Predictive Modeling of Spam Email Detection

Niranjana Kulkarni

Spring 2024

I. INTRODUCTION

These days, everyone knows the pain of spam emails clogging their inbox. But where did this word "spam" come from, and why do we use it to describe junk emails? The historical context of the word is quite interesting. Spam is a canned amalgam of the junk parts of ham (hence, the synonym for junk), and became popular during World War II as sustenance for soldiers on the front lines. In modern times, the definition lies in it being adopted to refer to unsolicited commercial electronic mail sent to a large number of addresses, in what was seen as drowning out normal communication on the internet. Dealing with this issue of spam emails by creating a better filter can help many industries, working professionals, and organizations. People can spend less time deleting spam, space in inboxes can be attributed to only important emails, and storage sizes of inboxes can be reduced. On a large scale, having an advanced filter for preventing spam emails from cluttering inboxes can have an enormous impact on the effectiveness of industries and can even increase profit margins. We decided to investigate: what makes the best spam filter? We wanted to discover what models we can train that will allow us to accurately identify spam in most cases, so that in the future we might be able to apply these models and see accurate filtering in real time. We obtained our dataset "Spam Email Data Classification" on which we will train our models from Kaggle.

Our main scientific question is: what are the written indicators that an email is considered as spam in a workplace setting? Many email platforms, including Google and Yahoo, have folders labeled as "Junk" or "Spam". This means that they are utilizing their own spam classification system to send spam mail into these folders, filtering spam out from regular emails. Given our dataset, our goal is to mimic such machine learning classification models and test the performance of our own models in hopes of obtaining a more successful algorithm.

II. BACKGROUND

In the past, researchers have attempted to tackle the problem of classifying spam emails through various methods. In some cases, preprocessing methods to extract the data were increasing in complexity, including word2vec and TF-IDF. Moreover, the next steps that these researchers took was to develop many different models,

such as large language models, deep learning, Lexicon-based algorithms, and graph-based models to aid in their performance of spam email classification. However, through variation in linguistic style and semantics of the textual data, issues arose around unlabeled datasets, class imbalances, and bias stemming from the collection method of this data. These precursors are concepts that we kept in mind throughout the evaluation of our own dataset and training of our own models.

III. DESCRIPTIVE ANALYSIS

Dataset

Our dataset "Spam Email Data Classification" contains 58 numerical variables (the file is named "spambase.csv"). We have 57 different predictor variables and 1 binary response variable (which is 0 if the email is not spam and 1 if the email is spam). Each row in our dataset represents one email that we have gathered data from. Our dataset had no null or missing values in any of the columns, so no rows needed to be removed from the dataset. However, our original dataset did have duplicate entries which we removed to preserve the uniqueness of the dataset. Looking at the descriptive statistics of the dataset (see **Figure 2**) and looking at our first 5 entries of our dataset (see **Figure 1** below) we can see that we have sparsity in our dataset. Checking our dataset for sparsity, we have that 77.12% of our dataset entries are the value zero. We will need to account for the sparsity of our data when choosing and training our models.

	WF_make	WF_address	WF_all	WF_3d	WF_our	WF_over	WF_remove	WF_internet	WF_order	WF_mail	...	CF_%3B	CF_%28	CF_%5B	CF_%21	CF_%24	CF_%23	CRL_average	CRL_longest	CRL_total	class
0	0.00	0.64	0.64	0.0	0.32	0.00	0.00	0.00	0.00	0.00	...	0.00	0.000	0.0	0.778	0.000	0.000	3.756	61	278	1
1	0.21	0.28	0.50	0.0	0.14	0.28	0.21	0.07	0.00	0.94	...	0.00	0.132	0.0	0.372	0.180	0.048	5.114	101	1028	1
2	0.06	0.00	0.71	0.0	1.23	0.19	0.19	0.12	0.64	0.25	...	0.01	0.143	0.0	0.276	0.184	0.010	9.821	485	2259	1
3	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	...	0.00	0.137	0.0	0.137	0.000	0.000	3.537	40	191	1
4	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	...	0.00	0.135	0.0	0.135	0.000	0.000	3.537	40	191	1

5 rows × 58 columns

Figure 1. First 5 entries of our original dataset

	WF_make	WF_address	WF_all	WF_3d	WF_our
count	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000
mean	0.104553	0.213015	0.280656	0.065425	0.312223
std	0.305358	1.290575	0.504143	1.395151	0.672513
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.420000	0.000000	0.380000
max	4.540000	14.280000	5.100000	42.810000	10.000000

Figure 2. First 5 columns of the descriptive statistics for our original dataset

Features

To provide context, let us discuss the variables in our dataset. Out of the 57 predictor variables, we have 3 general types or variables: "Word" Frequencies (48 variables), Character Frequencies (6 variables), and Length Variables (3 variables). First, we have word frequency variables. These are computed for each entry by finding the percentage of the number of times the word appears in the entire email. For example, for the word internet, the variable WF_internet stores the number of times the word appears in the email divided by the total number of words in the email multiplied by 100. We perform the same process for the character frequency variables. Therefore, for the frequency variables in our dataset, the numerical values for frequencies in the dataset are the percentage numbers (so instead of 0.25% we have 0.25 in our data). Note that "word" in this dataset refers to any string of alphanumeric characters bounded by alphanumeric characters or end of string characters such as "3internet123". Our Length Variables relate to the uninterrupted sequences of capital letters in the email: CRL_average = average length, CRL_longest = length of the longest sequence, and CRL_total = total number of capital letters. Our target variable (last column in **Figure 1**) is binary: 0 if the email is not spam and 1 if the email is spam. Our first 5 entries are all spam emails.

