserFollowers, HasHashtag), two are personal concern psychometrics (death, money). The last five include a LIWC linguistic dimension, a LIWC informal language metric, a drive psychometric, a social process, and a LIWC summary language variable (prep, nonflu, achieve, female, WPS). The results of Table 6.4 suggest that this information type has a greater chance of propagating if it is already a retweet, is a certain length, contains references to health, work, anxiety, numerical values, and analytical thinking. Using prepositions, nonfluencies (er, umm), hashtags, a certain number of words per sentence, having a certain number of followers, and containing references to death, money, and achievements negatively affect the propagation.

In Table 6.5, one can see the 19 features impact the retweet count for the Rumors & Counter-Rumors information type. Of these features, three Twitter Features (IsRetweet, TweetLength, TimeStamp), two drive psychometrics (power, risk), two LIWC informal language metrics (netspeak, assent), two LIWC summary language variables (Analytic, Authentic), a LIWC grammar metric (number) and a LIWC linguistic dimension (pronoun) all have positive correlations. The remaining features, including four Twitter features (HasUrl, HasHashTag,

Table 6.5: Salient Features Derived from Linear Regression of Type 3: Rumors & Counter Rumors Tweets wrt Retweet Count (r-squared=0.225; adj r-squared=0.207)

Feature	Category	Coefficient	P-value
IsRetweet	Twitter Feature	0.3124	0.000
TweetLength	Twitter Feature	0.1574	0.000
HasUrl	Twitter Feature	-0.0632	0.000
HasHashTag	Twitter Feature	-0.0966	0.000
risk	Drive	0.0818	0.001
netspeak	Informal Language	0.3297	0.001
power	Drive	0.0996	0.002
LogUserFollowers	Twitter Feature	-0.0671	0.005
Analytic	Summary Language Variable	0.0985	0.008
IsVerified	Twitter Feature	-0.0441	0.009
prep	Linguistic Dimension	-0.0887	0.012
number	Other Grammar	0.0361	0.018
ppron	Linguistic Dimension	-46.9191	0.027
assent	Informal Language	0.0548	0.029
Authentic	Summary Language Variable	0.0644	0.032
pronoun	Linguistic Dimension	43.8873	0.037
ipron	Linguistic Dimension	-27.4799	0.037
TimeStamp	Twitter Feature	0.0366	0.040
insight	Cognitive Process	-0.0477	0.047

LogUserFollowers, IsVerified), three LIWC linguistic dimensions (prep, ppron, ipron), and a cognitive process (insight), have a negative correlation. The results of Table 6.5 suggest that this information type has a greater chance of propagating if it is already a retweet, is a certain length, is posted in a specific time, contains common internet abbreviations (lol, btw, thx), has references to numbers, power, and risk, while avoiding the following: writing in the first person, prepositions, words that give insight, and Twitter features such as hashtags, URLs, a certain amount of followers, and account verification.

In Table 6.6, for the Generic Harm information type, there are a total of 24 features that impact the retweet count. Nine have a positive correlation (ipron, IsRetweet, TweetLength, percept, negemo, function, certain, sexual, drive). Ipron and function are LIWC linguistic dimensions. Percept, negemo, certain, and sexual are perceptual, psychological, cognitive, and biological processes. The other two are Twitter features. The remaining 15 features include: two LIWC summary language variables (WPS, SixItr), two LIWC linguistic dimensions (pronoun, prep), three drive psychometrics (reward, power, achieve), two Twitter features (HasUrl,

Table 6.6: Salient Features Derived from Linear Regression of Type 4: Generic Harm Tweets wrt Retweet Count (r-squared=0.269, adj r-squared=0.266)

Feature	Category	Coefficient	P-value
WC	Word Count	-0.0424	0.000
WPS	Summary Language Variable	-0.0314	0.000
pronoun	Linguistic Dimension	-1.8128	0.000
ipron	Linguistic Dimension	1.1279	0.000
reward	Drives	-0.0394	0.000
IsRetweet	Twitter Feature	0.3695	0.000
TweetLength	Twitter Feature	0.1786	0.000
HasUrl	Twitter Feature	-0.0610	0.000
HasHashTag	Twitter Feature	-0.1062	0.000
anx	Psychological Process	-0.0217	0.001
hear	Perceptual Process	-0.0617	0.001
focusfuture	Time Orientation	-0.0198	0.002
percept	Perceptual Process	0.0842	0.002
drive	Drive	0.0743	0.003
prep	Linguistic Dimension	-0.0409	0.007
see	Perceptual Process	-0.0436	0.007
power	Drives	-0.0389	0.015
function	Linguistic Dimension	0.0849	0.021
death	Personal Concern	-0.0150	0.022
negemo	Psychological Process	0.1121	0.027
certain	Cognitive Process	0.0217	0.036
sexual	Biological Process	0.0205	0.036
achieve	Drive	-0.0183	0.039
SixItr	Summary Language Variable	-0.0153	0.042

HasHashTag), two perceptual processes (hear, see), a LIWC word count metric (WC), a psychological process (anx), a time orientation psychometric (focusfuture), and a personal concern psychometric (death) all have a negative correlation. The results of Table 6.6 suggests that this information type has a greater chance of propagating if it is already a retweet, is a certain length, contains impersonal pronouns, words of sexual content, certainty (always, never), and function. The likelihood of propagation decreases if the Tweet has URLs and hashtags. Furthermore, using complex sentences with prepositions, pronouns (other than impersonal pronouns), and words containing six or more letters, including references to time, relativity, future, sight, sound, reward, anxiety, and death, negatively affect propagation.

We believe that one can use the results displayed in Tables 6.3 – 6.6 as directives for how to increase (or decrease) the propagation rate of an instance of an information type within the Twittersphere with respect to the COVID-19 infodemic.

6.4 Summary

In this chapter, we developed a process for developing an auto-labeler and a classifier for categorization a large number of Tweets into 4 information types. We then used our resulting labeled dataset to answer three questions related to: discovering the most salient features for classifying Tweets, determining the information types that tend to propagate the fastest, and gaining an understanding of the types of characteristics that may lead to faster propagation for a particular information type across the Twitter social network with respect to the COVID-19 infodemic.

Chapter 7

EMO-GAT: An Evolutionary Multi-Objective Method for Adversarial Training

7.1 Introduction

Adversarial examples are an effective method that can be leveraged to attack machine learning systems. The primary defensive method, adversarial training, is a simplistic approach when applied without optimization. In this paper we propose a novel method, evolutionary multi-objective optimization genetic adversarial training (EMO-GAT), that enables the development of a more robust model by performing evolutionary adversarial training using multiple types of adversarial examples. We demonstrate that EMO-GAT outperforms both traditional adversarial training and genetic adversarial training (GAT).

In this chapter, we introduce an extension of Genetic Adversarial Training (GAT) that utilizes evolutionary multi-objective optimization (EMO), referred to as Evolutionary Multi-objective Optimization Genetic Adversarial Training (EMO-GAT). We demonstrate the efficacy of EMO-GAT by comparing its performance against two different adversarial training (AT) methods: traditional AT and GAT. Additionally, we show that EMO-GAT can incorporate multiple types of adversarial examples, improve model robustness, discover ideal subsets of adversarial examples, discover an ideal quantity of adversarial examples to use within the AT routine, and can be used to facilitate the process of threat modeling ML systems.

7.2 Adversarial Examples

In this work, the adversarial examples are generated using two different grey box attacks: a Perturbation attack [22, 55] and a FX attack [2, 71]. Both of these attacks are leveraged against

a collection of ML classifiers independently in an effort to create a dataset of adversarial examples that will be used to objectively compare the performances of several different AT methods. The classifiers include: Random Forests (RF) [56], XGBoost (XGB) [57], k-Nearest Neighbors (KNN) [58], and Support Vector Machine with RBF kernel (SVM) [59].

Both attacks generate 200 false negative (FN) and 200 false positive (FP) adversarial examples targeting each classifier. The adversarial examples are then segmented into a training set and a test set where the split is 40/60. An overview of the generated adversarial examples is shown in Fig. 7.1.

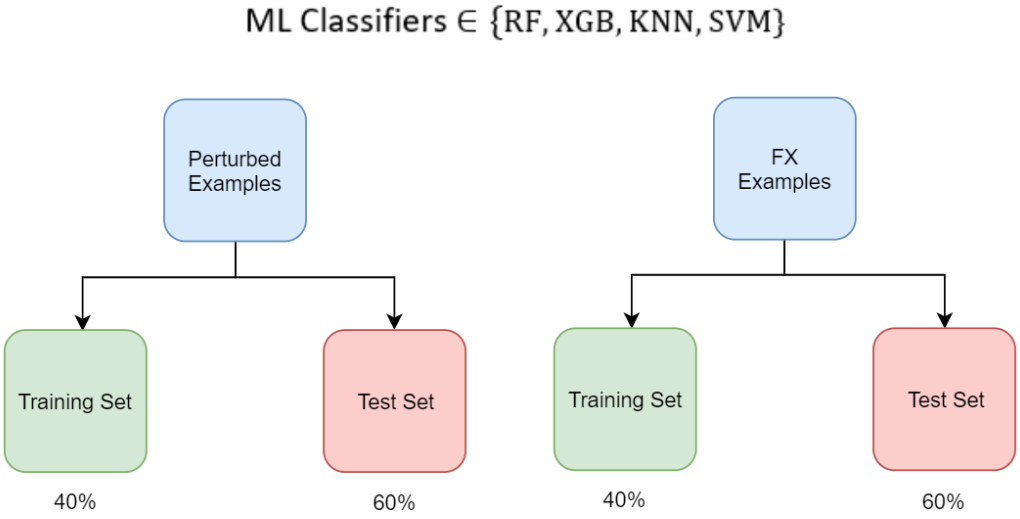

Figure 7.1: Adversarial Examples Overview

7.3 Model Robustness

Model robustness [72, 73] can be described as the process of posturing a model such that when the input variables are drastically changed, consistent and accurate output can still be achieved. The process of making a model more robust, model robustification [74], can generally be achieved using two different approaches: modifying the model structure and changing the training objective.

Modifying the model structure includes methods such as knowledge distillation [75] and adversarial detection [76]. Knowledge distillation aims to reduce the attack surface exploited by adversarial examples by reducing the size of the network. This is accomplished by transferring

knowledge from a large network to a small network. Research has shown [77] that transferred knowledge reduced model sensitivity and thereby reduced the susceptibility to adversarial examples generated using perturbation attack methods. While adversarial detection attempts to detect adversarial examples by analyzing the input prior to attempting to generate predictions. A few methods include statistical analysis [78] and training an additional classifier [79].

While changing the training objectives typically involves introducing adversarial examples into the training set. This process is referred to as adversarial training. Adversarial training is considered the most effective method to defend against adversarial examples and make a model more robust [80].

7.4 From AT to GAT to EMO-GAT

This section outlines the procedures of the different types of AT methods used in this work: AT, GAT, and EMO-GAT . Each method is evaluated using the performance on the models initial test set and the respective adversarial example test sets.

7.4.1 Traditional Adversarial Training

Traditionally, adversarial training (AT) [23,24] is the process of explicitly training a model with adversarial examples to make it more resilient or robust against adversarial attack.

While traditional AT is an effective method for making a model more robust, it is not without disadvantage. One issue is a phenomenon described as label leaking [81–83]. Label leaking is a issue that can emerge during AT where the model's classification accuracy becomes higher on adversarial examples than non-adversarial data. This potential shift in the adversarially trained models performance is why the quantity of adversarial examples used in the AT procedure is controlled and the change to the baseline performance is observed in the following experiment. The second issue is the inability to objectively evaluate the effects, performance, and optimizations of AT.

7.4.2 Genetic Adversarial Training

Genetic Adversarial Training (GAT) [2] aims to overcome several shortcomings present in traditional AT while enabling optimizing the performance of AT. GAT uses a steady state genetic algorithm [60] to evolve a population of training masks where a training mask is a binary string representing which adversarial examples will be used within the AT routine. A 1 means the adversarial example will be used while a 0 means it will be discarded. The training masks are evolved over a predefined number of function evaluations allowing the discovery of ideal subsets of adversarial examples that lead to a more robust model. GAT operates as follows:

Step 1: Generate a random population of training masks (individuals).

