Using Cluster Analysis to estimate Difficulty Level of Phishing Email Questions

School of Engineering and Computer Science

University of Ottawa

Master of Digital Transformation and Innovation

Graduate Project Report, Winter 2021

By Tauseef Tasin Chowdhury (300144686)

Submitted to Dr. David Knox in fulfilment of the requirement of DTI 6997

Abstract

There has been a significant boom in e-learning right after the pandemic. Students are relying on e-learning tool for the exams, workers are using it for training purposes. One of the key issues of e-learning is efficiency of learning. Depending on a user's knowledge and ability, their learning rate might be different. On an ideal scenario, each user gets assigned the ideal question based on their knowledge and difficulty of the question making the learning efficient. For that to happen, questions should be assigned a difficulty level. In this project, we use a tool called 'Phishducation' which is used to train employees about phishing emails. The gathered data was then used in a k-means clustering algorithm to cluster them based on their feature set. Clustering analysis was done to find relations within and between the clusters. The analysis was then used to assign difficulty levels to each cluster, validated by the user recorded difficulty level.

Table of Contents

Introduction

The way we learn and consume knowledge has changed quite significantly over the last few years. During the pandemic, students were forced to participate in online lectures, quizzes, and exams while employees had to participate in online meetings. Even before COVID, the e-learning market was starting to grow significantly. In US for example it was estimated to grow 6.22 Billion USD between 2017 and 2022 (*E-Learning Market in US by Product and End-user – Forecast and Analysis 2021-2025* 2021). The importance of online learning has increased dramatically over the last decade.

Online learning comes with both benefits and drawbacks. Students can learn from the comfort of their home and in a safe environment. However, there are many challenges that come with it as well, among them one of the most important ones is efficiency. Each individual has a different way of absorbing information and it varies greatly from person to person. Learning should be quick and efficient, where each individual's skill is taken into consideration when testing their knowledge.

Fairness is another problem that's very prominent in online learning. In most online exams, a student has to answer from a pool of questions. The questions are selected randomly or sometimes based on the professor's categorization. The problem is students could get a really hard question set or a really easy one based on what they get from that random pool. This can be frustrating for students.

Research Problem

In this research, we are attempting to improve the efficiency and speed of online learning. A research tool called 'Phishducation' is being used for this purpose. It's a website designed to teach users about the different types of Phishing attacks. An individual's knowledge in terms of phishing can be more or less than others. In an ideal scenario, 'Phishducation' would hand out the perfect question to the user based on their knowledge of the subject and also based on the difficulty of the question. This will help the users to improve their knowledge regarding phishing faster and also learn in an efficient way. The goal of this research is to figure out the difficulty levels of different phishing samples using cluster analysis on a small sample of human users.

Literature Review

Before figuring out a way to estimate the difficulty of phishing questions, significant research related to Phishing was done. Phishing is the process of obtaining sensitive information from the user which is then used against them. The term was first used in mid 1990s. Email is usually the disguised weapon used for phishing attacks.

Fette et al. (2007) suggested using PILFER to detect Phishing emails and compared its results with other traditionally used Machine Learning techniques to find no significant improvement. Kumaraguru et al. (2007) showed that embedded learning techniques where the users are presented the training material after the fell for a phishing attack. Cook et al. (2010) proposed a web-based framework to increase learner's efficiency. They also came to the conclusion that by grouping the learners, the system finds it easier to recommend what kind of materials should be given to each learner.

Different machine learning techniques were used and compared for classifying Phishing emails; where Support Vector Machine constantly achieved the best results (Basnet et al., 2008). Ma et al. (2009) used orthographic features and a modified global k-means clustering algorithm for provenance determination. Miyamoto et al. (2009) evaluated several machine learning-based detection methods for phishing website detection. AdaBoost showed the highest promise with SVM closely after. Steves et al., (2020) proposed a Phish Scale to rate the difficulty of phishing exercises. They put an emphasis on using the premise alignment as it played a significant role in rating the difficulty of each exercise.

Many authors have used K-means clustering to profile phishing emails (Toolan & Carthy, 2010; Xu et al., 2009; Yearwood et al., 2012). Hamid & Abawajy (2014) proposed an approach using two-step clustering algorithm to profile phishing emails. Dazeley et al. (2010) used k-means clustering algorithm in conjuncture with a classification algorithm to classify phishing emails.

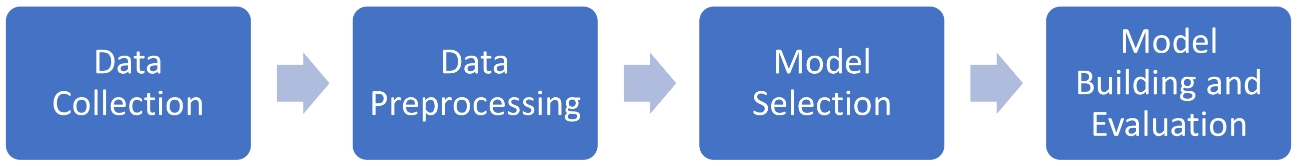
There have been lots of research done on Phishing email detection over the years but very few on how to give each phishing email a difficulty level based on how easy or hard it is to recognize. We plan on using k-means clustering to observe and analyze the results of those clusters, also we would like to gather some user feedback data to validate our clustering results.