We will proceed to investigate the relationships between our predictors. Looking at **Figure 3**, we can see that in the heatmap with the predictors being the axes that we have many variables that are positively correlated with each other, as evidenced by the central region and the bottom right of the heatmap. Since all the variables are not highly correlated with each other, we do not observe multicollinearity and thus will use all the variables for the model fitting.

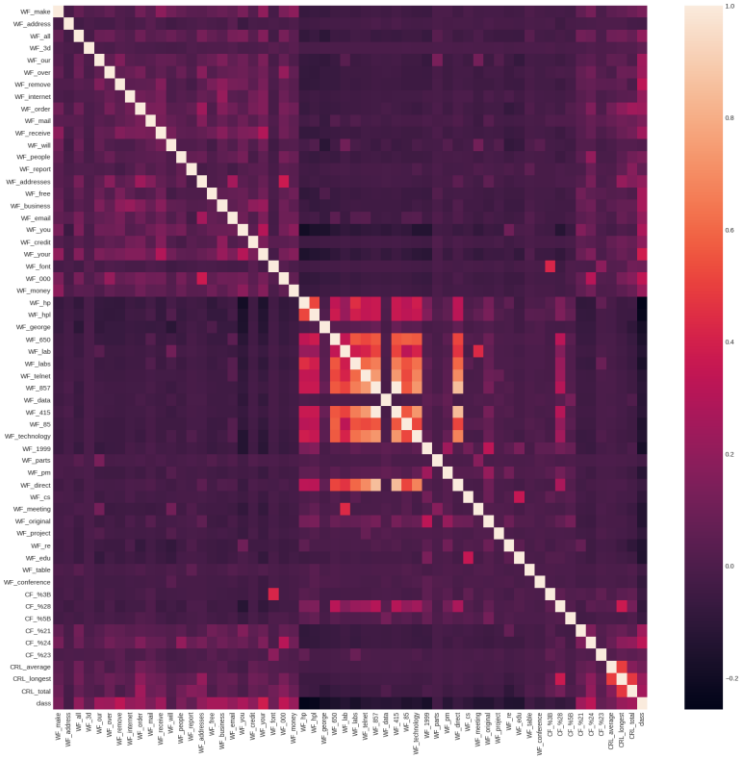

Figure 3. Heat maps of the original data (compressed (left) and whole (right)).

Let us continue to uncover information about the relationships in the data.

IV. EXPLORATORY DATA ANALYSIS

In order to extract more information about the data, we perform exploratory data analysis through visualizing the features and examining the relationships between them. Looking at the target variable called “class” in **Figure 4**, we can see that we have a slightly unbalanced ratio of spam to non-spam emails in the dataset. Out of the 4601 entries in our data set, we have that about two-thirds of the emails are non-spam and about one-third of the emails are spam.

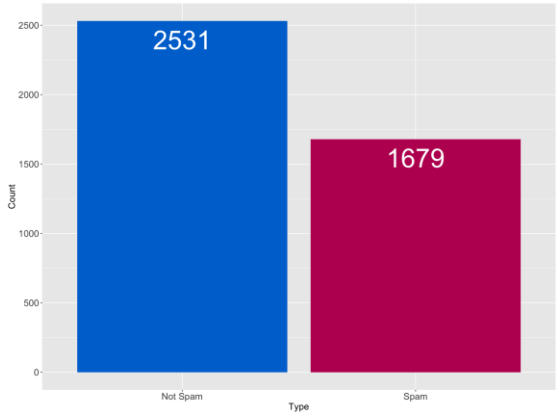

Figure 4. Spam Email Univariate Distribution

This makes sense for our data because we would expect a work email account to receive more work related emails than spam emails. Since we want models to be able to determine if the email is a spam email or not, it means we will have slightly higher accuracy for determining non-spam emails compared to spam emails.

