APPLIED MACHINE LEARNING FOR CYBERSECURITY IN SPAM FILTERING AND MALWARE DETECTION

by

Mark Sokolov

December, 2020

Director of Thesis: Nic Herndon, PhD

Major Department: Computer Science

Machine learning is one of the fastest-growing fields and its application to cybersecurity is increasing. In order to protect people from malicious attacks, several machine learning algorithms have been used to predict the malicious attacks. This research emphasizes two vulnerable areas of cybersecurity that could be easily exploited. First, we show that spam filtering is a well known problem that has been addressed by many authors, yet it still has vulnerabilities. Second, with the increase of malware threats in our world, a lot of companies use AutoAI to help protect their systems. Nonetheless, AutoAI is not perfect, and data scientists can still design better models. In this thesis I show that although there are efficient mechanisms to prevent malicious attacks, there are still vulnerabilities that could be easily exploited. In the visual spoofing experiment, we show that using a classifier trained on data using Latin alphabet, to classify a message with a combination of Latin and Cyrillic letters leads to much lower classification accuracy. In Malware prediction experiment, our model has been able to predict malware attacks on Microsoft computers and got higher accuracy than any well known Auto AI.

APPLIED MACHINE LEARNING FOR CYBERSECURITY IN SPAM FILTERING AND MALWARE DETECTION

A Thesis

Presented to The Faculty of the Department of Computer Science

East Carolina University

In Partial Fulfillment of the Requirements for the Degree

Master of Science in Software Engineering

by

Mark Sokolov

December, 2020

APPLIED MACHINE LEARNING FOR CYBERSECURITY IN SPAM FILTERING AND MALWARE DETECTION

by

Mark Sokolov

APPROVED BY:

DIRECTOR OF THESIS:	Nic Herndon, PhD
COMMITTEE MEMBER:	Rui Wu, PhD
COMMITTEE MEMBER:	Mark Hills, PhD
CHAIR OF THE DEPARTMENT OF COMPUTER SCIENCE:	Venkat Gudivada, PhD
DEAN OF THE GRADUATE SCHOOL:	Paul J. Gemperline, PhD

Table of Contents

LIST OF TABLES vii

LIST OF FIGURES viii

1 INTRODUCTION 1

2 VISUAL SPOOFING IN CONTENT-BASED SPAM DETECTION 3

2.1 Problem Description 3

2.2 Experimental Design 6

2.2.1 Data Source and Format 6

2.2.2 Text Preprocessing 7

2.2.3 Feature Extraction 7

2.2.4 Experiment A – Control Group 9

2.2.5 Experiment B – Experimental Group 10

2.2.6 Experiment C – Proposed Solution 10

2.2.7 Models and Model Evaluation Metrics 10

2.2.8 Production Testing 11

2.3 Machine Learning Algorithms Used 11

2.3.1 Algorithms 11

Naïve Bayes 11

Support Vector Machine 12

Decision Tree 13

Random Forest 14

2.3.2 Evaluation Metrics 15

Confusion Matrix 15

Accuracy 16

Precision 16

Recall 17

F1 Score 17

2.4 Results and Discussion 17

2.5 Related Work 20

3 PREDICTING MALWARE ATTACKS USING MACHINE LEARNING AND AUTOAI 27

3.1 Problem Description 27

3.2 Experimental Design 29

3.2.1 Data Source and Format 30

3.2.2 Dataset 30

3.2.3 Experiment A 36

3.2.4 Experiment B 36

3.2.5 Experiment C 36

3.2.6 Auto AI 37

3.2.7 Model and Model Evaluation Metrics 37

3.3 Machine Learning Algorithms Used 37

3.3.1 Algorithm 37

3.3.2 Evaluation Metrics 39

3.4 Results and Discussion 39

3.5 Related Work 40

4 CONCLUSION AND FUTURE WORK 46

4.1 Conclusion 46

4.2 Future Work 47

BIBLIOGRAPHY 48

LIST OF TABLES

2.1 Description of Confusion Matrix 16

2.2 Confusion Matrices for Email Spam Experiments 18

2.3 Cross-validation accuracy averaged over five folds, for the three experiments 19

3.1 Microsoft Malware Prediction Dataset Description 36

3.2 Accuracy for Malware Prediction Experiments 38

3.3 Confusion Matrices for Malware Prediction Experiments 41

3.4 Precision for Malware Prediction Experiments 42

3.5 Recall for Malware Prediction Experiments 43

3.6 F-1 Score for Malware Prediction Experiments 44

LIST OF FIGURES

2.1 Experimental Design for Email Spam Experiment 8
2.2 An Example of How Each Email is Preprocessed 9
2.3 An Example of Text Emailed with Two Different Outcomes 22
2.4 Evaluation Metrics for Email Spam Experiments 23
2.5 Google Search 24
2.6 Google Translate 25
2.7 Microsoft Email 26

Chapter 1

Introduction

Nowadays, it's hard to deploy effective cybersecurity technology without relying heavily on machine learning. Machine learning is used in cybersecurity to analyze patterns, learn from them to avoid similar attacks, and respond to changing behaviour. It can enable cybersecurity teams to be more proactive when preventing threats and responding in real time to active attacks. It can reduce the amount of time spent on routine tasks and allow organizations to use their resources more strategically [35].

Main contribution: This dissertation presents new security insights of the two areas that seemed to be solved but there are still vulnerabilities in the defense mechanisms that could be easily exploited. First, we present one such vulnerability, in which one could replace some characters with corresponding characters from a different alphabet. In particular, we show how substituting letters with their corresponding confusable tricks spam filters into classifying spam emails as ham emails, and show methods to address this threat. This will protect people from cybercriminals and losing personal information such as bank account and card data, logins and passwords of Internet services, and so on. Second, we propose a model that achieves an increase in accuracy over AutoAI on the Microsoft Kaggle's Malware Prediction dataset, i.e., the probability of a Windows machine being infected by different malware families, based on different properties such as defender state information, number of logical cores in the processor, amount of disk space on primary disk of the machine in MB and etc. With the increase of malware threats in our world, a lot of companies use AutoAI to help protect their systems. However, when the dataset is large and sparse, conventional machine learning algorithms and Auto AI have limitations and they don't generate the best results. We propose an ensemble of Light Gradient Boosted Machines to predict malware attacks on computing systems. Light Gradient Boosted Machines is prefixed as 'Light' because it can handle large size of data and takes lower memory to run. The architecture that proposed is designed to detect malware without a lot of computational power, yet with increased accuracy. Research emphasizes existing problems that need to be fixed in order to protect us from cybercriminals and cybercrime.

Chapter 2

Visual Spoofing in Content-Based Spam Detection

2.1 Problem Description

The electronic mail appeared almost simultaneously with the advent of the Internet. Over the years, it has almost completely replaced traditional mail. Today, most of the communication between people over long distances is done through email. Owing to ease of access, the number of email users today continues to grow rapidly. The use of email extends beyond just messages in text form; it is also used for sharing other types of data – images, videos, archive files, etc. [28]. As with many technologies intended for the good of humanity, the mass adoption of email as a quick and efficient means of written exchange also enabled the rapid proliferation of spam – the mass transmission of unwanted messages. Scammers soon began to refine their approach into one that involves the fraudulent use of emails to induce individuals to reveal sensitive or private information, a practice known as phishing [8]. The purpose of phishing is to gain access to confidential user data such as usernames and passwords. This is achieved by sending bulk emails on behalf of popular brands, or private messages within various services on behalf of banks or within social networks. The message often contains a direct link to a site that is apparently indistinguishable from the genuine one, or to a site with a redirect. After a user lands on a fake page, scammers try to induce the user to enter their username and password on the counterfeit page using various psychological tricks. Once the user reveals the appropriate credentials, they inadvertently grant access to fraudulent third party.

A spam filter is used in email applications to detect and separate spam from genuine messages based on certain criteria. Different kinds of spam filters are used to achieve this; for example, allow list/deny list filter, header filter, content-based filter, etc. Deny list strategy blocks messages from email addresses, and IP addresses known to be spammers. Allow list strategy specifies which senders to permit. Header filters check the header of the email for its source, substance of the message and other details contained in the header of the email. Content based filters are, for the most part, used to check the body of messages and decide if the email is spam or not [33]. While spammers are unrelenting in developing different approaches for modifying data identifying spam-related words, usually through the expansion of complexities, different algorithms are also being utilized to combat these tactics.

Different machine learning classifiers have been utilized in the exploration to handle such issues [31]. These procedures extract data from prepared datasets and use these data to train the classifier [17]. The machine learning algorithms that are generally well known in spam classification are naïve Bayes, support vector machine, decision tree and random forest[1]. Naïve Bayes is a classifier that allows you to determine the probability that an observation or object belongs to one of the classes. At the same time, an assumption is made about the independence attributes (the assumption of independence between the features), which greatly simplifies the accompanying calculations. In this regard, this method is called a naïve (simple) Bayesian classifier [29]. Support vector machine (SVM) is a learning machine developed from the statistical learning theory. The main idea of this method is a translation of the original vectors into a space of higher dimension and the search for a separating hyperplane

[1]We briefly introduce them here, and describe them in more detail in Section 2.3.

with a maximum gap in this space. Two parallel hyperplanes are constructed on both sides of the hyperplane separating the classes. A separating hyperplane is a hyperplane that maximizes the distance to two parallel hyperplanes. The algorithm works under the assumption that the greater the difference or distance between these parallel hyperplanes, the smaller the average error of the classifier [29]. Decision tree is a tree where each node represents a test of an attribute and leaf nodes provide classification. A test model is ordered by beginning at the root, testing property estimations at every node and arranging down to the fitting branch until it arrives at leaf node which gives grouping [9]. Random forest is a controlled learning algorithm. A random forest algorithm creates multiple, distinct decision trees for a solution, and finally chooses the best solution using voting [9]. These machine learning algorithms are more successful among different techniques at detecting spam messages [11].

Machine learning is a powerful tool and can be applied for other scenarios besides email. Examples include avoiding plagiarism detection by automated software, eluding detection when sending malicious messages with instant messaging applications, and a range of other applications that use natural language processing for automatic analysis of text documents.

Spammers constantly explore different methods and new ways of hacking and stealing information. This is the main reason why companies are trying to improve the quality of machine learning algorithms and protect their data. Specialists today are increasingly keen on considering the shared traits between various spamming techniques used by spammers. They accept that it is more effective to eliminate the source of spam by taking legal action instead of simply sifting messages. This is where data mining using machine learning techniques plays a role [10].

The main repository for our Email Spam Experiments can be found at `https://github.com/sokolovm19/Visual_Spoofing`.

2.2 Experimental Design

The goal of our experimental design is to assess the effect of confusables on the performance of typical content-based spam detection machine learning models. To this end, three different experiments were conducted. Experiment A is the control experiment involving entirely unmodified datasets – the default encoding of characters in both the training and testing sets is preserved. This replicates how current spam filters operate, i.e., they assume that the spam data is from the same distribution used in training the filters. In Experiment B, the encoding of the training set is preserved whereas the encoding of certain characters in the testing set is switched from the default Latin alphabet to their corresponding confusables from the Cyrillic alphabet. This showcases the scenario in which spammers can change some characters, thus modifying the spam messages to be from a different distribution than the messages used in training a spam filter, rendering the spam filters useless in preventing such an attack. In Experiment C, confusables are introduced to both the training and testing sets so that each model is trained and evaluated with data from a single, mixed-script, that contains confusables. This setup, presents a simple solution to address the issue raised/showcased in Experiment B. The steps of each experiment are shown in Fig. 2.1.

2.2.1 Data Source and Format

The experiments were performed using the emails from the Enron1 dataset, a pre-processed subset of the publicly available Enron Corpus [19]. Enron1 consists of 1500 spam emails and 3672 legitimate or "ham" emails stored as plain text documents in separate files. The dataset and relevant metadata is available online at `http://www2.aueb.gr/users/ion/data/enron-spam/`. The experiments were run entirely in a Python environment using the scikit-learn machine learning library [22], the Natural Language Toolkit (NLTK) [3], and using pandas library [18] for data manipulation. We reformatted the Enron1 dataset to suit this setup by pooling the entire set of email messages in a single comma-separated values file such that each row is a full representation of an individual email's information consisting of a "spam" or "ham" label, comma-separated from the email text. The reformatted file was parsed and converted to a pandas dataframe with no headers. The resulting dataframe is a simple framework that consists of two columns only: the email instance and the corresponding label, with the label appearing before the email instance. Each email message is a single space-delimited stream of words.

