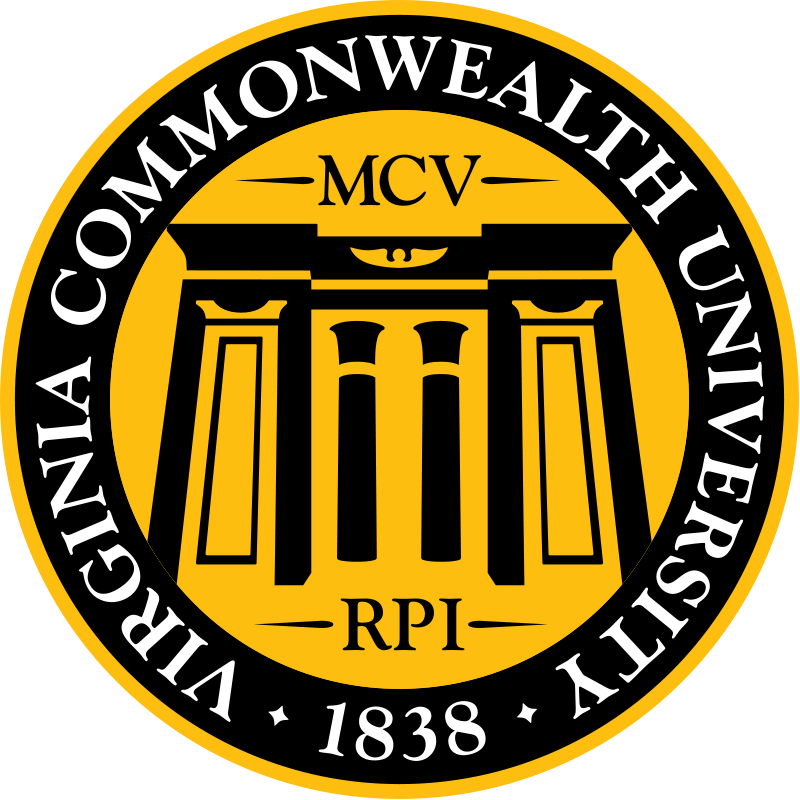
**EMAIL ANOMALY DETECTION**

**CAPSTONE SENIOR DESIGN 2019 - 2020**



### 

### **PREPARED BY**

Bharat Venkat

Jeremy Bailey

Liam Fernando

Mohammad Malik

**TABLE OF CONTENTS**

**Executive Summary 2**

**Project Proposal 3**

Technical Volume 3

Management Volume 3

Cost Volume 3

Resource Volume 3

**EXECUTIVE SUMMARY**

# The exfiltration of sensitive information is when an authorized individual extracts data from a secured system and either shares it with unauthorized third parties or moves to an insecure system. Organizations like Capital One contain large amounts of high-value data that outside threats would be eager to obtain. To reduce the risk of data exfiltration, organizations must integrate security awareness and implement tools to prevent exfiltration from happening. The Virginia Commonwealth University Senior Design group proposes to create a tool that can scan the contents of an email to detect potential data exfiltration attempts.

# The email anomaly scanning tool will be able to detect Personally Identifiable Information (PII) and sensitive data in email and email attachments. Utilizing machine learning concepts such as Natural Language Processing (NLP), a field focusing on computer interactions with human language, the tool will be able to detect data exfiltration attempts more quickly and efficiently. Attachments that are contained in emails will also be scanned to ensure that they do not contain any protected information. Image attachments specifically will be scanned using image scanning machine learning models that the tool utilizes. The tool will also contain additional features that would help Capital One protect their email traffic in various departments.

# The implementation of this tool will increase the defensive capabilities of Capital One and other companies trying to combat against data exfiltration via email. Incorporating a Machine Learning models to detect possible exfiltration attempts will increase the identification of sensitive data in emails. The tool will be able to filter through large volumes of emails and identify what type of information is being exfiltrated. Using this information a companies security team can create effective countermeasure to prevent further exfiltration and identify potential attackers.

## 

**PROJECT PROPOSAL**

## **Technical Volume**

This Section covers the setup and detailed description of the tool. Each feature implemented in the tool is described in detail, providing code level understanding of how it was created. When first creating the tool an environment needs to be set up to be able to run it. To do this we needed to use special libraries to achieve our goals.