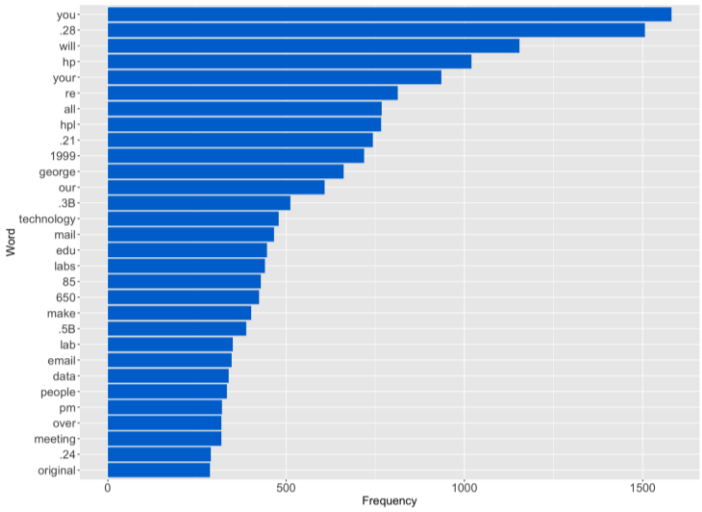

Figure 5(a). Frequent phrases in non-spam emails

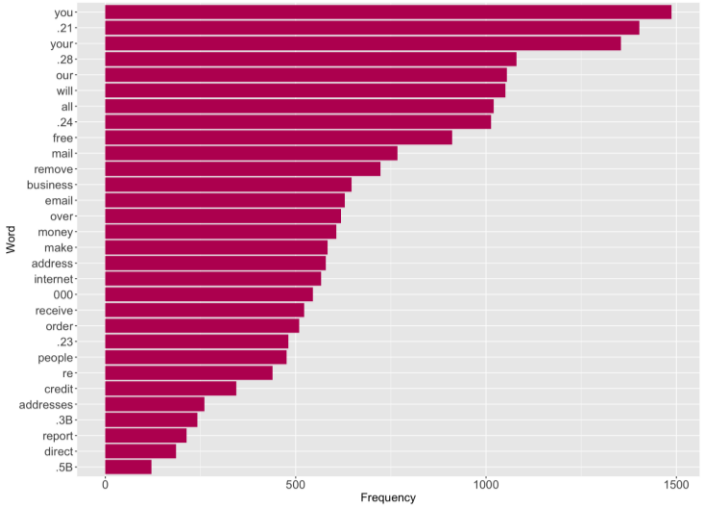

Figure 5(b). Frequent phrases in spam emails

In **Figure 5**, we wanted to see the relationship between the frequent phrases (including both word and character variables) present in the email in comparison to whether the email is spam or not. This frequency plot for non-spam emails (**Figure 5(a)**) shows how often each phrase appears in the sample of observations (email) as referenced before, we can see that the "george" and "650" are potential indicators of non-spam, as well as "hp" and "hpl". The frequency plot for the common phrases in spam emails (**Figure 5(b)**) is similarly created as the previous slide. We can see that the "free", "business", and "money" are potential indicators of spam. However, we can clearly see in both figures that "you", "your" and "21" are frequently used in spam as well as non-spam emails.

Transforming the Dataset

The raw distributions of our predictors, indicate that there are some problems we want to resolve with the dataset before we construct models. We can see that in the frequency variable histograms of **Figures 6** and **7** that they are zero-inflated and highly right-skewed. We must transform the data for model fitting.

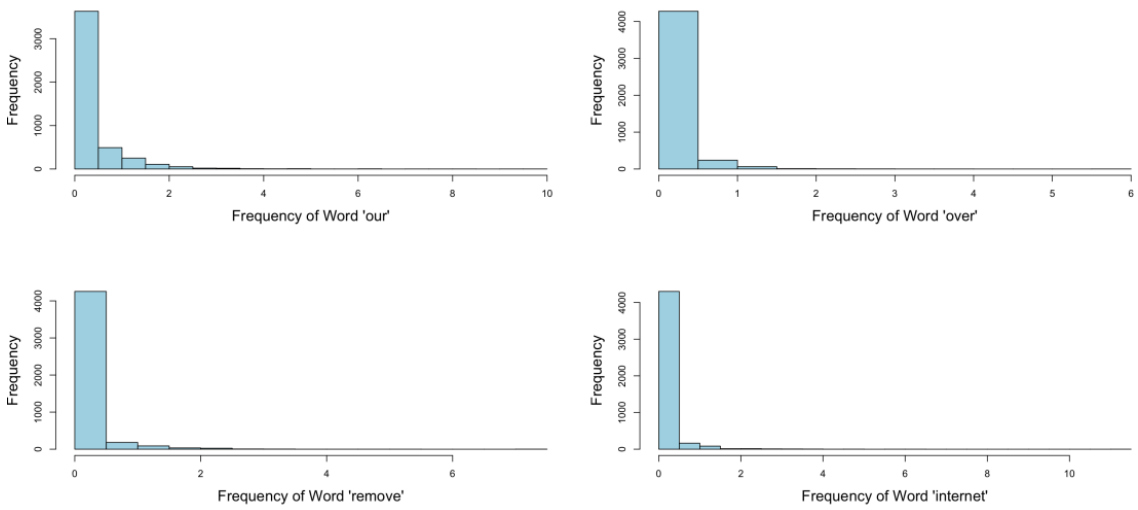

Figure 6. Word frequency variable percentage distribution

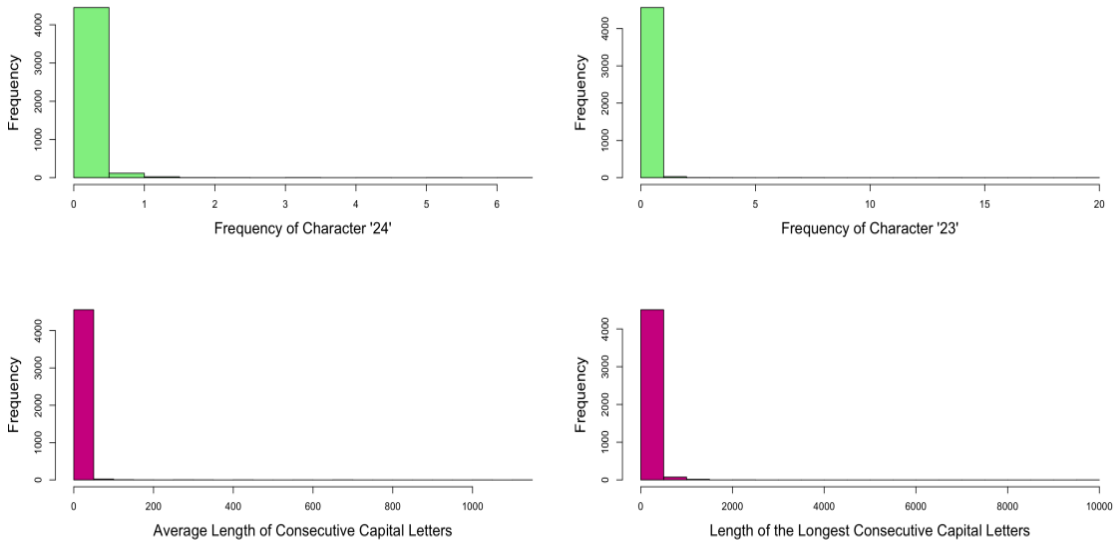

Figure 7. Character Frequency variable % distribution and Capital Run Length variable % distribution

