CSC 522 ALDA FINAL PROJECT
SPAM EMAIL DETECTION P27

- Data Cleaning
- Methods Development
- Exploration
- Results Analysis
- Conclusion Drawing
- Presentation Preparation
- Final Report Creation

Janelle Correia (jlcorrei) - Wrote code to generate graphs for data visualization, including email type percentage pie chart and histograms for spam/ham counts by date and word count frequency distribution, wrote the information within and added graphs for Section 3.2 (Pre-processing) and Section 4.2 (Plots) in the final report, and added this information to the relevant slides during our November 27th presentation (and presented the data accordingly).

Jayesh Gajbhar (jgajbha) - added the descriptions of TSNE, TFIDF, Word2Vec descriptions in the presentation along with the key takeaways. ran multiple test cases for different numbers of max features of TF-IDF for the algorithms Random Forest, KNN, SVM and Naive Bayes. helped with the TF-IDF and word2vec embeddings in the code. added the algorithm description for TSNE and added the result tables for TF-IDF over all the algorithms using different max features in the report.

Skanda Shastry(sshastr4) - Helped with the data preprocessing. Ran code to get the precision, recall and accuracy of KNN, Random Forest and SVM for multiple vector sizes of Word2Vec embedding. Added the algorithm description for Random Forest Classification .Made the result tables from Word2Vec embedding over all the algorithms using different vector sizes in the report. Helped maintain the readme file for the project's Github repository.

Tushar Kini (tkini) - Wrote the **result analysis** and **conclusion drawing** to be used in the **presentation** and **final report creation**. Wrote the literature review in the report. Wrote the code for T-SNE and model results **plot generation**. Also wrote the code for the TF-IDF and Word2vec embeddings to be used with the 4 algorithms in the **method development** part. Helped with tokenization of the dataset in the data **preprocessing and cleaning part.** Did the formatting for the final report. Added the hypothesis and experiment setup part in the report.

Spam Detection in Emails P27

Janelle Correia(jlcorrei), Jayesh Gajbhar(jgajbha), Skanda Shastry(sshastr4), and Tushar Kini(tkini)

Department of Computer Science, NC State University

Repo Link: [https://github.ncsu.edu/jlcorrei/engr-csc522-fall2023-P27]

1 Background

1.1 Problem Description

In an age where email communication has become an integral part of both personal and professional life, the influx of unsolicited and potentially harmful messages, known as spam, has grown to be a significant concern. Spam emails not only clutter inboxes but also pose security risks, making effective spam detection systems essential. Our project, "Spam Detection in Emails," endeavors to perform an analysis on the current methods of email classification and give insight to which method is the most impactful.

1.2 Literature Survey

We see that Mangena Venu Madhavan et a [3] experiments with SVM, KNN, Naive Bayes and Rough set classifiers. They describe in detail why these algorithms would perform well for email classification and how well they actually did. O. Olatunji (2019) [5] proposed a model based on support vector machines that are suggested for spam identification when carefully searching for optimized parameters for better results. Experimental findings indicate that all earlier models on the same common dataset used in this work are outperformed by the model suggested. Chhabra, Priyanka(2010) [2] also works with SVM for the same task. S. Muhammad Abdulhamid et al. in 2018 [4] suggests that Rotation forest that is a type of forest of decision tree performs the best in the outlined testbed. We see that in K. Agarwal and T. Kumar [1] an hybrid SVM-Naive Bayes mdoel is proposed for spam email detection. Sultana (2020) [6] uses TF-IDF as a method for text based email classification. We derive general intuitions for our project from the above .

2 Proposed Methodology

2.1 Algorithms:

2.1.1 SVM

Support Vector Machine (SVM) is a type of supervised learning method which is typically used for classification as well as regression problems. This method is known to be very efficient for a dataset with high dimensionality. The SVM algorithm attempts to locate a hyperplane in an n-dimensional space where n is the number of features which can distinctly classify any data point. The aim is to find such a plane while maximizing the margin so that a data point can be classified with a high level of confidence going forward.

