Email Phishing Detection using Machine Learning Models

An Information Security Group Project

November 09, 2022

Table of Contents

Abstract

Phishing extracts sensitive information from unwary victims and is a social engineering threat. The attackers appear as legitimate and credible individuals. The most used communication channels for such attacks are emails, and as such, we focus on this task domain.

Techniques used in prior works range from the simpler decision tree to complex deep learning methodologies including BiLSTM and transformer frameworks. As an NLP task, the identification of phishing emails also requires a significant amount of data preprocessing, using techniques including vectorisation, stopword removal, and feature representation.

The embedded vector representations of the input emails in text form are fed into our models, which include a decision tree, a pruned decision tree, boosting, k-nearest neighbours classification, and an SVM (Support Vector Machine).

We obtain a high accuracy of 99.79% using our best model, the SVM. This empirically demonstrates that deep learning models, even simpler ones such as the SVM, demonstrate higher accuracies as compared to machine learning methods, due to their better feature mapping and representation ability. This is due to their more complex and abstracted nature.

I. Introduction

One of the most essential ways in which people communicate with each other on a personal or business perspective is through the use of emails. They are reliable, easy to access and a quick way of communicating. As a result with the popular use of email, comes the rise of spam emails. These emails try to mimic recognized and genuine organisations and try to lure people into clicking on the fake email link which helps attackers steal confidential information of the victims.
Such cyber attacks are termed as phishing.

Our paper mainly focuses on discovering which machine learning or deep learning algorithm is a better method of phishing detection. The algorithms that are mainly focused on this paper include decision tree, decision tree with pruning, boosting, k-nearest neighbours classification, and Support Vector Machine.

Our system model pre-processes the data and is represented in the form of a vector.This dataset is then fed into each of the models and their AUC and ROC is evaluated which gives us an idea of which model provides higher accuracy for phishing detection.

The rest of the paper is organised as follows: Section 2 discusses the literature survey related to this paper. Section 3 explains the system model that has been proposed. Section 4 discusses the implementation of each algorithm. Section 5 presents the results and discussion. The conclusions of the paper are presented in Section 6.

II. Literature Survey

Publication Journal & Date	Major Techniques Used	Results/Outcomes	Drawbacks
Springer, 29 July 2021, "A hybrid DNN–LSTM model for detecting phishing URLs"[1]	Two types of feature extraction techniques used in combination: character embedding-based and manual natural language processing (NLP) feature extraction. Bidirectional LSTM nodes are used in conjunction with Deep Neural Networks, forming a hybrid deep neural network. In training, dropout, ReLU and Adam optimizer were used.	The Bidirectional LSTM provides better results than standard LSTM due to its ability to trace connections both forward and backward. Hybrid models are also found to be better due to their ability to simultaneously use both NLP features and character embedding features.	● Manual parameter tuning for NLP, character embedding ● Long training time due to hybrid model ● Use of custom dataset for testing
IEEE, 18 May 2022, "A Systematic Literature Review on Phishing Detection Using Natural Language Processing Techniques"[2]	● Supervised Classical ML Techniques(Decision Tree, Random Forest, Naive Bayes, SVM, Neural Networks, Linear Regression, K-Nearest Neighbours)	This survey paper aims to answer a list of queries Q1. Key research areas in phishing email detection using NLP A1. Feature extraction, optimisation techniques Q2. ML algorithms	● The findings of the paper have not been generalized to Non-English languages ● Complexity of the cybersecurity

	• Supervised Deep Learning Algorithms(CNN, RNN) • Unsupervised Learning Methods(K-Means Clustering) • Feature Extraction(PCA, Latent Semantic Analysis, Chi-Square, Mutual Information/Information Gain)	used in phishing detection email using NLP A2. SVM, Naive Bayes, Decision Tree, Random Forest, RNN, CNN Q3. Optimisations techniques used in phishing detection email using NLP A3. Adam optimiser, Sequential Minimal Optimisation Q4. Text features in phishing email detection using NLP A4. Bag-of-Words, Information Gained, Word2Vec, POS, PCA Q5. NLP techniques used in phishing email detection A5. Stopword removal, punctuations, special characters, stemming, Tokenisation, TF-IDF Q6. Datasets and resources A6. Nazario phishing corpus, SpamAssassin Public Corpus, Enron dataset, PhishTank,	domain has increased significantly, so there is a need to devise adaptable evaluation plans • Other model-related issues such as inadequate time validity and range • Accurate contrast of several phishing detection techniques is hard to produce because of the lack of basic benchmarks

		IWSPA-AP, and TREC corpus Q7. Evaluation criteria of the machine/deep learning techniques that were used in phishing email detection using NLP A7. Accuracy, precision, recall, and F1-score Q8. Tools used in phishing detection email using NLP A8. Python, Weka, Scikit-learn library, Java, TensorFlow Q9. Parts of the email that are the most widely used in phishing detection email using NLP A9. Email body text, header, URL	
IEEE, 17 February 2022, "Deep Learning for Phishing Detection: Taxonomy, Current Challenges and Future Directions"[3]	A Systematic Literature Review is adopted to analyze the relevant studies to support this research. ● Ensemble deep learning (EDL) models[Long	This study provided a comprehensive review of DL for phishing detection through an in-depth SLR approach. The paper also offered a significant insight	● There is no specific guideline for an optimal set of parameters that yield the best performance

	Short Term Memory(LSTM) and Bidirectional Long Short Term Memory (BiLSTM)] ● Discriminative DL models(Convolutional Neural Networks (CNN), Multilayer Perceptron (MLP)) ● Hybrid DL Model(Generative Adversarial Networks(GAN), Deep Neural Network(DNN))	into the current issues and challenges that DL faces in detecting phishing attacks by analyzing the trends and patterns of 81 selected articles from various sources. This research has drawn a taxonomy for phishing detection and DL to classify them into several classes based on a thorough analysis of the relevant studies. An empirical analysis was conducted in which accuracy of the various models are as follows : DNN-96.38% MLP-95.97% CNN-95.21% LSTM-LSTM:93.7%	accuracy in detecting phishing attacks. ● Researchers need to find-tune these parameters manually by conducting a very tedious and time-consuming series of experiments. ● Individual DL models might produce lower accuracy as compared to ensemble or hybrid models. ● Training duration is an important factor that needs to be

			taken into consideration.
ScienceDirect, 23 March 2021, "Spam Email Detection Using Deep Learning Techniques"[4]	• Makes use of word embedding (NLP technique) to classify spam emails. • Pre-trained transformer model BERT • DNN (deep neural network) model that contains a BiLSTM (bidirectional Long Short Term Memory) layer and two stacked Dense layers • set of classic classifiers K-NN and Naïve Bayes are also used to compare the results to. • Sci-kit learn for data cleaning and keras tokenization tool to split the words as tokens.	This paper aims to introduce the concept of word embedding in order to classify spam emails. For this purpose, a pre-trained BERT transformer model was used for this purpose. The results are also compared to a BiLSTM model and ML classifier algorithms K-NN and Naive Bayes. The BERT model was able to attain the highest accuracy of 98.67%, compared to the other models proving that the contextual representation for the input text does affect the spam classification positively. The models also	• 300 input sequence length limit inhibiting the performance. • Trained only in one language.