Let us discuss ways in which we transform the data. To combat the severity of the skewness in the data, we converted the original dataset from a frequency-dependent dataset to a binary (1 if the phrase is present and 0 if the

phrase is not in the email) dataset. We keep the last four columns of the dataset the same. To elaborate, if the entry has a nonzero value in a frequency variable column, this entry is replaced by 1 and stays 0 if the value contained in the entry is 0. We can see how the dataset is changed in **Figure 8** below. We use this binary dataset for some of our models. Additionally, we perform Yeo-Johnson scaling on the original dataset (using sklearn.preprocessing PowerTransformer) which increases low variance data and decreases high variance data to create a more uniform dataset. Also, we center and scale our binary dataset (using sklearn.preprocessing StandardScaler).

	WF_make	WF_address	WF_all	WF_3d	WF_our	WF_over	WF_remove	WF_internet	WF_order	WF_mail	...	CF_.3B	CF_.28	CF_.5B	CF_.21	CF_.24	CF_.23	CRL_average	CRL_longest	CRL_total	class
0	0	1	1	0	1	0	0	0	0	0	...	0	0	0	1	0	0	3.756	61	278	1
1	1	1	1	0	1	1	1	1	0	1	...	0	1	0	1	1	1	5.114	101	1028	1
2	1	0	1	0	1	1	1	1	1	1	...	1	1	0	1	1	1	9.821	485	2259	1
3	0	0	0	0	1	0	1	1	1	1	...	0	1	0	1	0	0	3.537	40	191	1
4	0	0	0	0	1	0	1	1	1	1	...	0	1	0	1	0	0	3.537	40	191	1

5 rows × 58 columns

Figure 8. First 5 entries of our new binary dataset

Now after preparing the data and exploring the associations between the predictors and the target variable, we can construct models or methods to deal with the data. Then we can analyze our results.

V. MODELING

We trained the following models: Stepwise Linear Regression , PCA with Logistic Regression, Logistic Regression, Elastic Net Logistic Regression, Naive Bayes, and K-Nearest Neighbors (KNN).

Stepwise Linear Regression

The first model that was developed was a Stepwise Linear Regression Model. This extension of a simple linear regression model used the original raw dataset as the data source. The direction of this stepwise regression was both, indicating some forward steps and some backward steps depending on variable significance. The model started with 57 predictors at the beginning and ended with 46, so 11 predictors were dropped. This model also has an AIC of -10264. Through investigation of the summary, the p-value of the model utility test is close to 0 indicating a significant model. However, the adjusted R-squared is 0.5547.

```
Residual standard error: 0.3261 on 4554 degrees of freedom
Multiple R-squared:  0.5591,    Adjusted R-squared:  0.5547
F-statistic: 125.6 on 46 and 4554 DF,  p-value: < 2.2e-16
```