2.1.2 Naive Bayes

The naive bayes classifier is based on the Bayesian theorem and is usually quite helpful when the number of input dimensions is large. A naive-bayes classifier assumes strong independence among the various attributes of data, but still manages to produce great results. This classifier generates a trained model with efficiency using very few data points given its simple design and oversimplified assumptions. The independence between attributes makes it easy to compute only variances instead of an exhaustive covariance matrix, and this estimation allows us to perform classification.

2.1.3 K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a simple yet effective supervised machine learning algorithm used for classification and regression tasks. It assigns a data point to the majority class among its K nearest neighbors, determined by a distance metric like Euclidean distance. KNN is intuitive, non-parametric, and doesn't assume any underlying data distribution. Its performance hinges on the choice of K and the distance metric. Despite its simplicity, KNN is widely applied in diverse fields, including recommendation systems and image recognition, for its ability to capture intricate patterns in data based on local neighborhood information.

2.1.4 Random Forest Classifier

Random Forest is a commonly used ML algorithm which combines the output of multiple decision trees to reach a single result, handling both classification and regression problems. It builds multiple decision trees during training using a random subset of features and a bootstrapped sample of the training data. The final prediction results from aggregating the predictions of individual trees through voting (for classification) or averaging (for regression), reducing overfitting, increasing accuracy, and offering insights into feature importance. Known for its versatility and robust performance across diverse data types, it is a popular choice in machine learning applications.

2.1.5 TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF, or Term Frequency-Inverse Document Frequency, is a widely used text processing technique for information retrieval and text mining. It evaluates a term's importance in a document relative to a collection of documents by multiplying the term frequency (TF) with the inverse document frequency (IDF). This calculation diminishes common word significance and highlights rare, distinctive terms, proving valuable in text classification, clustering, and search engine ranking. Despite its simplicity, TF-IDF effectively captures the semantic meaning of words within a document, aiding tasks that require understanding term relevance and context within a textual corpus.

2.1.6 Word2Vec

Word2Vec, a popular unsupervised learning algorithm used for natural language processing, focuses on word embedding. It represents words as dense vectors in a continuous space, capturing semantic relationships. Word2Vec learns embeddings based on word context in a large text corpus, operating on the principle that words in similar contexts have related meanings. Models like Continuous Bag of Words (CBOW) and Skip-gram predict surrounding words, preserving semantic information for tasks like text similarity, named entity recognition, and sentiment analysis. Word2Vec's ability to capture nuanced word meanings and relationships enhances the performance of downstream machine learning models in various natural language processing applications.

2.2 Intuition

Supervised Learning provides the system with certain inputs and corresponding outputs where a general rule is generated that maps input to its corresponding output. We try to use supervised methods to classify if an email is spam or not spam. We have a binary classification task at hand and after perusing the literature mentioned above we decide upon comparing 4 different algorithms. We choose SVM and Naive Bayes classifier as they have been shown to perform very well for such NLP tasks where classification is done based on the email text. We go ahead with using KNN as we wanted to test if classification can be done taking into consideration the K most similar email texts. We choose Random Forest Classifiers as we wanted to test the performance of a decision tree based classifier for this purpose. Not using a decision tree was done to reduce bias and variance in datasets that might come due to a single tree. A preliminary investigation of using Kmeans on tha dataset along with the t-SNE plots described below, we saw that simple unsupervised clustering methods like KMeans will not be well suited for the job due to the non separability of the 2 clusters in a linear way. The decision of using Word2vec and TF-IDF was because we wanted to compare the 2 most common embedding methods in NLP tasks. We also tried to include one embedding that was based on the frequency count of a word and one to include a higher dimensional vector representation of words. We also kept in mind the type of data produced by the embeddings. Word2Vec produces dense data whereas TD-IDF produces sparse data after transformation. These representation might influence the performance of the models selected above.