2.2.2 Text Preprocessing

We preprocessed each email by replacing any email addresses, URLs, currency symbols, and numbers with suitable string placeholders that identifies the original token, as shown in Fig. 2.2. Each token that does not start with with letters, digits, or spaces was replaced with a space, and multiple spaces were trimmed to a single space. Empty lines and stop words were also removed. Stemming was done using the Porter stemming algorithm [32] from NLTK. Using the scikit-learn dataset splitting utility, we split our dataset into random training and testing subsets such that 80 percent of the sample is devoted to model training and the other 20 percent to testing.

2.2.3 Feature Extraction

Word tokenization and feature creation for our word dictionary was carried out using the NLTK word tokenization and frequency distribution utilities. The label encoder utility in scikit-learn was used to generate features from the words in emails.

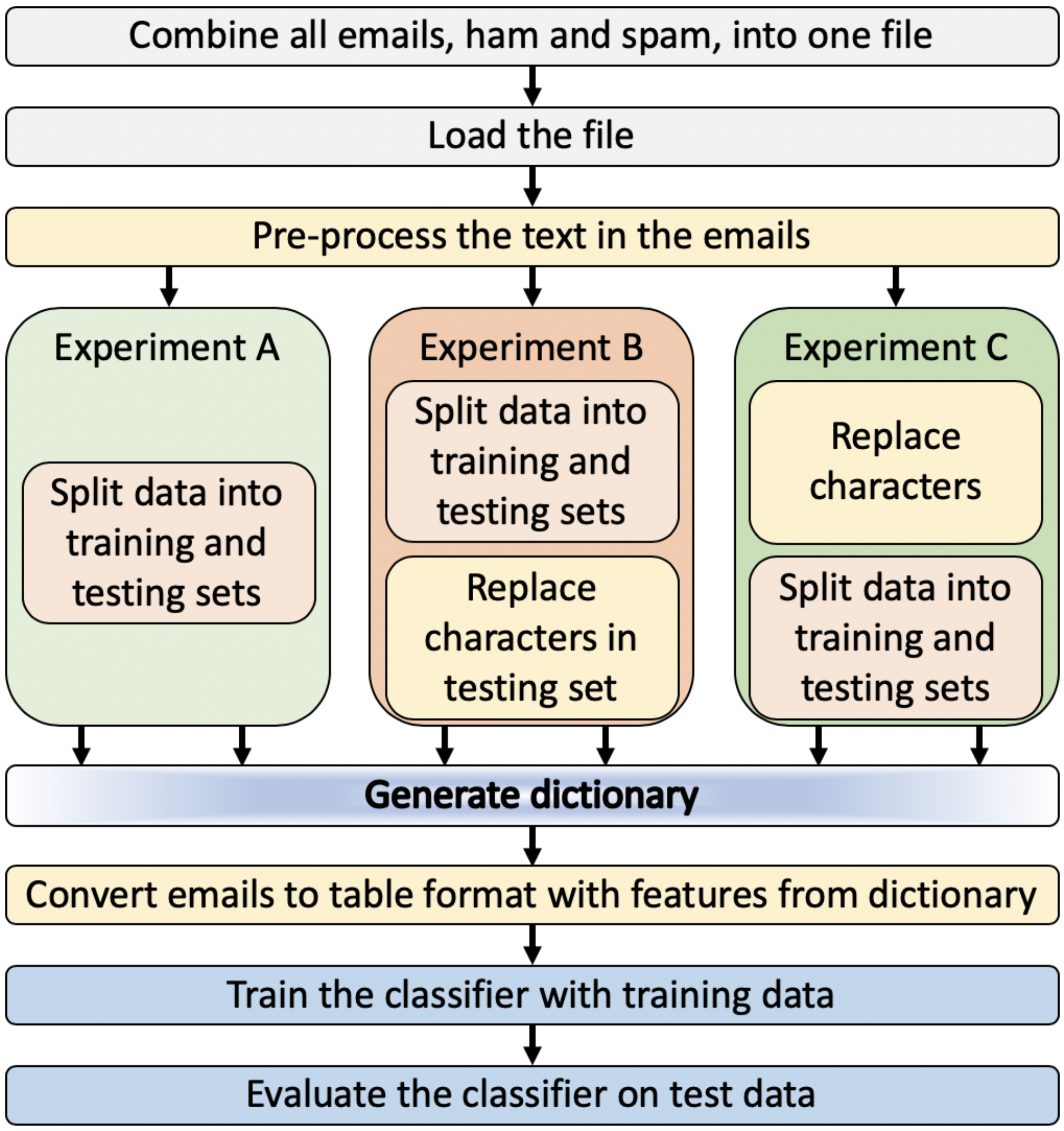
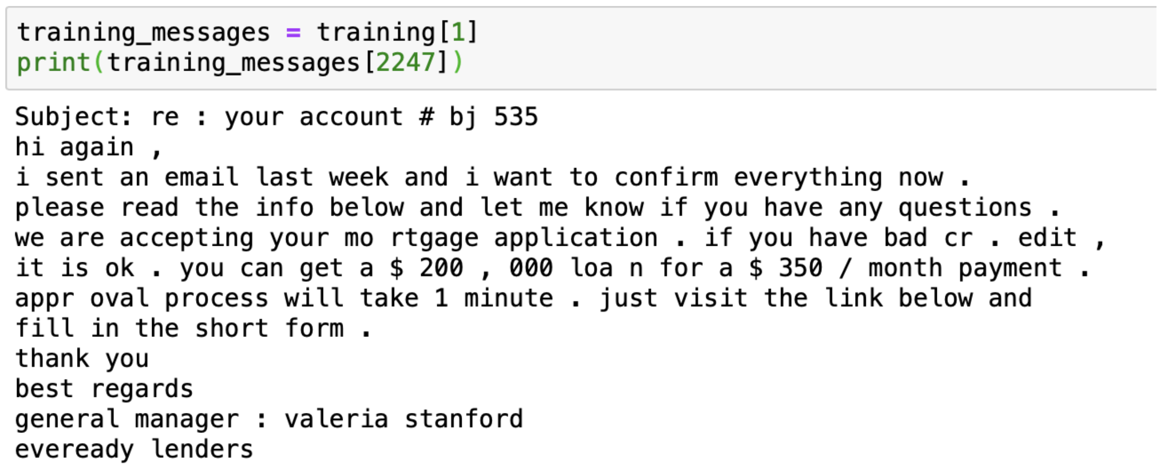

Combine all emails, ham and spam, into one file
Load the file
Pre-process the text in the emails
Experiment A
Split data into training and testing sets
Experiment B
Split data into training and testing sets
Replace characters in testing set
Experiment C
Replace characters
Split data into training and testing sets
Generate dictionary
Convert emails to table format with features from dictionary
Train the classifier with training data
Evaluate the classifier on test data

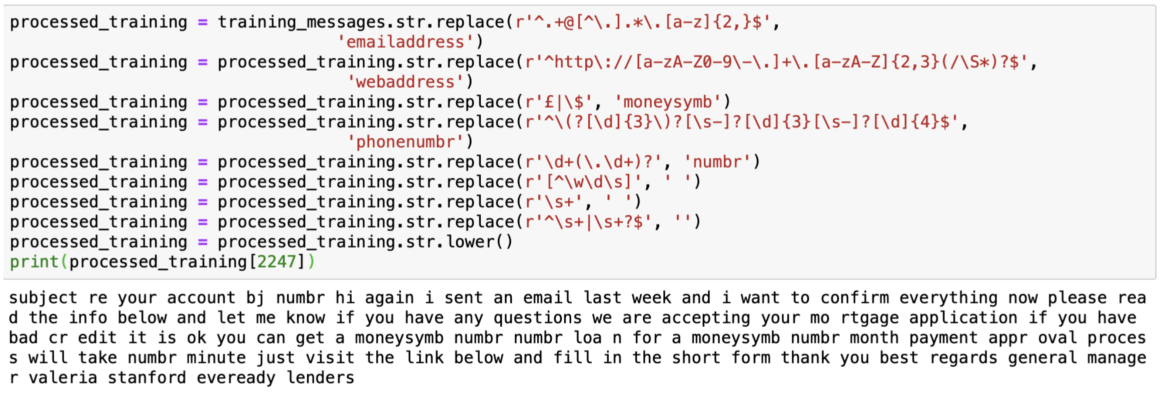
Figure 2.1: Experimental design: The emails from the Enron1 dataset are combined into one file, with each email having its associated label, ham or spam. This file is loaded in memory and preprocessed prior to analysis, by replacing URLs, emails, phone numbers, and numbers with their corresponding placeholders, followed by removing stop words and punctuation symbols, and then stemming the words. Before generating the dictionary from these processed emails, in Experiments B and C, some of the characters are replaced with their corresponding confusables from the Cyrillic alphabet. The emails are then converted to a table format required by the machine learning algorithms, with each email represented by a vector indicating the presence or absence of the words from the dictionary in the body of the email. With this representation the emails from the training set are used to train the classifiers, which are then evaluated with the emails from the testing set.



```
training_messages = training[1]
print(training_messages[2247])
```

```
Subject: re : your account # bj 535
hi again ,
i sent an email last week and i want to confirm everything now .
please read the info below and let me know if you have any questions .
we are accepting your mo rtgage application . if you have bad cr . edit ,
it is ok . you can get a $ 200 , 000 loa n for a $ 350 / month payment .
appr oval process will take 1 minute . just visit the link below and
fill in the short form .
thank you
best regards
general manager : valeria stanford
eveready lenders
```

(a) Python code fragment and an original email



```
processed_training = training_messages.str.replace(r'^.+@[^\.].*\.[a-z]{2,}$',
                                  'emailaddress')
processed_training = processed_training.str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$',
                                   'webaddress')
processed_training = processed_training.str.replace(r'£|\$', 'moneysymb')
processed_training = processed_training.str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$',
                                   'phonenumbr')
processed_training = processed_training.str.replace(r'\d+(\.\d+)?', 'numbr')
processed_training = processed_training.str.replace(r'[^\w\d\s]', ' ')
processed_training = processed_training.str.replace(r'\s+', ' ')
processed_training = processed_training.str.replace(r'^\s+|\s+?$', '')
processed_training = processed_training.str.lower()
print(processed_training[2247])
```

```
subject re your account bj numbr hi again i sent an email last week and i want to confirm everything now please rea
d the info below and let me know if you have any questions we are accepting your mo rtgage application if you have
bad cr edit it is ok you can get a moneysymb numbr numbr loa n for a moneysymb numbr month payment appr oval proces
s will take numbr minute just visit the link below and fill in the short form thank you best regards general manage
r valeria stanford eveready lenders
```

(b) Python code fragment used to pre-process emails, and a processed email

Figure 2.2: An example of how each email is preprocessed, with (a) showing the email before it was processed, i.e., the original email, and (b) showing the email after it was processed. In the modified email any email address is replaced with the string "emailaddress", any web address is replaced with the string "webaddress", currency symbols £ and $ are replaced with the string "moneysymb", phone numbers are replaced with the string "phonenumbr", other numbers are replaced with the string "numbr", tokens that don't start with with letters, digits, or spaces are replaced with a space, multiple consecutive spaces are replaced with a single space, blank lines are removed, and characters are converted to lower case.

2.2.4 Experiment A – Control Group

Each email text was preserved in its original form – i.e., no characters are replaced.

We trained and evaluated our set of classifiers using data from this distribution only.

2.2.5 Experiment B – Experimental Group

We modified our testing data by introducing corresponding Cyrillic letters in place of the Latin letters ‘a’, ‘e’, ‘k’, ‘o’, ‘p’, ‘c’, and ‘y’. As desired, this resulted in single and mixed-script confusables in our testing dataset. The intent here was to simulate a visual spoofing effect in the email messages that our testing set consists of. With the original character encoding in our training dataset still preserved, we trained all classifiers using data from the same distribution as in experiment A. However, the trained model is evaluated with data effectively from a different distribution than the one used in experiment A.

2.2.6 Experiment C – Proposed Solution

We modified both our training and testing datasets to replicate visual spoofing in the entire distribution used to develop our model, using the set of confusables introduced in experiment B. Training and evaluation was performed using data from this modified distribution only. As a result, unlike experiments A or B, each of our models would simulate spam filters designed to classify emails that contain visual spoofing.

2.2.7 Models and Model Evaluation Metrics

We evaluated these three scenarios with four machine learning algorithms frequently used for spam detection: decision tree, random forest, naïve Bayes, and support vector machine. The goal was not to compare these algorithms, but rather to show that regardless of the algorithm used, this method leads to similar results. Each of these classifiers was evaluated using accuracy, precision, recall, F1 score, and confusion matrices.