Step 2: Evaluate each individual in the population by augmenting the individual's selected adversarial examples and the target classifiers $\text{Baseline}_{\text{train}}$ set to create the adversarial training set. The classifier is then trained with this adversarial training set and is used to generate predictions on the adversarial test set. The resulting accuracy is used as the measure of fitness.

Step 3: Select two parents using binary tournament selection and create an offspring using uniform crossover.

Step 4: Evaluate the created offspring using the procedure outlined in Step 2.

Step 5: Replace the worst performing individual in the population with the offspring.

Step 6: Repeat until reaching the stopping condition.

To tackle the issue of label leakage, the GAT AT routine constrains the number of selected adversarial examples to 30. This number was chosen because it represents approximately 5% of the classifier training set. However, by constraining the number of selected adversarial examples, the search space for ideal subsets is also limited.

7.4.3 Evolutionary Multi-objective Optimization Genetic Adversarial Training

Evolutionary multi-objective optimization genetic adversarial training (EMO-GAT) is an extension of GAT that incorporates evolutionary multi-objective optimization (EMO). EMO-GAT uses the non-dominated sorting genetic algorithm II (NSGA-II) [36]. The non-dominated sorting genetic algorithm III (NSGA-III) [84] was also considered; however, for this problem, due to having only two objectives NSGA-II was chosen for use.

EMO-GAT overcomes the shortcomings of both traditional AT and GAT. To address the previous constraint that GAT imposed on the number of adversarial examples used in the AT procedure, EMO-GAT does not explicitly limit the number of selected adversarial examples. Instead, in EMO-GAT, the number of selected adversarial examples is controlled by introducing a penalty function [85, 86], P. The penalty function, P, is applied by measuring the degradation of the adversarially trained classifier's performance on the baseline test set ($\mathrm{A_{Adv}}$) when compared to the performance of the non-adversarial trained classifier ($\mathrm{A_{Non\text{-}Adv}}$) and reducing the fitness of each objective (F1, F2). Thus, given a change in accuracy:

$$\Delta A = A_{\text{Non-Adv}} - A_{\text{Adv}}$$

The penalty function can be represented by:

$$P = \begin{cases} 0 & \text{if } \Delta A \leq 0 \\ \Delta A & \text{if } \Delta A > 0 \end{cases}$$

Where the modified fitness, $F1'$ and $F2'$ can be calculated by:

$$F1' = F1 - P$$

$$F2' = F2 - P$$

The baseline test set represents represents "clean" or non-adversarial data and can illustrate how AT affects the baseline performance and the potential presence of label leaking. By

introducing a penalty function, the number of selected adversarial examples can by implicitly evolved while potentially mitigating the impact of label leaking.

The procedure of EMO-GAT is as follows:

Step 1: A population of individuals is generated. Each individual consists of two training masks: TM_{FX} and TM_{Pert}. The TM_{FX} use adversarial examples from the $Adversarial_{FX\text{-}train}$ set and the TM_{Pert} use adversarial examples from the $Adversarial_{Pert\text{-}train}$ set.

Step 2: Each individual in the population is evaluated to determine the fitness for each objective. The individuals are evaluated by conducting AT using the TM_{FX} and TM_{Pert} selected adversarial examples and the classifiers associated $Baseline_{train}$ set. Following training, the adversarially trained classifier is used to generate predictions on the associated $Baseline_{test}$ set, $Adversarial_{FX\text{-}test}$ set, and $Adversarial_{Pert\text{-}test}$. The resulting accuracies are used within the aforementioned penalty function to determine the fitness for each objective.

Step 3: The population is then sorted based on non-domination into fronts with respect to each objective.

Step 4: For each individual in the population, a diversity control mechanism, crowding distance, is measured by calculating the how close each individual is to its neighbors.

Step 5: A new population is filled by selecting the best N individuals from the current population, where N is the size of the population.

Step 6: Parents are selected using binary tournament selection by first considering the front level and next the crowding distance.

Step 7: Offspring are created using a crossover and mutation operation. The offspring are then evaluated to determine the fitness for each objective and are augmented to the population.

Step 8: Go to Step 3. This procedure is repeated until reaching the predefined stopping condition.

7.5 Our Experiment

7.5.1 Dataset

The dataset used in this work is BuzzFace [14, 37]. BuzzFace is a fake news detection data composed of news stories posted to Facebook during September 2016 and that relate to the 2016 United States presidential election.

7.5.2 Experiment Components

The experiment performed utilizes a collection of machine learning classifiers (e.g. RF, XGB, kNN, SVM) to compare the performance of several AT methods. Each classifier is trained using an 80/20 train and test split. The associated training sets and test sets are recorded for further use during the experiment. This data will be referred to as $\text{Baseline}_{\text{train}}$ and $\text{Baseline}_{\text{test}}$ respectively. Additionally, each classifier also has its own set of adversarial examples (FX, Perturbed). As previously mentioned, the adversarial examples are split into a training and test set using a 40/60 split. This data will be referred to as $\text{Adversarial}_{\text{FX-train}}$, $\text{Adversarial}_{\text{FX-test}}$, $\text{Adversarial}_{\text{Pert-train}}$, and $\text{Adversarial}_{\text{Pert-test}}$. A summary of the different components is summarized below.

- $\textbf{Baseline}_{\textbf{train|test}}$ - the initial data associated with each classifier. This data enables conducting adversarial training and provides a baseline for measuring the change in the original test accuracy (baseline accuracy).
- $\textbf{Adversarial}_{\textbf{FX-train|test}}$ - the FX adversarial examples generated for each classifier. This data will be used within the adversarial training routines and enable evaluating the performance of adversarially trained classifiers.
- $\textbf{Adversarial}_{\textbf{Pert-train|test}}$ - the Perturbed adversarial examples generated for each classifier. This data will be used within the adversarial training routines and enable evaluating the performance of adversarially trained classifiers.

Table 7.1: Baseline Classifier Performance

Classifier	Baseline	FX Examples	Perturbed Examples
RF	0.864	0.562	0.000
XGB	0.843	0.237	0.000
KNN	0.836	0.388	0.000
SVM	0.764	0.017	0.000

- **Baseline Accuracy**- the baseline accuracy represents the performance of the respective classifiers when generating predictions on the $\text{Baseline}_{\text{test}}$ data. This metric enables determining how the AT methods affect the classification performance on non adversarial data.

- **FX Accuracy** - the FX accuracy represents the performance of the respective classifiers when generating predictions on the associated $\text{Adversarial}_{\text{FX-test}}$ data. This metric enables evaluating how AT affects the classification performance on FX adversarial examples.

- **Pert Accuracy** - The Pert accuracy represents the performance of the respective classifiers when generating predictions on the associated $\text{Adversarial}_{\text{Pert-test}}$ data. This metric enables evaluating how AT affects the classification performance on Pert adversarial examples.

7.5.3 Classifier Baseline

The baseline performance for each classifier is outlined in Table 7.1. The first column presents the classifier used to generate the results. The second column presents the classifier accuracy on the $\text{Baseline}_{\text{test}}$ set. The third and forth columns represent the classifier performance when classifying the associated $\text{Adversarial}_{\text{FX-test}}$ and $\text{Adversarial}_{\text{Pert-test}}$ sets. These results provide a baseline to evaluate the different AT methods.

7.5.4 Experiment Layout

Traditional AT is conducted by selecting 30 random adversarial examples to augment to the training set to create the adversarially trained model. This process is performed for each model using their respective AT sets, $\text{Adversarial}_{\text{FX-train}}$ and $\text{Adversarial}_{\text{FX-test}}$, independently. Thus,

using the traditional AT method, there exists two sets of results representing the respective adversarial examples used within the AT routine: AT_{FX} and AT_{Pert}. Following execution, the generated results from each run will be used to measure the performance of traditional adversarial training.

GAT is conducted by evolving a population of training masks where each training mask represents the selected adversarial examples to use in the AT. Similar to traditional AT, GAT is also constrained to selected no more than 30 adversarial examples. The GAT procedure is performed independently for each classifier using their respective adversarial training sets: $Adversarial_{FX\text{-}train}$ and $Adversarial_{Pert\text{-}train}$. Thus, using GAT, there exists two sets of results representing the respective adversarial examples used within the GAT routine: GAT_{FX} and GAT_{Pert}. Following execution, the best individual will be selected from the evolved population for each run. These individuals will be used to measure the performance of GAT.

EMO-GAT is conducted by discovering pareto-optimal sets of solutions and mapping out the different pareto fronts. This is accomplished by utilizing NSGA-II to evolve a population of individuals that contain training masks for the respective adversarial example types. The EMO-GAT procedure is performed independently for each model using each classifier's respective adversarial training sets: $Adversarial_{FX\text{-}train}$ and $Adversarial_{Pert\text{-}train}$. Following execution, a single individual from the pareto-optimal front is selected from each run. These individuals will be used to measure the performance of EMO-GAT.

7.6 Results

The results presented in this section were generated by evaluating the performance of each AT method: traditional AT, GAT, and EMO-GAT. Each AT method's performance is evaluated over 30 runs, where a run is the process of completing the procedure of the AT method and generating results. The performance for each method is measured using 3 different metrics: average Baseline accuracy, average FX accuracy, and average Pert accuracy. The performance for each AT method is presented in Figs. 7.2, 7.3, 7.4. Within each figure, the bars denote the classifier performance associated with a particular AT method.

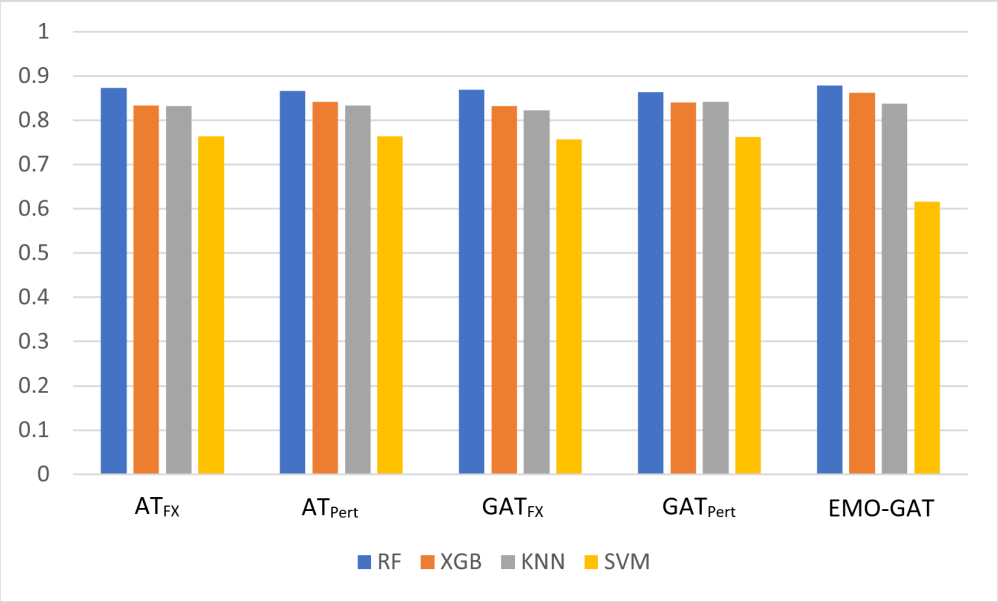

Figure 7.2: Average Baseline Accuracy

In Fig. 7.2, the average Baseline accuracy is presented for the experiment. The Baseline accuracy represents how well the adversarially trained classifiers performed on non-adversarial data. One can note that in all AT training methods, there is a deviation in the baseline performance when compared to the non adversarially trained classifier. This is of interest due to the phenomenon of label leaking that can emerge through the process of AT. Considering this, the results suggest that the methods used to control the number of adversarial examples used within the different AT routines is generally effective in mitigating the effects of label leaking. However, the EMO-GAT method when utilizing the KNN classifier appears to have approximately a 15% decrease in baseline accuracy. This can be addressed by introducing a more severe or advanced penalty function within the EMO-GAT procedure.

The average FX accuracy for the experiment is presented in Fig. 7.3. The FX accuracy represents how well the adversarial trained classifiers performed on the Adversarial$_{\text{FX-test}}$ data set. One can note that EMO-GAT performs slightly worse than the GAT$_{FX}$ method for several classifiers. This can be attributed to GAT$_{FX}$ being optimized on only a single objective.