Methodology

The methodology we followed could be divided into 6 steps. We will go through those steps one by one.

Figure 1

Methodology Steps



Data Collection

One of the earliest issues we had with this project was with the already existing data set. There were phishing samples with misidentified feature sets, the time it takes for a user to answer a question was really high or just didn’t make sense. For example: One sample was misidentified as a phishing sample whereas it was a non-phishing email. As a result, the data showed a failure rate of this particular sample to be around 90% where in reality people were answering correctly and it should’ve had a very low failure rate. For problems like these we decided to get rid of the existing data and collect new ones with properly identified feature set in a semi-controlled environment so that we know the data is correct.

Data was collected using ‘Phishducation’. There was a total of 20 sample of phishing and non-phishing emails that were selected for this research. Each sample had a total of 11 attributes (Example: Spelling error) and more were added later on. A user had to login to the website, create an account and take the quiz. The quiz gave users 10 samples and asked questions to test their knowledge on what features that particular sample had, or to distinguish between phishing and non-phishing emails. The time it takes for the user to answer a particular question and correctness of the answers were recorded. After the quiz was complete the users were told to visit a Google survey page where they would rate the difficulty of the questions, they answered on a scale of 1-10. This was done mainly to check the validity of our models results.

Data Preprocessing

Gathered data from the user was then transferred to a excel sheet where the dataset was organized on a sample-by-sample basis. Each sample had a total of 17 columns having the 11 attributes and other added on features that might help the machine learning algorithm in distinguishing between them. The attribute table listing all the attributes is shown on Table 1.

Table 1

Attribute table

Attribute 1: IP Based URL
Attribute 2: Multi-subdomain URL
Attribute 3: Wrong redirect
Attribute 4: Spoofed/Misspelled URL
Attribute 5: Language Demanding Urgent/ Immediate action
Attribute 6: URL contains many control/escape characters
Attribute 7: Email sender spoofing
Attribute 8: Text embedded in an Image
Attribute 9: Keywords in subject line (Incite people to act)
Attribute 10: Spelling/Grammatical errors
Attribute 11: General Language

Table 2

Feature Table with Description

Feature 12: Phishing email or Non-Phishing email. Summary: This is a Boolean value. Will be 0 if the sample is a non-phishing email and 1 if it’s a phishing email.

Feature 13: Number of Attributes Summary: This represents the number of attributes that each sample has.
Feature 14: Average Time in seconds Summary: The time value from all the user data is averaged based on sample-by-sample basis.
Feature 15: Failure Rate (Percentage) Summary: Failure rate represents the percentage of users who misidentified a sample or were unable to answer the questions correctly.
Feature 16: Clickable link in sample? Summary: This is also a Boolean value. Represents whether or not there is a clickable link on the email. Will be 0 if there is none and 1 if there is a clickable link.
Feature 17: Number of links Summary: The total number of links a phishing sample has.

The data was checked for missing data and outliers. There were 2 instances of outliers where the amount of time it took to answer a question was more than 5 minutes. Those were removed out of the dataset when calculating the average time in seconds for each sample. Failure rate was also calculated for each phishing sample. A summary of all the features is shown on Table 2.

Model Selection

We needed to find pattern in our data to help us determine which features contribute to making a phishing sample question easier/harder for the user to answer correctly. For that purpose, we thought using a machine learning algorithm would be the best approach.

There are a lot of algorithms to choose from but since our data was not labeled, we were limited to using unsupervised techniques. From a high-level standpoint, to categorize unlabeled phishing samples into different levels we needed to cluster them first. Find out the similarity in those clusters so we could understand what makes the samples easy or difficult. Clustering is an approach used in machine learning to group samples based on their datapoints. There are different types of clustering algorithms but by far one of the most used ones is K-Means clustering.

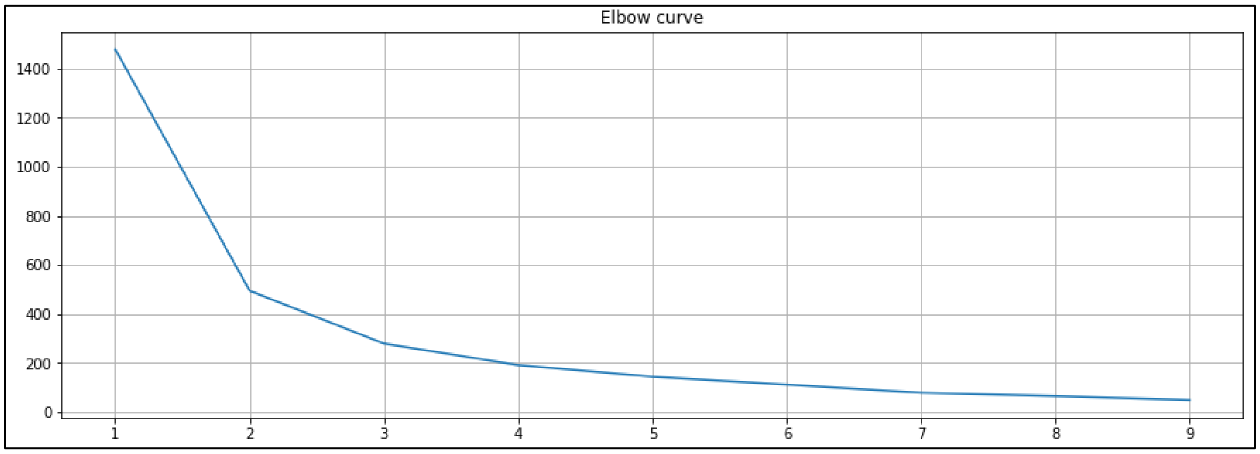
K-means clustering is one of the simplest and most popular unsupervised machine learning algorithms. In this algorithm, you have to define the number of clusters you want. This will define the number of randomly selected centroids, chosen as the beginning points for each cluster. Then iteration happens and on each iteration the positions of the centroids are optimized till they are stabilized or the user specified iterations are completed (Garbade, 2018). As this algorithm is relatively simple and has been used in profiling phishing emails and websites before, we thought it would be a great fit when clustering our sample of questions for pattern observation. So, that's the model we decided to use for our dataset.