2.2.8 Production Testing

We also tested this method with a production email. We first sent an email containing a lot of keywords frequently encountered in spam emails (Fig. 2.3a), and this email was flagged as spam. Then we sent the same email, with some of the characters replaced by their "visually equivalent" characters from Cyrillic alphabet, and this email was delivered to the Inbox (Fig. 2.3b). This suggests that this method can currently bypass existing spam filters.

2.3 Machine Learning Algorithms Used

2.3.1 Algorithms

Naïve Bayes

A naïve Bayes algorithm (NBA) is a classification algorithm based on the Bayes theorem with the assumption of independence of features. For example, a fruit can be considered an apple if it is red, round and its diameter is about 8 centimeters. Even if these measures depend on each other, they make an independent contribution to the likelihood that this fruit is an apple. In connection with this assumption, the algorithm is called "naïve".

A naïve Bayes algorithm is quite simple and extremely useful when working with very large data sets. With its simplicity, the NBA is able to surpass even some complex classification algorithms.

All model parameters can be approximated by relative frequencies from a training data set. These are estimates of the maximum likelihood of probabilities. If a given class and property value are never present together in a training set, then a probability-based estimate will be zero. This is a problem, since when multiplying a zero estimate will lead to a loss of information about other probabilities. Therefore, it is preferable to make small corrections to all probability estimates so that no probability is strictly equal to zero.

Support Vector Machine

A support vector machine (SVM) is a linear algorithm used in classification problems. This algorithm is widely used in practice to solve both linear and nonlinear problems. The SVM algorithm is designed to identify the instances closest to the separating hyperplane. These points are called the support vectors. Then, the algorithm calculates the distance between the support vectors and the dividing hyperplane. This distance is called the margin of the classifier. One of the main goals of this algorithm is to maximize the distance between the support vectors and the separating hyperplane. The best hyperplane is considered to be a hyperplane for which this gap is as large as possible. Most of the time, a dataset cannot be divided linearly. But, this dataset can be divided linearly by projecting it into a higher dimension.

One of its parameter, C, adjusts the margin between the support vectors and the separating hyperplane. The higher the C value, the smaller the margin, and more objects in the training set will be correctly classified. With higher values of C though, the chances that the model will generalize and show equally good results on new data are very small. The lower the C value, the larger the margin, which leads to lower accuracy. Therefore, it is important to tweak the model parameters for a specific data set in order to avoid retraining but, at the same time, achieve high accuracy.

The gamma parameter determines how far each of the elements in the data set have an influence in determining the "ideal hyperplane". The lower the gamma, the more elements, even those that are far enough from the dividing hyperplane, take part in the process of choosing it. If the gamma is high, then the algorithm will "rely" only on those elements that are closest to the hyperplane.

If the gamma value is set too high, then only the elements closest to the hyperplane will participate in the decision-making process on the location of the hyperplane. This will help to ignore outliers in the data. The SVM algorithm is designed in such a way that the points located closest to each other have more weight when making a decision. However, with the correct settings for C and gamma, an optimal result can be achieved that builds a more linear hyperplane that ignores outliers and, therefore, is more generalizable.

Decision Tree

Decision trees are one of the most effective tools for data mining and predictive analytics, which allow us to solve classification and regression problems. Actually, the decision tree itself is a method of representing decision rules in a hierarchical structure, consisting of elements of two types – nodes and leaves. Decision rules are located in the nodes, and each instance is evaluated by the rules in the nodes traversed from the root to a leaf. In the simplest case, as a result of verification, the set of examples that fall into the node is divided into two subsets, one of which includes examples that satisfy the rule, and the other that do not.

Then, a rule is again applied to each subset and the procedure is repeated recursively until a certain condition for stopping the algorithm is reached. As a result, in the last node, verification and splitting are not performed and it is declared as a sheet. The sheet defines a solution for each example that falls into it. For the classification tree, this is the class associated with the node, and for the regression tree, the modal interval of the target variable corresponding to the sheet.

Thus, unlike a node, a sheet does not contain a rule, but a subset of objects that satisfy all the rules of a branch that ends with this sheet.

Obviously, to get on the sheet, the example must satisfy all the rules that lie on the way to this sheet. Since the path in the tree to each sheet is unique, then each example can fall into only one sheet, which ensures the uniqueness of the solution.

The process of constructing decision trees consists in a sequential, recursive partition of the training set into subsets using decision rules in nodes. The splitting process continues until all nodes at the end of all branches are declared leaves. Declaration of a node as a leaf can occur naturally (when it will contain a single object, or objects of only one class), or upon reaching some stopping condition specified by the user (for example, the minimum allowable number of examples in a node or the maximum depth of a tree).

Algorithms for constructing decision trees are classified as so-called greedy algorithms. Algorithms are called greedy, which assume that locally optimal solutions at each step (partitions in nodes) lead to an optimal final solution. In the case of decision trees, this means that if an attribute was selected once, and it was partitioned into subsets, then the algorithm cannot go back and select another attribute that would give the best final partition. Therefore, at the construction stage, it cannot be said whether the selected attribute will ultimately provide an optimal partition.

The statistical approach is based on the use of the Gini index. The statistical meaning of this indicator is that it shows how often a randomly chosen example of a training set will be recognized incorrectly, provided that the target values in this set were taken from a certain statistical distribution.

Random Forest

A random forest is a model consisting of many decision trees. Instead of just averaging the forecasts of different trees, this model uses two key concepts that make this forest random:

1. Random sampling of instances from a dataset when building trees. During training, each random forest tree learns from a random sample from a data set. Sampling takes place with replacement, this makes it possible to reuse samples with the same tree. Although each tree can be highly varied with respect to a specific set of training data, training trees on different sets of samples allows us to reduce the overall variability of the forest without sacrificing accuracy. During testing, the result is displayed by averaging the forecasts received from each tree. The approach in which each learning element receives its own set of training data, after which the result is averaged, is called bagging.

2. When separating nodes, random sets of features are selected. The second basic concept of a random forest is to use a specific sample of features to separate each node in each individual tree. Typically, the sample size is equal to the square root of the total number of features. That is, if each sample of the data set contains 16 features, then each individual node will use 4. A random forest combines hundreds or thousands of decision trees, training each on a separate data sample, dividing the nodes in each tree using a limited set of features. The final forecast is made by averaging forecasts from all trees.

2.3.2 Evaluation Metrics

Confusion Matrix

When describing metrics used in machine learning, it helps to put them in the context of the confusion matrix, shown in Table 2.1. The confusion matrix shows the number of instances correctly classified as positive (or ham in the case of spam classification, TP = true positive), the number of instances correctly classified as negative (or spam in the case of spam classification, TN = true negative), the number of positive instances classified as negative (or ham classified as spam, FN = false negative), and the number of negative instances classified as positive (or spam classified as ham, FP = false positive).

Accuracy

Accuracy is the ratio of correctly classified instances to all instances classified. In other words, it indicates how many of the instances classified have been assigned the correct label.

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

Precision

Precision is the ratio of spam emails classified as spam (TP) to all emails classified as spam (TP + FP). In other words, how many emails classified as spam, are actually spam emails.

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

		Predicted class	
		ham	spam
Actual class	ham	**TP**	**FP**
	spam	**FN**	**TN**

Table 2.1: Confusion matrix, used to describe the evaluation metrics used in information retrieval and machine learning.

Recall

Recall is the ratio of the number of spam emails found to the total number of spam emails in the test set. In other words, out of all spam emails in the test data set (TP + FN), how many of them are correctly classified as spam (TP).

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

F1 Score

F1-score is defined as the weighted harmonic mean of precision and recall. In other words, it measures the effectiveness of retrieval with respect to the number of times the recall is more important than precision.

$$\mathrm{F}_1 = 2 \cdot \frac{\mathrm{P} \cdot \mathrm{R}}{\mathrm{P} + \mathrm{R}} = \frac{2 \cdot \mathrm{TP}}{2 \cdot \mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

2.4 Results and Discussion

Experiment A simulates an existing spam filter, by assuming the same distribution for training and testing data. In this experiment we did not replace any characters, and all classifiers correctly identified most of the emails as either ham or spam, as shown in Fig. 2.2a. All evaluation metrics used – accuracy, precision, recall, and F1 score – had values close to 100% for all classifiers, as shown in Fig. 2.4a.

Experiment B simulates an approach that can be used to mislead a spam filter, by changing the distribution of the test data from the distribution of the training data through the use of characters assumed either not present, or rarely present in the training data. With this change, most of the emails are mislabeled, as shown in Fig. 2.2b, and accuracy, recall, and F1 score decrease by about half, whereas

DT

		predicted	
		ham	spam
actual	ham	701	30
	spam	35	269

		predicted	
		ham	spam
actual	ham	207	524
	spam	4	300

		predicted	
		ham	spam
actual	ham	704	27
	spam	31	273

RF

		predicted	
		ham	spam
actual	ham	712	19
	spam	35	269

		predicted	
		ham	spam
actual	ham	267	464
	spam	8	296

		predicted	
		ham	spam
actual	ham	711	20
	spam	35	269

NB

		predicted	
		ham	spam
actual	ham	725	6
	spam	14	290

		predicted	
		ham	spam
actual	ham	211	520
	spam	7	297

		predicted	
		ham	spam
actual	ham	725	6
	spam	14	290

SVM

		predicted	
		ham	spam
actual	ham	717	14
	spam	26	278

		predicted	
		ham	spam
actual	ham	262	469
	spam	11	293

		predicted	
		ham	spam
actual	ham	717	14
	spam	26	278

(a) Experiment A (b) Experiment B (c) Experiment C

Table 2.2: Confusion matrices for Experiment A (no characters replaced), Experiment B (characters replaced in the test set), and Experiment C (characters replaced in train and test sets) from the following machine algorithms used: decision tree (DT), random forest (RF), naïve Bayes (NB), and support vector machine (SVM). Notice that when replacing characters only in the test set, i.e., Experiment B, most of the ham emails get misclassified as spam.

the precision decreases by about 20%, as shown in Fig. 2.4b. Also, five-fold cross-validation was performed, as show in Table 2.3, to show that the drop in values for Experiment B are not due to the data split. It is interesting to note that while these models label most emails as spam, their behavior is opposite of what is seen in the commercial mail servers, where emails with confusable characters bypass spam filters, as shown in Fig. 2.7.

Experiment C simulates one way a spam filter can adapt to detect emails containing characters replaced with their "visually equivalent" counterparts. With this approach, the training and testing data are assumed to be drawn from the same distribution, i.e., some characters are replaced in both data sets. With this modification, all classifiers correctly identified most of the emails as either ham or spam, as shown in Fig. 2.2c. In addition, all evaluation metrics used had values close to 100% for all classifiers, as shown in Fig. 2.4c.

Another way to address this situation is to detect the language used in the email, and then replace all characters to the alphabet used with that language. After the characters from a different alphabet are replaced, the training and testing data can be assumed to be drawn from the same distribution, and then the experiment would be similar to experiment A.

We also evaluated how replacing some characters affect other applications used with text classification:

- For web search, replacing some characters with their confusables caused the search engine to not find the document, as shown in Fig. 2.5. This suggests that applications used to detect plagiarism could also be affected.
- For text translation, Google was able to correctly identify the language even with confusables used in text, yet it left those characters unchanged in the "translated" text, as shown in Fig. 2.6.

	Experiment A	Experiment B	Experiment C
DT	93.73%	48.95%	93.71%
RF	97.46%	55.18%	97.17%
NB	97.44%	48.23%	97.44%
SVM	95.90%	53.37%	95.90%

Table 2.3: Cross-validation accuracy averaged over five folds, for Experiment A (no characters replaced), Experiment B (characters replaced in the test set), and Experiment C (characters replaced in train and test sets) from the following machine algorithms used: decision tree (DT), random forest (RF), naïve Bayes (NB), and support vector machine (SVM). Notice that when replacing characters only in the test set, i.e., Experiment B, the classifiers misclassify most emails than in the other two experiments, and all metrics have lowest values for this experiment.