In Fig. 7.4, the average Pert accuracy is presented for the experiment. The Pert accuracy represents how well the adversarially trained classifiers performed on the Adversarial$_{\text{Pert-test}}$ data set. Notice that for all classifiers, EMO-GAT out significantly outperforms all other AT methods

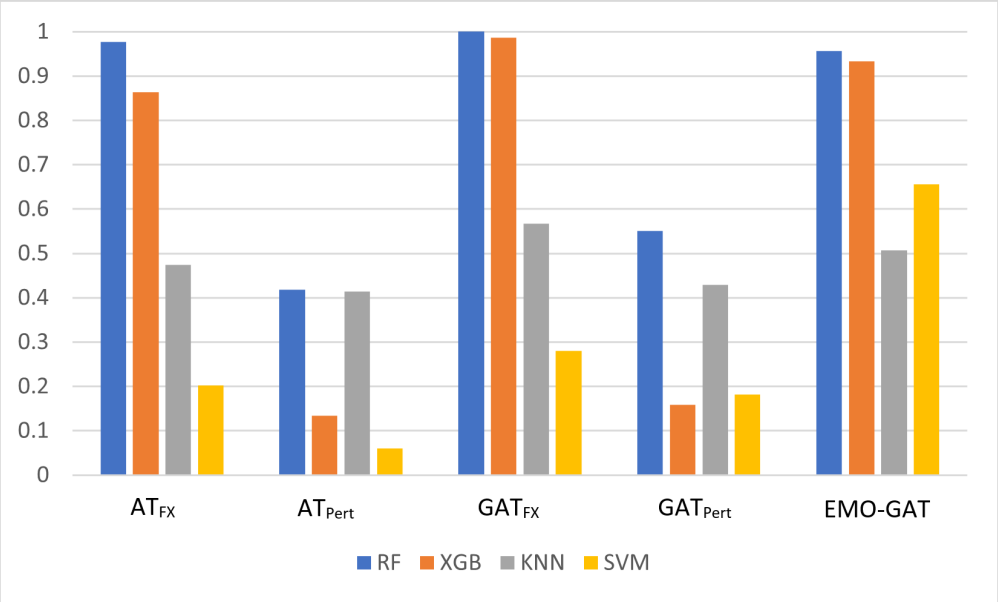

Figure 7.3: Average FX Accuracy

by at most 81% and at minimum 13%. This is significant because based on the baseline results, the $\text{Adversarial}_{\text{Pert}}$ examples are the most difficult to accurately detect.

The results presented suggest that the proposed extension, EMO-GAT, can perform nearly as well and better than the compared AT methods within each objective. However, this does not consider the most important feature of EMO-GAT, multi-objective optimization.

The only cases where EMO-GAT's performance was slightly lower than the other AT methods was when the respective accuracy measure was the same as the adversarial examples used in the AT. For example, consider the performance of GAT_{FX} and EMO-GAT when using the RF classifier and classifying $\text{Adversarial}_{\text{FX}}$ examples. GAT_{FX} out performs EMO-GAT by approximately 5%. However, if one were to examine the performance of GAT_{Pert} and EMO-GAT when using the RF classifier and classifying $\text{Adversarial}_{\text{Pert}}$ examples, a different result occurs. EMO-GAT out performs GAT_{Pert} by nearly 14% even when GAT_{Pert} is optimized on a single objective. This also alludes to the transfer of knowledge that can be gained from conducting AT on multiple types of adversarial examples. Such that, when a model is trained on a single type of adversarial example, the classification accuracy significantly increases when generating predictions other types of adversarial examples.

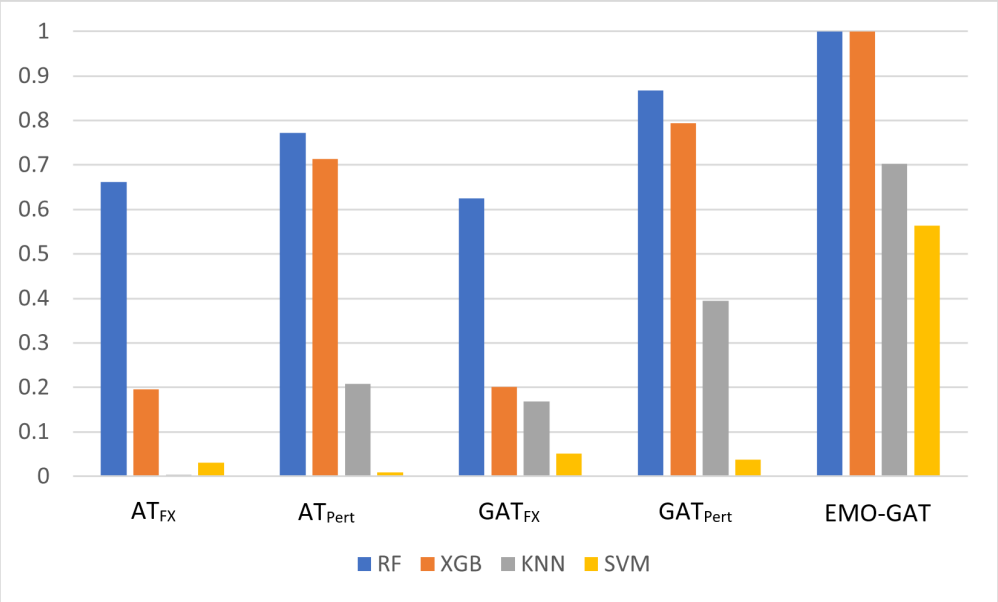

Figure 7.4: Average Pert Accuracy

7.7 Discussion

7.7.1 Performance Significance

To determine the top performers with respect to the results, the non-domination principle is applied to determine how the results compare when considering multiple objectives. In this analysis, only the adversarial example classification performance is considered. The Baseline accuracy is not included because it is not an objective that was sought to be maximized. Rather, the goal was to avoid a degradation in accuracy when classifying non-adversarial data. Because of this and the fact that including the Baseline accuracy would undermine the results of a non-domination sort, it is excluded from consideration. By conducting this type of analysis, the top performing solutions with respective to both objectives can be determined. Table 7.2 presents the top 5 fronts resulting from the non-domination sort. The first column represents the front and the second column presents the front distribution with respect to the AT method. Within the table, the first front represents the pareto-optimal set of solutions. Meaning that the solutions in the first front are no worse than any other individuals with respect to any objective and better in at least one objective. This principal holds true for the following fronts with respect to the preceding front. Fig. 7.5 depicts a graphical overview of the established fronts.

Table 7.2: Non-domination Sort Results

Front	Solutions
1	EMO-GAT$_{RF}$
2	EMO-GAT$_{XGB}$
3	AT$_{FX\text{-}RF}$, GAT$_{FX\text{-}RF}$, GAT$_{Pert\text{-}RF}$
4	AT$_{Pert\text{-}RF}$, GAT$_{FX\text{-}XGB}$, GAT$_{Pert\text{-}XGB}$, EMO-GAT$_{KNN}$, EMO-GAT$_{SVM}$
5	AT$_{FX\text{-}XGB}$, AT$_{Pert\text{-}XGB}$, GAT$_{Pert\text{-}KNN}$

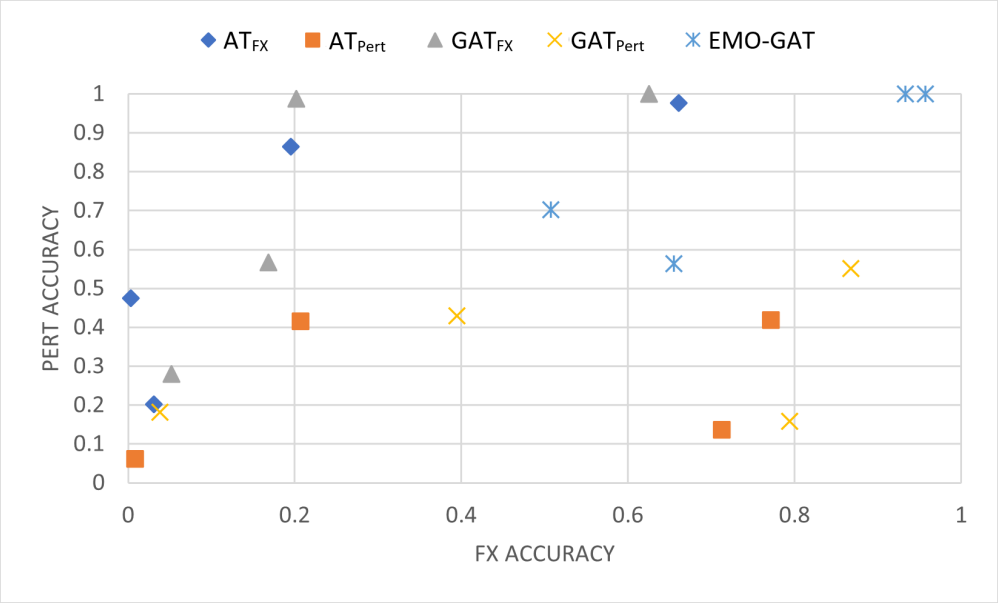

Figure 7.5: Front Overview

Notice, that within the top 5 fronts, solutions using EMO-GAT are more prevalent. Indicating that, EMO-GAT achieves the best trade between the two objectives independent of the classifier used within the AT routine. This is significant because the RF and XGB classifiers tend to have optimal performance correctly classifying adversarial examples. However, these optimal classifiers are dominated by lower performing solutions when using the EMO-GAT method.

7.7.2 Quantity of Adversarial Examples

One of the challenges associated with AT is determining an ideal quantity of adversarial examples to use. This issue is compounded when dealing with multiple types of adversarial examples. As previously discussed, having too many adversarial examples can induce label

Table 7.3: EMO-GAT Selected Adversarial Examples

Classifier	FX Examples	Pert Examples
RF	76.633 ± 8.060	87.900 ± 6.085
XGB	79.400 ± 4.841	89.033 ± 6.770
KNN	84.800 ± 6.306	80.900 ± 4.826
SVM	129.133 ± 4.410	132.000 ± 4.435

Table 7.4: Classifier Threat Modeling

	Pre-Adversarial Training		Post-Adversarial Training	
Classifier	FX Examples	Pert Examples	FX Examples	Pert Examples
RF	Med	High	Low	Low
XGB	High	High	Low	Low
KNN	High	High	Low	Med
SVM	High	High	Med	Med

leaking and hinder classifier performance on clean input. However, EMO-GAT enables discovering an ideal number of adversarial examples to use within the AT routines for each adversarial example type. The ideal quantity is implicitly discovered through the evolutionary search that EMO-GAT performs. The quantity of selected adversarial examples are presented in Table 7.3. The first column represents the specific classifier. The following columns present the average number and standard deviation of the quantity of selected adversarial examples with respect each adversarial example type. Notice that the average quantity of selected adversarial examples for the SVM classifier are significantly higher when compared to the other classifiers. The SVM also had a significant drop in the baseline performance. These two behaviors allude to the presence of label leaking. As previously discussed, this behavior can be mitigated by introducing a more severe or advanced penalty function. One can also see, that on average, the quantity of the $\text{Adversarial}_{\text{pert}}$ examples are selected in larger quantities than the $\text{Adversarial}_{\text{FX}}$ adversarial examples. This can be attributed to the $\text{Adversarial}_{\text{Pert}}$ examples being more effective at misleading the classifiers and requiring more data to better generalize the distribution.

7.7.3 Threat Modeling

Threat modeling is the process of identifying, quantifying, and analyzing potential threats to specific systems [87, 88]. A threat can be described as a vulnerability or point of weakness that

can be exploited by an adversary. By conducting an introspective evaluation of ML systems, threats can be identified, mitigated, and remediated. EMO-GAT facilitates this process by enabling constructing a trade-off between multiple objectives when posturing a ML system. For example, given the generated adversarial examples and the classifier performance both before and after EMO-GAT application, the potential attack surface can be analyzed.