3 Experiment setup

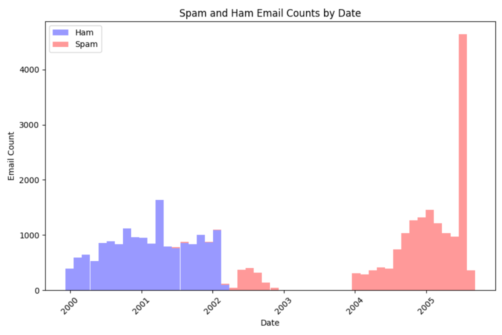
3.1 Dataset

Our project will make use of the Enron-Spam dataset link . It contains over total emails along with the subject and message, each labelled as either spam or ham.

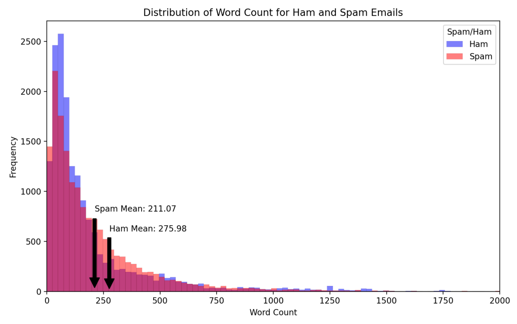
3.2 Pre-processing

Data pre-proccessing played a very important role in the performance optimization of our project so far. We did comprehensive data cleansing to ensure the quality and relevance of our dataset. Specifically, for forwarded emails, we performed a thorough check for null values to ensure that our dataset was complete and free of missing information. We also improved the quality of our subject headers by removing extraneous punctuation and words like "re:". These steps were essential in refining our dataset to allow our algorithms (see above) to operate effectively and to derive meaningful insights from the data. Proper data preprocessing is a critical step that allowed us to identify valuable information and data patterns.

Based on the resulting dataset following data preprocessing and tokenization, we see that the we have 17,171 ham emails and 16,545 spam emails. Though there is no class imbalance, we use class weights for training.



(a) Spam and Ham Email Counts by Date

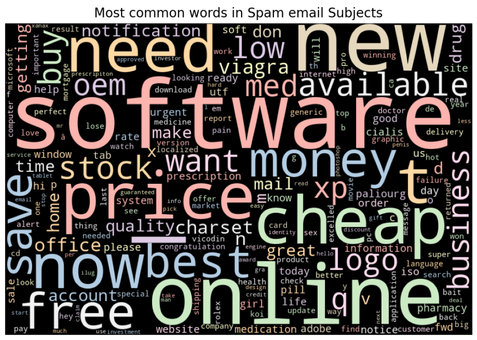


(b) Spam and Ham Word Count Frequency Distribution

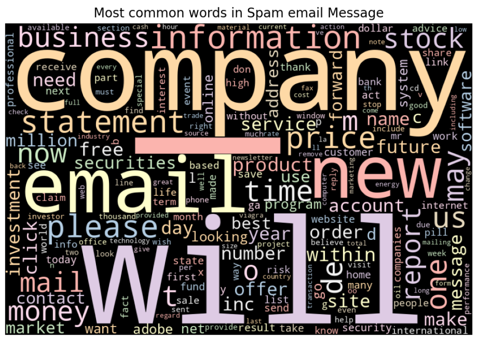
The Ham/Spam Email Counts by Date histogram indicates a concentration of ham data between 2000 and 2002, with some overlap of spam data. From mid-2002 to 2003, spam data becomes more concentrated, and from 2004 to 2005, it dominates. This shift suggests changing email dynamics and possibly increased reliance on spam filtering or email security measures during that period, reflecting a shift from a higher concentration of ham to spam emails. The observed patterns may also be influenced by data cleanup or processing changes in the Enron dataset during specific time frames.

We analyzed total word count distributions in both valid and spam emails. The mean word count for spam emails is 211.07, while ham emails have a mean word count of 275.98, indicating that ham emails are slightly longer on average. The shorter average length of spam emails may indicate a deliberate effort by spammers to convey messages more concisely for better reader attention.