Figure 9. Partial Summary Output for Stepwise Regression

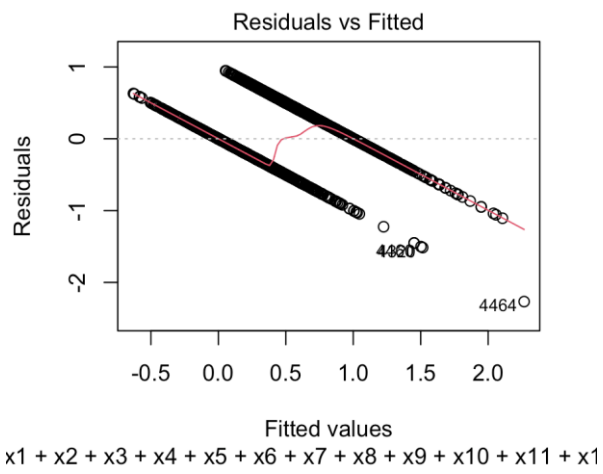

Figure 10. Residuals vs Fitted plot

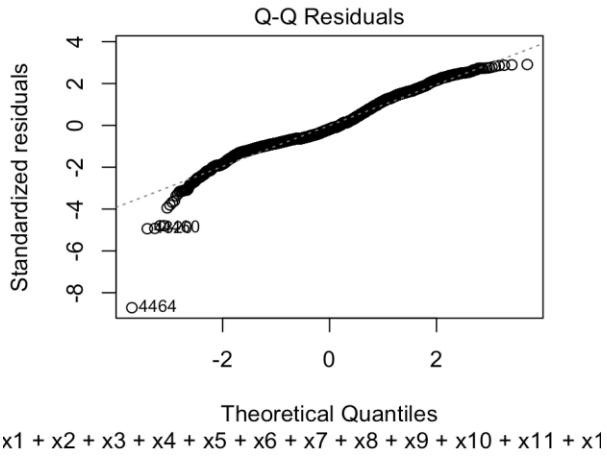

Figure 11. Q-Q Residuals Plot for Stepwise Regression

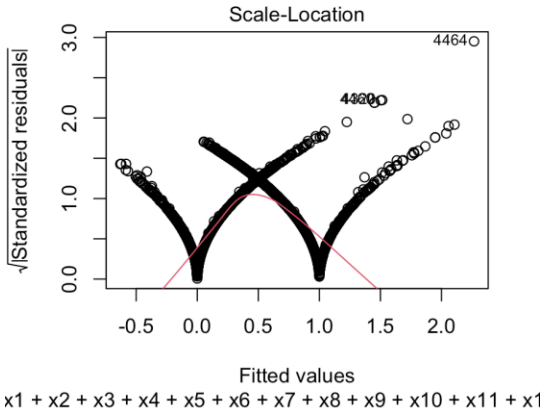

Figure 12. Scale-Location Plot

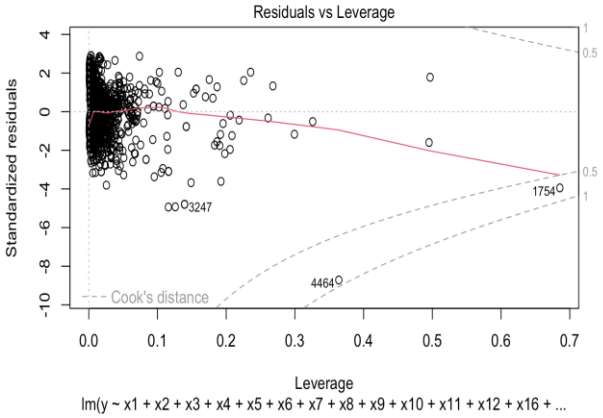

Figure 13. Residuals vs Leverage Plot

Furthermore, through investigation of the diagnostic plots, there is indication of heteroscedasticity and non-constant variance. Overall, we believe that we need more development of different models on a transformed dataset to obtain better accuracy and performance.

Principal Component Analysis with Logistic Regression

Due to the high dimensionality and sparsity in the dataset, we decided to perform Principal Component Analysis on the original raw data and perform Logistic Regression. Principal Component Analysis creates "components" or linear combinations of variables in the data set that capture the variance in the original dataset

variables (with duplicate rows removed). We wanted to keep in mind the uniqueness of the frequency entries of the original data but also consider our problems with the original dataset so we also performed Yeo-Johnson scaling (using ski.learn with PowerTransformer). Performing analysis with several components allows us to visualize and model the data in a much simpler way, which we shall demonstrate with the model of Logistic Regression.

In order to determine how many components to use for this dataset, we created a scree plot after scaling with Yeo-Johnson. A scree plot is a tool we use to determine how many components are the best for our data by showing how much variance is explained by adding another component. Looking at **Figure 14** below, we can see that at the "elbow" of the plot (where the plot turns sharply) the plot denotes 4 or 5 components. Therefore, for the purposes of the original dataset we will look at both cases just to see how there is a little difference between them. Additionally, in order to picture the data we include the case when we just have two components. Now let us discuss performing Logistic regression in these 3 cases.

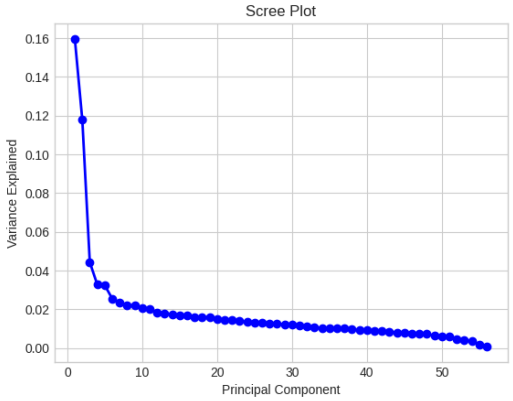

Figure 14. Scree Plot for Original Dataset

Let us compare the accuracy of the cases of PCA with 2, 4, and 5 components for Logistic Regression with Yeo-Johnson scaling. We perform a 80 - 20 (percent of data) split in the data for training and testing data respectively. We can see by **Figure 15** that we have higher accuracy of about 92% for Logistic Regression with 4 and 5 PCA components compared to Logistic Regression with 2 components. Also, we can notice that for the Precision and f1-score we have better results for 4 and 5 components relative to 2 PCA component Logistic Regression. We can see that in our confusion matrices for these 3 cases (**Figure 16**) also demonstrate that PCA with 4 and 5 components give us a much more accurate result in Logistic Regression for the data. Overall, we can see that we have a decently high accuracy score of 92% by performing Logistic Regression with PCA with 4 or 5 components. The accuracy score could be higher, so we will fit more models to see if we can get a better accuracy.

```
Logistic Regression Results for PCA components=2
 With Yeo Johnson Scaling
              precision    recall  f1-score   support

           0     0.9034    0.9253    0.9142       495
           1     0.8896    0.8588    0.8739       347

    accuracy                         0.8979       842
   macro avg     0.8965    0.8920    0.8940       842
weighted avg     0.8977    0.8979    0.8976       842

Logistic Regression Results for PCA components=4
 With Yeo Johnson Scaling
              precision    recall  f1-score   support

           0     0.9335    0.9354    0.9344       495
           1     0.9075    0.9049    0.9062       347

    accuracy                         0.9228       842
   macro avg     0.9205    0.9201    0.9203       842
weighted avg     0.9228    0.9228    0.9228       842

Logistic Regression Results for PCA components=5
 With Yeo Johnson Scaling
              precision    recall  f1-score   support

           0     0.9354    0.9354    0.9354       495
           1     0.9078    0.9078    0.9078       347

    accuracy                         0.9240       842
   macro avg     0.9216    0.9216    0.9216       842
weighted avg     0.9240    0.9240    0.9240       842
```

Figure 15. Comparison of Logistic Regression with 2, 4 and 5 PCA components

Confusion Matrix for 2 Components

	TP	TN
PP	**458**	37
PN	49	**298**

Confusion Matrix for 4 Components

	TP	TN
PP	**463**	32
PN	33	**314**

Confusion Matrix for 5 Components

	TP	TN
PP	**463**	32
PN	32	**315**

Figure 16. Confusion Matrices for Yeo-Johnson Scaled Logistic Regression with 2, 4 and 5 PCA components.