Table 7.5 presents a threat model of the generated adversarial examples and their respective metrics. Each of the metrics are derived from the creation method, required knowledge, and the threat for each type of adversarial example. The first column represents the specific metric that is being evaluated. The following columns present the respective adversarial example types. Each of the metrics are quantified with a low, medium, or high value. The generation cost represents the computational cost required to generate the respective adversarial examples. The $\text{Adversarial}_{\text{FX}}$ examples are assigned a low generation cost value due to the simple approach needed for generation. While the $\text{Adversarial}_{\text{Pert}}$ examples have a high generation cost assigned due to using an evolutionary strategy to generate the examples and resulting in a significant computational cost. The required knowledge represents how much information an adversary would need to successfully create the respective adversarial examples. Both, adversarial example types have a medium value due to requiring access to the trained classifier and the dataset. The threat, represents how vulnerable the classifiers were to the respective adversarial example types prior to AT. Both adversarial example types are given a high threat value due to the poor classification performance across the majority of the classifiers. The average accuracy performance (A_{Avg}) ranges for the assigned threat values are as follows:

$$\text{Threat} = \begin{cases} \text{High} & \text{if } A_{\text{Avg}} \leq 45 \\ \text{Med} & \text{if } 45 < A_{\text{Avg}} \leq 70 \\ \text{Low} & \text{if } A_{\text{Avg}} > 70 \end{cases}$$

Through analyzing the classifiers both before and after AT, the residual threat for each classifier with respect to the adversarial example type can be determined. Table 7.4 presents the classifier threat modeling both before and after conducting AT using the EMO-GAT method.

Table 7.5: Adversarial Example Threat Modeling

Type	FX Examples	Pert Examples
Generation Cost	Low	High
Required Knowledge	Med	Med
Threat	Med	High

The first column presents the specific classifier. The following columns present the threat assessment of the respective adversarial example types both before and after conducting AT using EMO-GAT. When examining the overall classifier threat modeling, one can notice that for all classifiers, the threat is substantially reduced following EMO-GAT AT. Specifically, all the high and medium threats were remediated following EMO-GAT application. Additionally, one can also determine which classifiers are more robust by inspecting the shift in the classifier attack surface. For example, the RF classifier had a medium and high threat assessment for the respective adversarial example types. Following EMO-GAT AT, both adversarial example types were reduced to a low threat value. While this behavior is consistent across the majority of the classifiers, the RF was slightly more robust when compared to the other classifier performance prior to AT. This slight difference may indicate that the RF classifier is better at generalizing the adversarial example distributions when compared the other classifiers. One can also see that some classifiers still maintain a sizeable attack surface and thereby are more vulnerable to attack.

7.8 Summary

In this chapter, we presented an extension of GAT, EMO-GAT and demonstrate its efficacy by comparing its performance against several AT methods. The results suggest that EMO-GAT is an effective method for conducing AT and outperforms the compared AT methods in promoting a more robust model. The results also suggest that EMO-GAT can incorporate and discover an ideal trade off between multiple types of adversarial examples, discover an ideal quantity of adversarial examples for AT, and facilitate the process of threat modeling.

Our future work will be dedicated towards improving the performance of EMO-GAT. This includes experimenting with different hyper-parameters within the EA routine and exploring more advanced penalty functions.

Chapter 8

An Extended Study of Social Network Messages During the COVID-19 Infodemic: Salient Features and the Propagation of Information Types

8.1 Introduction

Machine learning requires large quantities of quality data in order to develop an unbiased and well representative model. One challenge associated with this is acquiring data that is labeled with a high degree of fidelity and consistency. Coupled with the requirements for a large amount of data, the problem becomes more severe. The immediate objective for this work is to expand on our previous approach for generating labels on a large collection of Tweets. The secondary objective is to reevaluate the our previous research questions on a broader diversity of Tweet content.

8.2 Auto-Labeler

Multi-objective optimization is combined with feature-selection to improve upon previous efforts in constructing an auto-labeler. This combined strategy will be referred to as a $\text{NSGA}_{\text{GEFeS}}$ hybrid. The multi-objective strategy used is the non-dominated sorting genetic algorithm II [36] (NSGA-II) while the feature selection method used is genetic and evolutionary feature selection (GEFeS). The problem of labeling the collected Tweets is broken down into two phases. In the first phase, Tweets whose content falls outside of the desired classes are identified and removed. This non-relevant Tweet content will be referred to as non-essential information. In the second phase, Tweets are labeled in accordance to the defined classes. Both phases involve

iteratively evolving machine learning models by proving batches of labeled Tweets and reinforcing the model's predictions on unlabeled Tweets. Following the completion of both phases, a pipeline will be created to ingest Tweet content and generate labels.

In both phases NSGA-II is used to maximize the models accuracy in correctly classifying the Tweets and the Cohen's kappa coefficient [89, 90]. Accuracy was chosen as an objective to maximize due to the metric representing how well the model is at identifying relationships and patterns. While Cohen's kappa was selected due to the type of problem being solved. Cohen's kappa coefficient is a statistical measure of inter-rate agreement between two raters that is corrected for how often that the raters may agree by chance [91]. This metric is of relevance and value due to having both the predicted and actual labels serving as raters. By using this metric, the models can learn to exhibit a higher agreement between both series of data and thereby improve model performance. Cohen's kappa coefficient can be represented in several ranges that depict the quality of rater agreement. Table 8.1 presents the value ranges for the Cohen's kappa values and the corresponding strength of agreement.

Table 8.1: Cohen's Kappa Interpretation

Cohen's Kappa	Strength of Agreement
0.00 - 0.20	Poor
0.21 - 0.40	Fair
0.41 - 0.60	Moderate
0.61 - 0.80	Substantial
0.80 - 1.00	Almost Perfect

8.2.1 Data Processing

To process the Tweets and generate the features is broken down into two stages: processing and transforming the Tweet content and extracting the features.

In the first stage, the Tweet content is processed and transformed. The Tweet content includes the actual Tweet and the associated metadata. The metadata includes contextual information about the Tweet and the author (user) including: user followers, user following,

isVerified, isRetweet, retweet count, Tweet length, hasUrl, and hasHasTag. The metadata is processed to extract several features and the larger valued features such as the user followers, user following, and retweet count are log transformed. Next, the Tweet is processed to remove hashtags, non-ascii characters, user tagging, and URLs. There are 11 features generated in this stage.

In the second stage, the features are extracted using several frameworks: Syntax-Aware LexicaL Emotion Engine (SALLEE) [92], Cognition [93], Drives [94], Social Dynamics [95], Temporal and Orientation [96], Toxicity [97], Needs and Values [98], and Linguistic Inquiry and Word Count (LIWC) [38]. SALLEE is a framework that detects emotions and sentiment expressed in text to generate emotional scores. The Cognition framework examines text to quantify aspects of cognitive processing and analytical thinking to analyze how people process information and make decisions. The Drives framework provides insight into motivational or drive indicators within text. After combining all the features, 176 features are generated in this stage.

The features extracted in the both stages are combined to create the processed dataset. The processed dataset consists of 187 features for each processed Tweet.

8.2.2 Phase 1: Removing Non-essential Information

In the first phase, the non-essential information is identified and removed. This is accomplished by iteratively training a $\text{NSGA}_{\text{GEFeS}}$ hybrid model to identify the non-essential information. Each iteration (batch) involves evaluating the $\text{NSGA}_{\text{GEFeS}}$ hybrid performance on a batch of 250 Tweets. The $\text{NSGA}_{\text{GEFeS}}$ hybrid's performance is evaluated from 2 different perspectives: performance on the test set and the performance on Tweets sampled from the unlabeled dataset. The reason a second 'test' set is utilized is due to the complex and diverse feature space. The second 'test' set serves to objectively measure how the auto-labeler can be expected to perform when actually deployed. A baseline is also created to compare the advantages in using the $\text{NSGA}_{\text{GEFeS}}$ hybrid method. The baseline represents a vanilla approach and evaluates the model's performance with the absence of any auxiliary training methods such as feature selection.

As previously mentioned, the iterative approach to building the $NSGA_{GEFeS}$ hybrid model is achieved through multiple iterations or batches. A batch is the process of sampling 250 Tweets from the dataset, processing the Tweets, building and evaluating model performance, and feeding the labeled data into the following batches. Meaning that, the overall labeled dataset grows by 250 Tweets for each batch completed. For the baseline, the performance is measured by simply splitting the labeled dataset into an 80/20 split and training an XGB [57] model. This routine is completed for 30 runs and serves a comparative baseline for each batch. The routine to train, evaluate, and select a $NSGA_{GEFeS}$ hybrid for a single run is as follows.

Step 1: Sample 250 Tweets from the dataset. These tweets are processed, manually labeled, and are added to the current batch's dataset.

Step 2: Split the batch's dataset into a training and a test set using an 80/20 split.

Step 3: Create a population consisting individuals where each individual consists of a feature mask that will be applied to the training dataset to create a reduced training set in accordance with the GEFeS routine.

Step 4: Measure the fitness of each individual in the population. The fitness is calculated by training a XGB model with the reduced training set and generating predictions on the test set. The fitness measure consists of the model's accuracy and Cohen's kappa value.

Step 5: Domination sort the population into fronts, such that, each front is only dominated by the preceding front in accordance with the NSGA-II routine.

Step 6: Select parents using binary tournament selection and generate offspring using uniform crossover.

Step 7: Replace the worst performing individuals in the population with the offspring.

Step 8: Repeat until reaching the stopping condition.

For each batch, the $NSGA_{GEFeS}$ hybrid model is evaluated for 30 runs at 4K, 8K, and 16K function evaluations (FE). Meaning that, in each batch where a $NSGA_{GEFeS}$ hybrid model

routine is completed, there exists 30 collections of evolved solutions for each number of FEs. To select the best model, the collection of solutions are combined and domination sorted. By combining and domination sorting the evolved solutions, the better performing solutions with respect to each objective can be identified.

Tables 8.2 and 8.3 present the performance across each batch for the first phase. Specifically, Table 8.2 lists the performance on the test sets while Table 8.3 shows the performance for the sampled collections of 250 Tweets. In both tables, the first column represents the specific batch. The next two columns represent the performance of the vanilla approach over 30 runs with respect to accuracy and Cohen's kappa. The final two columns represent the performance of the 30 best individuals from the $\text{NSGA}_{\text{GEFeS}}$ hybrid runs with respect to accuracy and Cohen's kappa value. In the performance columns, the value outside of the parentheses denotes the highest value achieved for the associated metric. The values within the parentheses are the respective metric averaged over 30 runs/individuals and the standard deviation.

Notice that in both tables a general trend can be observed. With respect to the average accuracy, the Baseline test set performance and prediction performance both tend to follow an upward trend as the batches progress. While the $\text{NSGA}_{\text{GEFeS}}$ hybrid appears to be follow an inverted trend where the testing accuracy is decreasing while the prediction accuracy is increasing. The trend with the Baseline can attributed to the natural biases that are exhibited by machine learning algorithms. Due to the small amount of training data, effective generalization are difficult to achieve. This results in causing models to a lean towards simplistic learning strategies. This simplistic approach or naive behavior continues until better generalizations can be achieved with more data. On the other hand, the $\text{NSGA}_{\text{GEFeS}}$ hybrid exhibits an interesting behavior. This inverted trend may be attributed to how genetic algorithms iterate through a search space and more towards better solutions. Given the lack of training data, the $\text{NSGA}_{\text{GEFeS}}$ hybrid will also be driven towards simpler generalization approaches, but the performance will be optimized due to the nature of genetic algorithms. For example, given an strongly imbalanced dataset, simple models will tend to heavily favor the large class. However, with the inclusion of more auxiliary methods, more advanced decisions can be learned that also include favoring the larger class. Thereby, as the $\text{NSGA}_{\text{GEFeS}}$ hybrid receives more training data, it

Table 8.2: Phase 1: Test Performance

	Baseline		$NSGA_{GEFeS}$ Hybrid	
	Accuracy	Cohen's Kappa	Accuracy	Cohen's Kappa
Batch 1	0.720 (0.621 ± 0.062)	0.406 (0.199 ± 0.136)	0.960 (0.909 ± 0.023)	0.915 (0.807 ± 0.045)
Batch 2	0.780 (0.700 ± 0.044)	0.499 (0.331 ± 0.093)	0.890 (0.870 ± 0.013)	0.770 (0.710 ± 0.029)
Batch 3	0.773 (0.702 ± 0.039)	0.437 (0.301 ± 0.084)	0.867 (0.841 ± 0.013)	0.694 (0.634 ± 0.027)
Batch 4	0.775 (0.732 ± 0.027)	0.446 (0.307 ± 0.073)	0.895 (0.843 ± 0.016)	0.711 (0.592 ± 0.037)
Batch 5	0.768 (0.729 ± 0.019)	0.368 (0.289 ± 0.039)	0.856 (0.836 ± 0.010)	0.621 (0.564 ± 0.022)

Table 8.3: Phase 1: 250 Tweet Prediction Performance

	Baseline		$NSGA_{GEFeS}$ Hybrid	
	Accuracy	Cohen's Kappa	Accuracy	Cohen's Kappa
Batch 1	0.704 (0.665 ± 0.020)	0.333 (0.241 ± 0.047)	0.720 (0.647 ± 0.027)	0.380 (0.218 ± 0.057)
Batch 2	0.720 (0.683 ± 0.019)	0.343 (0.259 ± 0.045)	0.712 (0.679 ± 0.019)	0.352 (0.252 ± 0.047)
Batch 3	0.796 (0.766 ± 0.017)	0.404 (0.336 ± 0.042)	0.808 (0.773 ± 0.016)	0.436 (0.350 ± 0.042)
Batch 4	0.768 (0.742 ± 0.013)	0.326 (0.241 ± 0.036)	0.798 (0.754 ± 0.017)	0.332 (0.250 ± 0.045)
Batch 5	0.798 (0.752 ± 0.018)	0.311 (0.232 ± 0.048)	0.820 (0.771 ± 0.018)	0.352 (0.218 ± 0.061)

becomes more difficult to generalize. However, the information learned is effectively applied and is exhibited when generating predictions on the sampled Tweets. This trend suggests that the $NSGA_{GEFeS}$ hybrid models are effectively able to generalize on diverse samples of Tweets and can serve as auto-labelers.