		performed well against new test data, showing that the models are not overfitted and the preprocessing step was successful.	
ScienceDirect, 23 March 2021, "Phishing Email Detection Using Natural Language Processing Techniques: A Literature Survey"[5]	Binary search feature selection (BSFS) Doc2Vec and performed a parallel arrangement that employed "SVM, LR, RF, and Naive Bayes (NB) SVM classifier SVM, Neural Network Hybrid chi-square statistics and mutual information to improve dimensionality XGBoost, LightGBM, and Bernoulli Naive Bayes Bernoulli Naive Bayes followed by LightGBM with the TF-IDF	Accuracy of 97.41% F1 score of 81.6% and 76.6% F1-measure of 100% Extremely quick and are also marked by lower time unpredictability. Highest accuracy of 96.5% in 0.157 seconds and 95.4% in 1.708 seconds THEMIS has an accuracy of 99.848%	Cannot effectively detect new phishing scams, which needs significant manual feature engineering. One major flaw with the NLP phishing e-mail detection developed on ML is its high dependence on emails' surface text instead of deep semantics. Further work is required to employ modernized DL techniques in phishing email detection studies, for instance, Recurrent Neural Networks (RNNs), Convolutional neural networks (CNN), and Deep

	THEMIS		Reinforcement Learning models.
Ariyadasa, S., Fernando, S., & Fernando, S. (2020). Detecting phishing attacks using a combined model of LSTM and CNN. *International Journal of Advanced And Applied Sciences*, *7*(7), 56-67.[6]	• The paper provides a solution to detect phishing attacks using a combined model of LSTM and CNN deep networks with the use of both URLs and HTML pages. • The URLs are learned using an LSTM network with 1D convolutional, and another 1D convolutional network is used to learn the HTML features. • These two networks were trained separately and combined through a sigmoid layer by dropping the last layer of	• The proposed model reached an accuracy of 98.34%. • The previously recorded highest accuracy of models using similar methodology was 97.3%.	• HTML pages are still suffering from a requirement of expert knowledge for feature extraction.

	each model to have the proposed model. ● Feature extraction was performed only with the HTML pages. The URLs were directly fed with minimum pre-processing.		
Ho, G., Sharma, A., Javed, M., Paxson, V., & Wagner, D. (2017). Detecting credential spear phishing in enterprise settings. In *26th USENIX Security Symposium (USENIX Security 17)* (pp. 469-485).[7]	● This paper proposes a method which uses features derived from an analysis of fundamental characteristics of spear phishing attacks, combined with a new non-parametric anomaly scoring technique for ranking alerts. ● Introduces a new technique called Directed Anomaly Scoring (DAS). At a high level, DAS ranks all events by	● The proposed system successfully detects 6 known spear phishing campaigns that succeeded (missing one instance); an additional 9 that failed; plus 2 successful spear phishing attacks that were previously unknown. ● Establishes that the detector's false positive rate is low enough to be practical, on	● email and network activity conducted outside of LBNL's network borders will not get recorded in the NIDS logs. ● The system will miss attacks where the email links to an HTTPS website

	comparing how suspicious each event is relative to all other events. Once all events have been ranked, DAS simply selects the N most suspicious (highest-ranked) events, where N is the security team's alert budget.	average, a single analyst can investigate an entire month's worth of alerts in under 15 minutes. ● According to the data reported by the paper, Standard anomaly detection techniques using the same features would need to generate at least 9 times as many alerts to detect the same number of attacks. ● The system was tested against common anomaly detection techniques such as: Kernel Density Estimation (KDE), Gaussian	

		Mixture Models (GMM), and k-Nearest Neighbors (kNN). All three traditional techniques detected fewer than 25% of the attacks found by DAS.	
Hakim, Z. M., Ebner, N. C., Oliveira, D. S., Getz, S. J., Levin, B. E., Lin, T., ... & Wilson, R. C. (2021). The Phishing Email Suspicion Test (PEST) a lab-based task for evaluating the cognitive mechanisms of phishing detection. *Behavior research methods*, *53*(3), 1342-1352.[8]	• The proposed Phishing Email Suspicion Test (PEST), and a cognitive model to quantify behavior provides a framework for studying the cognitive neuroscience of phishing detection. • In PEST, participants rate a series of phishing and non-phishing emails according to their level of suspicion. • In the proposed computational	• In this test, participants evaluated a series of 160 emails regarding their level of suspicion. Only 62% of the participants classified the emails correctly. • By comparing real-world and lab-based efficacy, the paper was able to test, for the first time, whether email phishing susceptibility in	• PEST is not effective at determining the phishing susceptibility of individual people. • The approach makes use of relatively narrow demographics of our participants, who were undergraduate students at the University of Arizona with a mean age under 19

	model, behavior is quantified in terms of participants' overall level of suspicion of emails, their ability to distinguish phishing from non-phishing emails, and the extent to which emails from the recent past bias their current decision.	the lab is related to email phishing susceptibility in the real world.	years old.
Fitzpatrick, B., Liang, X., & Straub, J. (2021). Fake news and phishing detection using a machine learning trained expert system. *arXiv preprint arXiv:2108.08264*.[9]	• This paper presents the use of a machine learning trained expert system (MLES) for phishing site detection and fake news detection. • The paper aims to design a rule-fact network that allows a computer to make	• The paper has demonstrated that the MLES system is able to support network designs for both fake news and phishing detection, despite key differences in data and the network design approach taken by the	• The model has to be tested on more datasets to arrive at a better accurate conclusion regarding performance and accuracy. • Decisions have to be refined based on system

	explainable decisions like domain experts in each respective area. ● The phishing website detection study uses a MLES to detect potential phishing websites by analyzing site properties (like URL length and expiration time). ● The fake news detection study uses a MLES rule-fact network to gauge news story truthfulness based on factors such as emotion, the speaker's political affiliation status, and job.	researchers. ● The paper has also discussed how the MLES can be used with real-world data and shown how it can be used with existing datasets that were developed for use with neural network techniques.	performance after training.
Sun, Y., Chong, N., & Ochiai, H. (2021). Federated Phish Bowl: LSTM-Based Decentralized Phishing Email	● The paper proposes a decentralized phishing email detection framework called	● The paper show that FedPB can attain a competitive performance with a	● FedPB can encounter threats such as backdoor attacks and information-s

Detection. *arXiv preprint arXiv:2110.06025*.[10]	Federated Phish Bowl (FedPB) which facilitates collaborative phishing detection with privacy. ● Using LTSM for phishing detection, the framework adapts by sharing a global word embedding matrix across the clients, with each client running its local model with Non-IID data.	centralized phishing detector, with generality to various cases of FL retaining a prediction accuracy of 83%.	tealing attacks.
Osman, A. H., & Aljahdali, H. M. (2017). Feature weight optimization mechanism for email spam detection based on two-step clustering algorithm and logistic regression method. *International Journal of Advanced Computer Science*	● This paper proposed an improved filtering spam technique for suspected emails, messages based on feature weight and the combination of two-step clustering and logistic	● The paper indicates that the performance evaluation is enforcing the filtering accuracy results and proved better results after weighting process and feature selection.	● If the number of observations is lesser than the number of features, Then Logistic Regression might lead to overfitting. ● Logistic Regression

and Applications, *8*(10).[11]	regression algorithm. ● Unique, important features are used as the optimum input for a hybrid proposed approach. This study adopted a spam detector model based on distance measure and threshold value. ● Two-step clustering algorithm was used to generate a new feature called "Label" to cluster and differentiate the diversity emails and group them based on the inter samples similarity. ● Thereby the spam filtering process was simplified using the Logistic regression classifier in order	● The prediction of email filtering using the logistic regression method extracted average accuracy results with 94.96% for testing phase before feature weighting process. average accuracy results of 96.48 % in the testing phase was achieved after selecting significant features using the feature weight process.	assumes linearity between the dependent variable and the independent variables.