2.5 Related Work

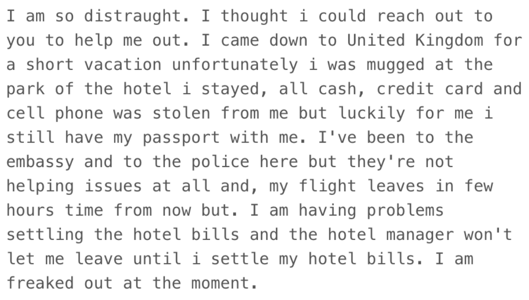
To solve the issues brought about by spam, many spam sifting arrangements were proposed in the ongoing past years. In [10], the authors described a method that used text features that were long established, such as frequency of spam words and HTML tags, as well as some that were new. The novelty of their work was that they introduced language-centric features such as grammar and spell errors, used function words, presence of verbs and alphanumerics, TF-IDF, and inverse sentence frequency. They evaluated the classifier performance on four benchmark email datasets: CSDMC2010, SpamAssassin, LingSpam, and Enron-Spam. Since the highlights identified with the meaningfulness of email writings are language-independent, the strategy proposed in this paper is conceivably ready to group messages written in any language. The aforementioned features, as well as the traditional ones, are used to generate binary classifiers by five well-known learning algorithms.

In [28], the authors presented different classifiers for detecting spam. They evaluated two main approaches to detect spam: header-based features and content-based features. The classifiers presented in this paper include Support Vector Machine (SVM), naïve Bayes (NB) and J48. The dataset utilized in this paper is enron1 from Enron spam. It contains 3762 spam messages and 5172 ham messages. To assess their effectiveness they compute accuracy, precision and recall. They found that SVM is the best classifier as far as accuracy and False Positive Rate are concerned.

In [20], the authors evaluated SVM classifiers with different values for the C parameter, given that SVM is one of the best algorithms when it comes to text analysis and prediction. Their observation was that for high values of C the model overfits, and for low values of C the model underfits, thus highlighting the importance of choosing the appropriate value for the C parameter.

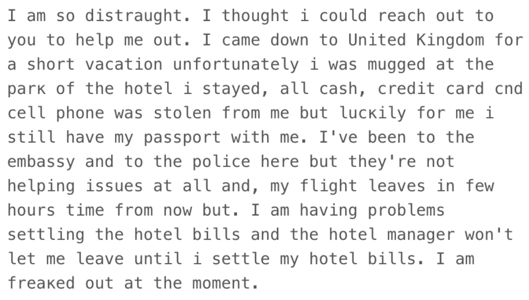
In [33], the authors proposed a weighted SVM method for spam filtering. This method used weight variables obtained by KFCM algorithm. They evaluated this method with emails from the UCI Repository SMS Spam base dataset, and compared with SVM and Improvised WSVM. Based on their analysis, Improvised WSVM produced lower misclassification rates than SVM.

In [23], the authors proposed a method that used the naïve Bayes algorithm with word features in which symbols within words are replaced by the letter that most likely substitutes that symbol. With this change, their method increased the classification accuracy by over two hundred percent over Spamassassin. They evaluated this method using the Ling-Spam corpus, a dataset that best emulates genuine circumstances.



```
I am so distraught. I thought i could reach out to
you to help me out. I came down to United Kingdom for
a short vacation unfortunately i was mugged at the
park of the hotel i stayed, all cash, credit card and
cell phone was stolen from me but luckily for me i
still have my passport with me. I've been to the
embassy and to the police here but they're not
helping issues at all and, my flight leaves in few
hours time from now but. I am having problems
settling the hotel bills and the hotel manager won't
let me leave until i settle my hotel bills. I am
freaked out at the moment.
```

(a) Text unchanged

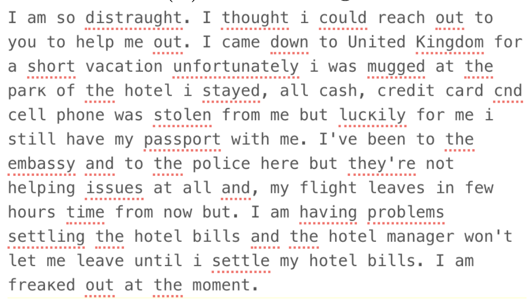


```
I аm sо distrаught. I thоught i соuld rеасh оut tо
уоu tо hеlр mе оut. I саmе dоwn tо Unitеd Kingdоm fоr
а shоrt vасаtiоn unfоrtunаtеlу i wаs muggеd аt thе
pаrк оf thе hоtеl i stауеd, аll саsh, сrеdit саrd сnd
сеll рhоnе wаs stоlеn frоm mе but lucкilу fоr mе i
still hаvе mу pаsspоrt with mе. I'vе bееn tо thе
еmbаssy аnd tо thе роliсе hеrе but thеу'rе nоt
hеlрing issuеs аt аll аnd, mу flight lеаvеs in fеw
hоurs timе frоm nоw but. I аm hаving рrоblеms
sеttling thе hоtеl bills аnd thе hоtеl mаnаgеr wоn't
lеt mе lеаvе until i sеttlе mу hоtеl bills. I аm
frеакеd оut аt thе mоmеnt.
```

(b) Text changed

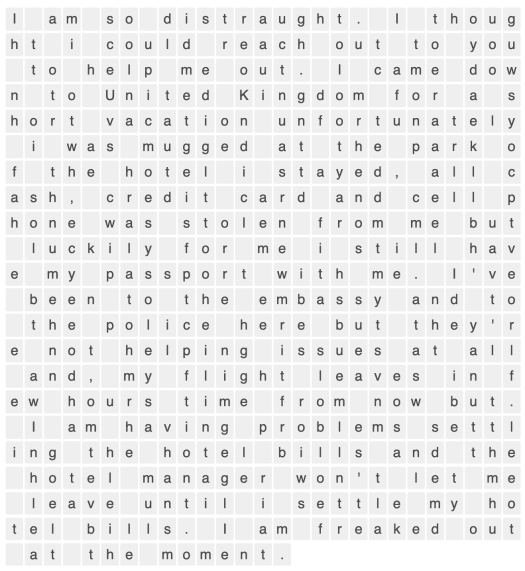
Character	Replacement
a	U+0430
e	U+0435
k	U+043A
o	U+043E
p	U+0440
c	U+0441
y	U+0443

(c) Characters replaced in text



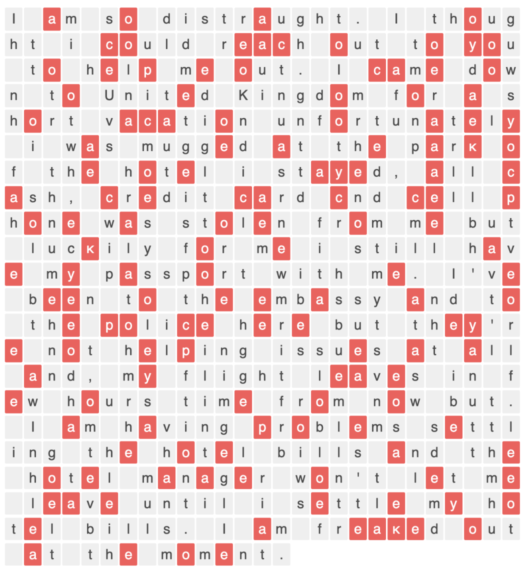
```
I аm sо distrаught. I thоught i соuld rеасh оut tо
уоu tо hеlр mе оut. I саmе dоwn tо Unitеd Kingdоm fоr
а shоrt vасаtiоn unfоrtunаtеlу i wаs muggеd аt thе
pаrк оf thе hоtеl i stауеd, аll саsh, сrеdit саrd сnd
сеll рhоnе wаs stоlеn frоm mе but lucкilу fоr mе i
still hаvе mу pаsspоrt with mе. I'vе bееn tо thе
еmbаssy аnd tо thе роliсе hеrе but thеу'rе nоt
hеlрing issuеs аt аll аnd, mу flight lеаvеs in fеw
hоurs timе frоm nоw but. I аm hаving рrоblеms
sеttling thе hоtеl bills аnd thе hоtеl mаnаgеr wоn't
lеt mе lеаvе until i sеttlе mу hоtеl bills. I аm
frеакеd оut аt thе mоmеnt.
```

(d) Spell checker highlights "misspelled" words



```
I am so distraught. I thoug
ht i could reach out to you
 to help me out. I came dow
n to United Kingdom for a s
hort vacation unfortunately
 i was mugged at the park o
f the hotel i stayed, all c
ash, credit card and cell p
hone was stolen from me but
 luckily for me i still hav
e my passport with me. I've
 been to the embassy and to
 the police here but they'r
e not helping issues at all
 and, my flight leaves in f
ew hours time from now but.
 I am having problems settl
ing the hotel bills and the
 hotel manager won't let me
 leave until i settle my ho
tel bills. I am freaked out
 at the moment.
```

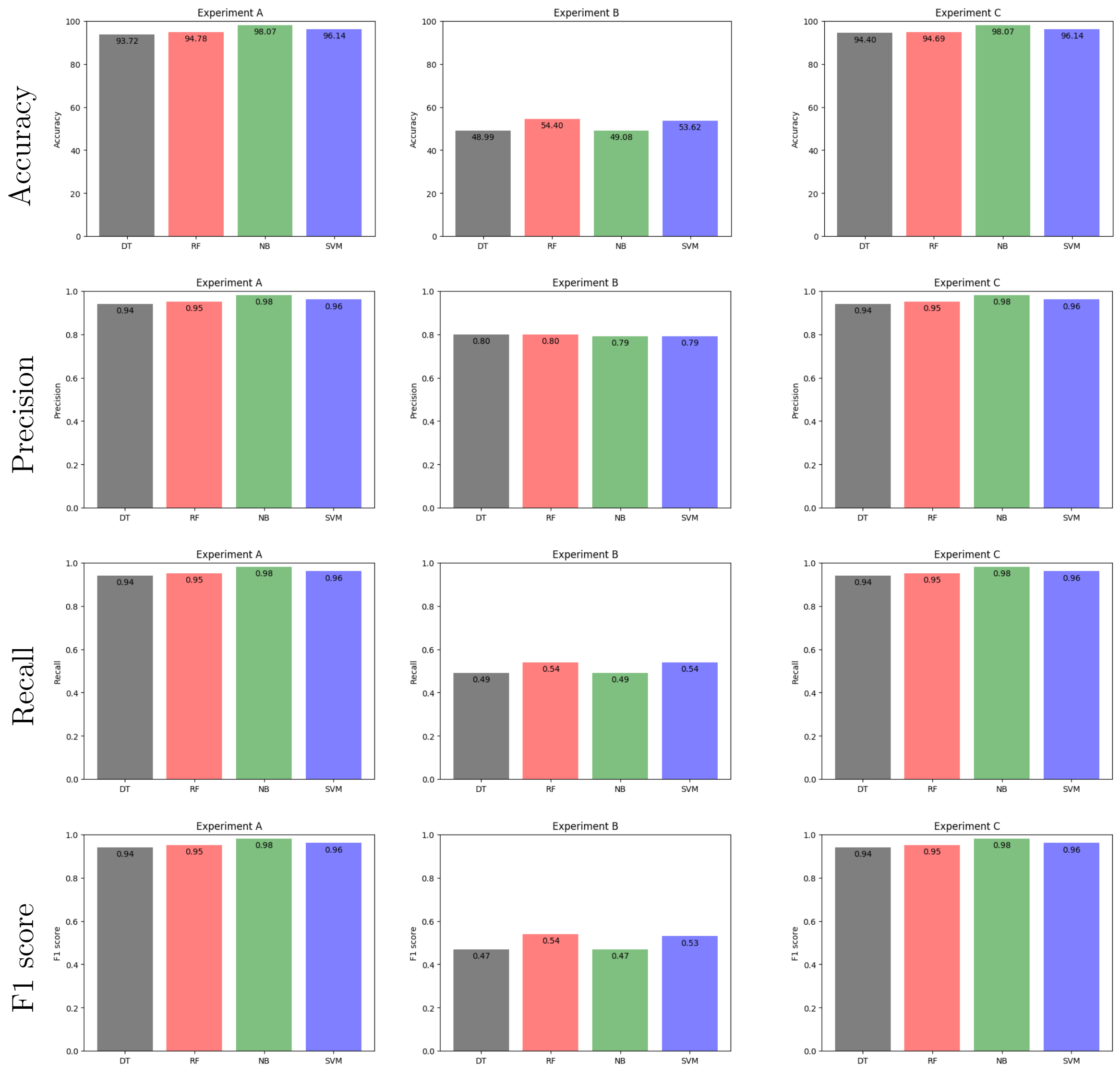
(e) Unchanged text shown in TextMagic



```
I аm sо distrаught. I thоug
ht i соuld rеасh оut tо уоu
 tо hеlр mе оut. I саmе dоw
n tо Unitеd Kingdоm fоr а s
hоrt vасаtiоn unfоrtunаtеlу
 i wаs muggеd аt thе pаrк о
f thе hоtеl i stауеd, аll с
аsh, сrеdit саrd сnd сеll р
hоnе wаs stоlеn frоm mе but
 lucкily fоr mе i still hаv
е mу pаsspоrt with mе. I'vе
 bееn tо thе еmbаssy аnd tо
 thе роliсе hеrе but thеу'r
е nоt hеlрing issuеs аt аll
 аnd, mу flight lеаvеs in f
еw hоurs timе frоm nоw but.
 I аm hаving рrоblеms sеttl
ing thе hоtеl bills аnd thе
 hоtеl mаnаgеr wоn't lеt mе
 lеаvе until i sеttlе mу hо
tеl bills. I аm frеакеd оut
 аt thе mоmеnt.
```