Model Building and Evaluation

One of the problems of k-means algorithm is the number of clusters isn't selected automatically. There are mainly two methods used for selecting the clusters in k-means. The most popular one is the elbow method followed by the silhouette scores. The elbow method runs k-means clustering on the dataset on a range of values and computes the average score for all the clusters. Scikit learn library calculates the distortion score by default. The scores are then plotted onto a graph, picking the elbow of the curve as the number of clusters to use in that dataset.

Figure 2

Elbow Curve



If we look at the curve on figure 2, we could see the elbow of the curve is somewhere between 2 and 3. Silhouette scores are used to measure how dense and well separated the clusters are. So, we decided to calculate the average silhouette scores for n clusters ranging from 2 to 6. The results are shown on table 3.

Table 3

Average silhouette score for clusters

Number of clusters	**Average silhouette score**
2	0.533
3	0.636
4	0.352
5	0.337
6	0.289

The value of the silhouette score ranges from [-1,1] where 1 means the clusters are dense and very well separated and 0 meaning they're overlapping. In our case, the ideal number of clusters should be 3 as that resulted in the highest score.

With the value of k=3 we fit the k-means model using our dataset. The clusters were created and saved in a new column of our dataset. Plotting the clusters turned out to be a difficult task, because we needed to plot 17 features on a 2-dimensional axis. For dimensionality reduction we decided to use Principal component analysis or PCA. It's a very popular and well-known technique used in machine learning. We reduced all 17 features into two, X and Y for plotting. The clustered datapoints were plotted and saved for further analysis.

Findings and Analysis

Some initial analysis of the data before clustering reveals some key findings. Those are presented here:

1. Phishing emails have a high failure rate and high average time compared to non-phishing ones. We could see from the bar chart shown in figure 3 that phishing emails have an average failure rate of about 31% where non phishing emails have a failure rate of approximately 10%. Also, the average time it takes for a user to answer a Phishing email sample question is almost double compared to non-phishing as shown on figure 4. Users take relatively less time in correctly identifying non-phishing emails as opposed to phishing ones. One of the reasons could be the familiarity of the non-phishing emails could prompt the user to easily recognize them and answer those questions quickly.

Figure 3

A bar chart of Failure Rate (Phishing vs Non-phishing emails)