Let us see where the model could be improved by looking at the case with PCA components is just 2. Our principal component plots denoted in **Figure 17** show us how accurate Logistic Regression using PCA with two components truly is when we randomly select 80% of the entries to be the training data and 20% of the entries to be testing data. We can see that the overlap between the red and green points in the testing and training plots shows us some issues in the model. These points are also represented in our confusion matrix for two components for Logistic Regression in **Figure 16** in the regions of the table which are not light green. In the future, we can look at the points

when this overlap happens as well as outlying points to construct more accurate models of Logistic Regression for the original data.

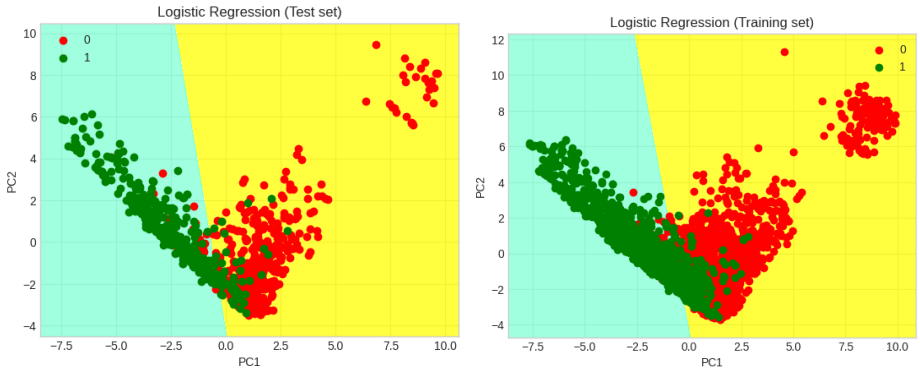

Figure 17. Test Set vs. Training Set for Logistic Regression with 2 Components

Comparing our ROC curves for the 3 cases of PCA Logistic Regression (see **Figure 18**), we can see that the cases of the components being 2, 4, and 5 that performing Logistic Regression with 4 components is the best choice. We can clearly see from the picture that PCA with 4 components is better than PCA with 2 components. We can see in the figure that the AUC for 4 components is the highest even though the curve for 4 components almost overlaps with the curve with 5 components. Thus, we will compare the Logistic Regression for 4 components with Yeo-Johnson scaling with the other models we create.

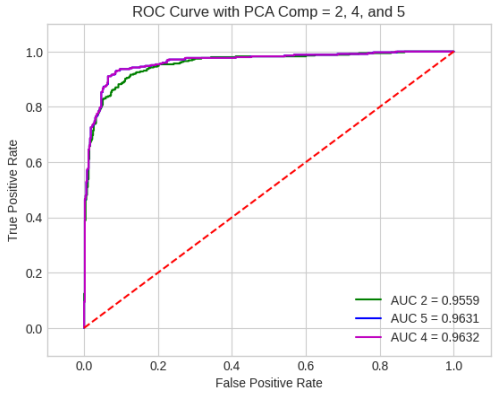

Figure 18. ROC Curves for Logistic Regression with PCA Components of 2, 4 and 5

Ordinary Logistic Regression

The next model that we created is an Ordinary Logistic Regression model. This model differs from the previous model in that the source data is the Binary Transformed data and no PCA preprocessing is implemented. A decision was made to include all 57 predictor variables so that no data is lost and to determine how this affects the

results. This Logistic Regression Model was also coded in Python using sklearn with 80% of the data partitioned into training data and 20% partitioned into test data. From the classification report in **Figure 19**, the accuracy of this model is 93%, which we deem to be a very successful model. The precision, recall, and F1-score are also suggestive of a highly performative model. The full classification report and confusion matrix are presented below.

	precision	recall	f1-score	support
0	0.93	0.96	0.94	538
1	0.94	0.89	0.91	383
accuracy			0.93	921
macro avg	0.93	0.93	0.93	921
weighted avg	0.93	0.93	0.93	921

Figure 19. Classification Report for Ordinary Logistic Regression

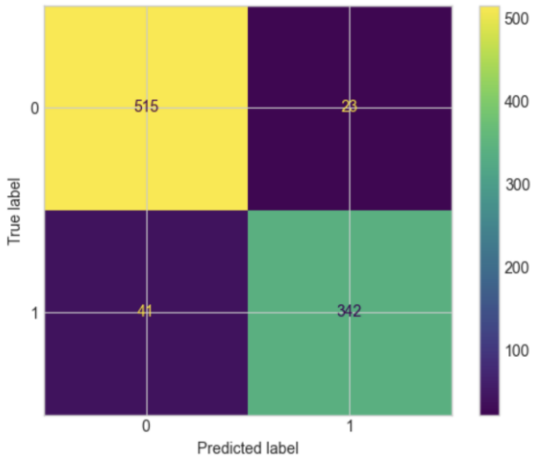

Figure 20. Confusion Matrix for OLR

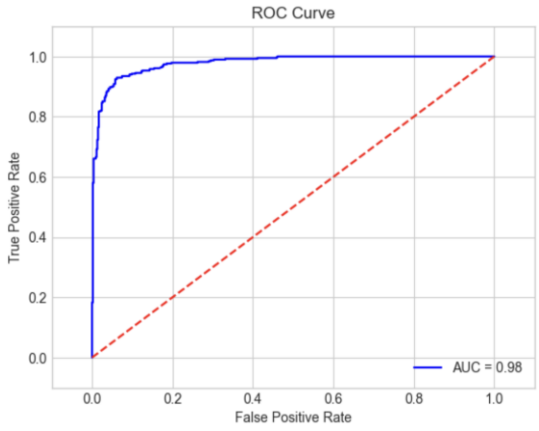

Figure 21. ROC Curve and AUC for OLR

After creating the ROC curve in **Figure 21** and finding the AUC to be 0.98, we determined that this ordinary Logistic Regression Model is one of our best performing models.

