DTonomy Internship Final Report

07.26.2020

R. Kevin Oberlag
Summer 2020
DSA 5900
Credit Hours: 4
Company: DTonomy
Company Sponsor: Peter Luo

Introduction

Digital technologies have revolutionized the world we live in, completely changing the way we communicate, work and learn. However, with the digital revolution and all of the benefits it can provide, also comes nefarious actors whose aims are to exploit vulnerabilities in either the digital systems themselves, or by exploitation of the users of these systems. The people, processes and technologies that are used to mitigate threats on digital systems and their users form the field of cybersecurity. Cybersecurity is a vastly growing and important domain, containing many facets and problems to which it attempts to resolve. The increase in digital technology use means an increase in the number of available targets and attack vectors. These threats require intervention of humans and technologies, and as the number of threats rise with increasing technology use, they can sometimes exceed the capacity of an organization's ability to address them. My project has been performed for and under the supervision of the cybersecurity company, DTonomy, whose goal has been to address such issues.

DTonomy aims to solve critical problems that security teams must deal with on a daily basis, utilizing automation and artificial intelligence to enable smarter organization of security alerts and incidents. The company describes their technology as follows:

> DTonomy's technology includes a unique adaptive learning engine which continuously learns, provides contextual insights that are not easily discoverable, and makes relevant recommendations and automated workflows to guide IT teams through steps and procedures, resulting in up to 10 times quicker resolution of incidents, decreased downtime, and reduced alert fatigue for staff.

In short, the business objectives of DTonomy are to deliver meaningful insights of security related threats and alerts to customers, reducing the level of convolution while also providing contextual recommendations that would otherwise be difficult to obtain by an individual or team. This project attempts to expand these insights by integrating with the G-Mail web and mobile email applications, which have the potential to reach a larger user base, therefore leading to a potentially larger customer base of DTonomy's main platform. The application that has been developed for this project is called Phish AIR, and is a G-Mail

add-in which provides users an easy to use tool for acquiring important information about suspicious and potentially malicious phishing emails.

Objectives

The insights and recommendations discussed above are currently delivered through DTonomy's online platform, however, it is desirable to also deliver meaningful insights closer to the source of some security incidents, such as through email platforms. Phishing emails are one of the many issues that security teams must face, so the goal in working with DTonomy has been to build an add-in for G-Mail's web and mobile email applications that will help analysts determine whether an email should be classified as phishing or not. The add-in application delivers meaningful security insights, such as IP Address, URL extraction, WHOIS information, screenshots of the effective web pages of the embedded URLs, and geolocation visualizations of the email server routing paths. Having this information available via a G-Mail add-in gives users direct and easy access to important information about the email that can help them determine its potential maliciousness, and take appropriate threat resolution actions.

The objectives included software development of the add-in application, data analysis of a legitimate/phishing dataset to determine important features and indicators of suspicious emails for future application feature development, visual representation by mapping geolocation markers of email routing paths, and implementation of recommended steps of malicious emails.

I. Email add-in software development of core features

The initial objective in developing the add-in was to build a few core features and publish the application to the G-Suite Marketplace. The core features included the ability to extract URLs from an email body and list them as options to the user for further scanning and analysis; scanning of web pages, which would allow the user to obtain information of the effective web page of the URL, such as a verdict on the maliciousness of the web page, the IP address of the web page host server, the country of the web page host server, the domain of the effective web page, and a screen shot of the effective web page, all without the user actually having to visit the

potentially malicious web pages themselves; lastly, functionality that allows the user to view the WHOIS website data for the domain of each URL in order to determine if the hyperlink in their email seems legitimate. Included along with the development of these core features was research on the G-Mail add-in project development platform, Apps Script, as well as research on the third-party APIs to be utilized for the core features. Upon completion of these core features, the application was published on Google's G-Suite Marketplace, which took significantly more research and effort than anticipated. There were weekly demo meetings in which I was able to discuss the progress and accomplishments of the application development, and it was in these meetings that I was able to get the feedback that I needed in order to determine if the specific objective had met the requirements and had been completed. There were several aspects of software engineering that were learned and improved on via this portion of the project, including coding via the Apps Script platform, and documentation research and integration with third-party APIs.

II. Analysis on important features of phishing emails

This objective involved a mix of programming and analysis, requiring data extraction and manipulation skills, along with exploratory data analysis, in order to find the most relevant features for insights into suspicious emails. The dataset used for analysis came from the UCI repository [1] and was accompanied with research details on the features that were engineered for phishing email analysis. The dataset included data from emails flagged as “Legitimate” and “Phishing,” and had features that were engineered based on results modeled from an Artificial Neural Network. The goal of this objective was to discover a few of the strongest features that contributed to the results of the dataset. Utilizing the Python programming language and packages, recursive feature elimination was applied to determine the feature importance score of the 30 provided features in the dataset. Reading the data from its CSV format, manipulating the data by removing specific attributes, modeling the data for recursive feature elimination, and visualizing the data were all aspects involved in the analysis of this dataset. Once the analysis was completed, a report was compiled and shared with the data science team at DTonomy. Feedback from the team helped to determine the completeness of the report. This objective

provided an excellent learning opportunity to apply analytic, reporting and data visualization skills learned in the DSA program.

III. Visual representation of information

The third objective that was completed for this project was the addition of a map visualization to the add-in, which serves the purpose of easily viewing geolocation data related to the email server routing paths. Creating visualizations to better understand a set of data is an important skill of a Data Scientist. This step was an opportunity to learn and develop more skills for visualizing data. This objective also included research on the documentations of Google's Static Map API, and integration of the API with the add-in, as well as research on the documentation of another third-party API, ipgeolocationapi.com, which was used for capturing geolocation data from IP addresses found within email headers.

IV. Implement a recommendation engine

The final objective was to provide recommendations on what actions users should take for potentially malicious emails. In the future, this will utilize insights from objective II, and be programmatically implemented into the add-in to provide dynamic recommendations. As of now, the recommendations are static and are displayed to the user if an email contains URLs which are deemed malicious by the urlscan.io API.

Data

The four objectives above were separated to show chronology of their implementation, however for the explanation of the data, I have separated the software development related tasks and the data analysis tasks into their own subgroups for ensuring a cohesive description of the related data.

I. Email add-in software development

The data environment for the add-in application consists of the following five data sources: G-Mail messages, urlscan.io results, whois.com results, ipgeolocationapi.com results, and Google Static Map API results.