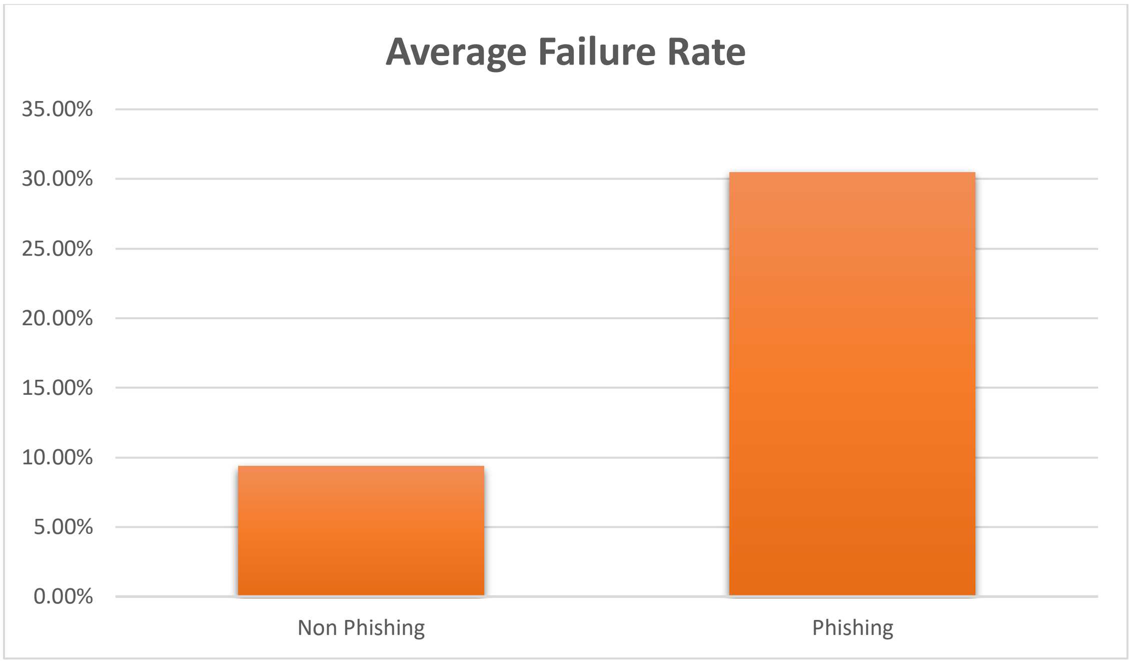
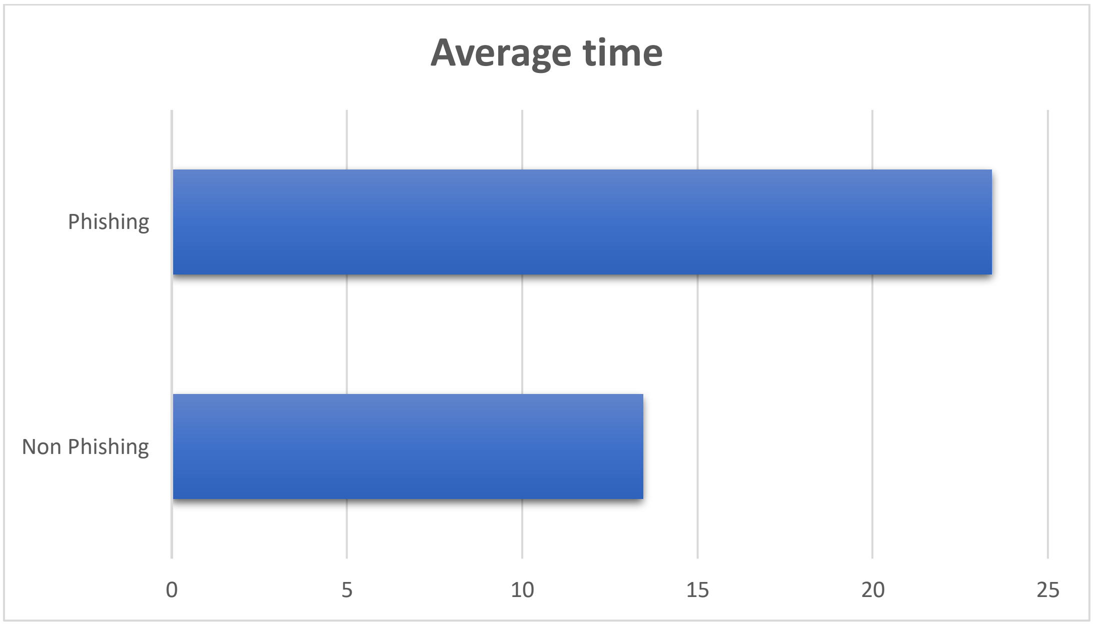


Figure 4

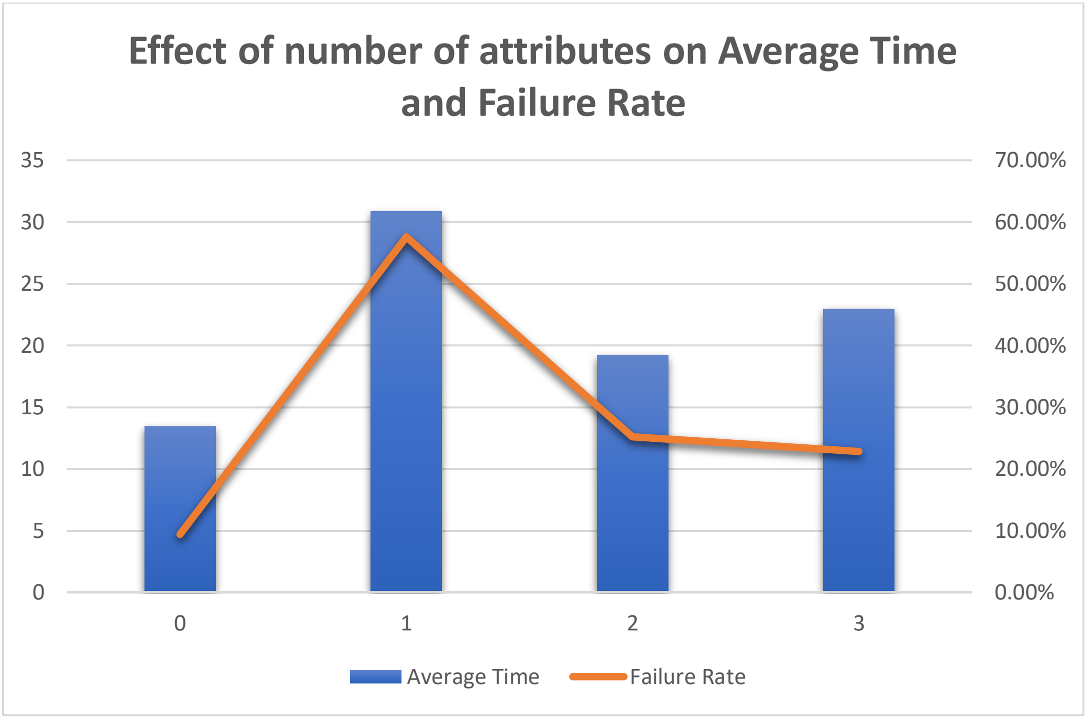
A bar Chart of Average Answering Time (Phishing vs Non-Phishing emails)



2. A sample questions number of attributes had an effect on the failure rate and average time. We could see from figure 5 that samples with 0 attributes have the lowest average time to answer and failure rate. The average time and failure rate goes up with the number of attributes and then goes down again. When a sample question has only 1 attribute it takes the highest amount of time to answer and also has the highest rate of failure at around 58%. This could be due to only one attribute being hard to find out for the user, whereas multiple attributes are easier to detect.