Our Word Cloud analysis offered an intuitive and visual way to discern the most significant terms in various aspects of email data, which enhanced our understanding of email content, provided valuable insights for distinguishing between spam and legitimate emails. This allowed us to identify patterns and frequently occurring keywords in spam email subjects and messages, which will allow us to understand the common themes and tactic used by spammers.

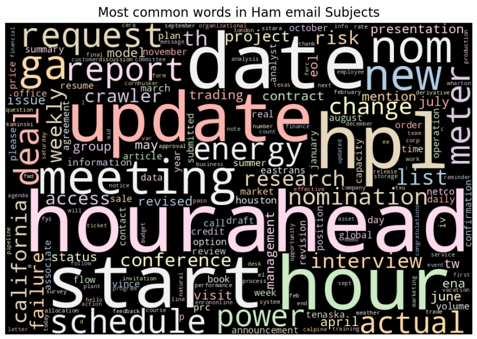


(a) Spam Subject Word Cloud

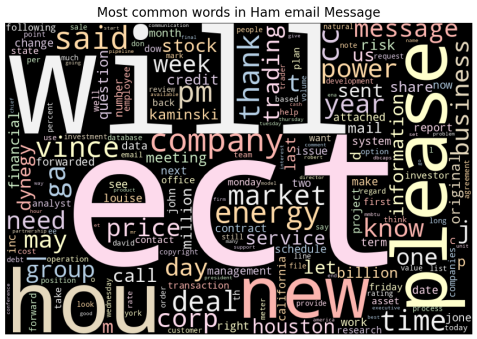


(b) Spam Message Word Cloud

Figure 2: Spam Dataset Word Clouds



(a) Ham Subject Word Cloud



(b) Ham Message Word Cloud

Figure 3: Ham Dataset Word Clouds

3.3 t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a dimensionality reduction method used for visualizing complex data patterns. Unlike PCA, t-SNE focuses on preserving pairwise similarities between data points, revealing local structures and clusters. It excels at capturing intricate patterns and is effective for exploratory data analysis, highlighting subtle relationships. t-SNE emphasizes local structures over global ones, making it suitable for revealing non-linear patterns. However, it is sensitive to hyperparameter choices and may not preserve global distances accurately.

3.4 Cross Validation

We used the Cross-Validation method in our project to assess the performance and generalization ability of our analysis models. The Cross-Validation technique helps to mitigate issues related to overfitting and gives a more reliable estimate of a model's performance. Specifically, we used a 5-fold Cross-Validation strategy, where the dataset was divided into 5 approximately equal-sized subsets. In each iteration, 4 subsets were used for training the model, while the remaining subset was used for testing. We repeated this process 5 times having each subset as the test set once. We then averaged the results from each iteration to obtain the 5-fold CV score which increases the confidence in the capacity of our models for future data generalization.

3.5 Hypothesis

Through the execution of this project, we not only want a supervised classifier that can effectively classify emails based on their text, we also want to see how our chosen algorithms behave in test conditions. We want to find which one from Word2vec and TF-IDF is suitable for our task. Also we want to experiment with the dimensionality of the data given to our 4 algorithms and verify if increasing the dimensions and thus using more compute gives better results. We also try to find the hyper-parameters for which the given algorithms would perform the best and try to find a rationale behind it. We also want to test if we can effectively use t-SNE as a visualization method for clusters instead of PCA given the drawback mentioned earlier.

3.6 Experimental Design

We use all the 4 algorithms and do the hyper-parameter tuning as described ahead. We use TF-IDF and set the maximum feature parameter to 5000, 2500 and 1000 to demonstrate decreasing dimensionality and feed it to all models. For Word2vec we select the number of dimensions for each word from a value from 50, 100 and 200 dimensions. We use the cosine and euclidean distance parameter for KNN and also experiment with 3, 5 and 10 nearest neighbors. For SVM we go ahead and use Grid Search from values of C and using RBF and Polynomial kernel. For Random Forest Classifiers we experiment with the maximum depth of the tree and the number of trees.