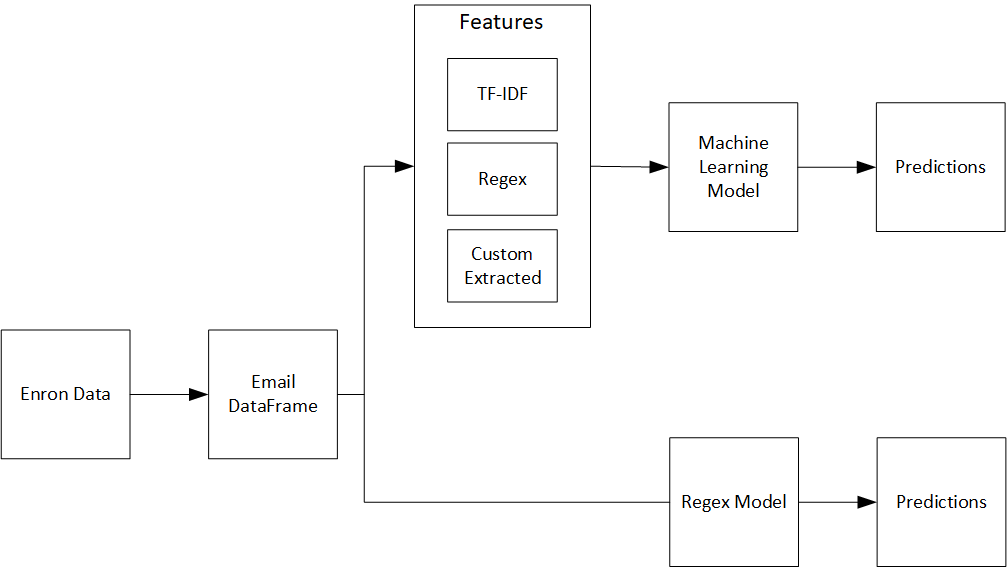
**Environment setup**

The first task in building the email anomaly detection tool was to create a virtual environment and set up some programs that would be used to create our tool. Listed below are the programs that are required to be set up before creating the tool.

* Anaconda
* Python3
* Spark specifically PySpark inside Anaconda
* Scala

Anaconda creates a virtual environment that is used for access and manage machine learning libraries, and packages. Python3 is a high-level, interpreted, interactive and object oriented scripting language that was used in development. Pyspark is the collaboration of Apache Spark and Python that acts as a clustered computing framework built around speed, ease of use, and streaming analytics. Finally, Scala is a general-purpose language providing support for functional programming and strong static type system.

## **Architecture Diagram**



**Dataset Creation**

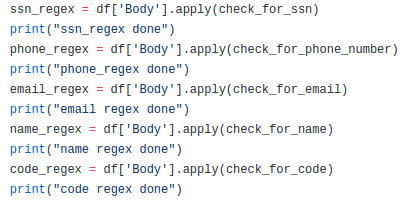
The dataset we are building our model from is the Enron Email dataset. This dataset is widely used because it is a very large repository of real emails. An issue that we have had with this dataset is that it has already been cleansed of sensitive information: what we want to predict with our model.

To process the dataset, we’ve parsed each email into a dataframe. Dataframes (both Pandas and PySpark) are easy to work with and process. Sensitive data has been injected (using the Python Faker library) into selected emails. These emails were then labeled as Anomaly emails. We also prescreen the emails before injecting data to see if they already have sensitive content. If they do they are also labeled as Anomaly emails.

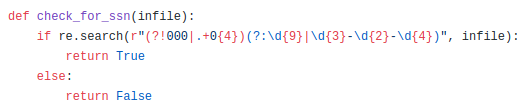
Our dataset is balanced (even number of positive and negative samples). The relevant data that we decided to work with is data from the body of the emails (necessary because our injection of data was random: using other info like sender would create false correlations that the model might pick up on).

**Regex Model**

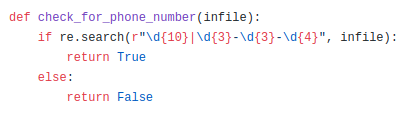
We have also created a regex model to be able to determine if sensitive information is within our dataset. An example of what type of information is being detected by the regex can be seen below. This model serves as a baseline for the machine learning model. We used pandas to create multiple series of each sensitive label and afterwards we would combine the series together to create a dataframe.

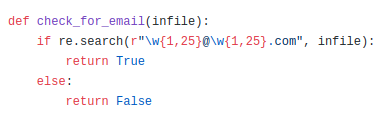


Each of these types of checks are done on the body of the email using lambda functions. These functions contain regex pattern matching to detect is a body of text contains a specific pattern. A detailed view on how the regex checks if an email contains a social security number is shown below.

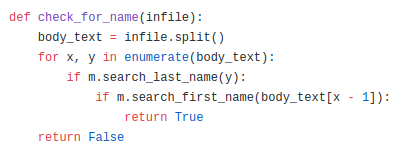


As the name implies, the following functions will detect phone numbers and emails respectively.