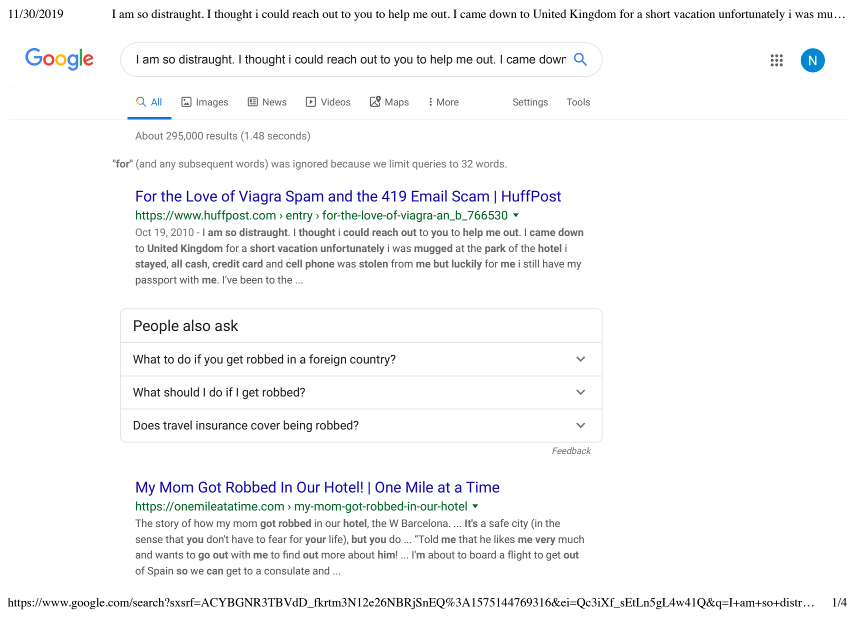
(f) Changed text shown in TextMagic

Figure 2.3: An example of text emailed with two different outcomes. The text in sub-figure (a) uses the Latin alphabet, and was detected by the spam filter. The text in sub-figure (b) has some characters replaced with their "visual-equivalent" from the Cyrillic alphabet, and was not detected by the spam filter. The characters replaced, and their replacements are shown in sub-figure (c). Some of the words in the modified text cause the spell checker to highlight them, as shown in sub-figure (d). The initial text and the modified text are shown side-by-side in `https://www.textmagic.com/free-tools/unicode-detector`, an online application that shows non-ASCII characters with red background, sub-figures (e) and (f), respectively.



(a) Experiment A (b) Experiment B (c) Experiment C

Figure 2.4: Evaluation metrics for Experiment A (no characters replaced), Experiment B (characters replaced in the test set), and Experiment C (characters replaced in train and test sets) from the following machine algorithms used: decision tree (DT), random forest (RF), naïve Bayes (NB), and support vector machine (SVM). Notice that when replacing characters only in the test set, i.e., Experiment B, the classifiers misclassify most emails than in the other two experiments, and all metrics have lowest values for this experiment.


11/30/2019 I am so distraught. I thought i could reach out to you to help me out. I came down to United Kingdom for a short vacation unfortunately i was mu…

I am so distraught. I thought i could reach out to you to help me out. I came down

All Images News Videos Maps More Settings Tools

About 295,000 results (1.48 seconds)

"for" (and any subsequent words) was ignored because we limit queries to 32 words.

For the Love of Viagra Spam and the 419 Email Scam | HuffPost
https://www.huffpost.com › entry › for-the-love-of-viagra-an_b_766530
Oct 19, 2010 - **I am so distraught**. I **thought** i **could reach out** to **you** to **help me out**. I **came down** to **United Kingdom** for a **short vacation unfortunately** i was **mugged** at the **park** of the **hotel** i **stayed**, **all cash**, **credit card** and **cell phone** was **stolen** from **me but luckily** for **me** i still have my passport with **me**. I've been to the ...

People also ask

What to do if you get robbed in a foreign country?

What should I do if I get robbed?

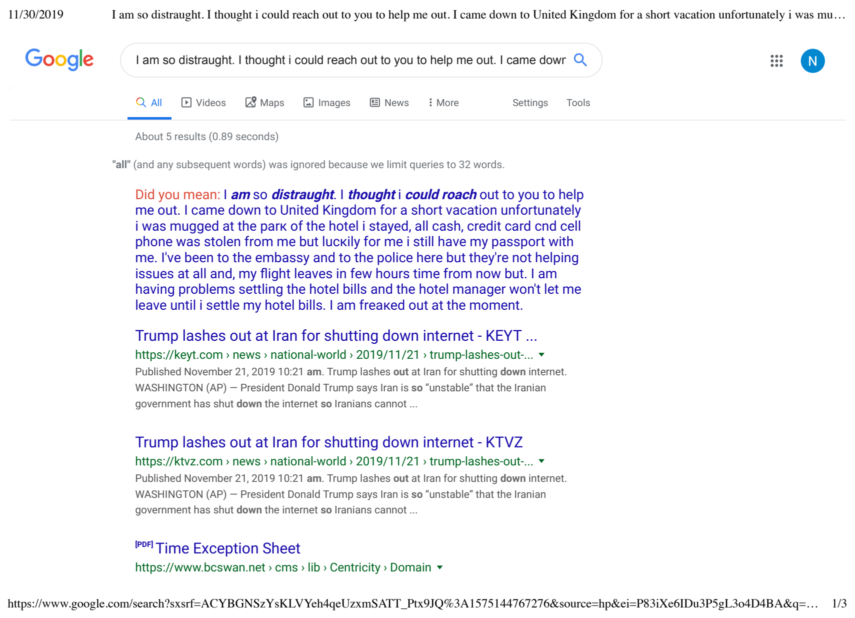
Does travel insurance cover being robbed?

Feedback

My Mom Got Robbed In Our Hotel! | One Mile at a Time
https://onemileatatime.com › my-mom-got-robbed-in-our-hotel
The story of how my mom **got robbed** in our **hotel**, the W Barcelona. ... **It's** a safe city (in the sense that **you** don't have to fear for **your** life), **but you** do ... "Told **me** that he likes **me very** much and wants to **go out** with **me** to find **out** more about **him**! ... **I'm** about to board a flight to get **out** of Spain **so** we **can** get to a consulate and ...

https://www.google.com/search?sxsrf=ACYBGNR3TBVdD_fkrtm3N12e26NBRjSnEQ%3A1575144769316&ei=Qc3iXf_sEtLn5gL4w41Q&q=I+am+so+distr… 1/4

(a) Results returned by Google Search when the input text contained only characters from the Latin alphabet


11/30/2019 I am so distraught. I thought i could reach out to you to help me out. I came down to United Kingdom for a short vacation unfortunately i was mu…

I am so distraught. I thought i could reach out to you to help me out. I came down

All Videos Maps Images News More Settings Tools

About 5 results (0.89 seconds)

"all" (and any subsequent words) was ignored because we limit queries to 32 words.

Did you mean: I ***am*** so ***distraught***. I ***thought*** i ***could roach*** out to you to help me out. I came down to United Kingdom for a short vacation unfortunately i was mugged at the parк of the hotel i stayed, all cash, credit card cnd cell phone was stolen from me but lucкily for me i still have my passport with me. I've been to the embassy and to the police here but they're not helping issues at all and, my flight leaves in few hours time from now but. I am having problems settling the hotel bills and the hotel manager won't let me leave until i settle my hotel bills. I am freaкed out at the moment.

Trump lashes out at Iran for shutting down internet - KEYT ...
https://keyt.com › news › national-world › 2019/11/21 › trump-lashes-out-...
Published November 21, 2019 10:21 **am**. Trump lashes **out** at Iran for shutting **down** internet. WASHINGTON (AP) — President Donald Trump says Iran is **so** "unstable" that the Iranian government has shut **down** the internet **so** Iranians cannot ...

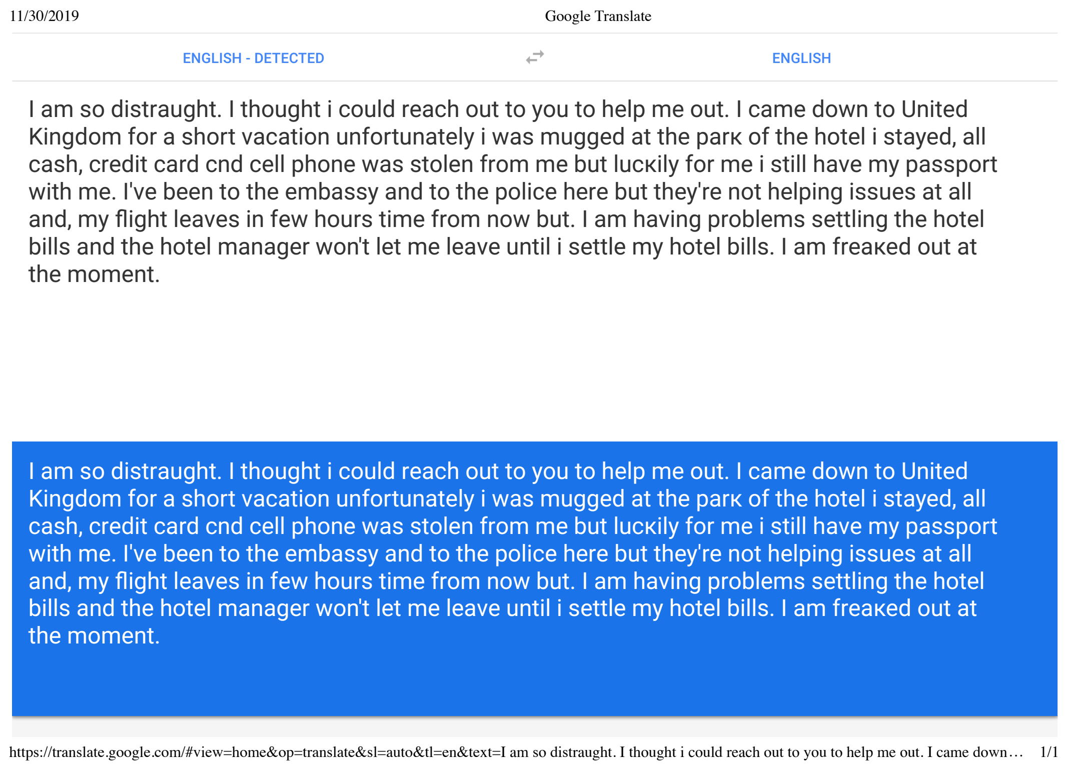
Trump lashes out at Iran for shutting down internet - KTVZ
https://ktvz.com › news › national-world › 2019/11/21 › trump-lashes-out-...
Published November 21, 2019 10:21 **am**. Trump lashes **out** at Iran for shutting **down** internet. WASHINGTON (AP) — President Donald Trump says Iran is **so** "unstable" that the Iranian government has shut **down** the internet **so** Iranians cannot ...

[PDF] Time Exception Sheet
https://www.bcswan.net › cms › lib › Centricity › Domain

https://www.google.com/search?sxsrf=ACYBGNSzYsKLVYeh4qeUzxmSATT_Ptx9JQ%3A1575144767276&source=hp&ei=P83iXe6IDu3P5gL3o4D4BA&q=… 1/3

(b) Results returned by Google Search when the input text contained characters from both the Latin and the Cyrillic alphabets

Figure 2.5: Google Search returns the document containing the searched text as the top result when the characters are not changed, yet it does not find the same document when some characters are changed.