The performance of the best performing $NSGA_{GEFeS}$ hybrid model is shown in Figs 8.1 and 8.2. This model was chosen because it exhibits the best performance while having a sizable amount of training data. This model will be used in the completed pipeline to identify Tweets that consist of non-essential information and flag them for removal.

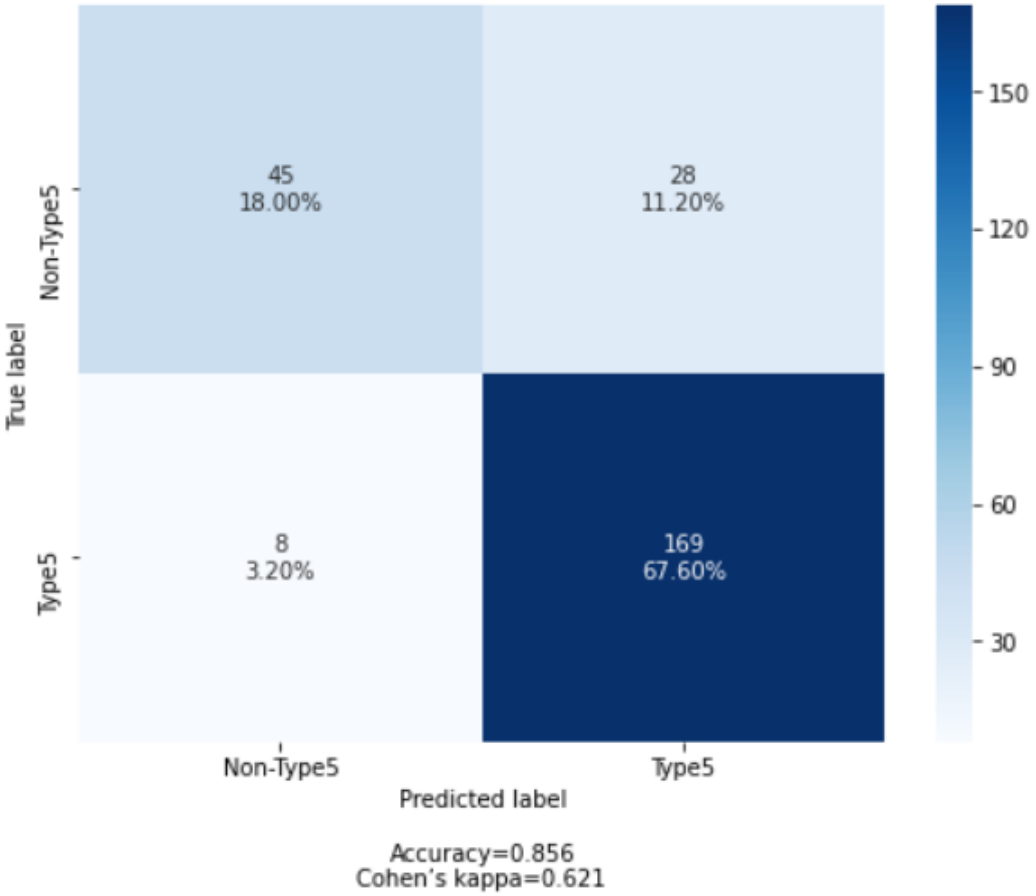

Figure 8.1: Phase 1: Auto Labeler Test Accuracy

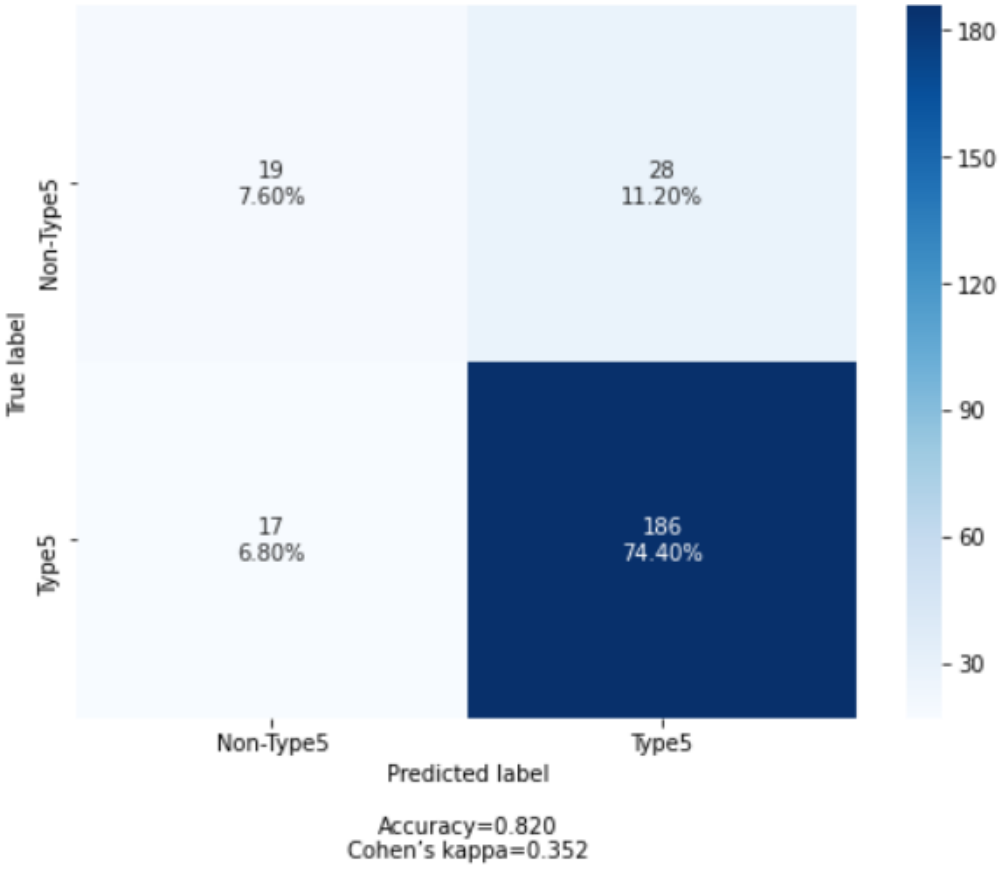

Figure 8.2: Phase 1: Auto Labeler Prediction Accuracy

8.2.3 Phase 2: Classifying Tweets

In the second phase, the Tweets are assigned a label. Similar to the first phase, this is accomplished by iteratively training a $\text{NSGA}_{\text{GEFeS}}$ hybrid model to classify Tweets by information type. The procedure is largely the same except that there are four classes to classify. However, due to having multiple classes and a larger data space to generalize, several different techniques were explored to improve the classification performance. The techniques include several different applications of ensemble learning [99, 100] techniques. Each technique is evaluated following the execution of each batch. Meaning that, the ensemble learning techniques include only the 30 best individuals from the $\text{NSGA}_{\text{GEFeS}}$ hybrid routine. The first ensemble learning technique applied is voting. Two voting methods are explored including: hard voting [101] and weighted voting [102]. Hard voting entails having each model in the voting scheme generate predictions and using a majority rule to determine the final predicted labels. While weighted

voting involves weighting the votes from the models in the voting scheme such that the importance given to a vote is proportional to the model's classification success. In this application, the weights used are the test accuracies associated with each model. Finally, the last technique is a feature aggregation method. The underlying premise is that each model was evolved and underwent an evolutionary process to identify essential features with different subsets of training data. Because of this, each model's feature mask represents the ideal features for different areas of the search space. By aggregating these feature masks, a broader feature mask can be identified that incorporates the ideal features from each of the model's observed search spaces. The aggregated feature mask approach involves taking the 30 best individuals from the $\text{NSGA}_{\text{GEFeS}}$ hybrid routine and analyzing the frequency each feature selection. From this, the top 50% features are retained and used to train a new XGB model. This new model is evaluated over 30 runs using the created feature mask.

Tables 8.4, 8.5, and 8.6 outline the performance of the models across the different batches for the second phase. Specifically, Table 8.4 presents the performance on the test sets while Tables 8.5 and 8.6 lists the performance on for the sampled collections of 250 Tweets. Similar to the first phase, in each table the performance representation for the model is accuracy and the Cohen's kappa value. Again, the Baseline measures the performance of an XGB model when trained on the respective batch's training set over 30 runs. The performance of the $\text{NSGA}_{\text{GEFeS}}$ hybrid includes the 30 best solutions from an aggregation of the 30 runs for each of the function evaluation amounts. Differing from the previous phase, the performance of the three ensemble based approaches are also presented. It's worth noting that due to both voting methods using a collection of pre-trained models and combining each of the models decisions, only a single performance is measured from the generated predictions.