The G-Mail message contains several data points, but the two main data points used within the application are the message body, and the email message headers. The email message body is easily accessible via the API within Google's add-in project development platform, Apps Script. Apps Script is essentially a small framework developed by Google, which exposes parts of the JavaScript language and is run with Google's V8 JavaScript engine. The Apps Script API allows the developer to retrieve the plain text or HTML versions of the email body. The add-in application uses the HTML version for easier identification and extraction of hyperlink elements embedded within the text. By using the HTML content, we are able to use an HTML parser to easily find <a> and <area> HTML elements which contain an "href" attribute and are therefore considered hyperlinks within the message. The API does not handle the email headers quite as easily, so a custom header parser was developed within the application, using regular expressions to parse the separate header properties and their constituent values. The hyperlinks are used for the urlscan.io third-party API and the headers are used to extract IP addresses to help determine geolocations of the email server routh paths. Details are explained in the *Methodology* section of the report.

The urlscan.io data is retrieved for each URL sent to the urlscan.io "submission" API. Once a URL is submitted, the servers for urlscan.io begin to scan the website and perform analysis on whether or not the web page is considered malicious or not. A JSON (JavaScript Object Notation) object is returned to the application with a UUID (universally unique identifier) which is used to obtain the scan result, once it has completed. The UUID is passed to the urlscan.io "result" API which returns a JSON object containing many data points on the analysis of the web page. The data points that are useful to the add-in application are extracted and cached into their own JavaScript object. The data points include a boolean value of whether the web page is considered malicious or not, the IP Address of the domain's host server, the country of the host server, the domain of the effective URL, and a URL of the captured screen shot of the web page. The urlscan.io documentation and

examination of the results provided the necessary understanding of the data objects and their value types.

I have included whois.com as a source of data, as it is presented to the user via the add-in application, however, the data itself is not specifically used within the application. The user is simply presented with a link to the domain results of the scanned URL.

The ipgeolocationapi.com results are retrieved for an email when the email is first opened. The application is specifically interested in the “Received” header, which details the IP address of each server from the source to the destination. By extracting the IP address for each instance of the “Received” header found in the text, we can find the geolocations of each server in the email route. Each IP address is passed to the ipgeolocationapi.com API which returns a JSON object with country and geolocation data. The coordinates and country name data points are cached in a JavaScript object within the add-in application.

Lastly, the Google Static Map API is used to create an image of a map, given coordinates and points of interest, which are used to place markers and set the bounds of visibility of the map. The coordinates returned from the ipgeolocationapi.com API are passed to the Google Static Map API, which then provides a URL to the application for retrieving the image of the newly created map.

The only preparation that was needed for sending and receiving the data for each API was to stringify the JavaScript objects for passing to the API, and parsing the response object once returned from the API, for use as a JavaScript object in the application.

II. Analysis of important features of phishing emails

The data environment for the analysis of important features of phishing emails consisted of a single CSV file from the UCI repo [1]. I had developed a parser in Python, for parsing mbox files (a file format for holding collections of emails) and extracting data points from a set of legitimate/phishing emails, however this analysis was halted and focus was instead shifted to the UCI dataset, as it was

already backed by research and had several engineered features of which to try to pull insights from.

The UCI dataset was accompanied with research documents which explained the process of the modeling that led to the dataset, as well as the rules that were created that represent the engineered features found in the dataset. This documentation helped explain how the features were engineered and the value types to be expected. A Jupyter Notebook with Python was used to work with the data and perform the analysis, and the primary tool for reading, manipulating and preparing the data for analysis was the Python package, Pandas. Using Pandas, the data was read from its CSV format, into a dataframe for easier viewing and manipulation. The data frame was then used to determine the shape of the data, which reveals the number of attributes and observations found in the dataset. Using the functions of Pandas it was determined that the UCI CSV file consists of 32 attributes and 11,055 observations. The data that was used to train the neural network which led to UCI dataset, contained categorical variables for “Legitimate”, “Suspicious”, and “Phishy”, which therefore had to be converted to the numerical values, 1,0 and -1, respectively. The dataset contains 1 attribute for the index, 30 feature attributes, and 1 attribute for the “Result”, or target label. The label attribute has 6,157 values of 1, or “Legitimate”, and 4,898 values of -1, or “Phishy.” The 30 feature attributes each contain either -1, 0, or 1. The results from Pandas, in combination with the research documentation [1], which accompanied the dataset, helped to prepare and understand the data. It was important to implement the techniques that utilized Pandas in order to understand the value types that the data consisted of, the numerical ranges, and distribution of those data points, particularly the labels. After viewing the value counts of the labels it was important to note that the data was fairly evenly spread, but that there could still be some bias, favoring “Legitimate” observations. The usage of this information on a possible bias will be explained in more detail in the *Methodology* section.

The final analysis report consists of two result sets. The same analysis was performed on each result set, however, the first set consists of only features that could be applied directly to an email, whereas the second set includes all features, which have the addition of those that would be applied to a possible phishing web

page. A separate analysis was necessary as the intention is to eventually apply this information to the add-in application, which primarily would use information directly from email messages. Having a subset of the features allows analysis of the most important features of that subset.

Methodology

I. Email add-in software development

A. Techniques

The software development cycle at DTonomy follows a Scrum project management framework, which is useful for small teams and helps break up software development into small goals that can be completed in short time frames, called *sprints*. With this framework, DTonomy has two weekly 15-30 minutes progress meetings on Mondays and Wednesdays, and one 2 hour long demo meeting on Fridays. The short meetings help keep the team apprised of what each team member is working on and provide an allotted time for an exchange of questions and ideas. The demo meeting occurs so that each developer can present and get feedback on the project that they are working on. This is the iterative process that was taken for Phish AIR 's application development.

There are few other "techniques" that I would say applied to the development process. I would have typically tried to implement an Object Oriented programming approach to the software development, but Apps Script does not really allow for this type of development. Apps Script is a subset of JavaScript, but does not allow the creation of modules. Considering that most add-ins are likely small applications, it isn't problematic to write the application as a script, rather than an Object Oriented application.

B. Procedure

As stated in the sections above, the initial task that I was assigned to do for DTonomy was to implement core functionality of the G-Mail add-in. This process involved initial discussions on the core features that should be included in the version 1.0 release, including discussions of other applications with similar functionality, recommended APIs to integrate, research on the G-Suite add-in development platform, Apps Script, and research on the publication and approval process of the G-Suite Marketplace.