	to distinguish the hidden patterns of spam and non-spam emails.		
Soon, G. K., On, C. K., Rusli, N. M., Fun, T. S., Alfred, R., & Guan, T. T. (2020, March). Comparison of simple feedforward neural network, recurrent neural network and ensemble neural networks in phishing detection. In *Journal of Physics: Conference Series* (Vol. 1502, No. 1, p. 012033). IOP Publishing.[12]	• The objective of this paper is to investigate the performance of feedforward neural network, recurrent neural network and ensemble neural network in phishing email detection.	• The parameter tuning experiment employed in the paper showed that FFNN and ENN require less hidden neurons compared to RNN due to the complexity in the RNN. • FFNN achieved an average accuracy of 97.11%, RNN: 97.11% and ENN: 96.8895.	• The training of the advanced neural networks made use of is resource intensive and slow. • The models face issues like Exploding or Gradient Vanishing.
Bergholz, A., Chang, J. H., Paass, G., Reichartz, F., & Strobel, S. (2008, August). Improved Phishing Detection using Model-Based Features. In	• The paper investigates the statistical filtering of phishing emails, where a classifier is trained on characteristic	• The CLTOM model achieved an accuracy of 94.93%. • The adaptive DMC approach introduced by the paper	• The paper does not use a representativ e set of legitimate and phishing emails from

CEAS.[13]	features of existing emails and subsequently is able to identify new phishing emails with different contents. ● Advanced email features are generated by adaptively trained Dynamic Markov Chains and by novel latent Class-Topic Models. ● A new statistical model was developed, the latent Class-Topic Model (CLTOM) which incorporates category information of emails during the model inference for topic extraction.	reduces the memory requirements compared to the standard DMC approach by two thirds almost without any loss in performance.	real users for training the model. ● For very large topic numbers CLTOM is prone to overfitting.
KAYTAN, M., & HANBAY, D. (2017). Effective classification of	● The paper proposed an intelligent model for detecting	● The average classification accuracy of ELM was	● An ELM cannot encode more than 1 layer

phishing web pages based on new rules by using extreme learning machines. *Computer Science*, *2*(1), 15-36.[14]	phishing web pages based on Extreme Learning Machine. ● We must use a specific web page features set to prevent phishing attacks. We proposed a model based on machine learning techniques to detect phishing web pages.We have suggested some new rules to have efficient features.	measured as 95.05% and the highest classification accuracy was measured as 95.93%.	of abstraction. ● Although ELM can be trained really fast, they have very slow evaluation and poor performance.
Rao, R. S., & Ali, S. T. (2015, April). A computer vision technique to detect phishing attacks. In *2015 Fifth International Conference on Communication Systems and Network Technologies* (pp. 596-601). IEEE.[15]	● The paper proposes a novel solution to defend zero-day phishing attacks based on a combination of whitelist and visual similarity based techniques. ● The method uses computer vision technique called SURF detector to extract	● Bychoosing threshold t = 0.7, the model was able to correctly classify all the phishing websites with an acceptable percentage of false positives and false negatives i.e. 20.11% and	● SIFT offers better performance than SURF in different scale images. ● Does not take CSS style into account.

	discriminative key point features from both suspicious and targeted websites.	15.23% respectively.	
Feng, F., Zhou, Q., Shen, Z., Yang, X., Han, L., & Wang, J. (2018). The application of a novel neural network in the detection of phishing websites. *Journal of Ambient Intelligence and Humanized Computing*, 1-15.[16]	• The paper proposes a novel phishing detection model based on the Monte Carlo algorithm. • This detection model can achieve high accuracy and has good generalization ability by design risk minimization principle.	• The proposed detection model can achieve a high Accuracy of 97.71% and a low FPR of 1.7%.	• The number of layers can be increased in the neural network algorithm to further improve accuracy. • The paper does not prioritize the time efficiency of the proposed model in detail.

Table 2.1. Literature Survey

Limitations in Existing Methodologies

From our extensive literature survey, certain recurring drawbacks or gaps showed in most papers. They are as follows

1. Long training time
2. Lack of generalizability
3. No specific guideline for an optimal set of parameters that yield the best performance

4. Cannot effectively detect new phishing scams, requires manual feature engineering
5. Lack of diverse datasets
6. Dataset imbalance
7. ML approaches give better results but with scalability trade-off and time-consuming even on the small-sized datasets
8. Lack of accessible real world data due to privacy concerns
9. High computation and storage complexity of dataset

Considering these gaps in the current state of research in this domain, we have designed our model to overcome the aforementioned limitations.

III. System Model

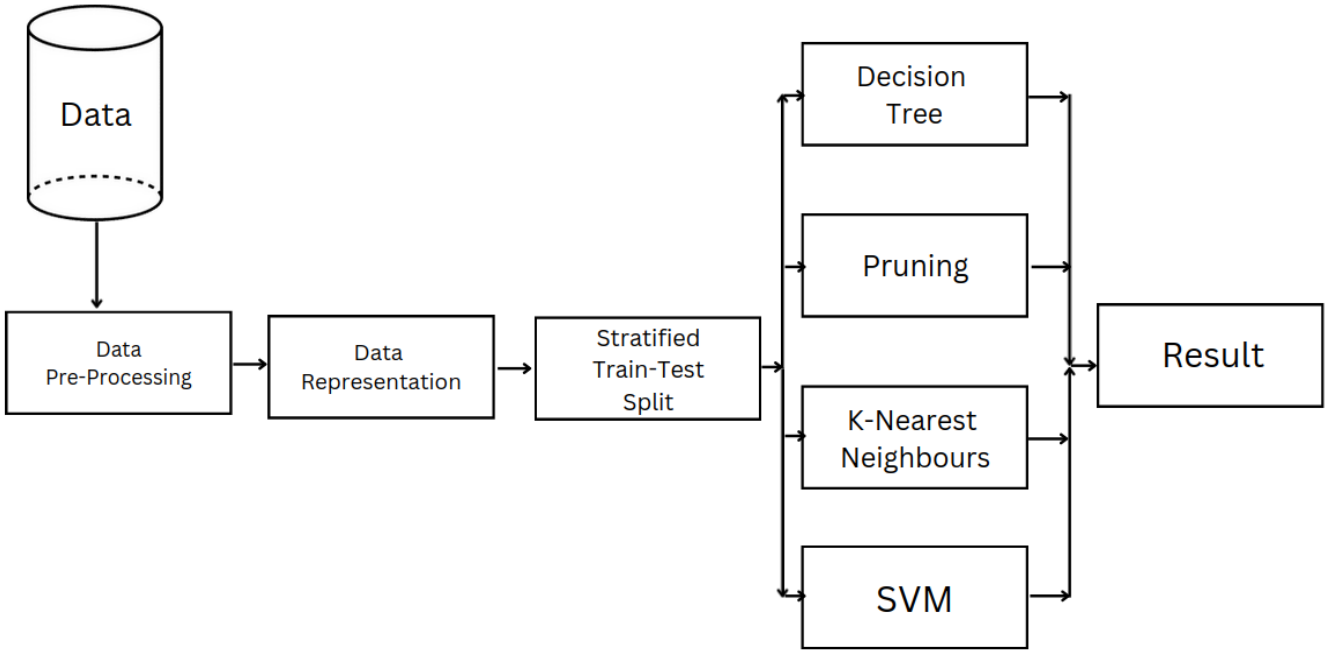

Fig. 3.1. Overall Architecture Diagram

Data Preprocessing

Data preprocessing is the process through which data is altered, or encoded, to make it more easily understandable to a machine. In other words, the algorithm can now quickly decipher the data's features. Similar to this, the dataset needs to be prepared before any machine learning technique is used. Fortunately, there are no null values in our Kaggle dataset, therefore the null values do not need to be filtered. As a result, we can examine each feature or area that contributes to a website becoming a phishing website.