From the Phase 2 results several observations can be made. First, one can note that the same performance trends observed in Phase 1 are also generally present in Phase 2. However, the upward performance trend with respect to average accuracy is lost after several Batches. This behavior can be attributed to the more complex problem being solved. Phase 2 has four classes while Phase has only 2. The increase in problem complexity makes it harder for the

Table 8.4: Phase 2: Test Performance

	Baseline		$NSGA_{GEFeS}$ Hybrid		Aggregation	
	Accuracy	Cohen's Kappa	Accuracy	Cohen's Kappa	Accuracy	Cohen's Kappa
Batch 1	0.548 (0.505 ± 0.022)	0.336 (0.274 ± 0.031)	0.900 (0.832 ± 0.027)	0.851 (0.753 ± 0.038)	0.660 (0.545 ± 0.064)	0.376 (0.273 ± 0.039)
Batch 2	0.584 (0.549 ± 0.021)	0.391 (0.341 ± 0.030)	0.830 (0.768 ± 0.019)	0.749 (0.657 ± 0.027)	0.670 (0.553 ± 0.047)	0.508 (0.345 ± 0.067)
Batch 3	0.628 (0.595 ± 0.020)	0.448 (0.401 ± 0.030)	0.807 (0.753 ± 0.018)	0.715 (0.641 ± 0.025)	0.647 (0.577 ± 0.048)	0.486 (0.383 ± 0.070)
Batch 4	0.635 (0.594 ± 0.022)	0.462 (0.402 ± 0.033)	0.760 (0.737 ± 0.009)	0.644 (0.613 ± 0.012)	0.665 (0.602 ± 0.034)	0.509 (0.412 ± 0.050)
Batch 5	0.696 (0.597 ± 0.032)	0.556 (0.404 ± 0.048)	0.752 (0.730 ± 0.011)	0.634 (0.602 ± 0.016)	0.652 (0.594 ± 0.030)	0.489 (0.400 ± 0.045)
Batch 6	0.650 (0.598 ± 0.021)	0.484 (0.405 ± 0.031)	0.753 (0.722 ± 0.011)	0.636 (0.589 ± 0.016)	0.650 (0.601 ± 0.027)	0.480 (0.407 ± 0.038)
Batch 7	0.654 (0.603 ± 0.031)	0.486 (0.404 ± 0.049)	0.751 (0.713 ± 0.012)	0.630 (0.572 ± 0.018)	0.649 (0.605 ± 0.024)	0.475 (0.410 ± 0.036)

Table 8.5: Phase 2: 250 Tweet Prediction Performance

	Baseline		Hybrid		Aggregation	
	Accuracy	Cohen's Kappa	Accuracy	Cohen's Kappa	Accuracy	Cohen's Kappa
Batch 1	0.640 (0.527 ± 0.056)	0.472 (0.315 ± 0.075)	0.560 (0.492 ± 0.026)	0.355 (0.257 ± 0.038)	0.576 (0.503 ± 0.026)	0.376 (0.273 ± 0.039)
Batch 2	0.630 (0.550 ± 0.043)	0.447 (0.341 ± 0.062)	0.580 (0.530 ± 0.025)	0.387 (0.315 ± 0.036)	0.588 (0.543 ± 0.021)	0.398 (0.332 ± 0.031)
Batch 3	0.667 (0.590 ± 0.037)	0.513 (0.403 ± 0.052)	0.636 (0.589 ± 0.021)	0.462 (0.391 ± 0.031)	0.648 (0.601 ± 0.020)	0.477 (0.407 ± 0.030)
Batch 4	0.604 (0.569 ± 0.019)	0.409 (0.358 ± 0.028)	0.596 (0.561 ± 0.017)	0.400 (0.348 ± 0.026)	0.604 (0.560 ± 0.021)	0.414 (0.347 ± 0.030)
Batch 5	0.596 (0.544 ± 0.020)	0.396 (0.318 ± 0.030)	0.584 (0.544 ± 0.019)	0.378 (0.316 ± 0.029)	0.596 (0.549 ± 0.022)	0.399 (0.324 ± 0.033)
Batch 6	0.612 (0.579 ± 0.019)	0.393 (0.349 ± 0.027)	0.628 (0.573 ± 0.021)	0.426 (0.338 ± 0.032)	0.608 (0.575 ± 0.018)	0.382 (0.344 ± 0.026)
Batch 7	0.612 (0.572 ± 0.016)	0.416 (0.356 ± 0.025)	0.608 (0.566 ± 0.018)	0.407 (0.346 ± 0.028)	0.608 (0.575 ± 0.019)	0.411 (0.360 ± 0.028)

models to effectively generalize. Aside from the Baseline, each performance evaluation is related to the $NSGA_{GEFeS}$ hybrid models. Thereby, when comparing the Baseline to the other performance representations, it suggests that on average, is not effective in solving this problem. However, due to the baseline also exhibiting the same performance trend, it suggests that the utilized feature representations may not be sufficient for this problem. To address this, more advanced feature representations can be explored.

The final model chosen as Phase 2's auto labeler is the $NSGA_{GEFeS}$ hybrid model from batch 6. This model is chosen because it offers the best performance when generating predictions on the sampled Tweets while having a substantial strength of agreement based on the Cohen's kappa value. The performance of the chosen $NSGA_{GEFeS}$ hybrid model is shown in

Table 8.6: Phase 2: 250 Tweet Prediction Performance with Voting

	Weighted Voting		Hard Voting	
	Accuracy	Cohen's Kappa	Accuracy	Cohen's Kappa
Batch 1	0.496	0.258	0.496	0.258
Batch 2	0.548	0.339	0.548	0.339
Batch 3	0.620	0.437	0.620	0.437
Batch 4	0.556	0.341	0.552	0.337
Batch 5	0.572	0.356	0.576	0.362
Batch 6	0.58	0.342	0.580	0.342
Batch 7	0.588	0.379	0.588	0.379

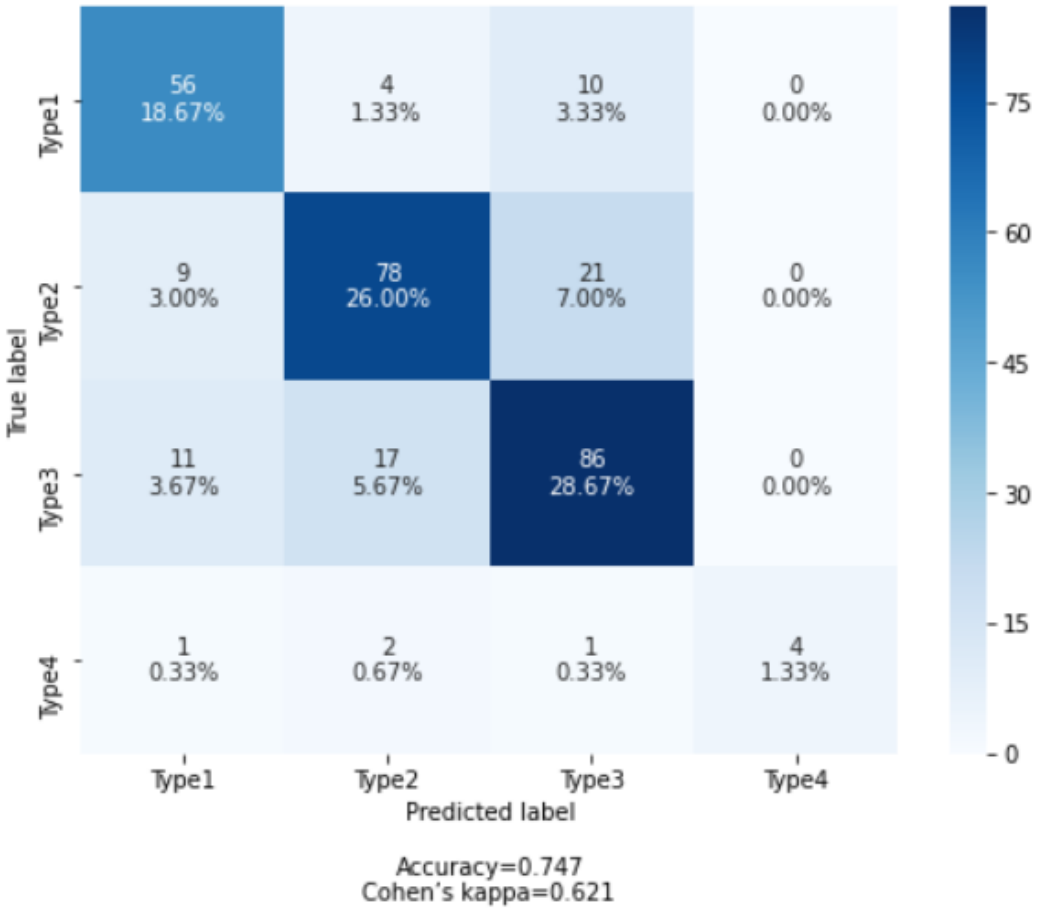

Figure 8.3: Phase 2: Auto Labeler Test Accuracy

Figs 8.3 and 8.4. This model is used in the completed pipeline to classify Tweets with respect to an information type.

8.2.4 Final Pipeline

Following the completion of the auto-labeler construction in both phases, the pipeline is created to ingest and process Tweets. Using the pipeline approximately 36K Tweets were ingested and auto-labeled. From the 36K Tweets, approximately 33K Tweets were identified as non-essential information and discarded. Thereby, the final collection of labeled Tweets processed from the pipeline is 3,666 Tweets. Combining this with the datasets built from the batches, the final dataset contains 5,664 Tweets. Fig 8.5 presents an overview of the class distribution of the final collection of labeled Tweets.

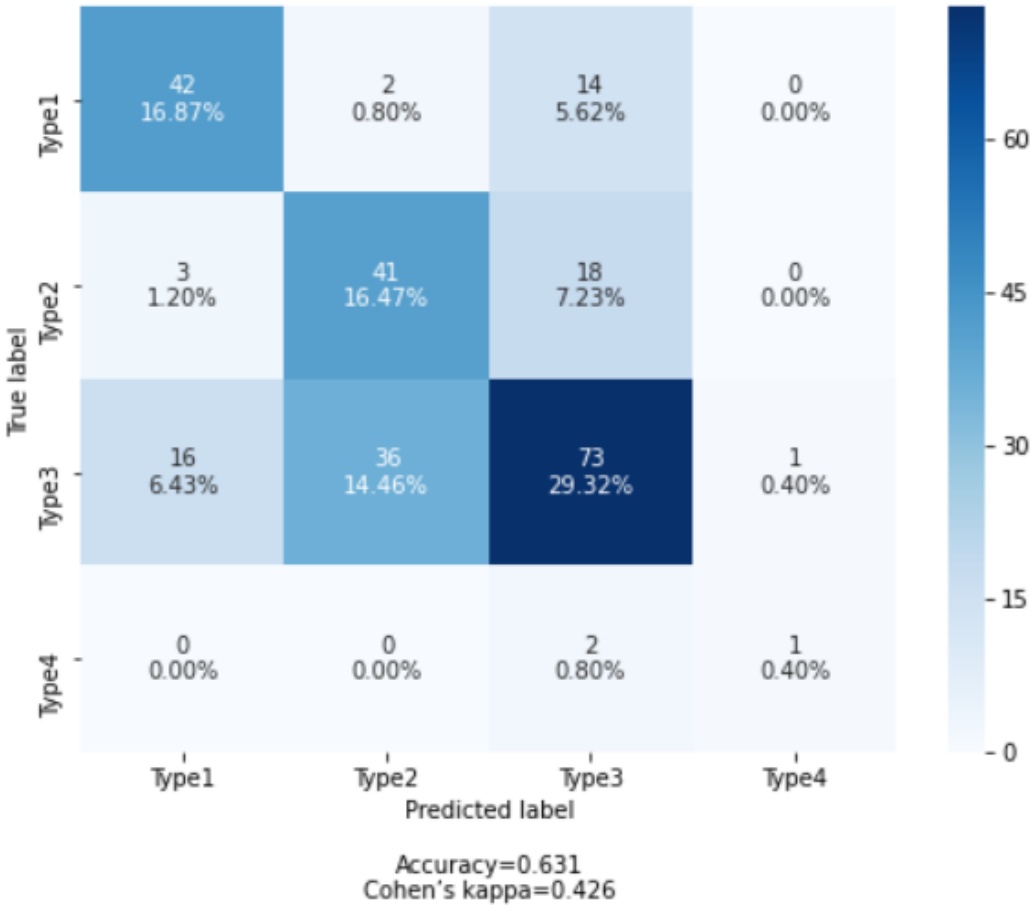

Figure 8.4: Phase 2: Auto Labeler Prediction Accuracy

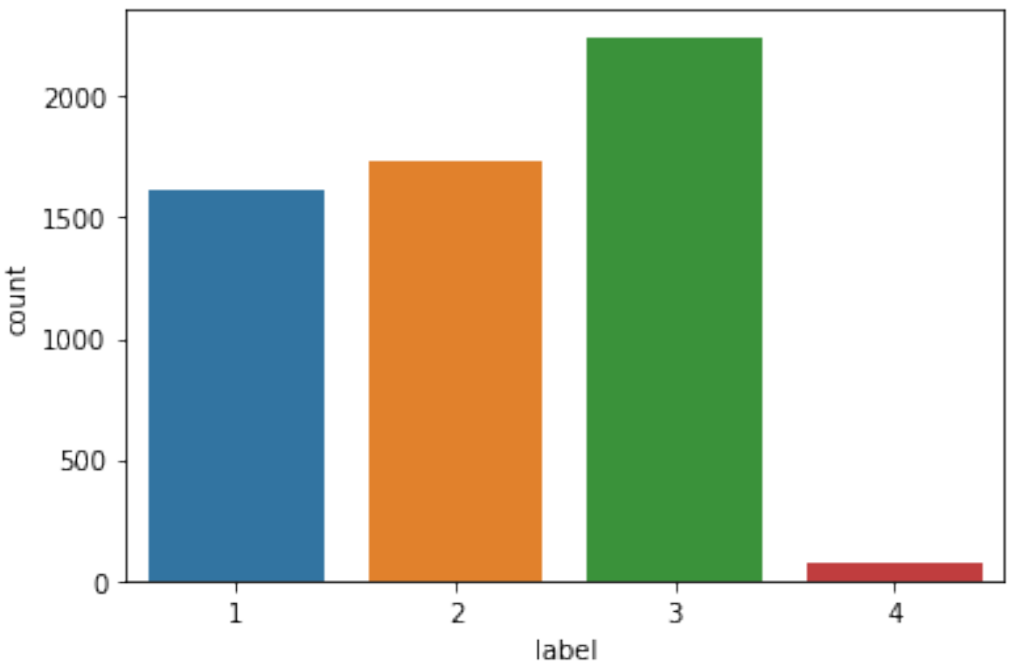

Figure 8.5: Final Dataset Class Distribution

Table 8.7: Extended Top 10 Salient Features

Feature Index	Feature	Representation	Saliency
105	she_he	Linguistic	80%
157	focus_future	Temporal and Orientation	76%
176	periods	Linguistic	76%
46	social	Personality	73%
67	tentative	Cognitive	73%
74	goodfeel	Emotions	73%
145	health	Linguistic	73%
184	apostrophes	Linguistic	73%
11	extraversion	Personality	70%
34	anxiety_prone	Personality	70%