Figure 5

A bar chart representing the effect of number of attributes on the Average Time and Failure Rate

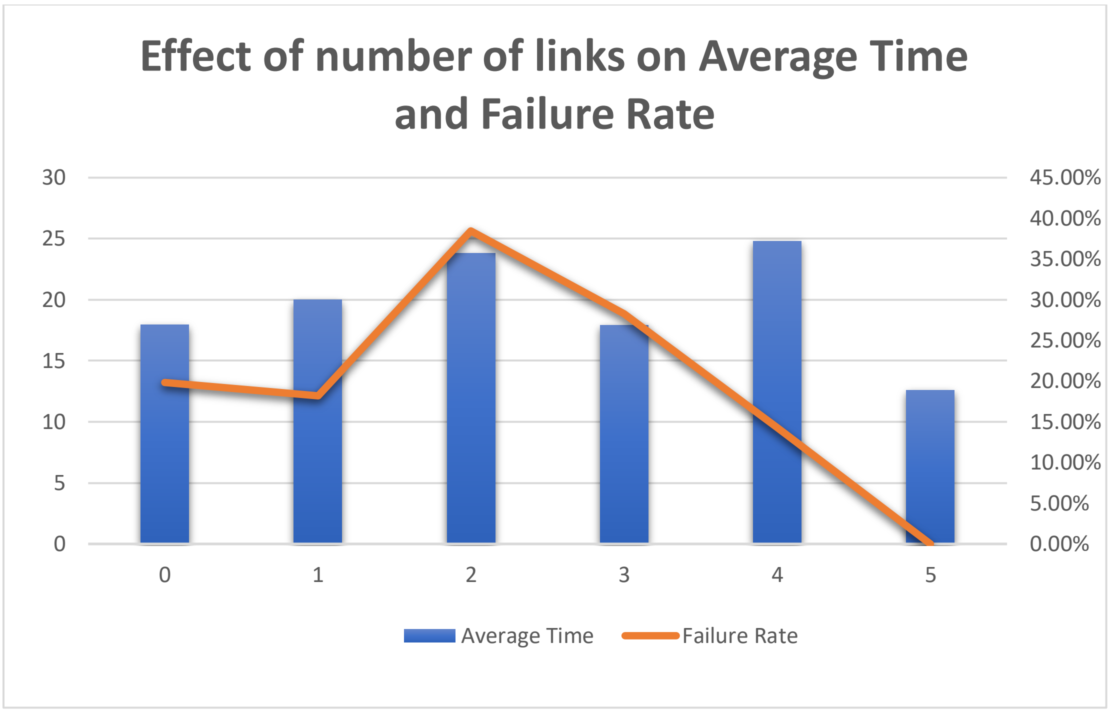


3. A sample question having more links leads to a high failure rate for the user. From figure 6 we could see the failure rate peaks at around 39% with 2 links and then slowly drops

off to 0% with 5 links in the question. The users might find it difficult to recognize a phishing email when there’s a small number of links in the email, whereas having more links with probable phishing links make it easier to recognize.

Figure 6

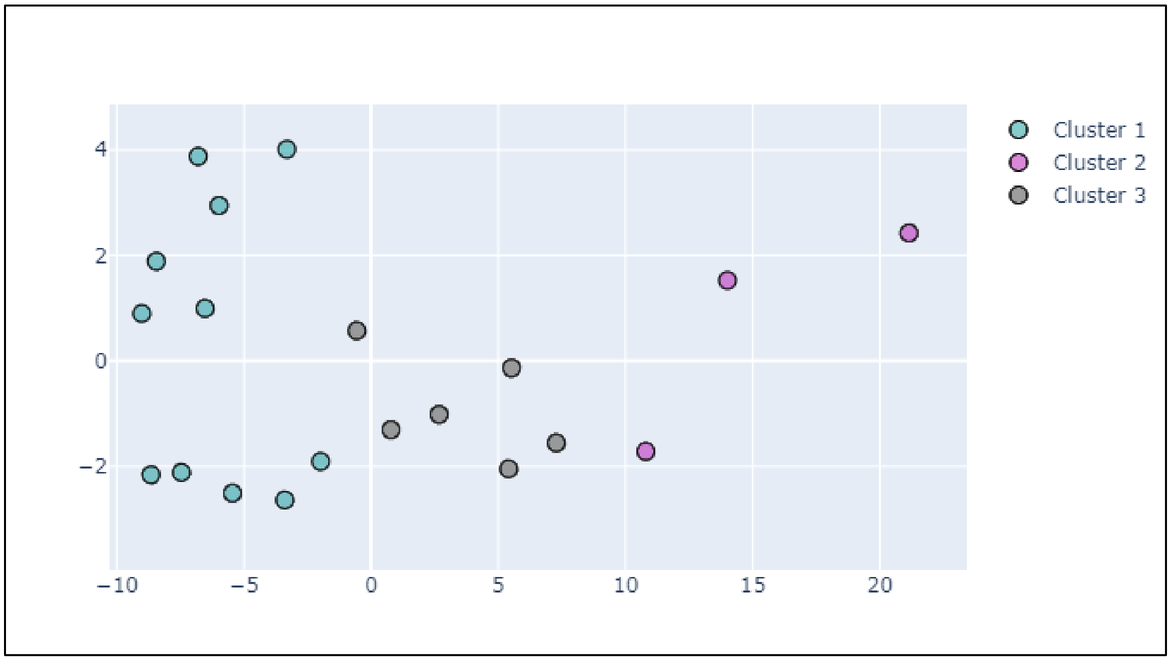
A bar chart representing the effect of number of links on the Average Time and Failure Rate