The first column is the index number of the respective row. We can delete this column or field from our dataset because it is not necessary for the output to depend on the row's index number. The fields in the second column through the thirty-first column are crucial to our output, hence these columns or fields need to be in our dataset. The dataset must be split into two pieces (input columns and

output columns). within our fresh dataset. Both the input data frame and the output data frame must be split into two test and training data frames. In the train dataset, 80% of the rows should be present, and in the test dataset, 20%. The "train test split" method from the "sklearn" library makes this simple.
First, the raw input is taken in the form of a string and is converted into a vector of integers. This vector is then passed to the NLP pipeline
The raw data is cleaned and preprocessed using the NLP pipeline. The pipeline consists of the following steps:

1. Tokenization: The text is split into words, punctuation marks etc. This is required as many machine learning models can only process numerical data.
2. Stopword removal: Stopwords are words which are filtered out before or after processing of natural language data (text). Although “stop words” usually refers to the most common words in a language, there is no single universal list of stop words used by all natural language processing tools, and indeed not all tools even use such a list. Some tools specifically avoid removing these stop words to support phrase search.
3. Lemmatization: Lemmatization is the process of grouping together the different inflected forms of a word so they can be analysed as a single item. For example, the words “better”, “good”, and “best” are all forms of the word “good”.
4. Stemming: Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root.
5. Vectorization: The text is converted into a vector of integers. This vector is then passed to the NLP pipeline

Data Representation

All words present in the dataset are represented in the form of a vector.
These vectors/words are then given a statistical value which denotes how important it is with respect to the document/dataset itself.
This is done with the help of TF-IDF(Term Frequency - Inverse Document Frequency)

The TF-IDF statistic measures a word's importance to a document in a corpus or group of documents. In information retrieval, text mining, and user modelling searches, it is frequently employed as a weighting factor. To account for the fact that some words are used more frequently than others overall, the tf-idf value rises according to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the term.

Term Frequency(TF)

$$tf(t,d) = count\ of\ t\ in\ d \div number\ of\ words\ in\ d$$

Document Frequency(DF)

$$df(t) = occurence\ of\ t\ in\ documents$$

Inverse Document Frequency(IDF)

$$idf(t) = log(N \div (df + 1))$$

Term Frequency-Inverse Document Frequency(TF-IDF)

$$tf - idf(t,d) = tf(t,d) * log(N \div (df + 1))$$

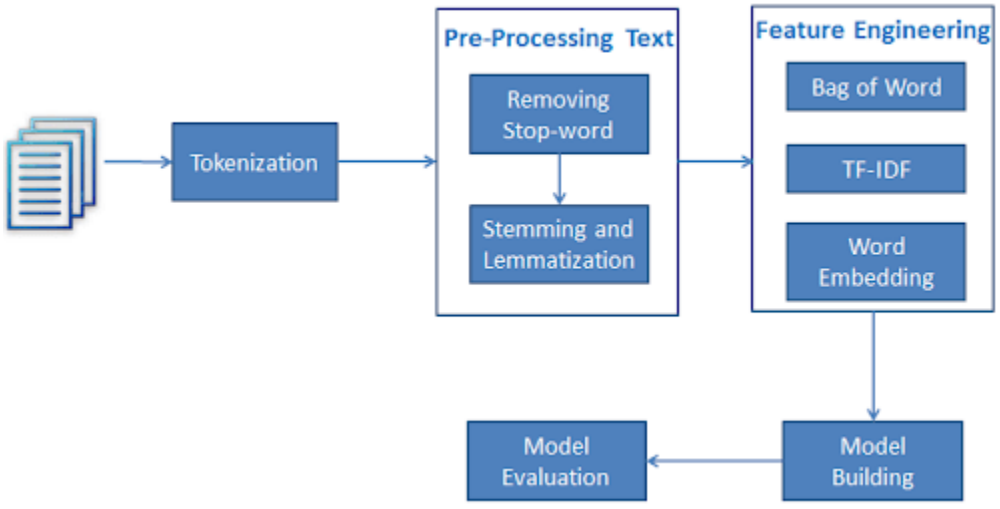

Fig. 3.2. Data Preprocessing and Representation
Source: https://devopedia.org/natural-language-toolkit

Stratified Train-Test Split

When machine learning algorithms are used to make predictions on data that was not used to train the model, their performance is estimated using the train-test split technique.

The process entails splitting the dataset into two subsets. The training dataset is the first subset, which is used to fit the model. The model is not trained using the second subset; rather, it is given the input element of the dataset, and its predictions are then made and contrasted with the expected values. The test dataset is the second dataset in question. The size of the train and test sets serves as the procedure's key configurable parameter. For either the train or test datasets, this is most frequently given as a percentage that ranges from 0 to 1.

For some classification issues, including the one we are working on, there are not an equal number of samples for each class label. In order to maintain the proportions of samples in each class that were seen in the original dataset, it is preferable to divide the dataset into train and test sets in this manner.

This method of splitting in a manner where the dataset is divided into the train and test splits while maintaining the proportions of the entire dataset into each individual split is known as Stratified Train-Test Split.