8.3 Results

8.3.1 Question #1: What are the most salient features for classifying social media messages?

To determine the most salient features for classifying social media messages, the $NSGA_{GEFeS}$ hybrid results from Phase 2 Batch 6 can be used. Previously, it was discussed that for each batch the $NSGA_{GEFeS}$ hybrid run results are aggregated and the best individual chosen. However, instead of taking the best individual, the 30 best performing individuals can be analyzed. Each individual consists of an evolved feature masks that contributes to better performance when generating predictions. By analyzing the collection of feature masks, the salient features that contribute to classifying social messages can be derived. Table 8.7 shows the top 10 features used to classify social messages by type. In the table, the first column represents the feature index. The second column represents the specific feature. The third column represents the features respective category. Finally, the last column represents how consistently the feature was selected within the 30 best individuals.

When examining 8.7, one can note that the most common feature representation consists of linguistic features. This is of value because linguistic features do not tend to require an enriched corpus for information retrieval. Meaning that, simple text content contains ample enough information to successfully extract linguistic information.

Table 8.8: Extended Key Feature Summary

Types	Amount	Verified	Log Follower Amount	Log Following Amount	Retweet Count
Type 1 - Caution and Advice	1607	33 (2.5%)	10,554.77 (6.57)	10,668.70 (6.63)	9,360.04 (5.82)
Type 2 - Doubt and Criticism	1735	14 (0.8%)	11,325.73 (6.52)	11,922.45 (6.87)	12,502.39 (7.20)
Type 3 - Rumors and Counter-Rumors	2244	32 (1.42%)	14,767.22 (6.58)	15,358.71 (6.84)	15,768.01 (7.02)
Type 4 - Generic Harm	78	0 (0%)	530.24 (6.79)	542.24 (6.94)	531.16 (6.80)

8.3.2 Question #2: Which types of messages, on average, propagate the fastest?

Table 8.8 presents summary of key features with respect to information type. In Table 8.8 the first column presents the information type while the following columns present the metrics with the associated features. For the the second column, the value in parentheses represents the percentage of verified user. In the following columns, the value in parentheses denotes the average value for the associated metric. From the table, several observations can be made. First, Tweets consisting of Doubt & Criticism tend to be propagated more on average while Caution & Advice are propagated the least. It can also be seen that Rumors and Counter-Rumors are the most common information type and are propagated by users that have a large social following.

8.3.3 Question #3: Given an information type can one determine the characteristics that will increase its propagation across a social network?

Tables 8.9 - 8.12 present the results for conducting linear regression with with respect to information type to measure the impact that features have to the number of retweets. In each table, the first column shows the respective feature. The second column shows the feature category. While the last columns show the coefficient, standard error, and P-value resulting from the linear regression. The coefficient denotes the size of the effect that changes to the feature will cause to the number of retweets. For example, if there is a positive coefficient, increasing the associated feature can expect to result in a high number of retweets. The standard error represents how accurate the coefficient is where a lower error value denotes a higher accuracy. The P-value represents a measure of statistical significance where a P-value < 0.05 denotes

Table 8.9: Top 15 Salient Features Derived from Linear Regression of Type 1: Caution & Advice Tweets wrt Retweet Count (r-squared=0.689; adj r-squared=0.652)

Feature	Category	Coefficient	Std Err	P-value
has_hashtag	Twitter Feature	-0.7044	0.125	0
has_url	Twitter Feature	-0.7929	0.131	0
is_retweet	Twitter Feature	4.7596	0.205	0
user_is_verified	Twitter Feature	1.5554	0.433	0
six_plus_words	Linguistic	-3.9874	1.003	0
tweet_length	Twitter Feature	0.0052	0.001	0.001
clout	Linguistic	-0.0587	0.018	0.001
has_mentions	Twitter Feature	-0.5747	0.181	0.002
risk_seeking	Drives	-4.4296	1.43	0.002
power	Linguistic	0.0286	0.011	0.008
inward_focus	Social Dynamics	-0.7085	0.272	0.009
negations	Linguistic	0.0139	0.006	0.017
authentic	Linguistic	0.0217	0.01	0.031
social	Social Dynamics	-0.0179	0.008	0.032
achievement	Drives	0.0125	0.006	0.048

statistical significance. In each table the features are sorted with an ascending P-value and only the top 15 statistically significant features with the lowest standard error are presented.

In each of the tables, the top 15 salient features are shown in relation to the number of retweets. Inspecting this information enables identifying which features are negatively and positively correlated with the number of retweets with respect to the information type. By applying the results shown, messages can be crafted such that they promote successful propagation. For example, given a need to propagate Caution & Advice related messages, one can craft their message to optimize on the positively correlated features while minimizing the negatively correlated features. This may take the form of ensuring the message does not contain has tags, does not have any URLs, is a retweet, is originated from a verified user, does not contain mentions, and does not exhibit any risk seeking characteristics.

8.4 Summary

In this chapter, we have presented and demonstrated the capability and feasibility of developing and leveraging an auto-labeler to generate labeled data. Additionally, we have extended

Table 8.10: Top 15 Salient Features Derived from Linear Regression of Type 2: Doubt & Criticism Tweets wrt Retweet Count (r-squared=0.444; adj r-squared=0.384)

Feature	Category	Coefficient	Std Err	P-value
has_mentions	Twitter Feature	-0.4587	0.127	0
is_retweet	Twitter Feature	5.0052	0.479	0
risk_seeking	Drives	-3.9171	1.038	0
risk_aversion	Drives	3.9353	1.038	0
reward	Linguistic	3.9431	1.041	0
tweet_length	Twitter Feature	0.005	0.001	0.001
social	Personality	-0.0224	0.007	0.001
certainty	Cognition	9.8626	3.367	0.003
has_hashtag	Twitter Feature	-0.3677	0.129	0.004
user_followers	Twitter Feature	-0.1298	0.047	0.006
auxiliary_verbs	Linguistic	-8.5721	3.212	0.008
causation	Cognition	9.5792	3.637	0.009
boredom	Emotions	-8.425	3.502	0.016
discrepancies	Linguistic	8.2923	3.888	0.033
comparisons	Cognition	4.4221	2.234	0.048

Table 8.11: Top 15 Salient Features Derived from Linear Regression of Type 3: Rumors & Counter Rumors Tweets wrt Retweet Count (r-squared=0.495; adj r-squared=0.453)

Feature	Category	Coefficient	Std Err	P-value
has_url	Twitter Feature	-0.813	0.11	0
has_hashtag	Twitter Feature	-0.8497	0.118	0
has_mentions	Twitter Feature	-0.7177	0.122	0
is_retweet	Twitter Feature	6.4088	0.273	0
fear	Emotional	3.3837	0.844	0
personal_concerns	Linguistic	32.4812	5.987	0
adverbs	Linguistic	7.5905	2.339	0.001
user_followers	Twitter Feature	-0.1452	0.046	0.002
conjunctions	Linguistic	8.7859	3.18	0.006
curiosity	Emotional	-8.8921	3.22	0.006
auxiliary_verbs	Linguistic	7.5404	2.872	0.009
insight	Linguistic	11.1216	4.821	0.021
risk	Linguistic	-12.0883	5.339	0.024
six_plus_words	Linguistic	-2.0442	0.933	0.029
function_words	Linguistic	-5.3258	2.484	0.032

Table 8.12: Top 15 Salient Features Derived from Linear Regression of Type 4: Generic Harm Tweets wrt Retweet Count (r-squared=1.00; adj r-squared=1.00)

Feature	Category	Coefficient	Std Err	P-value
is_retweet	Twitter Feature	-1.613	0	0
has_hashtag	Twitter Feature	-4.0118	0	0
has_url	Twitter Feature	2.7717	0	0
has_mentions	Twitter Feature	2.6632	0	0
analytical_thinking	Cognition	-1.2483	0	0
sentiment	Emotions	1.3952	0	0
badfeel	Emotions	-0.6198	0	0
ambifeel	Emotions	0.514	0	0
anger	Emotions	2.2165	0	0
disgust	Emotions	1.7713	0	0
fear	Emotions	-1.2284	0	0
sadness	Emotions	1.283	0	0
analytical_thinking.1	Linguistic	-1.2483	0	0
clout.1	Linguistic	0.6075	0	0
emotional_tone	Linguistic	0.9483	0	0

our prior research on the propagation of information in social media by analyzing an unrestricted and broad diversity of Tweets. We identified the most salient features for classifying social media messages, evaluated which type of messages propagate the faster, and explored the relationship between Tweet characterises and propagation. Future work will be dedicated to exploring more advanced feature representations and improving the classification performance of the auto-labeler in Phase 2.

Chapter 9

Conclusion

In this work, several challenges relating to fake news detection have been addressed including the large feature space available for fake new detection related features, the adversarial effects on fake news classification systems, and the wide scale propagation of information.

To address the large feature space available for fake news detection systems, we have demonstrated the efficacy in using genetic feature selection to identify essential features that contribute to successful classification. By incorporating the ideal features, significant improvements to classification accuracy can be achieved while substantially reducing the number of required features. In order to protect fake news detection systems, two methods are proposed including GAT and EMO-GAT. Our work demonstrates the utility in defensively posturing machine learning systems using adversarial examples with GAT. Furthermore, the defensive posturing is strengthened when utilizing EMO-GAT to overcome the shortcomings inherent to GAT. Finally, we explore the propagation of information across social media to identify what message characteristics lead to classification by information type. Additionally, we examine social media messages to determine which information type propagates the fastest and what characteristics contribute to successful propagation.

References

[1] M. Smith, A. Richardson, B. Brown, G. Dozier, M. King, and J. Morris. A study of the impact of evolutionary-based feature selection for fake news detection. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1859–1865, 2020.

[2] Marcellus Smith, Brandon Brown, Gerry Dozier, and Michael King. Mitigating attacks on fake news detection systems using genetic-based adversarial training. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 1265–1271, 2021.

[3] Marcellus Smith, Brandon Brown, Gerry Dozier, and Michael King. Emo-gat: An evolutionary multi-objective method for adversarial training. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, 2021.

[4] David Lazer, Matthew Baum, Yochai Benkler, Adam Berinsky, Kelly Greenhill, Filippo Menczer, Miriam Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, Michael Schudson, Steven Sloman, C. Sunstein, Emily Thorson, Duncan Watts, and Jonathan Zittrain. The science of fake news. *Science*, 359:1094–1096, 03 2018.

[5] William Weir. *History's greatest lies: the startling truths behind world events our history books got wrong*. Crestline, 2018.

[6] Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31(2):211–36, May 2017.

[7] Kelly A. Stahl. Fake news detection in social media. 2018.

[8] Kay Michel, Marcellus Smith, Brandon Brown, Michael King, and Gerry Dozier. A study of social network messages during the covid-19 infodemic: Salient features and the propagation of information types. In *SoutheastCon 2021*, pages 1–8, 2021.

[9] Sabri M. Saidam. On route to an e-society: Human dependence on technology and adaptation needs.