We also wanted to be able to check for full names, this would be more difficult as there are many different names and a decent chance of false positive matches. We found a name-dataset containing over one hundred thousand first and last names to cross reference to. Using pip install we added this module to run on the body of text for full names.



After running the regex, the series are combined together into a dataframe and then the dataframe is exported as a CSV file for reading.



**Feature Extraction**

Using PySpark we created User Defined Functions (UDFs) to extract features from our dataset such as, length of email, types of sensitive data, and feel of an email to name a few. Each feature will be described below and information on how the feature was implemented.

**1. Email Details BB**

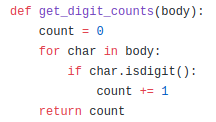
*Length of an email using the built in len() function in python3*



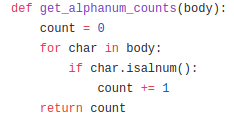
*Number of lines in an email by looking for the count() function in python3*

**

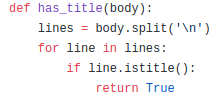
*Number of numeric characters by iterating through each character to see if it is a digit*

**

*Number of alphanumeric characters by iterating through each character and checking with the built in isalnum() function in python3*

**

*If there is a title in an email using the built in istitle() function in python3*



// Template for future implementation.

Regex Based Features

In addition to the aforementioned features we also took elements from the rule based model described earlier, and incorporated each of the regex checks as a feature.

TF-IDF

The last type of feature set we used was Term Frequency - Inverse Document Frequency. TF-IDF is very popular in NLP tasks, and is a measure of how important a given word is to a document in a collection. We thought TF-IDF would be a good feature set to play with because it would give us information on specific words that possibly could indicate some type of exfiltration attempt.

Modelling

PySpark has out of the box support for a number of classification algorithms. We tried Logistic Regression for its simplicity first: it is easily understandable and we thought it would be a good starting point. We then moved on to Support Vector Machine, a more powerful, but black box algorithm. The final algorithm we tried was Gradient Boosted Trees (GBT). GBT is one of the more powerful classical machine learning algorithms, providing good results. This comes at a tradeoff at being very computationally expensive. We were limited in our modeling architectures because the Capital One team wanted to work with PySpark models exclusively. We wanted to explore some deep learning models but PySpark doesn’t have support for these

Evaluation:

Our best performing model was the Gradient Boosted Trees trained on a combination of Regex features and custom extracted features. We evaluated our model using 5 fold cross-validation. We parameterized our model using a grid search. The optimal model had results of:

Test Area Under ROC 0.79991494991495

TP: 78

TN: 553

FP: 19

FN: 107

Sensitivity: 42.16216216216216

Specificity: 96.67832167832168

predictiveACC: 83.35535006605019

MCC: 0.4993564628245596

We think this is a good starting model and aim to improve it in further iterations.

## Management Volume

The implementation and management of this tool will be for Capital One and its security managers. This tool can be added to the list of tools that Capital One utilizes to ensure that data exfiltration does not occur. The use of this tool can be useful to other organization that needs to implement machine learning tools that combat against data exfiltration. This tool can continue to evolve into a more powerful tool able to quickly identify PII in emails and their attachments.

Stakeholders include employees at Capital One and clients of capital one with sensitive data.

Users include management/oversight personnel at Capital One (Blackbear team) whose responsibility is the protection of sensitive data.

## Cost Volume

This section outlines the costs and tools that we used during the course of the project to help save time. Each tool provides a description of what the tool is and how it was used during the course of the project.