After determining the core features, the first step was to research Apps Script. As stated before, Apps Script is essentially a subset of JavaScript. I already had extensive experience working with JavaScript, so my main research objective with Apps Script was to learn the API that it exposes. It wasn't apparent at first that certain functions of JavaScript were not accessible via Apps Script, which created a little bit of a learning curve. G-Suite add-in projects create both a web and mobile version of the add-in, automatically. So, in order to accomplish a continuity of design, and ease the burden on the developer of having a web and mobile version, G-Suite add-ins do not allow for client side JavaScript, or HTML. All user interface functionality is created via pre-made components and exposed via Google's Card Service. This can make development easier, but can also be very limiting. Since add-ins do not allow client side JavaScript, this means everything is rendered on the server, and that asynchronous function calls are not possible. The restriction has made for a slightly less than desirable user experience in regards to scanning URLs. When a user clicks to scan a URL, the entire add-in must be reloaded. The ideal approach would be an asynchronous AJAX call to the server for each scan action, but it is not possible with Apps Script.

The next step was research of the recommended APIs, which included urlscan.io, and whois.com. After learning how to access the APIs and the content they provided, I began writing the initial code. The first step was to get a simple add-in running, which could read an email message. By following tutorials and using the G-Mail Service API that Apps Script exposes,

this was a fairly simple task. The next step was to write the code for extraction of URLs from a message body. I researched various methods, including regex and HTML parsing, and determined that it would be safer to retrieve the HTML version of the message body, and then parse the elements which would represent a hyperlink. Since catching edge cases of embedded URLs in text can actually be somewhat complex, a regular expression would not be as reliable. I researched the possible HTML elements that could be used as a link to a new web page, and determined that the <a> and <area> tags were the relevant HTML elements to parse. By running the HTML formatted message body through an HTML parser, I was able to create a list of all URLs found within the message body, and create a UI for listing the hyperlinks as options to be scanned via the urlscan.io API.

The next step was to integrate with urlscan.io. Using methods exposed by the Apps Script API for making HTTP requests, it was fairly trivial to set up a RESTful API request to the urlscan.io API, and send each of the requested hyperlinks to be scanned. Urlscan returns the majority of the data points that were determined to be essential for the first release, such as a verdict on the maliciousness of the web page, the IP address of the web page host server, the country of the web page host server, the domain of the effective web page, and a screen shot URL of the effective web page. Upon receiving this data, the application caches the data so that subsequent requests for the same URL do not access the API unnecessarily. This helps avoid reaching a rate limit set by urlscan.io, and also makes the application more efficient and responsive. Lastly, the UI components were set up to display the data.

Next, the whois.com integration was set up. The domain information retrieved from urlscan.io is used to create a link in the add-in which allows the user to navigate directly to the WHOIS information for that domain. The only complication here was realizing that hyperlinks to external web pages must be whitelisted in the add-in's manifest file, for security protections enforced by Apps Script.

The above features make up the core functionality of the initial release, so the next step was publication to the G-Suite Marketplace. This was not an easy process, as the documentation was convoluted and confusing. The Apps Script project needed to be converted to a Google Cloud Platform Standard project first. So, I met with my supervisor to have access and permissions granted for this process. Upon converting the project to be hosted on the Google Cloud Platform, the application then needed to go through a two step approval process. One was a security verification, in which we needed to make a YouTube video detailing various aspects of the application which would help Google in their security assessment. The next phase was a branding verification, in which they make sure the application meets their design specifications and guidelines. In addition, they require links to a separate application description page, and drafting of a privacy policy and a terms of service agreement. After preparing the text and having the pages published, the application was finally approved for release to the Marketplace. I worked closely with a graphic designer intern, also working for DTonomy, in order to help with the branding and layout of the application and for creating branding images to be used in the Marketplace listing. Throughout this process, I obtained feedback from the team in the demo meetings, and was able to make the desired modifications and recommendations that were offered. These mostly consisted of simple design and layout changes.

It was at this point that I shifted focus to the analysis of important features which I will describe in the next subsection, however, the next process for the application development involved implementing a map visualization. In my research on analysis of phishing emails, I came across an article that mentioned the usefulness of the email headers in helping to discover potentially malicious emails [2]. Particularly the “Received” header is important, as it can help track the geolocations of the servers which the email has been routed through prior to being received by the user. In order to obtain the appropriate IP addresses from the header, I had to develop a header parser, which would first get the header section of the raw email

content, then parse the separate headers into JavaScript data structures. Since the “Received” header can contain more than one occurrence, it was set up as an array object. Having the parsed objects then allowed me to more easily write code to parse the IP address from the specified “Received” header elements in the array. I created regular expressions to match IP4 and IP6 address formats. Now that the IP addresses of the email routing path were easily accessible, they could be used to find the geolocations of the servers.

In order to get the geolocations of each server, another third-party API, ipgeolocationapi.com was utilized. A simple RESTful API that accepts an IP address as a request parameter, and returns information on the country, and its coordinates. After the coordinates are received, they are used in Google’s Static Map API, the final API of mention for the application.

The Static Map API is accessed fairly easily with the help of the Maps class and its methods, which are exposed via Apps Script. By creating a new static map object with the Maps class, the coordinates can be used to create markers, visibility points and paths on a map. The geolocation coordinates returned above, include precise coordinates and bounding coordinates, which can be used to help set the range of visibility for the world map.

The final process to complete the objectives described above was to implement recommendations to the users when they were presented with URLs that were deemed to be malicious by URL scan. This will eventually become a dynamic and more intelligent feature, but as of now, a static list of steps are presented to the user.

II. Analysis of important features of phishing emails

A. Techniques

Analysis of phishing emails was performed on a dataset found on the UCI Machine Learning Repository [1], which contained 30 engineered features

related to phishing. The following techniques were used to create feature rankings to determine a subset of the features which provided the most importance to the predictive model: Random forest classification, stratified k-fold cross validation, and recursive feature elimination. These techniques paired with some exploratory data analysis made up the full analysis of important features of phishing email.

Random Forests are often used for feature selection in Data Science because of how the mechanics of Random Forests naturally rank by how well they improve the purity of a given node. According to C. Saunders et al., “due to the random exploration of features, Random Forest lends itself to feature selection well and the measure of feature importance is the average information gain achieved during forest construction” [3]. Essentially, nodes with the greatest decrease in impurity, or highest information gain, happen at the top of the trees, while nodes with the least decrease in impurity, or least information gain, happen at the bottom of the trees. By pruning trees below a particular node, a subset of the most important features can be obtained. It is due to these characteristics of Random Forest that it was chosen as the classifier for feature selection.