After doing the k-means clustering we analyzed the individual clusters and their features. Here are some key findings from the k-means clustering results:

Figure 7

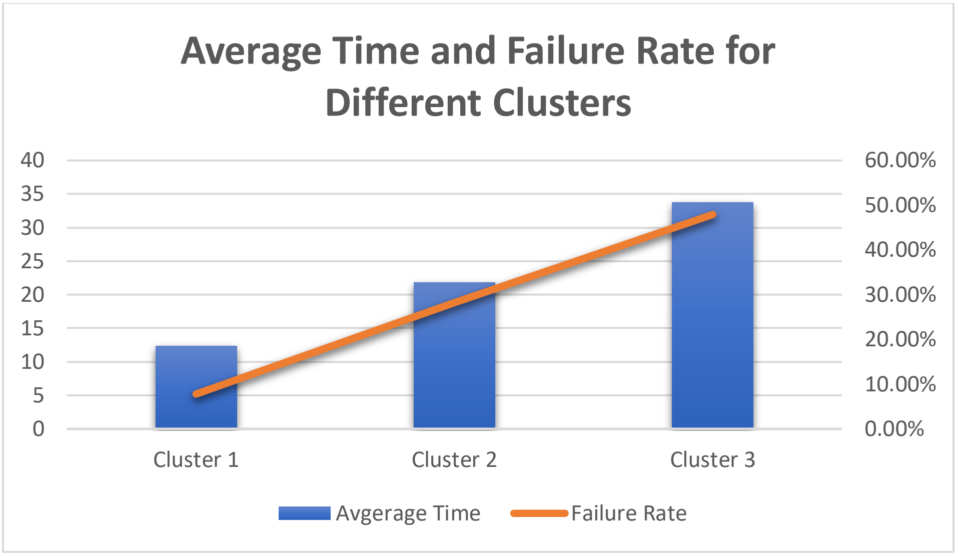
A scatter plot representing the different clusters



1. We could see 3 clusters shown in figure 7. Out of the 20 samples, the algorithms put 11 of those samples in cluster 1, whereas cluster 2 has the lowest number of samples with only 3.
2. From figure 8 we could see that cluster 1 has the lowest failure rate and average time, with cluster 3 having the highest. There seems to be a linear relation when it comes to time and rate of failure in the different clusters. The algorithm seemingly has put the easiest sample questions in cluster 1 and the hardest ones in cluster 3.

Figure 8

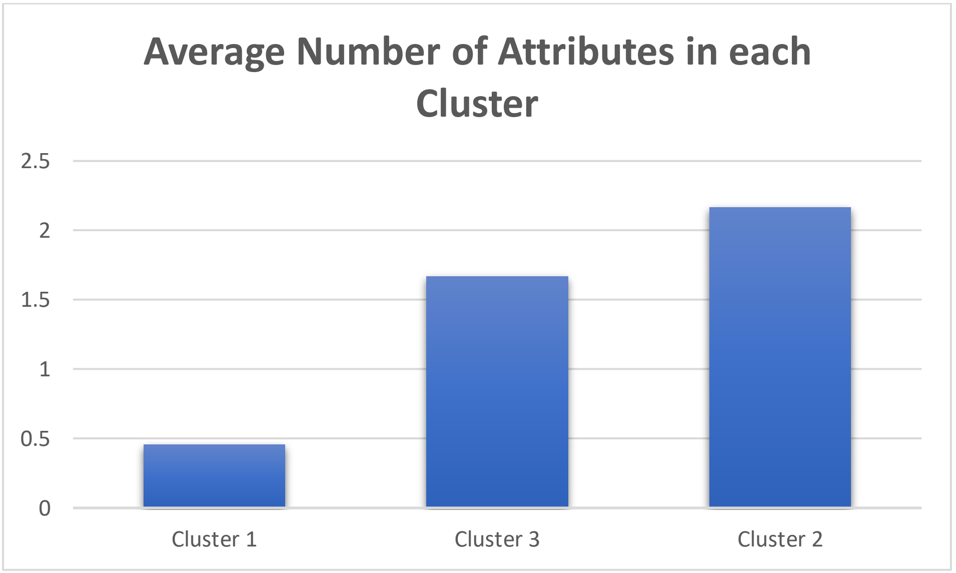
Average time and Failure rate for different clusters



3. If we look at the bar chart at figure 9, we could see that that the medium difficulty level cluster or cluster 2 has the highest average number of attributes in the phishing samples.