4 Experiment Results

4.1 Metric Values

We use Accuracy as a metric to give us an general indication of how well our model performs. Out of Precision and Recall we decide that for our use case at hand, Recall is more important as we need to reduce the False Positives in our prediction as labelling spam emails as not spam could be potentially more dangerous that labelling a non-spam email as a spam one.

Max Features	Recall	Accuracy
5000	0.9876	0.9815
2500	0.9843	0.9777
1000	0.9794	0.9667

Table 1: Naive Bayes Results for TF-IDF

Vector Size 50			
Kernel	C	Recall	Accuracy
Poly	0.1	0.9883	0.9747
RBF	0.1	0.9885	0.9790
Poly	1	0.9889	0.9807
RBF	1	0.9893	0.9825
Vector Size 100			
Kernel	C	Recall	Accuracy
Poly	0.1	0.9893	0.9757
RBF	0.1	0.9891	0.9798
Poly	1	0.9901	0.9820
RBF	1	0.9909	0.9836
Vector Size 200			
Kernel	C	Recall	Accuracy
Poly	0.1	0.9895	0.9758
RBF	0.1	0.9901	0.9795
Poly	1	0.9911	0.9832
RBF	1	0.9918	0.9849

Table 2: Word2vec results

Max Features: 5000			
Kernel	C	Recall	Accuracy
Poly	0.1	1.0	0.9994
RBF	0.1	0.9945	0.9849
Poly	1	1.0	0.9997
RBF	1	0.9932	0.9897
Max Features: 2500			
Kernel	C	Recall	Accuracy
Poly	0.1	1.0	0.9994
RBF	0.1	0.9924	0.9830
Poly	1	0.9903	0.9869
RBF	1	0.9907	0.9875
Max Features: 1000			
Kernel	C	Recall	Accuracy
Poly	0.1	1.0	0.9994
RBF	0.1	0.9897	0.9757
Poly	1	1.0	0.9997
RBF	1	0.9901	0.9836

Table 3: TF-IDF results

Table 4: SVM results

Vector Size	Recall	Accuracy
200	0.9887	0.9820
100	0.9874	0.9817
50	0.9880	0.9812

Table 5: Word2vec results

Max Features	Recall	Accuracy
5000	0.9868	0.9826
2500	0.9851	0.9814
1000	0.9785	0.9752

Table 6: TF-IDF Results

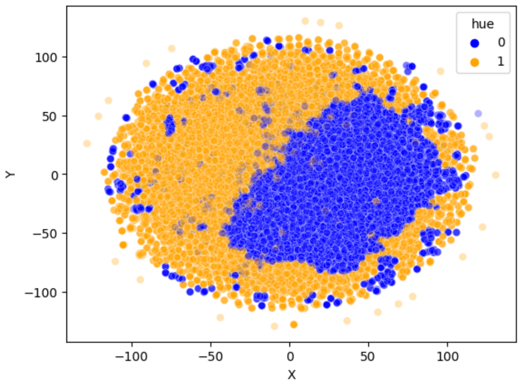
Table 7: Random Forest Classifier results