Cross-validation is an important technique for evaluating a machine learning model, in which a model is trained using a subset of the dataset and then evaluated using a complementary subset of the same dataset. “In *k*-fold cross-validation, the available learning set is partitioned into *k* disjoint subsets of approximately equal size. Here, *fold* refers to the number of resulting subsets. This partitioning is performed by randomly sampling cases from the learning set *without* replacement” [4]. Stratified k-fold cross-validation is important when there is a risk of sampling bias, and seeks to ensure that each fold is representative of all strata of the data. Since the observations of the phishing dataset were slightly biased toward “Legitimate” results, it was determined that Stratified k-fold cross-validation would be the most appropriate validation technique.

Recursive feature elimination is a method of feature selection which fits a model and removes the weakest feature(s). This technique was used in combination with the Random Forest classification model, and stratified k-fold cross-validation, in order to easily determine the values of feature importance.

B. Procedure

In order to apply the above techniques to the analysis of determining important features of the phishing dataset, I utilized Jupyter notebook, Python and Python packages. Specifically, I used the Pandas, Numpy, Matplotlib, Sci-kit-learn, and Yellowbrick Python packages.

Pandas was used to help with reading the data from its source CSV file, manipulating the data, and performing exploratory analysis on the data. I first read the data into a data frame, then viewed the attributes contained within the data frame. Pairing the research document's descriptions of the engineered features with the data frame visualization helped me to gain a better understanding of the available attributes and which attributes were features and which was the target label. Next, observing the shape of the data was important to understand the number of observations and total attributes found within the dataset. Applying the value_counts method from Pandas allowed me to see that the target label attribute, "Result", consists only of values -1 and 1, with 4898 values of -1, and 6157 values of 1. This shows a slight mis-balance in the data and is a reason for choosing the Stratified k-fold cross-validation technique described above. The value_counts method also revealed that the feature attributes contain only values of -1, 0, and 1. The analysis of feature importance was performed twice in this report. The first consisted of a subset of features, which would apply directly to emails, whereas the second included the features that apply directly to emails, as well as those that apply to web pages. It was important to get a distinction, since this analysis will be applied to the add-in, which focuses on email data. Pandas was used to get the email related subset.

Next, a 10-fold Stratified cross-validation object, and a Random Forest classifier, both created with sklearn, were used in the RFECV method provided by the Yellowbrick package. The RFECV method creates a visualizer for recursive feature elimination, which can produce a visualization of the scores achieved with a varying number of features, and the optimal number of features found for the dataset to which the model was fit. With phishing data, the False Negatives and False Positives are crucial, as we wouldn't want to incorrectly identify an observation as legitimate when it is in fact phishing, and we also wouldn't want to incorrectly state that legitimate observations are phishing, therefore the F1 scoring measure is determined to be appropriate and is used as the scoring measure for the RFECV method. I used the feature importance data returned from the visualizer to obtain the top 3 significant features from the model and visualize them in table and plot formats. The same analysis procedures were run for both the email specific features only, and for the email and web page specific features together.

Results and Analysis

I. Email add-in application

There is not any analysis to show for the add-in application development objectives, so instead I will simply show screenshots and details of the product.

The following screenshot shows a list of all embedded URLs that are available for scanning. This is the product as released in version 1.0.

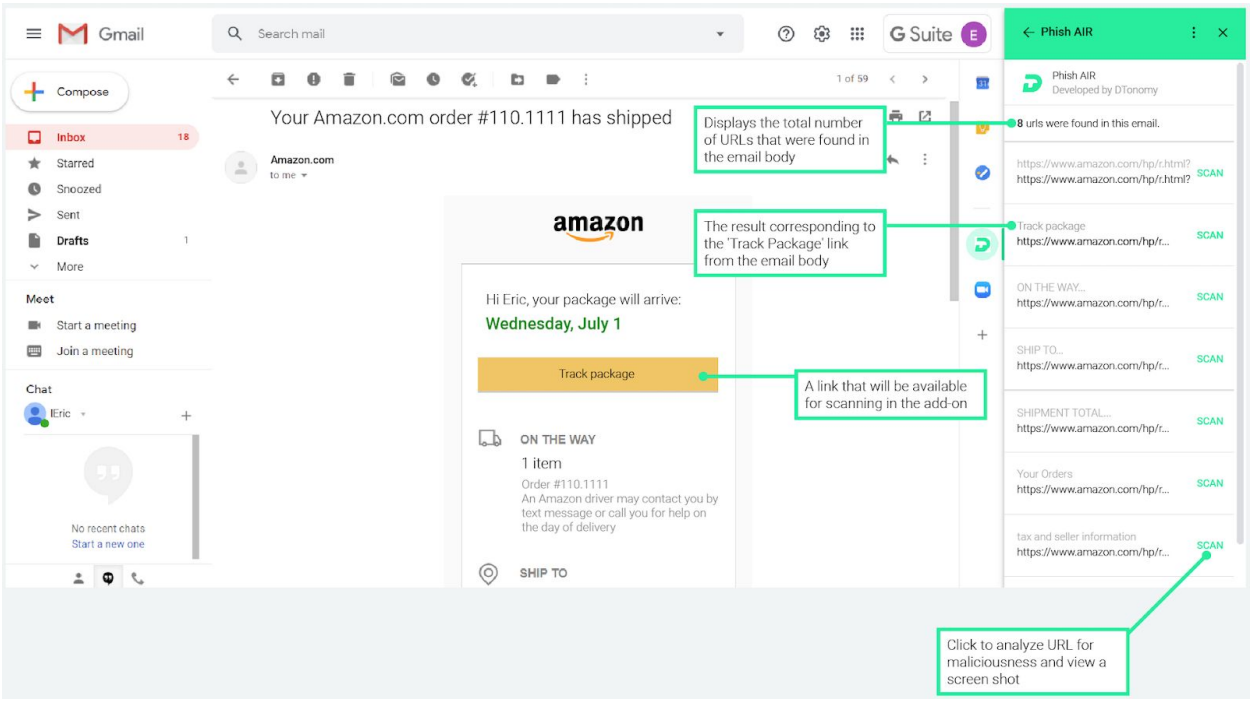

Figure 1. Phish AIR URL Scan Options V1.0.

The following screenshot shows what a user will see once they have completed a scan of a URL. We can see details on the maliciousness of the URL as determined by the urlscan.io API, the IP Address, the country, domain, and screenshot of the effective web page. This is the product as released in version 1.0.