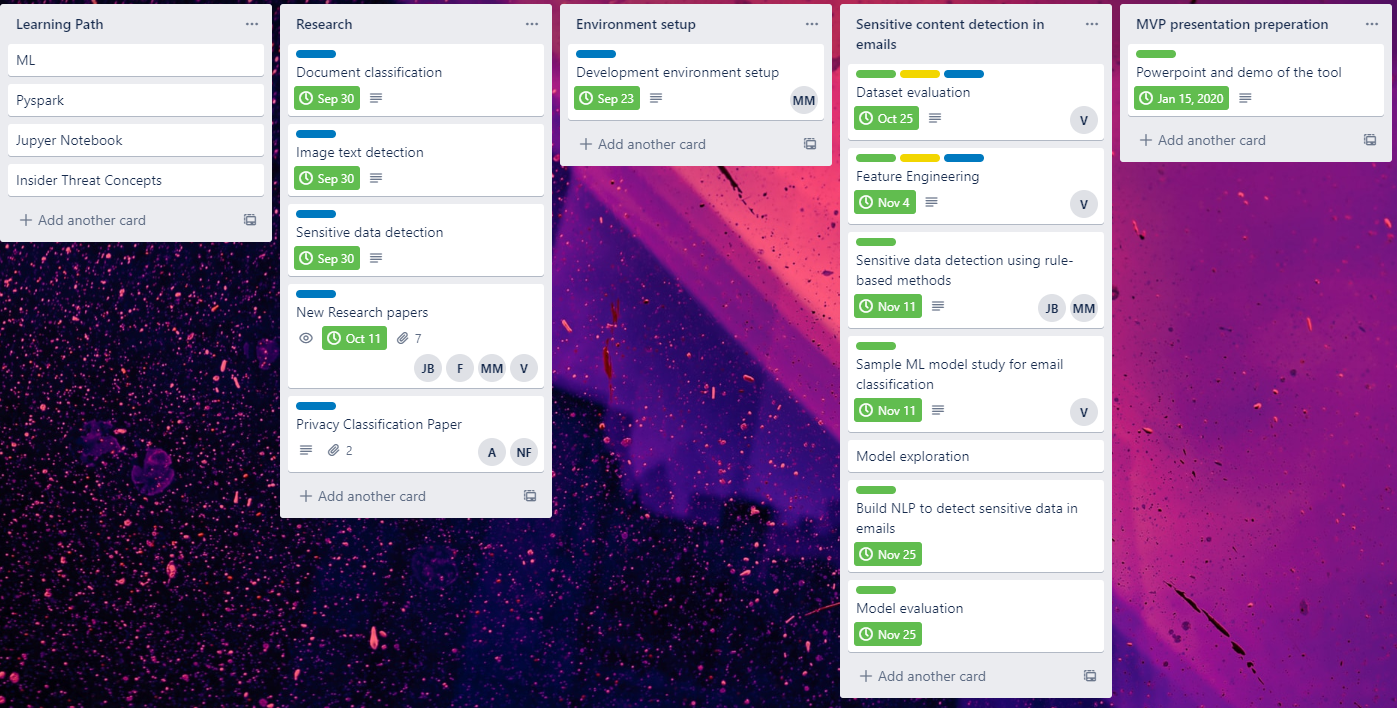
**Faker**

A Python package that helps generate fake data. Using this package we were able to create sensitive data that our tool could find. The data was injected into our clean dataset so that we could use regex to detect.

**Trello**

Trello is a collaboration tool that helps organize projects into boards so that a glance you will know who is working on what and where something is in a process. We utilized this board to help all parties connected to the team what was being worked on and what needed to be done.

A sample of what the board looked like during the project is as follows,



**Enron Email Dataset**

The Enron email dataset contains data from about 150 users, mainly senior management of Enron. The dataset combined with faker helped us create a dataset of emails with sensitive information that we could use to teach our machine.

Currently the development of this tool has not required any monetary costs to enable us to create the product. With time and effort the current state of the development can be achieved.

**Jupyter Notebook**

The Jupyter Notebook is an open-source web application that allowed us to create and share documents that contain live code, equations, visualizations, and narrative text. It was used throughout the development process to help visualize what parts of the code were doing so that members of the team could follow.

**Pandas and Scikit-Learn**

We started off using the Pandas python library for data manipulation. Once the data was formatted correctly we used the Scikit-Learn library to model the data. These libraries are very powerful but because of the sheer volume of data the tool must be able to handle, we transitioned to PySpark, a tool that can be used to manipulate and model the data in a distributed fashion.

**PySpark**

PySpark is the Python library for Apache Spark, a distributed computing platform. PySpark has built in support for working with massive data, as well as common data science functionality and algorithms. We used PySpark because the volume of data going to be fed into this tool is very large.

// Notes

*-cost to make this happen*

*-tools that we used to speed up the process*

## **Resource Volume**

During the course of this project various support in the development process came from the sponsors at Capital One. They provided guidance to clarify the requirements and expectations of the project. Maintaining clear lines of communication with the sponsors helped steer the project away from misunderstandings.

The biggest hurdle during this project was learning about machine learning and data science. Some of us were in the Intro to Data Science course during the semester and so were able to learn a significant amount while working on the project, but there was still a tremendous amount of knowledge we had to learn on our own. Because Natural language Processing is a subset of machine learning and data science, it created an even wider knowledge gap we had to overcome. The sponsors recommended having taken the Intro to Natural Language Processing course at VCU, however, it is only offered in the Spring so none of us had the chance to take it.

*-learning about NLP*

*-Sponsor support*

*-team support*