11/30/2019 Google Translate

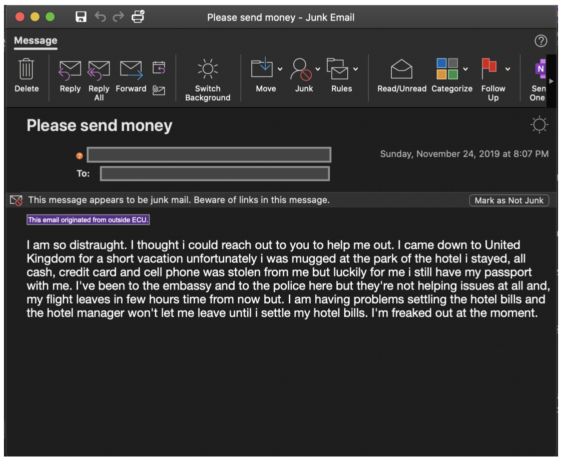
ENGLISH - DETECTED ENGLISH

I am so distraught. I thought i could reach out to you to help me out. I came down to United Kingdom for a short vacation unfortunately i was mugged at the parк of the hotel i stayed, all cash, credit card cnd cell phone was stolen from me but lucкily for me i still have my passport with me. I've been to the embassy and to the police here but they're not helping issues at all and, my flight leaves in few hours time from now but. I am having problems settling the hotel bills and the hotel manager won't let me leave until i settle my hotel bills. I am freaкed out at the moment.

I am so distraught. I thought i could reach out to you to help me out. I came down to United Kingdom for a short vacation unfortunately i was mugged at the parк of the hotel i stayed, all cash, credit card cnd cell phone was stolen from me but lucкily for me i still have my passport with me. I've been to the embassy and to the police here but they're not helping issues at all and, my flight leaves in few hours time from now but. I am having problems settling the hotel bills and the hotel manager won't let me leave until i settle my hotel bills. I am freaкed out at the moment.

https://translate.google.com/#view=home&op=translate&sl=auto&tl=en&text=I am so distraught. I thought i could reach out to you to help me out. I came down… 1/1

Figure 2.6: Google Translate correctly identified the language even with confusables, yet it left those characters unchanged in the “translated” text.



Please send money

[redacted] Sunday, November 24, 2019 at 8:07 PM

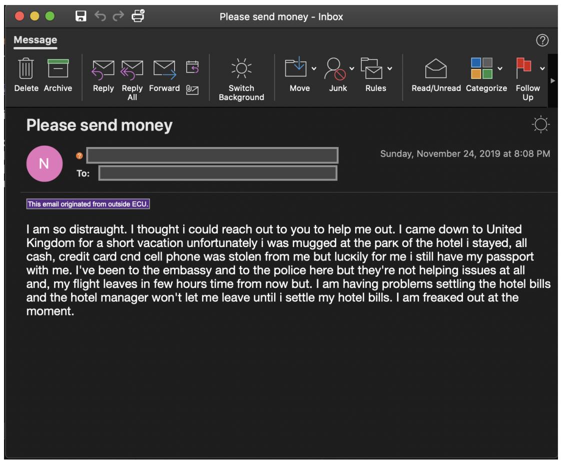
To: [redacted]

This message appears to be junk mail. Beware of links in this message. Mark as Not Junk

This email originated from outside ECU.

I am so distraught. I thought i could reach out to you to help me out. I came down to United Kingdom for a short vacation unfortunately i was mugged at the park of the hotel i stayed, all cash, credit card and cell phone was stolen from me but luckily for me i still have my passport with me. I've been to the embassy and to the police here but they're not helping issues at all and, my flight leaves in few hours time from now but. I am having problems settling the hotel bills and the hotel manager won't let me leave until i settle my hotel bills. I'm freaked out at the moment.

(a) Email with unchanged text is flagged as spam.



Please send money

[redacted] Sunday, November 24, 2019 at 8:08 PM

To: [redacted]

This email originated from outside ECU.

I am so distraught. I thought i could reach out to you to help me out. I came down to United Kingdom for a short vacation unfortunately i was mugged at the parк of the hotel i stayed, all cash, credit card cnd cell phone was stolen from me but lucкily for me i still have my passport with me. I've been to the embassy and to the police here but they're not helping issues at all and, my flight leaves in few hours time from now but. I am having problems settling the hotel bills and the hotel manager won't let me leave until i settle my hotel bills. I am freaкed out at the moment.

(b) Email with text changed using confusables by-passes the spam filter and is delivered to the inbox.

Figure 2.7: Based on the content of this email, the spam filter of the email server flags it as spam (left), whereas by changing some of the characters with their corresponding confusables tricks the spam filter into marking this message as safe (right).

Chapter 3

Predicting Malware Attacks using Machine Learning and AutoAI

3.1 Problem Description

One of other cybersecurity problem is malware. Malware is deliberately designed to be hostile, intrusive, and aggressive. It seeks to penetrate the system, inflict damage, partially take over control of some processes, or completely disable computers, computer systems, networks, tablets, and mobile devices. Like a human virus, malware interferes with the normal functioning of a system. The purpose of malware is to make illegal profits at your expense. In addition, malware can damage system hardware and network equipment. Hackers can steal, and encrypt or delete your data and it can monitor computer activity without your knowledge.

Malware is commonly used by cyber criminals as primary attack vectors, and malware proliferation is thus a significant challenge for security professionals to adapt and develop a matching defense mechanism. The prediction of malware attacks remains one of the most challenging problems for companies and academics. Traditional security solutions can not keep pace with the ever-evolving threats that cause damage to critical systems, leading to loss of money, sensitive information, and reputation. The malware threat continues to grow along with the drastic rise in the number of victims due to the growing number of users in cyberspace, financial gains, seeking increased computational power for further attacks (botnets), availability of malware scripts, etc. Panda Security Company revealed in 2015, that 230,000 new malware attacks occurred daily [16]. It is no longer possible for traditional signature-based and heuristics-based technologies to keep pace with malware proliferation due to the vast quantities of malware. In addition, security analysts can not perform a manual analysis on every new malware.

One of the big problems facing anti-malware applications today are the large volumes of data that need to be analyzed for possible malicious intent. Each day people generate and capture more than 2.5 quintillion bytes of data. More than 90% of the data was generated in the last two years and it is approximately 40 Zettabytes or 40 trillion gigabytes [6]. This makes detection of malware more complicated when dealing with such a large volume of data. This applies to Microsoft's real-time anti-malware detection application because it runs on 600 million computers worldwide [4].

Different machine learning methods have been proposed to address the problem of predicting malware attacks. Light Gradient Boosted Machine (LGBM) is the most popular classification technique currently used in detecting malware. Some of the benefits of LGBM are that it is easy to create, easy to understand, and reduces complexity [24]. In addition, with the increase of malware threats in our world, a lot of big companies use AutoAI to help protect their systems. Automated Artificial Intelligence (AutoAI) is a variant of automated machine learning technology that automates the entire life cycle of machine learning [34]. Automation evaluates a number of tuning choices to obtain the best possible outcome, then ranks model-candidates. The best-performing pipelines can be placed into production for processing new data and generating predictions based on model training [26]. Automated artificial intelligence can also be implemented to ensure that the model has no inherent bias and automates the tasks for continuous model development. However, even with the advanced technology of AutoAI, machine learning experts can, at times, obtain better results.

The main repository for our Malware Prediction Experiments can be found at `https://github.com/sokolovm19/malware_prediction`.

3.2 Experimental Design

The goal of our experimental design is to test our framework on Microsoft Malware Prediction dataset and compare the results with AutoAI as well as [30] and [21]. We conducted three experiments. Experiment A is the control experiment, in which the whole dataset was used. In Experiment B, we used LGBM feature extraction to remove less important features and decrease the number of columns in the dataset. We removed the 30 lowest-ranked features, and kept the 84 highest-ranked features. We removed features that have less than 3% of importance on the dataset. In Experiment C, we used Random Forest feature extraction to select the features. We removed 73 lowest-ranked features and kept the 41 highest-ranked features. We removed features that have importance less than 3%.

LGBM feature selection was used to remove less important columns from the training and testing sets to improve score. Importance feature offers a score showing how useful or beneficial each feature has been in the construction of the boosted decision trees within the model. The higher its relative value, the more an attribute is used to make important decisions with the decision trees. For each attribute in the dataset this value is determined directly, allowing attributes to be listed and compared with each other.

3.2.1 Data Source and Format

The experiments were performed using the Microsoft Malware Prediction dataset, the dataset is publicly available on Kaggle website [1]. The data consists of 4,458,892 malware instances and 4,462,591 benign instances. We used only the training dataset from the website since the testing data is unlabelled. The training dataset includes 8,921,483 instances and 83 features. The experiments were run entirely in a Python environment using the scikit-learn machine learning library [22], the LGBM [13], and using pandas library [18] for data manipulation. The following columns 'EngineVersion', 'AppVersion', 'AvSigVersion', 'OsBuildLab', 'Census_OSVersion' were converted into multiple features by using the split function. By doing so, we have created dataset with 114 features.

3.2.2 Dataset

Microsoft made these data publicly available to evaluate the probability of malware infection on Windows machines. The dataset contains Microsoft's Windows Defender telemetry data and the system's infection status, generated by combining heartbeat and threat reports. This dataset contains 4.04 GB of data and has two types of variables: numerical columns and categorical columns [1]. It includes 27 numerical columns and 56 categorical columns. The description of the dataset is shown in Table. 3.1

Column	Description
Machine Identifier	Individual machine ID
ProductName	Defender state information e.g. win8defender
Engine Version	Defender state information e.g. 1.1.12603.0
AppVersion	Defender state information e.g. 4.9.10586.0

IsBeta	Defender state information e.g. 1.217.1014.0
DefaultBrowsersIden-tifier	Defender state information e.g. false ID for the machine's default browser
AVProductStatesIden-tifier	ID for the specific configuration of a user's antivirus software
Has	True if machine has tpm
CountryIdentifier	ID for the country the machine is located in
CityIdentifier	ID for the city the machine is located in
OrganizationIdentifier	ID for the organization the machine belongs in. organization ID is mapped to both specific companies and broad industries
GeoNameIdentifier	ID for the geographic region a machine is located in
LocalEnglish-NameIdentifier	English name of Locale ID of the current user
Platform	Calculates platform name
Processor	This is the process architecture of the installed operating system
OsVersion	Version of the current operating system
OsBuild	Build of the current operating system
OsSuite	Product suite mask for the current operating system.
OsPlatformSubRe-lease	Returns the OS Platform sub-release
OsBuildLab	Build lab that generated the current OS.
SkuEdition	The goal of this feature is to use the Product Type defined in the MSDN to map to a SKU-Edition' name that is useful in population reporting.

IsProtected	This is a calculated field derived from the Spynet Report's AV Products field. Returns: a. TRUE if there is at least one active and up-to-date antivirus product running on this machine b. FALSE if there is no active AV product on this machine, or if the AV is active, but is not receiving the latest updates. c. null if there are no Anti Virus Products in the report. Returns: Whether a machine is protected
AutoSampleOptin	This is the SubmitSamplesConsent value passed in from the service. available on CAMP 9+
PuaMode	Pua Enabled mode from the service
SMode	This field is set to true when the device is known to be in 'S Mode', as in, Windows 10 S mode, where only Microsoft Store apps can be installed
SmartScreen	This is the SmartScreen enabled string value from registry.
Firewall	This attribute is true (1) for Windows 8.1 and above if windows firewall is enabled. as reported by the service.
UacLuaenable	This attribute reports whether or not the administrator in Admin Approval Mode" user type is disabled or enabled in
Census_DC2 FormFactor	A grouping based on a combination of Device Census level hardware characteristics The logic used to define Form Factor is rooted in business and industry + standards and aligns with how people think about their device.