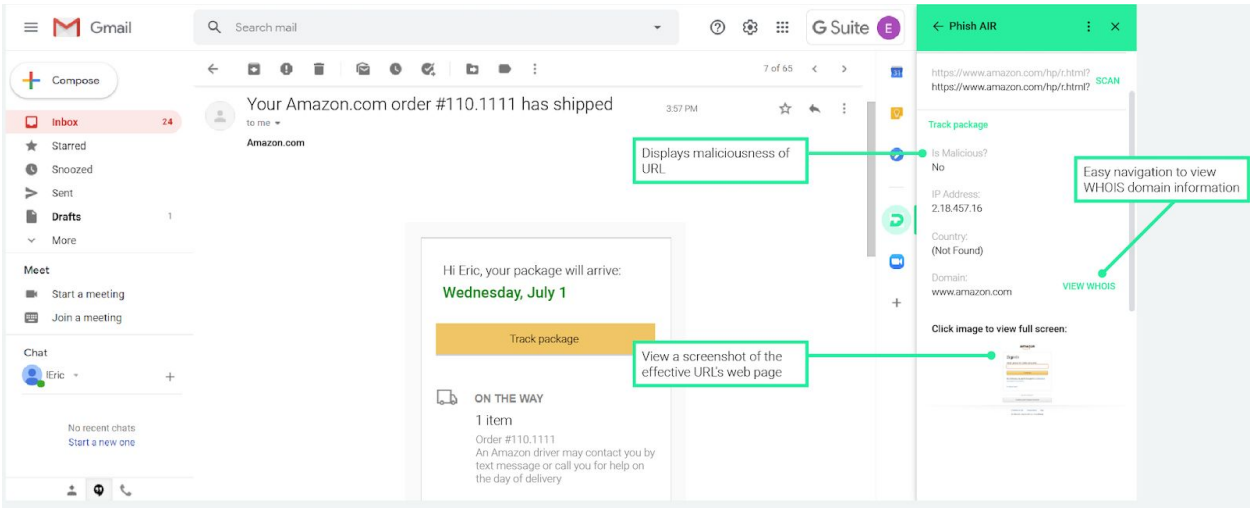

Figure 2. Phish AIR URL Scan Results V1.0.

The following screenshot shows the email routing path that was determined from the IP Addresses of the “Received” email headers. This particular email passed through a single server prior to reaching the destination inbox, so there is not a path, but we can see where the email came from. This is the product as released in version 1.1.

Figure 3. Phish AIR Email Routing Path Map V1.1.

The following screenshot shows the recommended steps for a malicious URL, as determined by the urlscan.io API. This is the product as released in version 1.1.

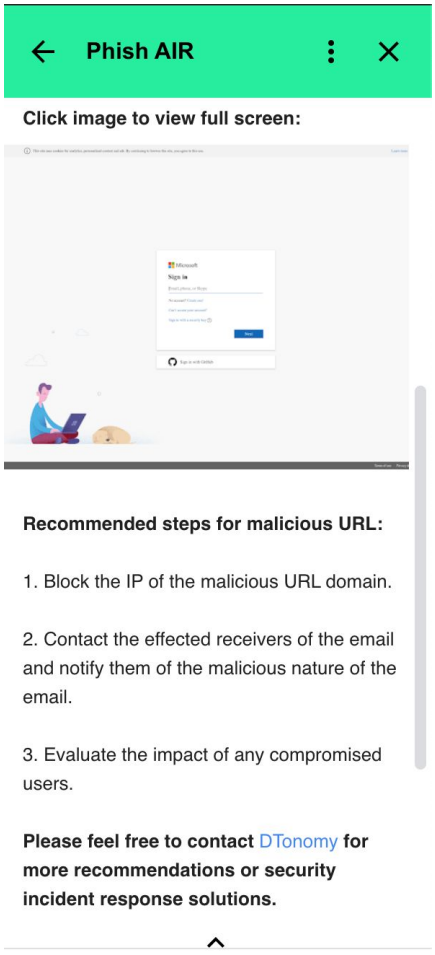

Figure 4. Phish AIR Recommended Step For Malicious URL.

II. Analysis of important features in phishing emails

The following descriptions and visualizations describe the analysis of the important features of phishing emails.

The figure below shows the curve resulting from recursive feature elimination for the random forest classification model. The model was fit with the data that contained a subset of features that could apply directly to email data. The subset consists of only 9 of the original 30 features. We can see that the curve finds that all 9 of the features are used for the optimal RFE cross-validation score of 0.739, however the optimal score is nearly reached with just 2-3 features. The shaded area represents the variability of cross-validation, one standard deviation above and below the mean accuracy score drawn by the curve. This is the type of result we would be hoping to find considering that we are only wanting to find the top 2-3 most important features of the model.

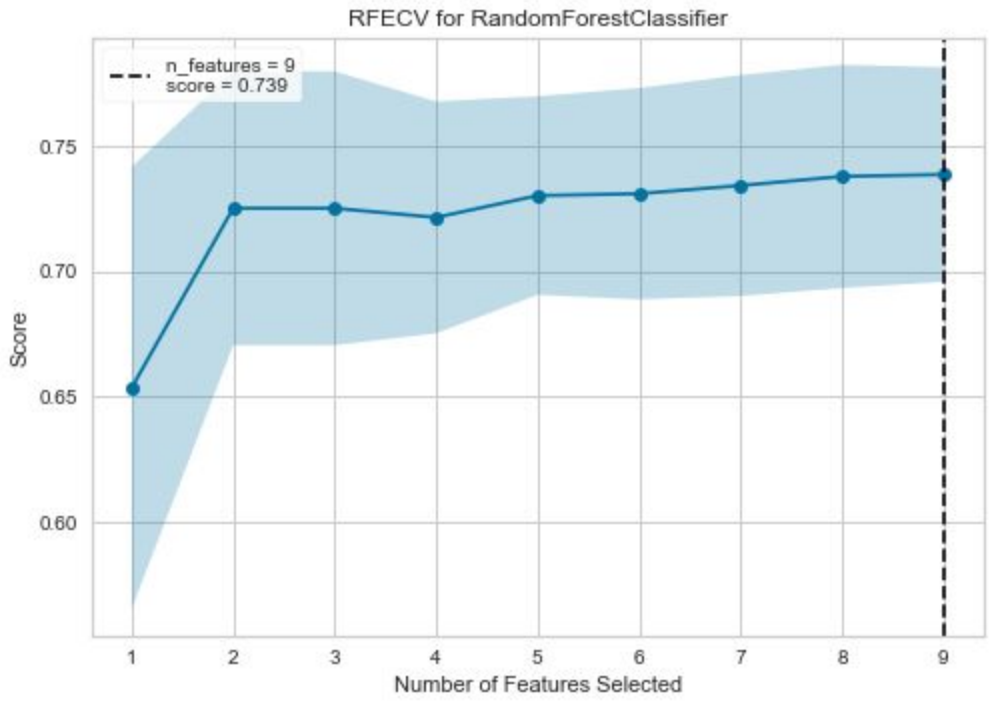

Figure 5. RFECV feature importance scores for email features subset.