Elastic Net Regression

Our next model is Elastic Net regression. Elastic Net regression is a regression method with additional penalizing terms in the loss function of classical regression. The additional term in Elastic Net regression comprises a combination of two different penalization methods called Ridge and Lasso penalization.

Ridge penalization uses the L2 norm of the estimated coefficients along with a parameter called lambda, which indicates the magnitude of the penalty. On the other hand, Lasso penalization uses the L1 norm of the estimated coefficients and a respective magnitude parameter, Lambda. These terms are added to the model's loss

function to restrict the criterion for choosing the best coefficient estimation. Since our goal here is classification, we will add the penalty term to the log-likelihood so that the estimated coefficients would be adjusted according to the penalties as the following formula :

$$\log L(p) - \lambda(\alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2)$$

Log Likelihood | Lasso term | Ridge term

Figure 22. Loss Function of Elastic Net Regression

Ridge and Lasso penalization have different advantages. When the Ridge term is added, it will not zero out any coefficients of predictors, but it can reduce the multicollinearity in high dimensional data by making the coefficients estimates smaller than ordinary regression. Because of this property, Ridge is usually used when prediction is the main interest. On the other hand, the Lasso term can cancel some coefficients because it utilizes the L1 norm, which is a sum of absolute values of estimated coefficients. While it could lose some information by zeroing out some predictors, Lasso penalization is used for its property of identifying significant predictors related to the supervisor. Elastic net regression utilizes some advantageous properties of both penalization by combining the two terms proportionally. As a result, Elastic Net regression can identify the significant predictors related to the supervisor while managing high correlation within the data.

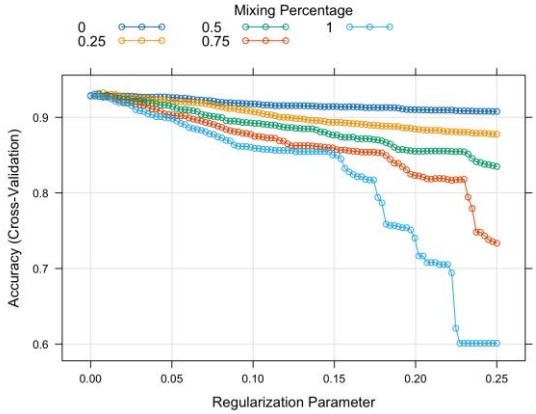

Figure 23. Parameter Estimation for Elastic Net Regression (Lambda and Alpha)

Using 10-fold cross-validation, we estimated the parameters lambda and alpha. Lambda is a parameter for the magnitude of penalization and alpha is a proportion allocation of the penalty terms. Our model estimated the

Alpha to be 0.25, which means the weight of the lasso term is 0.25 and that of the ridge term is 0.75. We saw that the estimated optimal Lambda value is 0.00505. This is indicating the magnitude of the penalization is very small at the optimal level. Since this value is very close to zero, we can assume that the elastic net model will not be very different from the classical regression model without penalization. From the plot below, we can see that the highest cross-validation accuracy occurs when the lambda value is close to zero. Also, when lambda is very small, there is no distinguishable difference between alpha values. Therefore, we see that penalizing with the Elastic Net is not necessary and it may overcomplicate our regression model.

This observation is also reflected in the accuracy of the model. Although the penalty terms could have reduced some coefficients of predictors, the model yielded an accuracy of 0.924, which is smaller than the accuracy of the logistic regression model which is 0.93. While Elastic Net regression could have reduced some multicollinearity and zeroed out some predictors, it fails to improve the performance of prediction.

Naive Bayes

We wanted to fit a model that utilizes the idea of Bayesian inference; we chose to implement a Naive Bayes classifier model. Although Naive Bayesian classification is not necessarily a model that fully utilizes Bayesian inference concepts, it does bring an idea from Bayes Theorem of the conditional distribution. In this model, assuming conditional independence of predictors, the posterior distribution can be simply expressed by the product of the probability distribution of each predictor conditioned on the response category.

$$P(C_k|X) = \frac{P(C_k) \cdot P(X|C_k)}{P(X)} \longrightarrow P(C_k|X) = \prod_{i=1}^{n} P(x_i|C_k)$$

Figure 24. Conditional Independence of Naive Bayesian Classifier

With this assumption of conditional independence and knowledge about the conditional distribution of each predictor variable given a supervisor label, we can easily implement a Naive Bayesian classifier. The reason why this method is called ‘Naive’ is that the assumption of conditional independence of all the predictors is a very unrealistic environment. However, if the condition is met, this method often works as a quite powerful prediction tool. While there are some choices of different kernels and adjustment parameters, we fitted 10-fold cross-validation again to obtain the optimal Naive Bayes classifier.