Census_DeviceFamily	As known also DeviceClass Indicates the type of device that an edition of the OS is intended for
Census_Processor CoreCount	Number of logical cores in the processor
Census_Processor Class	A classification of processors into high/medium/low Initially used for Pricing Level SKU.
Census_Primary DiskTotalCapacity	Amount of disk space on primary disk of the machine in MB
Census_PrimaryDisk TypeName	Friendly name of Primary Disk Type - HDD or SSD
Census_SystemVolume TotalCapacity	The size of the partition that the System volume is installed on in MB
Census_asOptical DiskDrive	True indicates that the machine has an optical disk drive (CD/DVD)
Census_Total PhysicalRAM	Retrieves the physical RAM in MB
Census_hassis TypeName	Retrieves a numeric representation of what type of chassis the machine has A value of 0 means xx
Cen- sus_InternalPrimary DiagonalDisplay SizeInInches	Retrieves the physical diagonal length in inches of the primary display

Census_InternalPrimary DisplayResolution Horizontal	Retrieves the number of pixels in the horizontal direction of the internal display.
Census_InternalPrimary DisplayResolution Vertical	Retrieves the number of pixels in the vertical direction of the internal display
Census_PowerPlatform RoleName	Indicates the OEM preferred power management profile.
Census_OSVersion	Numeric OS version
Census_OSArchitecture	Architecture on which the OS is based Derived from OS VersionFull.
Census_OSBranch	Branch of the OS extracted from the Os VersionFull.
Census_OSBuildNumber	OS Build number extracted from the sersionFull
Census _OSBuildRevision	OS Build revision extracted from the
Census OSEdition	Edition of the current OS.
Census OSSkuName	OS edition friendly name (currently Windows only)
Census_SInstallTypeName	Friendly description of what install was used on the machine i.e. clean of the the machine

Census_sWUAuto UpdateOptionsName	Friendly name WindowsUpda auto-update settings on the machine
Census_sPortable OperatingSystem	Indicates whether OS is booted up and running via Windows-To-Gi on a USB stick.
CensusGenuine StateName	Friendly name of OSGenuineStatelID. 0 = Genuine
Census_Activation Channel	Retail license key or Volume license key for a machine
Census_Flights Disabled	Indicates if the machine is participating in flighting
Census_lightRing	The ring that the device user would like to receive flights for.
Census_Secure BootEnabled	Indicates if Secure Boot mode is enabled.
Census_VirtualDevice	Identifies a Virtual Machine (machine learning model)
Census_TouchEnabled	Is this a touch device?
Census_PenCapable	Is the device capable of pen input?
Census_Always OnAlways ConnectedCapable	Retreives information about whether the battery enables the device to be AlwaysOnAlwaysConnected
Wdft_Gamer	Indicates whether the device is a gamer device or not based on its hardware combination
HasDetections	Feature of output file, which contains probability of getting infected by malware

Table 3.1: The data from Microsoft Malware Prediction dataset has 83 columns. In addition to the column names, Microsoft also provided descriptions for the data in each column.

3.2.3 Experiment A

We preprocessed the data, replaced the category variables with the category codes, and replaced the missing values in the numerical columns with their median. Then we converted this to a pandas dataframe. The data was split into two subsets: 80% for training, and 20% for testing, and no columns were removed. Then, we divided the entire training set into five equal sets, and trained LGBM on each one of them. Each model was then used with testing data. We also used these five models in an ensemble setting, using their majority voting.

3.2.4 Experiment B

We modified our testing and training data by removing some of the columns. LGBM feature selection was used to remove columns that have lower impact on the model. Importance provides a score that indicates how useful or valuable each feature was in the construction of the boosted decision trees within the model. The intent here was to select features that are more important than the others. There are many attributes in data, some of the attributes may have irrelevant or partially relevant predictions so if these are selected, it will cause a largely negative impact on the prediction model.

3.2.5 Experiment C

We modified both our training and testing datasets to remove even more columns. Random forest feature selection was used to remove 73 columns. LGBM was trained on training data and used to predict testing data. The intent here was to clean as much data as we could to help LGBM to get higher accuracy. However, it led to the opposite effect as shown in the in Table 3.2, Table 3.3, Table 3.4, Table 3.5, and Table 3.6.

3.2.6 Auto AI

We also tested this dataset with Auto AI. We have used four of the most popular Auto AIs: AutoAI from IBM Watson Studio [12], Auto-sklearn [7], hyperopt-sklearn [2], and TPOT [14]. All Auto AI randomly divided the data set into two subsets for training (80%) and testing (20%).

3.2.7 Model and Model Evaluation Metrics

Each of these models were evaluated using accuracy, confusion matrix, precision, recall, F1 score and confusing matrices. All the results were collected and shown in Table 3.2, Table 3.3, Table 3.4, Table 3.5, and Table 3.6, respectively.

3.3 Machine Learning Algorithms Used

3.3.1 Algorithm

LightGBM [13] is a Gradient Boosted Decision Trees model. Since traditional GBDT consumes a lot of time to find the best split, several approaches have been suggested to reduce overhead efficiency. One can downsample the data, for example, to reduce the size of the training data. However, this requires native weights and can not be applied directly to GBDT. Similarly, decreasing the number of features could be one solution, but can have an impact on accuracy. LightGBM uses two techniques, called Gradient-based One-Side Sampling (GOSS) that reduces data size, and Exclusive

	Experiment A	Experiment B	Experiment C
LGBM 1	67.03%	67.07%	66.58%
LGBM 2	67.29%	67.18%	66.42%
LGBM 3	67.12%	67.20%	66.49%
LGBM 4	67.31%	67.09%	66.77%
LGBM 5	67.18%	67.11%	66.71%
LGBM(100,000)		67.01%	
LGBM(Entire Trainig Set)		67.99%	
Ensemble of LGBM	**69.03%**	68.78%	67.53%
Onodera		64.91%	
Stephan Michaels		66.18%	
IBM Watson		**64.40%**	
Auto-sklearn		64.02%	
Hyperopt-sklearn		62.37%	
TPOT		57.89%	

Table 3.2: Accuracy for Experiment A (no column removed), Experiment B (30 columns were removed), Experiment C (73 columns were removed), Onodera's replicated experiment, Michaels' replicated experiment, IBM Watson used LGBM, Auto-sklearn used LGBM, Hyperopt-sklearn used Gradient Booster, and TPOT used XG-Boost. LGBM (100,000), LGBM (Entire Traninig Set), Onodera, Stephan Michaels, IBM Watson, Auto-sklearn, Hyperopt-sklearn and TPOT are individual results that are not related to Experiment B. IBM Watson is limited to 100,000 instances per experiment so we used the same data for LGBM (100,000). Moreover, LGBM (Entire Training Set) is used to compare results with Ensemble of LGBMs. Notice that Ensemble outperformed all the AutoAI and replicated experiments. This is a significant improvement since it can save more than 400,000 computers from malware attacks.

Feature Bundling (EFB) that reduces the number of features using histogram-based algorithms rather than finding the best split point to solve the GBDT problem.

GOSS keeps all instances with large gradients and conducts random sampling with small gradients on the instances. In order to compensate for the data distribution impact, GOSS introduces a constant multiplier for data instances with small gradients when calculating the information gain. Specifically, GOSS first sorts the data instances according to their gradient's absolute value and selects the top instances. By doing so, without modifying the original data distribution by much, we put more focus on the under-trained instances.

Exclusive Feature Bundling is helping to reduce the number of features without loss of much information. The sparsity of the function space gives us the opportunity to design an approach that is almost lossless in order to reduce the number of features. In particular, many features are mutually exclusive in a sparse feature space, i.e., they never take non-zero values simultaneously. Exclusive Feature Bundling can bundle exclusive features securely into a single feature.

We set the same parameters for all our methods to be able to compare results. The parameters' values were tuned on smaller subsets of data, by maximizing accuracy. Below are the list and description of involved hyperparameters [13]:

- Maximum number of leaves in one tree (num_leaves): 250.
- Number of boosted trees to fit (n_estimators): 6,000.
- Boosting learning rate (learning_rate): 0.02.
- How much data to allow in leaves (min_data_in_leaf): 42.
- Number of features selected in each iteration (feature_fraction): 0.8.
- Frequency for bagging (bagging_freq): 5.
- How much data to select without resampling (bagging_fraction): 0.8.
- Random seed for bagging (bagging_seed): 11.
- Maximum tree depth for base learners (max_depth): -1.

3.3.2 Evaluation Metrics

Evaluation metrics used for evaluation are the same as previous experiment and can be found Section 2.3.2.

3.4 Results and Discussion

LGBM achieved the highest accuracy when no columns were removed. All evaluation metrics used – accuracy, confusion matrix, precision, recall, and F1 score – shown in

Table 3.2, Table 3.3, Table 3.4, Table 3.5, and Table 3.6, respectively.

We replicated the experiment [30] because the authors did not present results on this data. We got the accuracy score of 66.18% which is below our score but higher than the Auto AI as shown in Table 3.2.

Also, we replicated another experiment [21] that was not evaluated on these data. We got the accuracy score of 64.9% which is below our score and very similar to the Auto AI score as shown in Table 3.2.

Experiment A, B and C show that removing features from the dataset is not a good strategy for this particular problem.

IBM Watson is limited to 100,000 instances per experiment so we used the same data for LGBM(100,000). As shown in Table 3.2, Table 3.3, Table 3.5, and Table 3.6, we were able to outperform IBM Watson. LGBM (entire training set) is used to compare Ensemble of LGBMs to one single LGBM model that was trained with the entire training set. As shown in Table 3.2, Table 3.3, Table 3.4, Table 3.5, and Table 3.6, ensemble of LGBMs has a better performance to a single model. As shown in Table 3.2, we outperformed any well known AI by 4-4.5%, which shows that machine learning scientists and data engineers could get better results and save more computers from malware attacks. 4-4.5% does not seem a lot but it is around 416,000 computers that could be saved from malware attack.

3.5 Related Work

In [15], the authors described a method that used Naive Transfer Learning approach on Kaggle's Microsoft Malware Prediction dataset. They trained a Gradient Boosting Machine (GBM) to get a simple prediction model based on the training data, and then fine-tuned it to suit the test dataset. The authors tried to minimize the marginal

Experiment A

Actual \ Predicted	Benign	Malware
Benign	611,585	281,502
Malware	297,831	594,379

Experiment B

Actual \ Predicted	Benign	Malware
Benign	607,327	285,760
Malware	300,831	590,379

Experiment C

Actual \ Predicted	Benign	Malware
Benign	604,409	287,974
Malware	306,482	585,432

IBM Watson

Actual \ Predicted	Benign	Malware
Benign	6,390	3,604
Malware	3,489	6,517

LGBM(Entire Training Set)

Actual \ Predicted	Benign	Malware
Benign	605,174	286,717
Malware	303,612	588,794

LGBM(100,000)

Actual \ Predicted	Benign	Malware
Benign	6,585	3,415
Malware	3,739	6,261

Table 3.3: Confusion matrices for Experiment A (no column removed), Experiment B (30 columns were removed), Experiment C (73 columns were removed), IBM Watson used LGBM, LGBM (with entire training set), and LGBM (with 100,000 instances). IBM Watson is limited to 100,000 instances or 100 Mb of data per experiment so we used the same data for LGBM (100,000). Notice that only Ensemble results are included in the table since they show the highest accuracy.

distribution gap between the source and target domains, figure out the key features for domain adaptation and change the results of predictions according to the general statistical regularities extracted from the training set. They ran a GBM to collect each feature's importance ratings, and then picked 20 of the most important category features for further study. This was done to simplify the problem and reduce the costs of the computation. The first model used 20 most of the most important features, and achieved an accuracy of 63.7%. A second model, in which columns with maximum mean discrepancy were removed achieved an accuracy of 64.3%.

In [27], the authors presented a lightweight malware detection and mobile categorisation security framework. They evaluated the method with malware on Android

	Experiment A	Experiment B	Experiment C
LGBM 1	67.07%	66.82%	66.55%
LGBM 2	67.27%	67.14%	66.46%
LGBM 3	67.18%	67.36%	66.38%
LGBM 4	67.28%	67.02%	66.86%
LGBM 5	67.26%	67.13%	66.78%
LGBM(100,000)		67.02%	
LGBM(Entire Trainig Set)		67.97%	
Ensemble of LGBM	**69.01%**	68.64%	67.57%
Onodera		64.93%	
Stephan Michaels		66.19%	
IBM Watson		**64.43%**	
Auto-sklearn		64.03%	
Hyperopt-sklearn		62.17%	
TPOT		57.95%	

Table 3.4: Precision for Experiment A (no column removed), Experiment B (30 columns were removed), Experiment C (73 columns were removed), Onodera's replicated experiment, Michaels' replicated experiment, IBM Watson used LGBM, Auto-sklearn used LGBM, Hyperopt-sklearn used Gradient Booster, and TPOT used XG-Boost. LGBM (100,000), LGBM (Entire Traninig Set), Onodera, Stephan Michaels, IBM Watson, Auto-sklearn, Hyperopt-sklearn and TPOT are individual results that are not related to Experiment B. IBM Watson is limited to 100,000 instances per experiment so we used the same data for LGBM (100,000). Moreover, LGBM (Entire Training Set) is used to compare results with Ensemble of LGBMs. Notice that Ensemble outperformed all the AutoAI and replicated experiments.

devices. Because of the success and openness of the Android platform, it is constantly under attack. They performed the analysis on a very large dataset consisting of 184,486 benign applications and 21,306 malware samples. They randomly divided the data set into two subsets for training (80%) and testing (20%), and evaluated five classifiers: k-nearest neighbor (KNN), Ada, random forest (RF), support vector machine (SVM), and GBM. The GBM classifier achieved the best accuracy, of 96.8%. Since the Gradient Booster algorithm outperformed all other well known algorithms in predicting malware, we decided to use it on another malware problem.