The figure below plots out the feature importance score for each of the 9 email related features. We can see from the plot that there are two features that clearly standout as providing the most information gain.

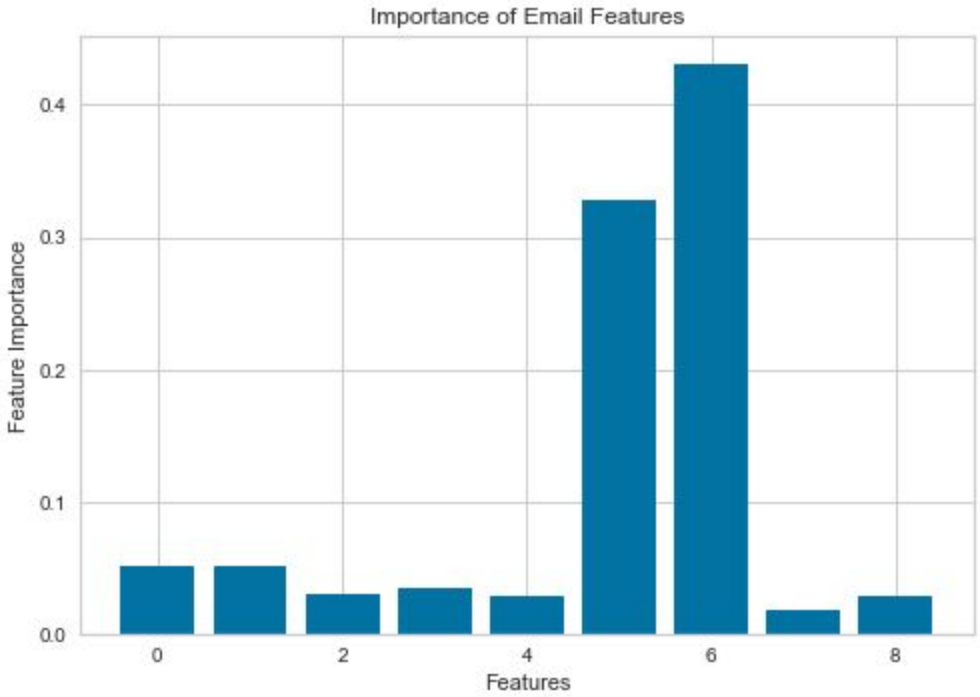

Figure 6. Plot of individual feature importance scores for email features subset.

The table below shows the feature importance scores for each of the 9 features. This table and Figure 7, following the table, help us to see that the features, having_Sub_Domain and Prefix_Suffix would provide the most benefit to the classification model, and would therefore be the most interesting features for further analysis and usage for recommendations and insights for the email add-in application.

Table 1. Individual feature importance scores for email features subset

Feature	Importance
having_Sub_Domain	0.430904
Prefix_Suffix	0.327949
URLURL_Length	0.051535
having_IPhaving_IPAddress	0.051028
having_At_symbol	0.034060
Shortining_Service	0.029805
double_slash_redirecting	0.028943
Abnormal_URL	0.028078
HTTPS_token	0.017696

The following plot reduces the number of features found in Figure 6, in order to better show specifically which were the top 3 features. Along with the table above, Figure 7 helps us to see that the features, having_Sub_Domain and Prefix_Suffix would provide the most benefit to the classification model, and would therefore be the most interesting features for further analysis and usage for recommendations and insights for the email add-in application.

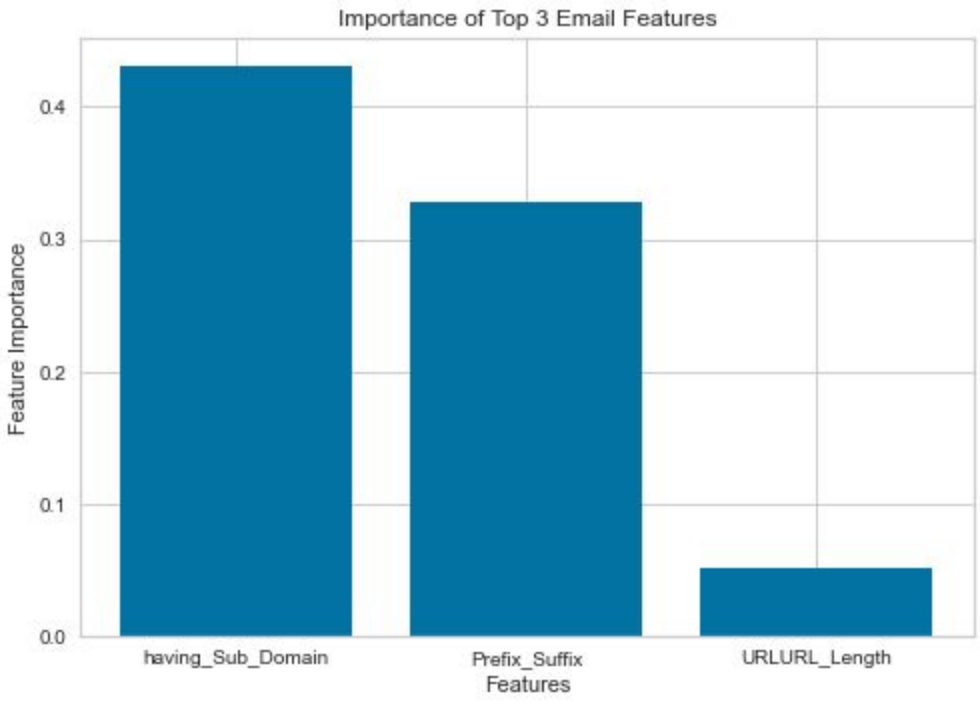

Figure 7. Plot of top 3 individual feature importance scores for email features subset.

The following are the engineering descriptions of the top three email related features, as described by the research notes of the accompanying research for the dataset. The descriptions help to give a better understanding of what the top features represent, and how their values can be determined for a dataset.

> **having_Sub_Domain:** Let us assume we have the following link: http://www.hud.ac.uk/students/. A domain name might include the country-code top-level domains (ccTLD), which in our example is “uk”. The “ac” part is shorthand for “academic”, the combined “ac.uk” is called a second-level domain (SLD) and “hud” is the actual name of the domain. To produce a rule for extracting this feature, we firstly have to omit the (www.) from the URL which is in fact a sub domain in itself. Then, we have to

remove the (ccTLD) if it exists. Finally, we count the remaining dots. If the number of dots is greater than one, then the URL is classified as “Suspicious” since it has one sub domain. However, if the dots are greater than two, it is classified as “Phishing” since it will have multiple sub domains. Otherwise, if the URL has no subdomains, we will assign “Legitimate” to the feature.