[10] Douglas Ahlers. News consumption and the new electronic media. *Harvard International Journal of Press/Politics*, 11(1):29–52, 2006.

[11] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *SIGKDD Explor. Newsl.*, 19(1):22–36, September 2017.

[12] Przemyslaw M. Waszak, Wioleta Kasprzycka-Waszak, and Alicja Kubanek. The spread of medical fake news in social media â€“ the pilot quantitative study. *Health Policy and Technology*, 7(2):115 – 118, 2018.

[13] Xinyi Zhou and Reza Zafarani. Fake news: A survey of research, detection methods, and opportunities, 2018.

[14] J. C. S. Reis, A. Correia, F. Murai, A. Veloso, and F. Benevenuto. Supervised learning for fake news detection. *IEEE Intelligent Systems*, 34(2):76–81, 2019.

[15] Issa Traore and Sherif Saad. Detection of online fake news using n-gram analysis and machine learning techniques. pages 127–138, 10 2017.

[16] William Yang Wang. "liar, liar pants on fire": A new benchmark dataset for fake news detection, 2017.

[17] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale, 2017.

[18] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 372–387, 2016.

[19] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning, 2018.

[20] B. Brown, A. Richardson, M. Smith, G. Dozier, and M. C. King. The adversarial ufp/ufn attack: A new threat to ml-based fake news detection systems? In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1523–1527, 2020.

[21] Vivek B. S., Konda Reddy Mopuri, and R. Venkatesh Babu. Gray-box adversarial training, 2018.

[22] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

[23] Xabier Echeberria-Barrio, Amaia Gil-Lerchundi, Ines Goicoechea-Telleria, and Raul Orduna-Urrutia. Deep learning defenses against adversarial examples for dynamic risk assessment, 2020.

[24] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S. Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free!, 2019.

[25] Ninghao Liu, Mengnan Du, Ruocheng Guo, Huan Liu, and Xia Hu. Adversarial attacks and defenses: An interpretation perspective, 2020.

[26] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[27] Wentao Wang, Han Xu, Xiaorui Liu, Yaxin Li, Bhavani Thuraisingham, and Jiliang Tang. Imbalanced adversarial training with reweighting, 2021.

[28] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy, 2019.

[29] Matteo Terzi, Gian Antonio Susto, and Pratik Chaudhari. Directional adversarial training for cost sensitive deep learning classification applications, 2019.

[30] G. Dozier, K. Purrington, K. Popplewell, J. Shelton, T. Abegaz, K. Bryant, J. Adams, D. L. Woodard, and P. Miller. Gefes: Genetic evolutionary feature selection for periocular biometric recognition. In *2011 IEEE Workshop on Computational Intelligence in Biometrics and Identity Management (CIBIM)*, pages 152–156, 2011.

[31] Henry Williams, Joi Carter, Willie Campbell, Kaushik Roy, and Gerry Dozier. Genetic & evolutionary feature selection for author identification of html associated with malware. *International Journal of Machine Learning and Computing*, 4:250–255, 06 2014.

[32] A. Alford, C. Steed, M. Jeffrey, D. Sweet, J. Shelton, L. Small, D. Leflore, G. Dozier, K. Bryant, T. Abegaz, J. C. Kelly, and K. Ricanek. Genetic evolutionary biometrics: Hybrid feature selection and weighting for a multi-modal biometric system. In *2012 Proceedings of IEEE Southeastcon*, pages 1–8, 2012.

[33] T. Blickle and L. Thiele. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, 4(4):361–394, 1996.

[34] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer Berlin Heidelberg, 2015.

[35] Thomas Bäck, Frank Hoffmeister, and Hans-Paul Schwefel. A survey of evolution strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann, 1991.

[36] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[37] Giovanni Santia and Jake Williams. Buzzface: A news veracity dataset with facebook user commentary and egos, 2018.

[38] James W Pennebaker, Ryan L Boyd, Kayla Jordan, and Kate Blackburn. The development and psychometric properties of liwc2015. Technical report, 2015.

[39] Jigsaw. Perspectiveapi. `https://www.perspectiveapi.com/`, 2017.

[40] Mike Thelwall, K Buckley, G Paltoglou, C Cai, and A Kappas. Sentistrength. *URL: http://sentistrength. wlv. ac. uk*, 2014.

[41] echen102. https://github.com/echen102/covid-19-tweetids.

[42] Emily Chen, Kristina Lerman, and Emilio Ferrara. Tracking social media discourse about the covid-19 pandemic: Development of a public coronavirus twitter data set. *JMIR Public Health Surveill*, 6(2):e19273, May 2020.

[43] S. B. Parikh and P. K. Atrey. Media-rich fake news detection: A survey. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 436–441, 2018.

[44] Joseph P Forgas and Roy Baumeister. *The Social Psychology of Gullibility: Conspiracy Theories, Fake News and Irrational Beliefs*. Routledge, 2019.

[45] Edson C Tandoc Jr, Zheng Wei Lim, and Richard Ling. Defining "fake news" a typology of scholarly definitions. *Digital journalism*, 6(2):137–153, 2018.

[46] Nicholas W. Jankowski. Researching fake news: A selective examination of empirical studies. *Javnost - The Public*, 25(1-2):248–255, 2018.

[47] Elise M. Stevens and Karen McIntyre. The layers of the onion: The impact of satirical news on affect and online sharing behaviors. *Electronic News*, 13(2):78–92, 2019.

[48] Ian Brodie. Pretend news, false news, fake news: The onion as put-on, prank, and legend. *The Journal of American Folklore*, 131:451, 10 2018.

[49] The Onion. Congress takes group of schoolchildren hostage, Oct 2017.

[50] B. Xue, M. Zhang, W. N. Browne, and X. Yao. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation*, 20(4):606–626, 2016.

[51] Giovanni C. Santia and Jake Ryland Williams. Buzzface: A news veracity dataset with facebook user commentary and egos. In *ICWSM*, 2018.

[52] Gilbert Syswerda. Uniform crossover in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, page 2–9, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[53] B. Darvish Rouani, M. Samragh, T. Javidi, and F. Koushanfar. Safe machine learning and defeating adversarial attacks. *IEEE Security Privacy*, 17(2):31–38, 2019.

[54] Anthony D. Joseph, Blaine Nelson, Benjamin I. P. Rubinstein, and J. D. Tygar. *Adversarial Machine Learning*. Cambridge University Press, USA, 1st edition, 2019.

[55] Reuben Feinman, Ryan R. Curtin, Saurabh Shintre, and Andrew B. Gardner. Detecting adversarial samples from artifacts, 2017.

[56] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[57] Tianqi Chen and Carlos Guestrin. Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug 2016.

[58] Padraig Cunningham and Sarah Delany. k-nearest neighbour classifiers. *Mult Classif Syst*, 04 2007.

[59] William S Noble. What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567, 2006.

[60] Lawrence Davis. *Handbook of genetic algorithms*. Van Nostrand Reinhold, 1996.

[61] Daniel H. Solomon, Richard Bucala, Mariana J. Kaplan, and Peter A. Nigrovic. The infodemic of covid-19. *Arthritis & Rheumatology*, 72(11):1806–1808, 2020.

[62] Richard J. Medford, Sameh N. Saleh, Andrew Sumarsono, Trish M. Perl, and Christoph U. Lehmann. An "infodemic": Leveraging high-volume twitter data to understand public sentiment for the covid-19 outbreak. *medRxiv*, 2020.

[63] www.merriam webster.com. Words we're watching: 'infodemic'. *https://www.merriam-webster.com/words-at-play/words-were-watching-infodemic-meaning. Last Accessed 1/7/2021*.

[64] Alan Black, Christopher Mascaro, Michael Gallagher, and Sean P. Goggins. Twitter zombie: Architecture for capturing, socially transforming and analyzing the twittersphere. In *Proceedings of the 17th ACM International Conference on Supporting Group Work*, GROUP '12, page 229–238, New York, NY, USA, 2012. Association for Computing Machinery.

[65] Axel Bruns, Jean Burgess, and Tim Highfield. *A â€˜Big Dataâ€™ Approach to Mapping the Australian Twittersphere*, pages 113–129. 01 2014.

[66] Matteo Cinelli, Walter Quattrociocchi, Alessandro Galeazzi, Carlo Michele Valensise, Emanuele Brugnoli, Ana Lucia Schmidt, Paola Zola, Fabiana Zollo, and Antonio Scala. The covid-19 social media infodemic. *Scientific Reports*, 10(1), Oct 2020.

[67] J. Gaston, M. Narayanan, G. Dozier, D. L. Cothran, C. Arms-Chavez, M. Rossi, M. C. King, and J. Xu. Authorship attribution via evolutionary hybridization of sentiment analysis, liwc, and topic modeling features. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 933–940, 2018.

[68] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[69] Anthony Viera and Joanne Garrett. Understanding interobserver agreement: The kappa statistic. *Family medicine*, 37:360–3, 06 2005.

[70] S. Weisberg. *Applied Linear Regression*. John Wiley & Sons, Hoboken, New Jersey, 2005.

[71] B. Brown, A. Richardson, M. Smith, G. Dozier, and M. C. King. The adversarial ufp/ufn attack: A new threat to ml-based fake news detection systems? In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1523–1527, 2020.

[72] Tsui-Wei Weng, Pu Zhao, Sijia Liu, Pin-Yu Chen, Xue Lin, and Luca Daniel. Towards certificated model robustness against weight perturbations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):6356–6363, Apr. 2020.

[73] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty, 2019.

[74] Ninghao Liu, Mengnan Du, Ruocheng Guo, Huan Liu, and Xia Hu. Adversarial attacks and defenses: An interpretation perspective, 2020.

[75] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks, 2017.

[76] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples, 2017.

[77] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks, 2016.

[78] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features, 2019.

[79] Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. Adversarial and clean data are not twins, 2017.

[80] Chuanbiao Song, Kun He, Jiadong Lin, Liwei Wang, and John E. Hopcroft. Robust local features for improving the generalization of adversarial training. *CoRR*, abs/1909.10147, 2019.

[81] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale, 2017.

[82] Haichao Zhang and Jianyu Wang. Defense against adversarial attacks using feature scattering-based adversarial training, 2019.

[83] Jianyu Wang and Haichao Zhang. Bilateral adversarial training: Towards fast training of more robust models against adversarial attacks, 2019.

[84] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014.

[85] Alice Smith, David Coit, Thomas BÃ¤ck, David Fogel, and Zbigniew Michalewicz. Penalty functions. 07 1998.

[86] William Crossley, Edwin Williams, William Crossley, and Edwin Williams. *A study of adaptive penalty functions for constrained genetic algorithm-based optimization*.

[87] A. S. Sodiya, S. A. Onashoga, and Beatrice Oladunjoye. Threat modeling using fuzzy logic paradigm. *Issues in Informing Science and Information Technology*, 4:053–061, 2007.

[88] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the science of security and privacy in machine learning, 2016.

[89] J.A. Cohen. A coefficient of agreement for nominal scales. *Psychological Bulletin*, 20:37–, 01 1960.

[90] Mary McHugh. Interrater reliability: The kappa statistic. *Biochemia medica : Äasopis Hrvatskoga druÅ¡tva medicinskih biokemiÄara / HDMB*, 22:276–82, 10 2012.

[91] Susana Vieira, Uzay Kaymak, and JoÃ£o Sousa. Cohen's kappa coefficient as a performance measure for feature selection. pages 1–8, 07 2010.

[92] Receptiviti. Emotions framework.

[93] Receptiviti. Cognition framework.

[94] Receptiviti. Drives framework.

[95] Receptiviti. Social dynamics framework.

[96] Receptiviti. Temporal and orientation framework.

[97] Receptiviti. Toxicity framework.

[98] Receptiviti. Needs and values framework.

[99] Robi Polikar. *Ensemble Learning*, pages 1–34. Springer US, Boston, MA, 2012.

[100] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[101] Nadja Betzler, Robert Bredereck, Jiehua Chen, and Rolf Niedermeier. Studies in computational aspects of voting. In *The Multivariate Algorithmic Revolution and Beyond*, pages 318–363. Springer, 2012.

[102] Thomas G Dietterich et al. Ensemble learning. *The handbook of brain theory and neural networks*, 2(1):110–125, 2002.