Metric		Max Features: 5000		Vector Size 200	
Neighbors	Distance Metric	Recall	Accuracy	Recall	Accuracy
3	Cosine	0.9800	0.9720	0.9829	0.9807
5	Cosine	0.9705	0.9671	0.9808	0.9787
10	Cosine	0.9657	0.9662	0.9769	0.9779
3	Euclidean	0.9955	0.8341	0.9864	0.9793
5	Euclidean	0.9961	0.8030	0.9868	0.9795
10	Euclidean	0.9959	0.7698	0.9847	0.9790
Metric		Max Features: 2500		Vector Size 100	
Neighbors	Distance Metric	Recall	Accuracy	Recall	Accuracy
3	Cosine	0.9742	0.9647	0.9849	0.9817
5	Cosine	0.9673	0.9616	0.9787	0.9780
10	Cosine	0.9617	0.9623	0.9765	0.9781
3	Euclidean	0.9940	0.8131	0.9847	0.9785
5	Euclidean	0.9955	0.7804	0.9864	0.9793
10	Euclidean	0.9965	0.7495	0.9833	0.9789
Metric		Max Features: 1000		Vector Size 50	
Neighbors	Distance Metric	Recall	Accuracy	Recall	Accuracy
3	Cosine	0.9702	0.9523	0.9837	0.9807
5	Cosine	0.9599	0.9482	0.9798	0.9786
10	Cosine	0.9560	0.9502	0.9779	0.9781
3	Euclidean	0.9889	0.7945	0.9856	0.9795
5	Euclidean	0.9893	0.7615	0.9862	0.9796
10	Euclidean	0.9943	0.7253	0.9849	0.9799

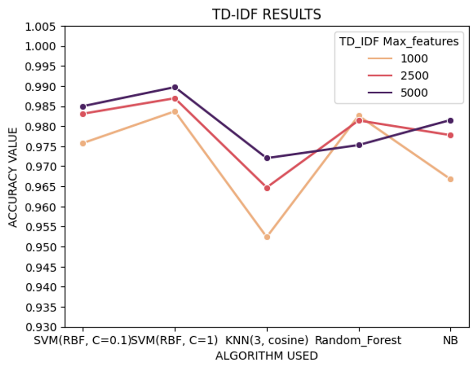
Table 8: KNN results for TF-IDF and Word2vec embeddings

4.2 Plots

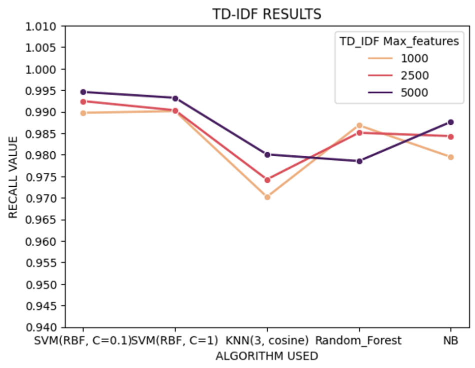
The t-SNE plot generated reveals noticeable patterns in the distribution of ham/valid (yellow) and spam (blue) email data. The higher concentration of ham/valid data in the combined t-SNE plot highlights the prevalence of legitimate emails in the Enron dataset, but the concentrated nature of both the ham and spam clusters suggests that each class has distinct internal structures, i.e., the data is clearly separable, meaning that there are features unique to each subset of data (spam vs. ham).



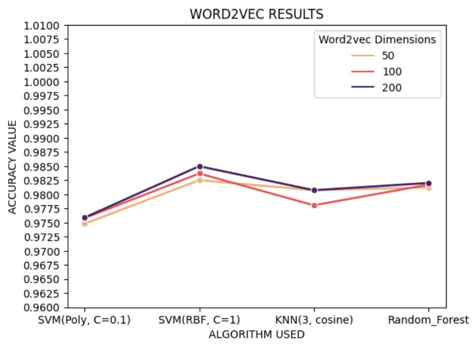
(a) Ham / Spam Data t-SNE Plot



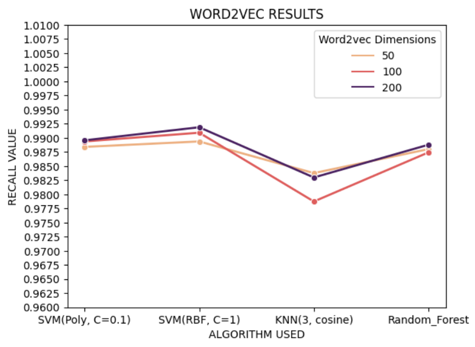
(a) Accuracy Values for TF-IDF



(b) Recall Values for TF-IDF



(a) Accuracy Results for Word2vec Embeddings



(b) Recall Results for Word2vec Embeddings

4.3 Critical Evaluation

We make use of the frequency distribution plots to see that the spam and ham emails are well segregated by date. We can see a cluster of spam emails at a later date. Also looking at the word count distribution, we can deduce that the average spam email is shorter than the ham emails . Also the t-SNE plot shows that the 2 clusters are pretty much separable. This has resulted in our metric scores being so high and we were not surprised by them.