Prefix_Suffix: The dash symbol is rarely used in legitimate URLs. Phishers tend to add prefixes or suffixes separatedby (-) to the domain name so that users feel that they are dealing with a legitimate web page. For example http://www.Confirme-paypal.com/

URLURL_Length: Phishers can use a long URL to hide the doubtful part in the address bar. For example: http://federmacedoadv.com.br/3f/aze/ab51e2e319e51502f416dbe46b773a5e/?cmd=_home&dispatch=11004d58f5b74f8dc1e7c2e8dd4105e811004d58f5b74f8dc1e7c2e8dd4105e8@phishing.website.html.

The figure below shows the curve resulting from recursive feature elimination for the random forest classification model. The model was fit with the data that contained all 30 features of the dataset, those relating directly to emails, as well as those relating to web pages. We can see that the curve finds that all 30 of the features are used for an optimal RFE score of 0.972, however an excellent score is reached with just 2-3 features. The shaded area represents the variability of cross-validation, one standard deviation above and below the mean accuracy score drawn by the curve. This is the type of result we would be hoping to find considering that we are only wanting to find the top 2-3 most important features of the model.

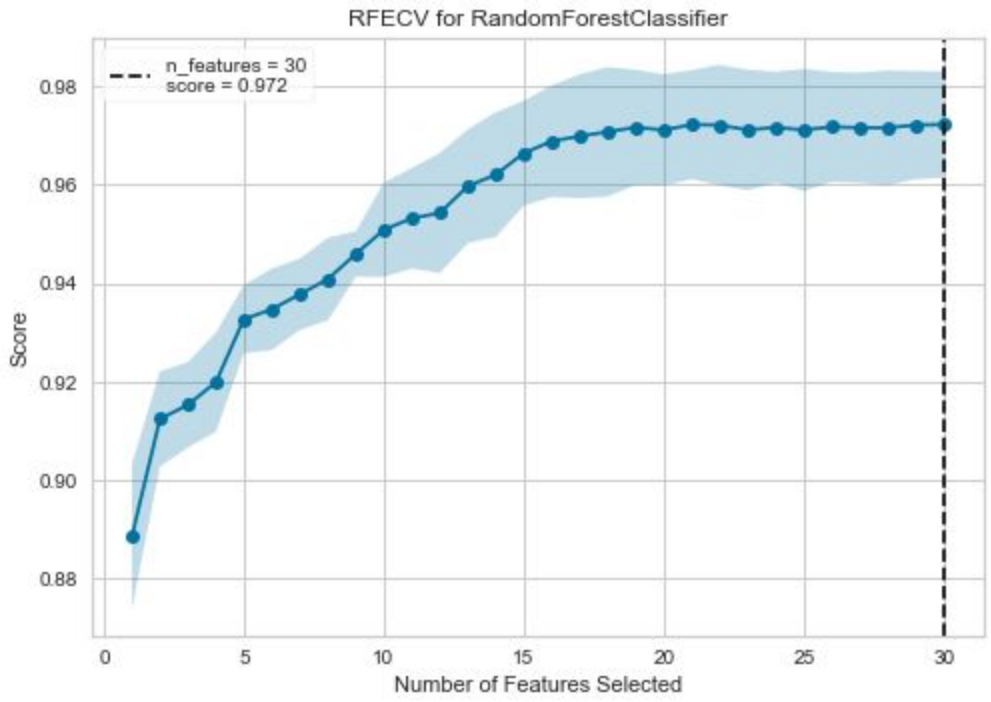

Figure 8. RFECV feature importance scores for all features.

The figure below plots out the feature importance score for each of the 30 features. We can see from the plot that there are two features that clearly standout as providing the most information gain, and a few others that provide moderate information gain.

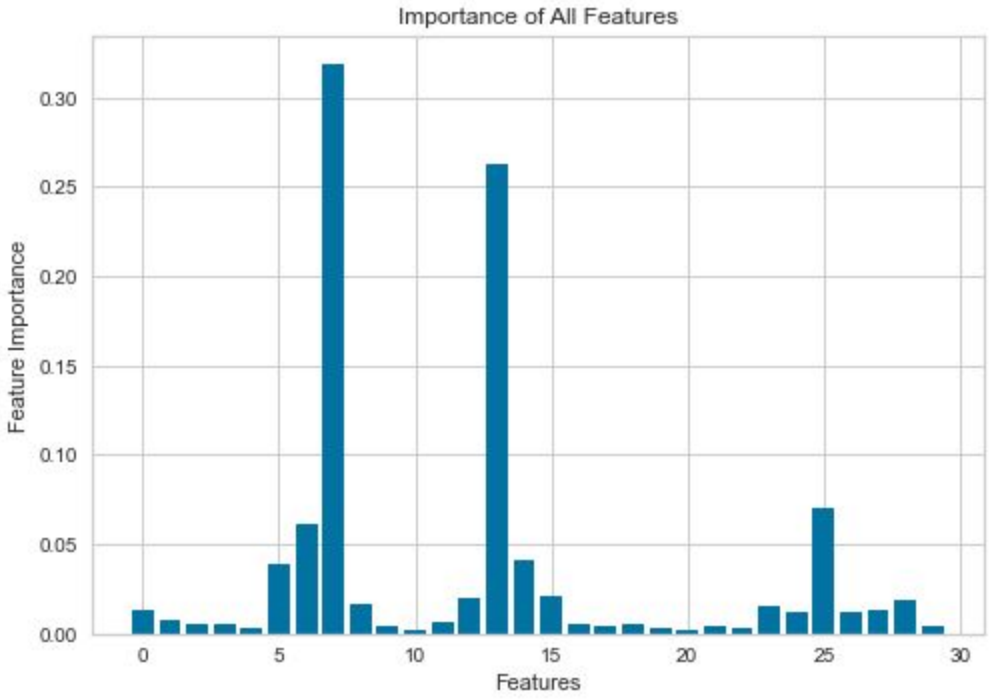

Figure 9. Plot of individual feature importance scores for all features.

The following plot reduces the number of features found in Figure 9, in order to better show specifically which were the top 3 features. Figure 10 helps us to see that the features, SSLfinal_State, URL_of_Anchor, and web_traffic would provide the most benefit to the classification model, and would therefore be the most interesting features for further analysis and usage for recommendations and insights for the email add-in application.

Figure 10. Plot of top 3 individual feature importance scores for all features.