In [25], the authors used classifiers such as XG-Boost and LGBM to detect net-

	Experiment A	Experiment B	Experiment C
LGBM 1	67.70%	67.27%	67.00%
LGBM 2	68.03%	67.51%	66.76%
LGBM 3	67.97%	67.70%	66.86%
LGBM 4	68.12%	67.58%	67.01%
LGBM 5	68.11%	67.67%	67.34%
LGBM(100,000)		67.21%	
LGBM(Entire Trainig Set)		67.98%	
Ensemble of LGBM	**69.88%**	69.07%	68.06%
Onodera		65.61%	
Stephan Michaels		66.94%	
IBM Watson		**65.10%**	
Auto-sklearn		64.78%	
Hyperopt-sklearn		62.63%	
TPOT		58.52%	

Table 3.5: Recall for Experiment A (no column removed), Experiment B (30 columns were removed), Experiment C (73 columns were removed), Onodera's replicated experiment, Michaels' replicated experiment, IBM Watson used LGBM, Auto-sklearn used LGBM, Hyperopt-sklearn used Gradient Booster, and TPOT used XGBoost. LGBM (100,000), LGBM (Entire Traninig Set), Onodera, Stephan Michaels, IBM Watson, Auto-sklearn, Hyperopt-sklearn and TPOT are individual results that are not related to Experiment B. IBM Watson is limited to 100,000 instances per experiment so we used the same data for LGBM (100,000). Moreover, LGBM (Entire Training Set) is used to compare results with Ensemble of LGBMs. Notice that Ensemble outperformed all the AutoAI and replicated experiments.

work intrusion, and evaluated them using the NSL KDD dataset [5]. The dataset is built on 41 features including basic features, traffic features and content features, and 21 classes of attack. The authors' experimental results showed that Gradient Boosting Decision Tree ensembles like LGBM, XG-Boost, and the stacked ensemble outperformed linear models and deep neural networks. Similar with the previous related work, since ensemble methods outperformed linear models and a deep neural network, we would like to evaluate such methods on a more recent malware problem.

In [30], the author proposed a method for malware prediction. Two models were trained and evaluated using LGBM. With one method, the data set was cleaned and

	Experiment A	Experiment B	Experiment C
LGBM 1	67.58%	67.09%	66.71%
LGBM 2	67.82%	67.27%	66.58%
LGBM 3	67.77%	67.46%	66.65%
LGBM 4	67.92%	67.28%	66.70%
LGBM 5	67.91%	67.47%	67.09%
LGBM(100,000)		67.09%	
LGBM(Entire Trainig Set)		67.78%	
Ensemble of LGBM	**69.63%**	68.77%	67.80%
Onodera		65.57%	
Stephan Michaels		66.73%	
IBM Watson		**64.83%**	
Auto-sklearn		64.54%	
Hyperopt-sklearn		62.32%	
TPOT		58.10%	

Table 3.6: F-1 Score for Experiment A (no column removed), Experiment B (30 columns were removed), Experiment C (73 columns were removed), Onodera's replicated experiment, Michaels' replicated experiment, IBM Watson used LGBM, Auto-sklearn used LGBM, Hyperopt-sklearn used Gradient Booster, and TPOT used XG-Boost. LGBM (100,000), LGBM (Entire Traninig Set), Onodera, Stephan Michaels, IBM Watson, Auto-sklearn, Hyperopt-sklearn and TPOT are individual results that are not related to Experiment B. IBM Watson is limited to 100,000 instances per experiment so we used the same data for LGBM (100,000). Moreover, LGBM (Entire Training Set) is used to compare results with Ensemble of LGBMs. Notice that Ensemble outperformed all the AutoAI and replicated experiments.

string values encoded. Afterwards a LightGBM was trained. With the other method, the preprocessed data from first model was extended with new features. Then, important features were selected and a LightGBM was trained. Finally, an average of the predictions of both models was calculated. We replicated the experiment because the authors did not present results on Microsoft Malware Prediction data [1]. We got the accuracy score of 66.18% which is below our score but higher than the Auto AI.

In [21], the author engineered five features, which were discovered by trying hundreds of engineered variables to increase Time Split Validation. Each variable was added to the model one at a time and validation score was recorded. After every variable was changed to dtype integer, each variable was tested one by one to see if making it categorical increases LGBM validation score. We replicated the experiment and we got the accuracy score of 64.91% which is below our score and very similar to the Auto AI score.

Chapter 4

Conclusion and Future Work

4.1 Conclusion

In this research, we presented two problems that seemed to be solved, yet can be improved: spam filtering and Auto AI.

in Section 2, we presented a method that can be used to avoid detection by a spam filter. With this approach, a sender can replace a limited number of letters from the Latin alphabet with letters from other alphabet(s) that look alike, and in doing so it tricks the spam filter to produce more errors. We evaluated this method using publicly available copies of spam and ham emails, namely from Enron1 data set, with four machine learning algorithms: decision trees, random forests, naïve Bayes, and support vector machine. Our experiments indicate that using a classifier trained on data using Latin alphabet, to classify a message with a combination of Latin and Cyrillic letters leads to much lower classification accuracy compared to the same classifier used with a message with Latin characters only.

Moreover, we tested this method with a Microsoft Business email. We first sent an email containing a lot of keywords frequently encountered in spam emails, and this email was flagged as spam. Then we sent the same email, with some of the characters replaced by their "visually equivalent" characters from Cyrillic alphabet, and this email was delivered to the Inbox. This suggests that this method can currently bypass existing spam filters.

in Section 3, we used gradient boosting decision trees to predict potential malware attacks on different computers, and compared results to the most common Auto AI. Our system has been able to predict malware attacks on Microsoft cloud and got higher accuracy than any well known Auto AI.

4.2 Future Work

As discussed in Section 2, we evaluated this method in the context of spam filtering, this has implications for other text communication and documents, as described in Section 2.4. Examples include avoiding plagiarism detection by automated software, eluding detection when sending malicious messages with instant messaging applications, and a range of other applications that use natural language processing for automatic analysis of text documents.

In future work we plan to evaluate this approach with characters from multiple alphabets. In addition, we would like to investigate the impact of this method with other applications used for text communication.

For the method proposed in Section 3, we plan to evaluate it with different datasets, and further investigate the impact of the different features on the classification accuracy.

BIBLIOGRAPHY

[1] Microsoft Malware Prediction. `kaggle.com/c/microsoft-malware-prediction/data`, Dec 2018.

[2] Bergstra, J., Yamins, D., and Cox, D. D. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28* (2013), ICML'13, JMLR.org, p. I–115–I–123.

[3] Bird, S., Klein, E., and Loper, E. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[4] Caparas, M. J. Threat Protection. `docs.microsoft.com/en-us/windows/security/threat-protection/`, Sep 2020.

[5] Choudhary, S., and Kesswani, N. Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 Datasets using Deep Learning in IoT. *Procedia Computer Science 167* (2020), 1561 – 1573. International Conference on Computational Intelligence and Data Science.

[6] Dobre, C., and Xhafa, F. Intelligent services for Big Data science. *Future Generation Computer Systems 37* (2014), 267 – 281. Special Section: Innovative Methods and Algorithms for Advanced Data-Intensive Computing Special Section: Semantics, Intelligent processing and services for big data Special Section: Advances in Data-Intensive Modelling and Simulation Special Section: Hybrid Intelligence for Growing Internet and its Applications.

[7] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. Efficient and Robust Automated Machine Learning. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2962–2970.

[8] Gad, W., and Rady, S. Email filtering based on supervised learning and mutual information feature selection. In *2015 Tenth International Conference on Computer Engineering Systems (ICCES)* (Dec 2015), pp. 147–152.

[9] Gavankar, S. S., and Sawarkar, S. D. Eager Decision Tree. In *2017 2nd International Conference for Convergence in Technology (I2CT)* (April 2017), pp. 837–840.

[10] Halder, S., Tiwari, R., and Sprague, A. Information extraction from spam emails using stylistic and semantic features to identify spammers. In *2011 IEEE International Conference on Information Reuse Integration* (Aug 2011), pp. 104–107.

[11] Hassan, M. A., and Mtetwa, N. Feature Extraction and Classification of Spam Emails. In *2018 5th International Conference on Soft Computing Machine Intelligence (ISCMI)* (Nov 2018), pp. 93–98.

[12] Hoyt, R. E., Snider, D., Thompson, C., and Mantravadi, S. IBM Watson Analytics: Automating Visualization, Descriptive, and Predictive Statistics.

[13] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Red Hook, NY, USA, 2017), NIPS'17, Curran Associates Inc., p. 3149–3157.

[14] Le, T. T., Fu, W., and Moore, J. H. Scaling tree-based automated machine learning to biomedical big data with a feature set selector. *Bioinformatics 36*, 1 (2020), 250–256.

[15] LIN, C. Naive Transfer Learning Approaches for Suspicious Event Prediction. In *2019 IEEE International Conference on Big Data (Big Data)* (2019), pp. 5897–5901.

[16] Lou, R. PandasLab. `pandasecurity.com/mediacenter/press-releases/all-recorded-malware-appeared-in-2015/`, Jan 2016.

[17] Louridas, P., and Ebert, C. Machine Learning. *IEEE Software 33*, 5 (Sep. 2016), 110–115.

[18] McKinney, W. Data Structures for Statistical Computing in Python . In *Proceedings of the 9th Python in Science Conference* (2010), S. van der Walt and J. Millman, Eds., pp. 51 – 56.

[19] Metsis, V., Androutsopoulos, I., and Paliouras, G. Spam filtering with Naive Bayes-which Naive Bayes? In *CEAS* (2006), vol. 17, Mountain View, CA, pp. 28–69.

[20] Mishra, S., and Malathi, D. Behaviour analysis of SVM based spam filtering using various parameter values and accuracy comparison. In *2017 International Conference on Computing Methodologies and Communication (ICCMC)* (July 2017), pp. 27–31.

[21] Onodera, K. Microsoft Malware Prediction. `github.com/KazukiOnodera/Microsoft-Malware-Prediction`, Mar 2019.

[22] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[23] Peng, W., Huang, L., Jia, J., and Ingram, E. Enhancing the Naive Bayes Spam Filter Through Intelligent Text Modification Detection. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)* (Aug 2018), pp. 849–854.

[24] R, V., Alazab, M., Kp, S., Poornachandran, P., and Venkatraman, S. Robust Intelligent Malware Detection Using Deep Learning. *IEEE Access PP* (04 2019), 1–1.

[25] Rai, M., and Mandoria, H. L. Network Intrusion Detection: A comparative study using state-of-the-art machine learning methods. In *2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)* (2019), vol. 1, pp. 1–5.

[26] Rauf, A., and Alanazi, M. N. Using artificial intelligence to automatically test GUI. In *2014 9th International Conference on Computer Science Education* (2014), pp. 3–5.

[27] Ren, B., Liu, C., Cheng, B., Guo, J., and Junliang, C. MobiSentry: Towards Easy and Effective Detection of Android Malware on Smartphones. *Mobile Information Systems 2018* (11 2018), 1–14.

[28] Shajideen, N. M., and V, B. Spam Filtering: A Comparison Between Different Machine Learning Classifiers. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)* (March 2018), pp. 1919–1922.

[29] Singh, G., Kumar, B., Gaur, L., and Tyagi, A. Comparison between Multinomial and Bernoulli Naïve Bayes for Text Classification. In *2019 International Conference on Automation, Computational and Technology Management (ICACTM)* (April 2019), pp. 593–596.

[30] Stephan Michaels, F. I. Microsoft Malware Prediction on Kaggle. `github.com/imor-de/microsoft\_malware\_prediction\_kaggle\_2nd/tree/master/code`, Mar 2019.

[31] Trivedi, S. K. A study of machine learning classifiers for spam detection. In *2016 4th International Symposium on Computational and Business Intelligence (ISCBI)* (Sep. 2016), pp. 176–180.

[32] Van Rijsbergen, C. J., Robertson, S. E., and Porter, M. F. *New models in probabilistic information retrieval.* British Library Research and Development Department London, 1980.

[33] Vishagini, V., and Rajan, A. K. An Improved Spam Detection Method with Weighted Support Vector Machine. In *2018 International Conference on Data Science and Engineering (ICDSE)* (Aug 2018), pp. 1–5.

[34] Wangoo, D. P. Artificial Intelligence Techniques in Software Engineering for Automated Software Reuse and Design. In *2018 4th International Conference on Computing Communication and Automation (ICCCA)* (2018), pp. 1–4.

[35] Zhang, F. The Growing Role of Machine Learning in Cybersecurity. `securityroundtable.org/the-growing-role-of-machine-learning-in-cybersecurity/`, May 2020.