For the KNN results, we notice that increasing the number of neighbors considered in making the classification, results in a lower accuracy and recall score. This prompts us to look in a small neighborhood for classification. Also we notice that using cosine as a distance metric gives us better results for both embeddings. This difference is more pronounced in TF-IDF than the Word2vec results. For the same number of neighbors and distance metric , we see that the Word2vec performs better than TF-IDF as we can assume that cosine measure would be better for vector representation in a higher dimension than euclidean distance.

Considering the SVM results we can clearly see that the RBF kernel trick performs better than the polynomial kernel that can be attributed to the concentric nature of clusters as shown in t-SNE plots. We can also see that a higher parameter C gives better results for both kernels.

For the Random Forest Classifier result we can note an important point that increasing the number of dimensions for Word2vec and features in TF-IDF, does not give better results. So we can say that high dimensionality doesn't always give better results considering the higher cost of resources utilized. On the contrary, we see that the metrics decrease by increasing the number of max features using TF-IDF. This can be attributed to the fact that the random forest would overfit the dataset given more data and increase the variance. For Naive Bayes we see that increasing the number of maximum features for TF-IDF increases the accuracy as we can assume that more data would give us a better estimate of prior probabilities.

5 Conclusion

We can see that using higher dimensional data and using more computational resources is not always beneficial to increase the accuracy metric. We can conclude that a choice for the dimensionality is very dependent on the algorithm being used as the results would vary a lot. Using more resources and using higher dimensions for an algorithm that gives good enough accuracy with a lower dimension is not justified. Also more data might introduce noise in our classification and reduce metrics owing to increased variance by overfitting. We see that increasing the number of neighbors in KNN classification increases the noise in calculation and is not always advised for better generalization. Choosing the distance metric for KNN is an important factor and that should be decided by the semantic meaning of the metric with respect to the data. Cosine distance makes more sense in higher dimensional vector space whereas Manhattan does not. Choosing a kernel trick for the SVM classifier should always be backed with some kind of support from the type of clusters we are dealing with. Complex clusters would probably do well with RBF, whereas linearly separable clusters would warrant a simpler linear kernel. Keeping in mind that PCA is not a great visualization tool as the first 2 or 3 dimensions used for visualization might not capture the required explained variance from the original dataset. Instead methods like t-SNE that preserve the similarity between data points should be used. We should also consider the type of data used for an algorithm. Some algorithms might work better for dense data whereas others might not. Computational times are also decided by this factor.

References

[1] Kriti Agarwal and Tarun Kumar. Email spam detection using integrated approach of naïve bayes and particle swarm optimization. In *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 685–690, 2018.

[2] Wadhvani R. & Shukla S. Chhabra, P. "spam filtering using support vector machine.". *International Journal of Engineering Research*, 9:6, 2010.

[3] Mangena Venu Madhavan et al. Title of the paper. *IOP Conf. Ser.: Mater. Sci. Eng.*, 1022:012113, 2021.

[4] Shafi'i Muhammad Abdulhamid, Maryam Shuaib, Oluwafemi Osho, Idris Ismaila, and John K. Alhassan. Comparative analysis of classification algorithms for email spam detection. *International Journal of Computer Network and Information Security*, 10(1):60–67, January 2018.

[5] Sunday Olusanya Olatunji. Improved email spam detection model based on support vector machines. *Neural Computing and Applications*, 31(3):691–699, June 2017.

[6] T. Sultana. Email based spam detection. *International Journal of Engineering Research*, 9, 2020.