The following are the engineering descriptions of the top three features, as described by the research notes of the accompanying research for the dataset. The descriptions help to give a better understanding of what the top features represent, and how their values can be determined for a dataset.

> **SSLfinal_State:** The existence of HTTPS is very important in giving the impression of website legitimacy, but this is clearly not enough. It has been suggested to check the certificate assigned with HTTPS including the extent of the trust certificate issuer, and the certificate age. Certificate Authorities that are consistently listed among the top trustworthy names include: “GeoTrust, GoDaddy, Network Solutions, Thawte, Comodo,

Doster and VeriSign". Furthermore, by testing out our datasets, we find that the minimum age of a reputable certificate is two years.

URL_of_Anchor: An anchor is an element defined by the `<a>` tag.

This feature examines:

1. If the <a> tags and the website have different domain names.

2. If the anchor does not link to any web page, e.g.:

   A. <a href="#">

   B. <a href="#content">

   C. <a href="#skip">

D. <a href="JavaScript ::void(0)">

web_traffic: This feature measures the popularity of the website by determining the number of visitors and the number of pages they visit. However, since phishing websites live for a short period of time, they may not be recognized by the Alexa database. By reviewing our dataset, we find that in worst case scenarios, legitimate websites ranked among the top 100,000. Furthermore, if the domain has no traffic or is not recognized by the Alexa database, it is classified as "Phishing". Otherwise, it is classified as "Suspicious".

Deliverables

I. Email add-in application

The primary final deliverable for my work with DTonomy was the development of a G-Mail plugin which delivers augmented intelligence to security analysts as an easy to use tool for investigating suspicious and potentially malicious phishing emails. When security analysts are notified of a suspicious email there are typical steps that are taken that can help them determine if an email is legitimate or not, including reviewing the domain of the URLs to ensure that it isn't a spoofed web page, viewing

a screen shot to help determine legitimacy of a web page, and viewing the email routing path to understand where an email has come from prior to reaching the destination. Delivering this easy to use functionality that is accessible directly in their email platform was the primary goal of this project. In addition to developing an initial version, the add-in needed to be published to Google's G-Suite Marketplace, for public access and usage.

I provided DTonomy with detailed documentation on accessing the coding environment, which for Apps Script projects is a cloud hosted IDE from Google; code documentation for explaining the functionality of the program; deployment steps for production deployment of new versions. In order to publish the application initially, significant research had to be performed on publication of a G-Mail add-in to the Marketplace. So, in addition to delivering the application itself, I also provided DTonomy with insights and documentation on the necessary steps and components involved in preparing an application for verification and approval in the Marketplace. As mentioned earlier in the report, this process was not simple, and had confusing (and sometimes contradictory or outdated) documentation spread around multiple web pages. I created a short-list of the steps involved for preparing an application for its initial publication and delivered it to my supervisor.

The application, Phish AIR, is accessible to the public via Google's G-Suite Marketplace (https://gsuite.google.com/marketplace/app/phish_air/682516813925), and provides the benefit of delivering analysis and investigative tools to users that need to inspect suspicious emails, while also providing another platform that can increase recognition of DTonomy as a company, and the types of cybersecurity and threat analysis services they can provide.

II. An analysis report on important features of phishing emails

Another deliverable that was requested by DTonomy was to provide analysis and extract a few important features that can be used to help determine "phishiness" of an email. I conducted research on various papers found from academia and shared interesting findings to the AI team at Dtonomy. A dataset from the UCI Machine Learning Repository [1] backed with detailed research was used to perform analysis,

and extract 2-3 important features out of a set of 30 engineered features. I delivered an organized and documented report, which shows meaningful explanations and visualizations on the most important features.

Determining these important features helped to create a starting point for delivering insights and reasons about an email's "phishing" verdict. This analysis can be applied to the add-in for future versions, as well as be used in future analysis of DTonomy's main platform, DTonomy AIR.

References

[1] Mohammad, R. A., McCluskey, T. L., & Thabtah, F. (2015, March 26). *UCI machine learning repository: Phishing websites data set*. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/Phishing+Websites

[2] Hale, M. L., Gandhi, R., & Morrison, B. B. (2017). *Phishing - Email header analysis · Nebraska-gencyber-modules*. Phishing - Email Header Analysis. https://mlhale.github.io/nebraska-gencyber-modules/phishing/email-headeranalysis

[3] C. Saunders et al. (2005). *Subspace, Latent Structure and Feature Selection, Lecture Notes in Computer Science vol. 3940*, pp. 173–184.

[4] Berrar, D. (2019). *Cross-Validation*. ScienceDirect.com | Science, health and medical journals, full text articles and books. https://www.sciencedirect.com/topics/medicine-and-dentistry/cross-validation

Self-Assessment

When starting this project I had hoped to augment my software engineering skills as well as my data analysis skills. My internship with DTonomy has allowed me to not only practice and improve on these skills, but also develop real-world experience developing a usable tool that is now available to the public in the G-Suite Marketplace. In addition, I have provided insights and analysis on important features of phishing data for the DTonomy team to further implement.

I accomplished augmenting my software engineering skills through the hands-on experience of the application development process, as well as by learning a new, useful and job marketable platform for developing G-Suite add-ins. I had to learn the new platform independently, as I had no prior experience working with Apps Script. In addition, I gained useful experience working with documentation and third-party API integration, which also had to be learned independently, as they were APIs with which I had no prior experience. Another important learning experience that I was exposed to is how a small team at a start-up functions and collaborates, utilizing the Scrum framework, and GIT source control tools. Collaboration and code sharing via GIT and BitBucket source control tools allowed me to gain an understanding of the branching and pull-request workflow that is often found with development teams.

I was able to practice my data analysis skills and implement knowledge gained from the DSA program to perform important programming and analysis on phishing data. Particularly, practice working in Jupyter notebooks, and working with Python and associated Data Science packages, such as Pandas and Sci-kit Learn. By using these tools I created reports with visualizations and implemented machine learning models to better understand important features within a dataset. I independently sought out techniques for feature selection and applied those techniques with the tools learned from the DSA program. I was able to use the knowledge from the DSA program to interpret the results of my analysis and create easy to understand reports.

This project was performed for and under the supervision of DTonomy and was used to meet the four credit-hour, Master of Data Science and Analytics *Professional Practice* curriculum requirements at the University of Oklahoma. I was hired by DTonomy as an unpaid Machine Learning Software Engineer Intern, on June 3rd, and will work until August 7th. My supervisor, Peter Luo, is the CEO of DTonomy, Inc. and can be reached by email at pchluo@dtonomy.com.

.