Figure 9

A bar chart representing the average number of attributes on different clusters



4. There are some attributes which are specific to the clusters. For example: Cluster 2 has attribute 2 and 3 in most of its samples. Whereas cluster 3 or the cluster with the highest failure rate has attribute 10 in most of its samples. Attribute 10 is spelling/grammatical errors. So, we could say that with confidence that spelling/grammatical mistake is a hard attribute for users to spot in phishing samples.

From our clustering results one could assume that the algorithm clustered the questions based on the difficulty. Cluster 1 having the easiest samples, cluster 2 in between and cluster 3 had the hardest questions. We wanted to verify the difficulty level of the clusters based on the users score. Each user who took the quiz had to rate the difficulty of the questions on a scale of 1 to10. We then averaged all the ratings for each sample and used those to come up with the "User defined difficulty level" for each of the groups. The user defined difficulty level for each cluster is shown in table 4.

Table 4

Comparison between difficulty levels

Cluster Number	**Difficulty Level from Cluster Analysis**	**Average User defined difficulty level (Out of 10)**
1	1	2.91
2	2	4.4
3	3	5.66

From the result in table 4 we could clearly see that; the user defined difficulty level matches up very well with the difficulty level we assumed from our cluster analysis. The k-means algorithm did in fact, divide the sample questions based on their difficulty level.

Limitations

One of the main problems we faced with this project was with data. A lot of time was spent in going through the phishing samples, correctly labeling them and collecting the new data. 20 samples are definitely not ideal to use in a machine learning model. The more data can be fed to the model the better it performs. Other than that, more data can lead to better analysis results as well where we could find more patterns in the dataset. This is not possible when working with a small dataset. The amount of data was definitely a shortcoming for the whole project.

Future Work

Using the existing methodology on a much bigger dataset would be something we'd be interested to do in the future. Also, the results from the clustering could be used to label the difficulty level of each of the samples. That could be further used in conjecture with an SVM model for predicting the difficulty of a new sample.

Conclusion

Clustering analysis in most cases is done to group up datapoints without having concern for any specific outcome. In our case, we wanted to group up the sample questions based on their difficulty level using the features we extracted. Even though we didn't have the ideal amount of data required, I think our result showed promise. With the help of more data and by using SVM, a complete difficulty level estimation system for phishing emails could be made.

References

Basnet, R., Mukkamala, S., & Sung, A. H. (2008). Detection of phishing attacks: A machine learning approach. *Studies in Fuzziness and Soft Computing*, *226*, 373–383. https://doi.org/10.1007/978-3-540-77465-5_19

Cook, D. A., Levinson, A. J., & Garside, S. (2010). Time and learning efficiency in Internet-based learning: A systematic review and meta-analysis. *Advances in Health Sciences Education*, *15*(5), 755–770. https://doi.org/10.1007/s10459-010-9231-x

Dazeley, R., Yearwood, J. L., Kang, B. H., & Kelarev, A. V. (2010). Consensus clustering and supervised classification for profiling phishing emails in internet commerce security. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6232 LNAI*, 235–246. https://doi.org/10.1007/978-3-642-15037-1_20

E-Learning Market in US by Product and End-user - Forecast and Analysis 2021-2025. Technavio. (2021, April). https://www.technavio.com/report/e-learning-market-in-the-us-analysis-share-2018?pk_vid=2a69e1dd3fbc7b381591262233ffe154.

Fette, I., Sadeh, N., & Tomasic, A. (2007). Learning to detect phishing emails. *16th International World Wide Web Conference, WWW2007*, 649–656. https://doi.org/10.1145/1242572.1242660

Garbade, D. M. J. (2018, September 12). *Understanding K-means Clustering in Machine Learning*. Medium. https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1.

Hamid, I. R. A., & Abawajy, J. H. (2014). An approach for profiling phishing activities. *Computers and Security*, *45*, 27–41. https://doi.org/10.1016/j.cose.2014.04.002

Kumaraguru, P., Rhee, Y., Sheng, S., Hasan, S., Acquisti, A., Cranor, L. F., & Hong, J. (2007). *Getting users to pay attention to anti-phishing education*. 70–81. https://doi.org/10.1145/1299015.1299022

Ma, L., Yearwood, J., & Watters, P. (2009). Establishing phishing provenance using orthographic features. *2009 ECrime Researchers Summit, ECRIME '09*. https://doi.org/10.1109/ECRIME.2009.5342604

Miyamoto, D., Hazeyama, H., & Kadobayashi, Y. (2009). *<Fulltext63.Pdf>*. 539–546.