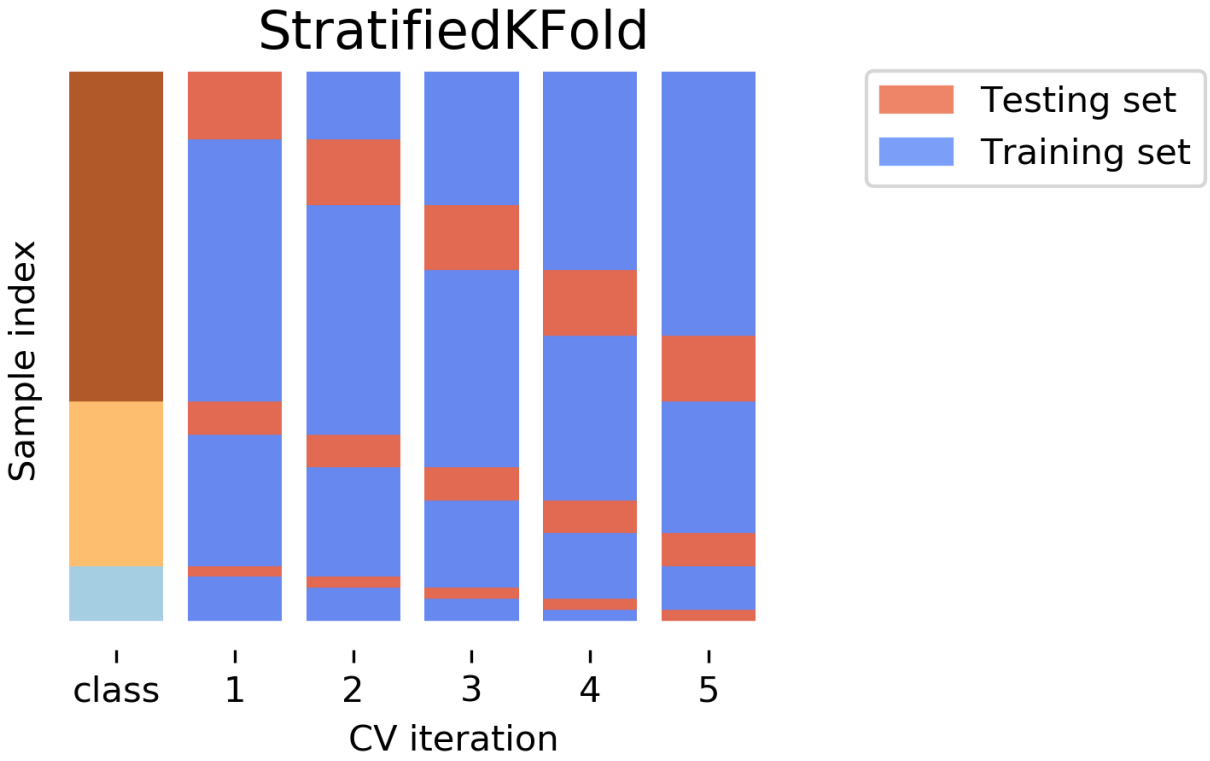

Fig. 3.3. Stratified K Fold

Source: https://amueller.github.io/aml/04-model-evaluation/1-data-splitting-strategies.html

IV. Implementation

1. Decision Tree

Similar to a tree structure, a decision tree creates or builds classification or regression models. It separates a data set into chunks of subsets while simultaneously creating a related decision tree. A tree containing decision and leaf nodes is the end result. A classification or decision is indicated by a decision node that has two or more branches as well as a leaf node. Decision trees may produce complex trees that are difficult to generalise, and they may be unstable due to small alterations in the data that might produce an entirely different tree. Decision trees can handle both numerical and categorical data and are often easy to comprehend and depict. They also need minimum data preparation.

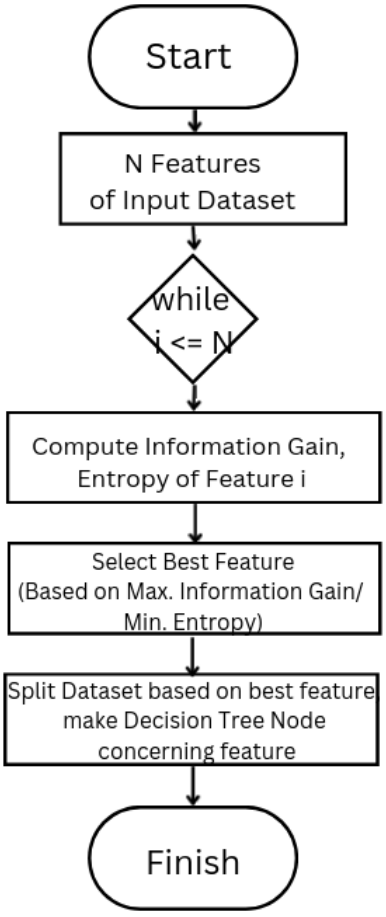

Fig. 4.1. Decision Tree Algorithm

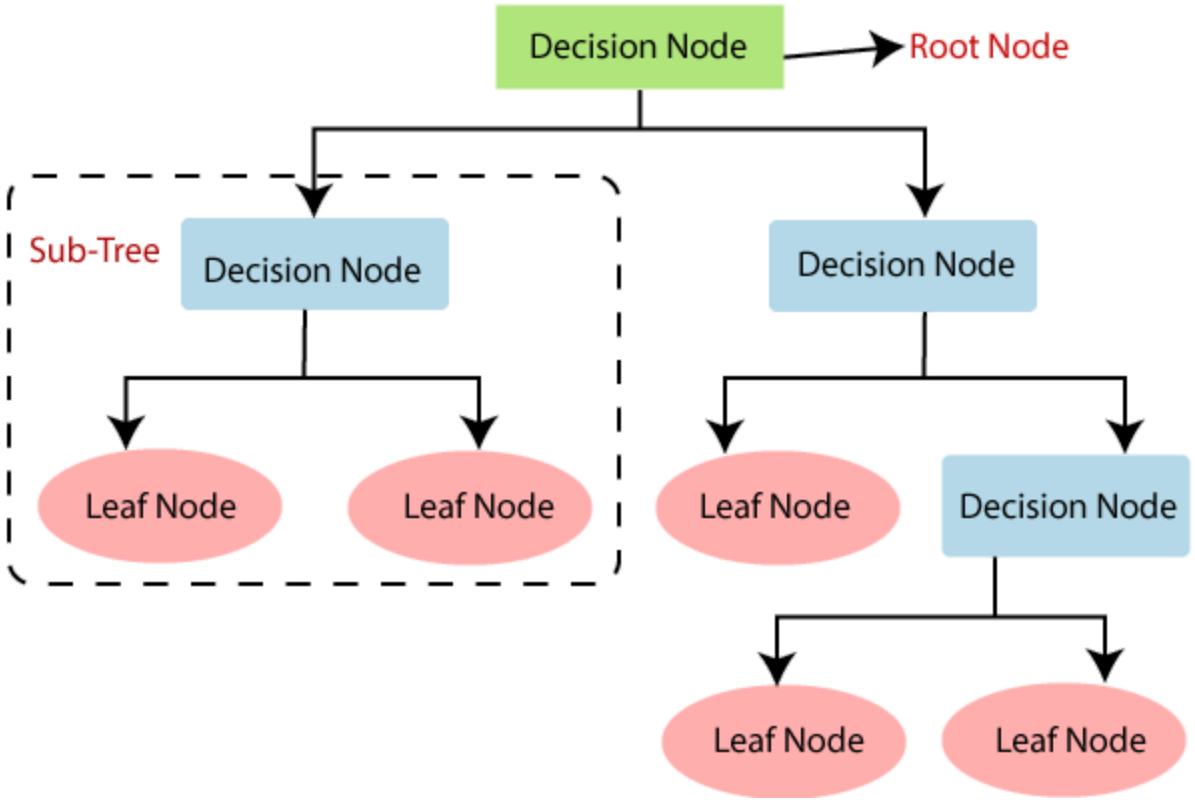

Fig. 4.2. Decision Tree

Source: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

2. Decision Tree with Pruning

By deleting parts of the tree that have minimal ability to classify instances, pruning is the process of lowering the size of decision trees. This size reduction should not result in any reduction. predicting capability. Pruning lowers the final classifier's complexity, which increases predicted accuracy by reducing overfitting. The parameter that regulates the decision tree's post-pruning procedure is confidence factor. The decision tree's performance will be enhanced by determining the ideal value for this parameter.