The performance of the Naive Bayes classifier was not as impressive as our expectation. We observed an accuracy of 0.8941736, the area under the ROC curve was 0.954, the Precision was 0.8617647, the Recall was 0.8746269, and the F1 Score was 0.8681481. These values are low compared to our previous models. We suspect that the assumption of conditional independence was not sufficient in our case even after taking a PCA preprocessing to reduce the multicollinearity between predictors. While the Naive Bayes classification method is very well known as an excellent tool for natural language processing data, our given dataset did not have abundant information about the email contents themselves. Therefore, the Naive Bayes classifier failed to become a candidate for our best-performing models.

K-Nearest Neighbors

Lastly, we wanted to try fitting some simple machine learning models to test the performance in prediction. We chose to utilize the K Nearest Neighbors (KNN) method and took a 10-fold cross-validation for estimating the parameter, K. The KNN model has a very simple logic for classification. Using a distance metric such as Euclidean distance, the model identifies the K nearest neighbors and predicts the supervisor label of new data as the most common class label among the K neighbors. Because the increase in dimensionality expands the computational cost of calculating the distance between data points, it is heavily susceptible to the curse of dimensionality. Our dataset has a lot of predictors resulting in high dimensionality. Therefore, we had to bear with the time to fit the KNN model.

From the cross-validation, we see that the optimal number of neighbors for making a decision is 7. Considering the high dimensionality and the sparsity of sample space, number 7 seems to be quite a small number. However, it does show a very high accuracy compared to other K values as the below plot shows.

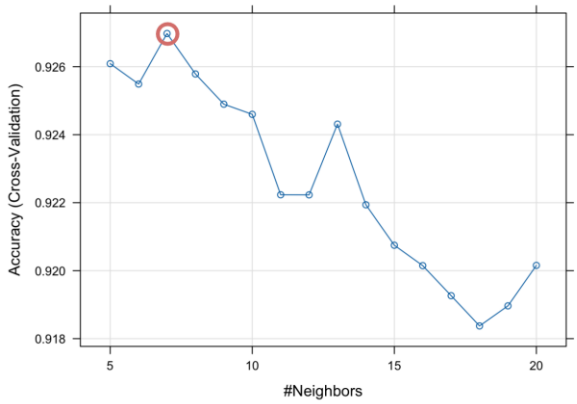

Figure 25. Parameter Estimation for KNN Classifier (K; number of nearest neighbor)

The prediction accuracy was found to be 0.9357907, which is the highest among all of our models. Other accuracy criteria such as Precision, Recall, F1 score, and the area under the ROC curve showed a very high performance. We are surprised to see that the simple implementation of KNN methods resulted in the highest accuracy.

VI. CONCLUSION

In conclusion, there were three main points that we took away from this experience:
Overall, the spam email classification dataset required transformations, such as Yeo-Johnson, Center and Scaling, and Binary Transform. The many categorical predictors lent themselves useful in simpler models, such as Logistic Regression with and without PCA, but also more complex classifiers, such as Naive Bayes and K-Nearest Neighbors. KNN was the most effective. The second most accurate models were the ordinary Logistic Regression with the Binary Transformed dataset and extensions, such as Elastic Net. Logistic Regression with PCA on the original dataset was not as accurate. We learned the importance of data preparation as the models created on the raw data set were not performing as well as those created on the transformed data set. In addition, we also learned about the concepts and nuances of these models. Lastly, we thought about how applications and future development of deep learning models in this spam email classification field can affect businesses in their endeavors.

VII. FUTURE APPLICATIONS

One thing we have kept in mind while progressing through this project is the application of spam email classification in the real world and how models, such as those that are existing or those that we have created ourselves can be made better. Businesses and the corporate world need to have applications that can filter out spam emails and decipher phishing schemes that will harm infrastructure and revenue. Gathering more data, such as IP addresses, and records on the number of times a spam email is sent can aid in making these spam filters more accurate and personalized to the recipient. Large language models and deep learning can also be utilized to streamline the classification of emails from sources that may not be trustworthy. Overall, security of data structures and information is at risk with the numerous cybersecurity threats on the rise. However, through the training of high-performance models, the aim is to mitigate these risks and in return, fortify sensitive intelligence so it is more protected.

VIII. REFERENCES

"Email Spam." *Wikipedia*, Wikimedia Foundation, 3 Apr. 2024, en.wikipedia.org/wiki/Email_spam.

"History of Email Spam." *Wikipedia*, Wikimedia Foundation, 22 Apr. 2024, en.wikipedia.org/wiki/History_of_email_spam.

Kaddoura, Sanaa, et al. "A Systematic Literature Review on Spam Content Detection and Classification." PeerJ. Computer Science, U.S. National Library of Medicine, 20 Jan. 2022, www.ncbi.nlm.nih.gov/pmc/articles/PMC8802784/.

"Principal Component Analysis(Pca)." *GeeksforGeeks*, GeeksforGeeks, 6 Dec. 2023, www.geeksforgeeks.org/principal-component-analysis-pca/.

Sharma, Somesh. "Spam Email Classification." *Kaggle*, 9 July 2020, www.kaggle.com/datasets/somesh24/spambase?resource=download.

"Sklearn.Linear_model.Logisticregression." *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.

Srivastava, Tavish. "A Complete Guide to K-Nearest Neighbors (Updated 2024)." *Analytics Vidhya*, 4 Jan. 2024, www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/#:~:text=The%20K%2DNearest%20Neighbors%20(KNN)%20algorithm%20is%20a%20popular,training%20dataset%20as%20a%20reference.

"What Are Naïve Bayes Classifiers?" *IBM*, 8 Apr. 2024, www.ibm.com/topics/naive-bayes#:~:text=NaÃ¯ve%20Bayes%20is%20part%20of,important%20to%20differentiate%20between%20classes.