Steves, M., Greene, K., & Theofanos, M. (2020). Categorizing human phishing difficulty: A Phish Scale. *Journal of Cybersecurity*, *6*(1), 1–16. https://doi.org/10.1093/CYBSEC/TYAA009

Toolan, F., & Carthy, J. (2010). Feature selection for Spam and Phishing detection. *General Members Meeting and ECrime Researchers Summit, ECrime 2010*. https://doi.org/10.1109/ecrime.2010.5706696

Xu, K. S., Kliger, M., Chen, Y., Woolf, P. J., & Hero, A. O. (2009). Revealing social networks of spammers through spectral clustering. *IEEE International Conference on Communications*, *Section II*, 1–6. https://doi.org/10.1109/ICC.2009.5199418

Yearwood, J., Mammadov, M., & Webb, D. (2012). Profiling phishing activity based on hyperlinks extracted from phishing emails. *Social Network Analysis and Mining*, *2*(1), 5–16. https://doi.org/10.1007/s13278-011-0031-y

Appendix

Code for running the K-Means algorithm

```
!pip install sklearn

from sklearn.cluster import KMeans

import pandas as pd


#from sklearn.preprocessing import MinMaxScaler

from matplotlib import pyplot as plt

import numpy as np

from numpy import arange

from sklearn.decomposition import PCA

import plotly.graph_objs as go

from plotly.offline import init_notebook_mode, iplot

import plotly.graph_objects as go

#init_notebook_mode()

def configure_plotly_browser_state():

    import IPython

    display(IPython.core.display.HTML('''

      <script src="/static/components/requirejs/require.js"></script>

      <script>

          requirejs.config({

            paths: {

              base: '/static/base',
      plotly: 'https://cdn.plot.ly/plotly-1.5.1.min.js?noext',
     },
    });
    </script>
    '''))
from google.colab import files
uploaded = files.upload()
#df= pd.read_csv("Data.csv")
df= pd.read_csv("KMeans2.csv")
#df = df.loc[:, ~df.columns.str.contains('^Unnamed')]
df.head()
#df=df.drop(['Difficulty Level','Avg Time'], axis=1)
#df=df.drop(['Difficulty Level 3','Time*Failure Rate','Failure Rate','Avg Time in Seconds','Total
Attributes','Sample Number'], axis=1)
#df=df.drop(['Difficulty Level 3','Time*Failure Rate','Total Attributes','Sample Number'], axis=1
)
df=df.drop(['Normalized Time','Sample Number'], axis=1)
df.head()
df.info()
cols = df.columns[0:]
cols
#Elbow Method
distorsions = []
```

```
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(df[df.columns[2:]])
    distorsions.append(kmeans.inertia_)


fig = plt.figure(figsize=(15, 5))
plt.plot(range(1, 10), distorsions)
plt.grid(True)
plt.title('Elbow curve')
from sklearn.metrics import silhouette_score
#Silhouette scores
range_n_clusters = [2, 3, 4, 5, 6]
for n_clusters in range_n_clusters:
    clusterer = KMeans(n_clusters=n_clusters)
    preds = clusterer.fit_predict(df[df.columns[2:]])
    centers = clusterer.cluster_centers_


    score = silhouette_score(df[df.columns[2:]], preds)
    print("For n_clusters = {}, silhouette score is {})".format(n_clusters, score))
km=KMeans(n_clusters=3)
#y=km.fit_predict(df)
df["cluster"] = km.fit_predict(df[df.columns[2:]])
df.tail()
```

```
# Principal component separation to create a 2-dimensional picture
pca = PCA(n_components = 2)
df['x'] = pca.fit_transform(df[cols])[:,0]
df['y'] = pca.fit_transform(df[cols])[:,1]
df = df.reset_index()
df.tail()
trace0 = go.Scatter(x = df[df.cluster == 0]["x"],
                    y = df[df.cluster == 0]["y"],
                    name = "Cluster 1",
                    mode = "markers",
                    marker = dict(size = 10,
                                  color = "rgba(15, 152, 152, 0.5)",
                                  line = dict(width = 1, color = "rgb(0,0,0)")))
trace1 = go.Scatter(x = df[df.cluster == 1]["x"],
                    y = df[df.cluster == 1]["y"],
                    name = "Cluster 2",
                    mode = "markers",
                    marker = dict(size = 10,
                                  color = "rgba(180, 18, 180, 0.5)",
                                  line = dict(width = 1, color = "rgb(0,0,0)")))
trace2 = go.Scatter(x = df[df.cluster == 2]["x"],
                    y = df[df.cluster == 2]["y"],
                    name = "Cluster 3",
                    mode = "markers",
                    marker = dict(size = 10,
                                color = "rgba(132, 132, 132, 0.8)",
                                line = dict(width = 1, color = "rgb(0,0,0)")))
# trace3 = go.Scatter(x = df[df.cluster == 3]["x"],
#                   y = df[df.cluster == 3]["y"],
#                   name = "Cluster 4",
#                   mode = "markers",
#                   marker = dict(size = 10,
#                               color = "rgba(122, 122, 12, 0.8)",
#                               line = dict(width = 1, color = "rgb(0,0,0)")))
#Required Code to run plotly in a cell
configure_plotly_browser_state()
init_notebook_mode(connected=False)
data = [trace0, trace1, trace2]


iplot (data)
from google.colab import drive
drive.mount('drive')
df.to_csv('drive/My Drive/KMeansResultWith3Clusters.csv')
```