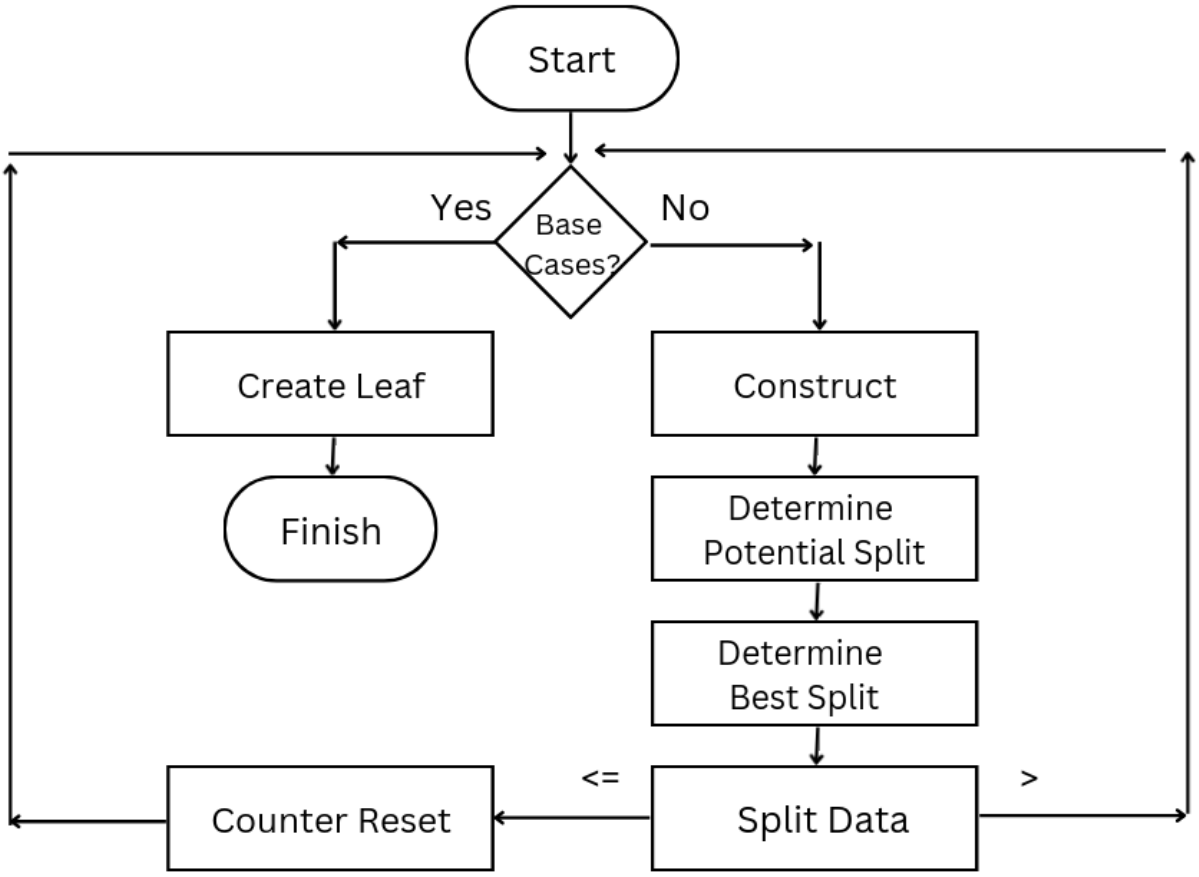

Fig. 4.3. Decision Tree with Pruning Algorithm

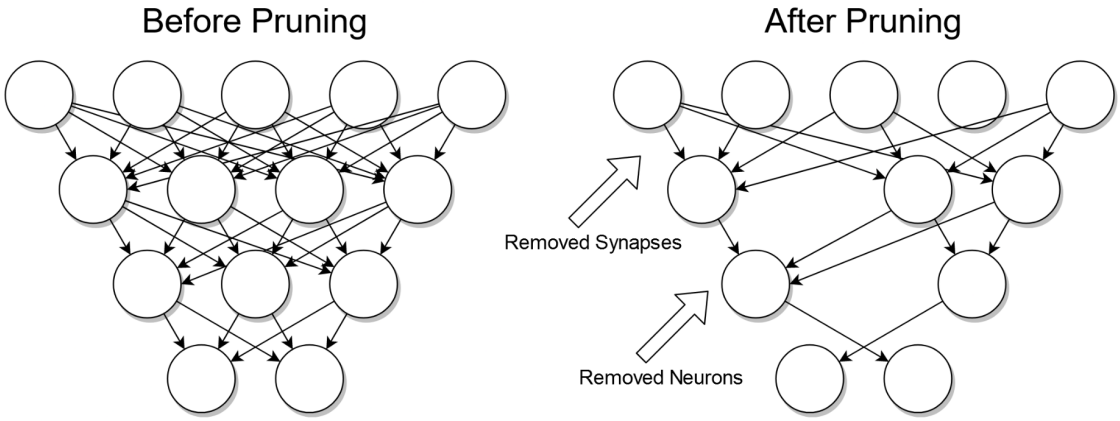

Fig 4.4. Pruning

Source:https://en.wikipedia.org/wiki/Decision_tree_pruning

3. Boosting

Ada-boost combines weak classification models into a single strong classifier. A single model could classify items incorrectly. However, combining many classifiers by choosing a set of samples for each iteration and giving the final vote adequate weight can improve the classification as a whole. Sequentially generated trees serve as weak learners, correcting wrongly predicted samples by giving them more weight after each prediction round. The model is gaining knowledge from prior mistakes. The weighted majority vote, or weighted median in the event of regression issues, makes the final prediction. In essence, the Ada-Boost process is repeated by choosing the training set in accordance with the precision of the prior training. Each classifier's weight is determined by the accuracy of the prior iterations when training it.

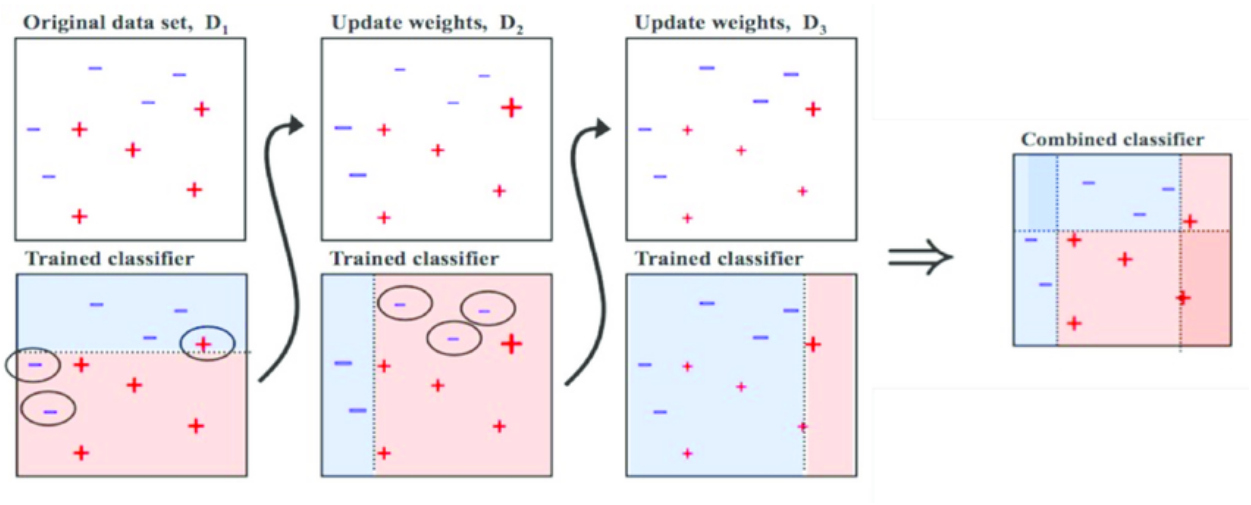

Fig. 4.5. Boosting

Source: https://towardsdatascience.com/understanding-adaboost-for-decision-tree-ff8f07d2851

4. k-Nearest Neighbours

The supervised classification method known as the k-nearest neighbours algorithm employs proximity to determine if two objects are the same. The algorithm utilises a large number of triggered points to teach itself how to trigger new points. When nonparametric and non-statistical techniques are required in real-world applications, KNNs are applied. These methods don't assume anything about how the distribution of the data is carried out. The KNN categorises the coordinates that are identified by a certain characteristic when we are provided primary data. The cost of calculation is considerable since it must calculate the distance between each instance and all of the training samples, but this is the sole drawback.

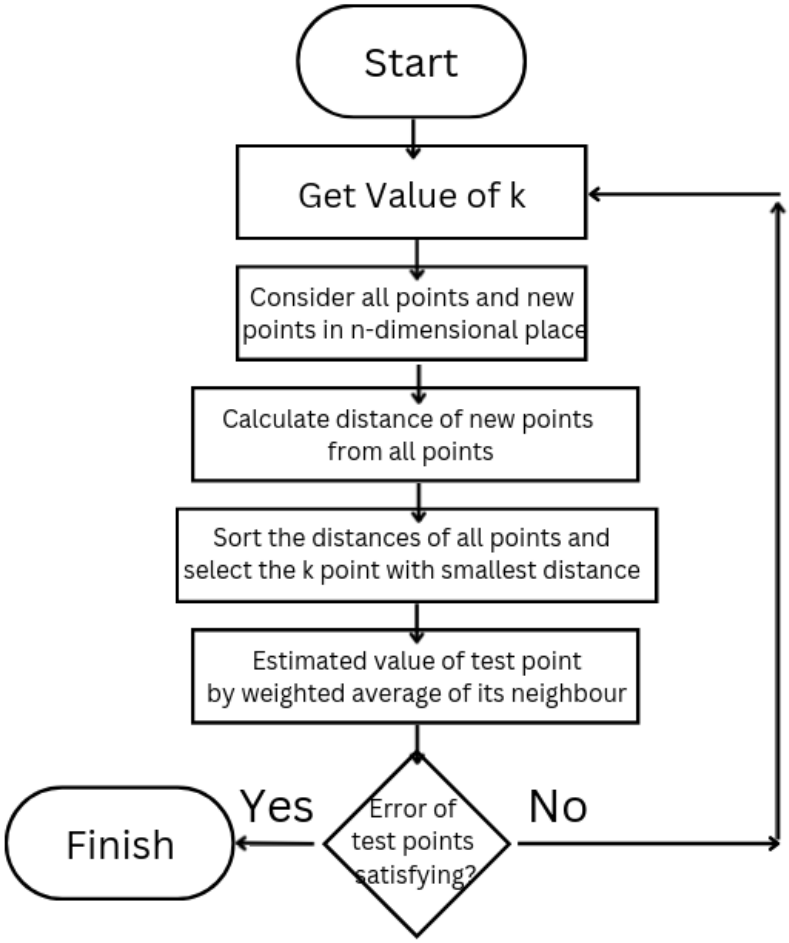

Fig. 4.6. K-Nearest Neighbours Algorithm

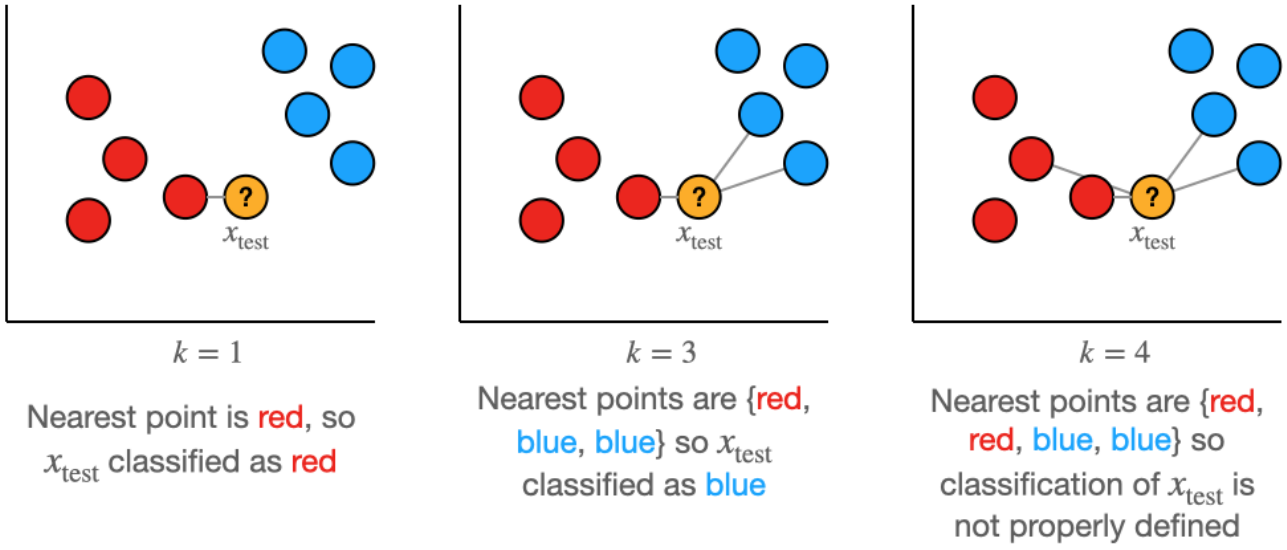

Fig. 4.7. K-Nearest Neighbours

Source:
https://towardsdatascience.com/why-does-increasing-k-decrease-variance-in-knn-9ed6de2f5061

5. Support Vector Machine

A supervised machine learning technique called a support vector machine provides data analysis for regression and classification. SVM is frequently employed in classification. Every feature has a value equal to the value of the given coordinate. The ideal hyperplane that distinguishes between the two classes is then found. Support vector machines express categories as gaps as large as feasible between points in space contrasted with the training data. It employs a small subset of training points in the decision function and is effective and efficient in high dimensional environments; as a result, it is also renowned for its memory efficiency. Probability estimates are given by the method in an indirect manner, and they are computed using five-fold cross-validation.

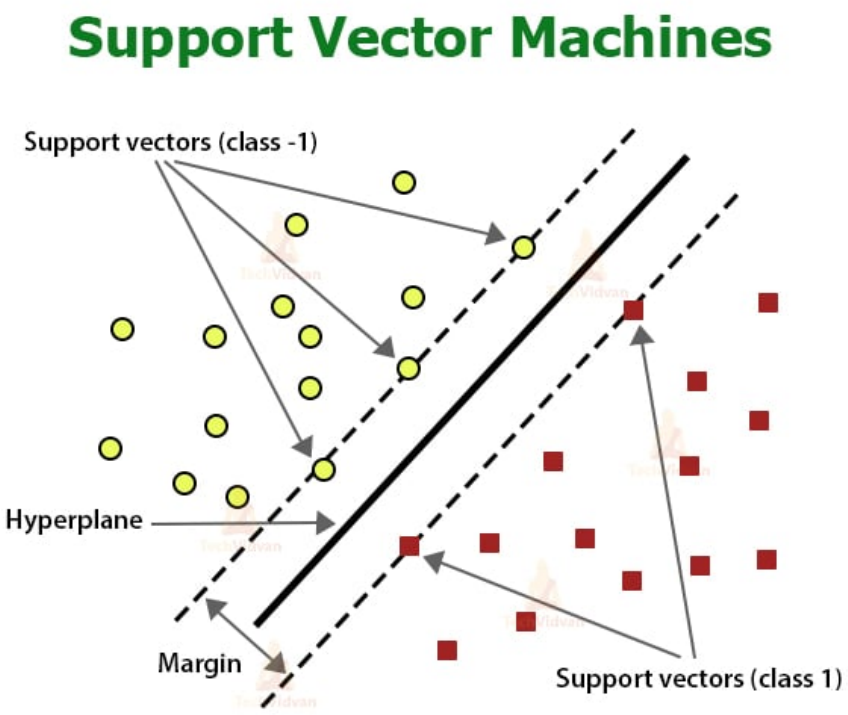

Fig. 4.8. Support Vector Machines

Source: https://techvidvan.com/tutorials/svm-in-r/

Learning appropriate values for each weight and bias from the input dataset constitutes the process of training a model. Empirical risk minimization is the technique through which a machine learning algorithm builds a model by validating several samples and looking for a model that minimises loss. The consequence of a poor prediction is termed as loss. The loss is a numerical measure of how poorly the model predicted a particular case. If the model's forecast is accurate, there will be no loss; if not, there will be a loss. Finding weights and biases that have minimal loss across all of the instances in a dataset is the main goal of training a model. An algorithm uses training data to learn throughout the process of training a machine learning model. The training procedure results in a model antiquity, which is referred to as an ML model. The right response, or goal, must be present in the training data. An ML model that has been acclimated to these patterns is produced by the learning algorithm, which finds patterns in the training data that transfer the qualities of the input data to the target. In order to

train all the machine learning models for this issue, we utilised the function "fit," passing the train input and train output datasets as input and output parameters.

V. Results and Discussion

Dataset

A fraud email dataset from Kaggle is used, the emails are present in a csv format, which affords portability and easier model training. Deployment is also made easier due to the smaller dataset size.

Metrics of Evaluation

AUC (Area Under The Curve) ROC (Receiver Operating Characteristics) is utilised as the evaluation metric of choice. It is also commonly used as the evaluation metric for 2-class classification problems, particularly in fraud identification algorithms such as our model. AUC represents the degree of separability between classes (in this case, phishing and non-phishing). A higher AUC denotes a better model, with an AUC of 1 denoting perfect separability, i.e., no overlap in classes. An AUC of 0 means that the model predicts phishing emails as non-phishing emails and vice-versa, while a score of 0.5 denotes complete non-separability, i.e., model accuracy would be the same as a randomly selected prediction.

A plot of True Positive Rate (TPR) is made against False Positive Rate (FPR) with the two respectively denoting the y-axis and the x-axis. The distribution curves of the two classes, phishing and non-phishing are then made.

$$TPR/Recall/Sensitivity = TP \div (TP + FN)$$
$$FPR = 1 - Specificity$$
$$= FP \div (TN + FP)$$

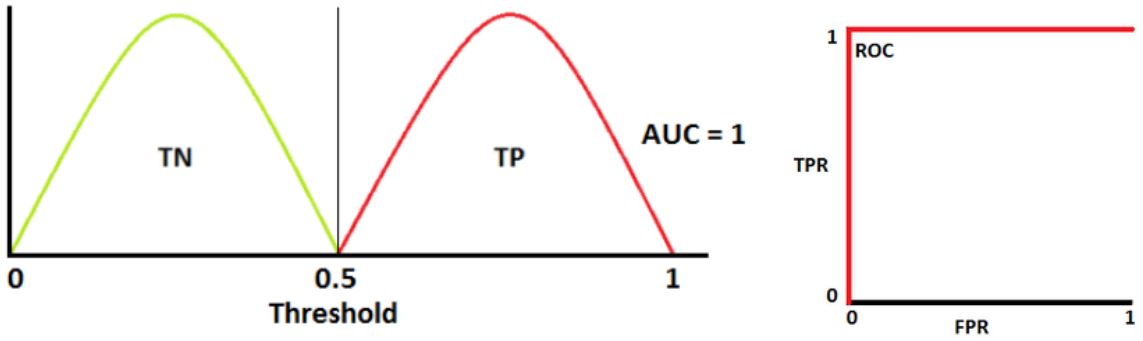

Fig. 5.1. Visualisation of an AUC score of 1. Note that the red distribution curve is that of the phishing class, while the green is that of non-phishing

Source: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

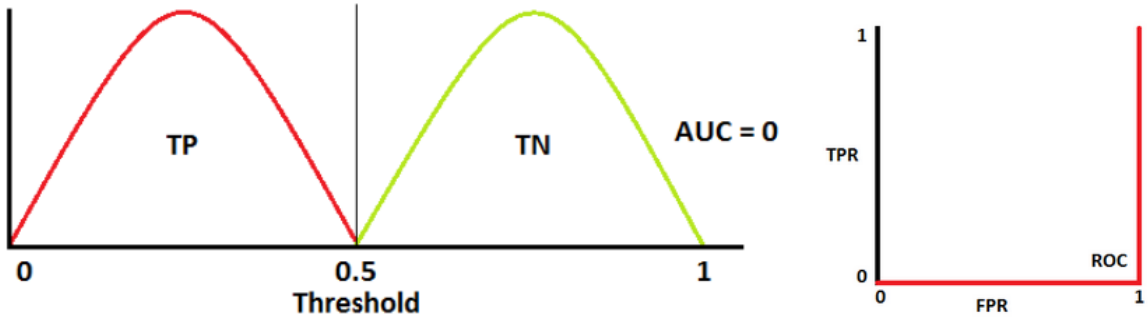

Fig. 5.2. Visualisation of an AUC score of 0, False positive rate is also 0

Source: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

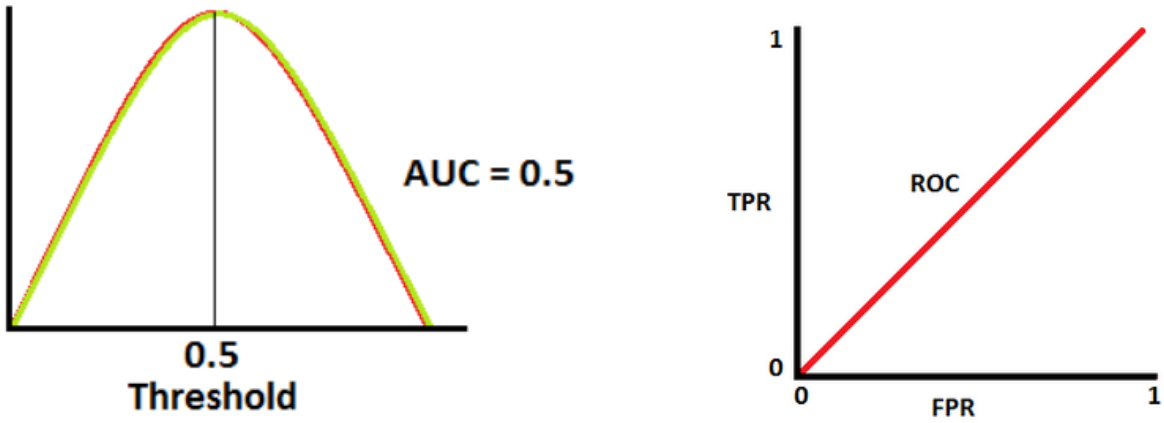

Fig. 5.3. Visualisation of an AUC score of 0, TPR and FPR demonstrate a linear correlation.

Source: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

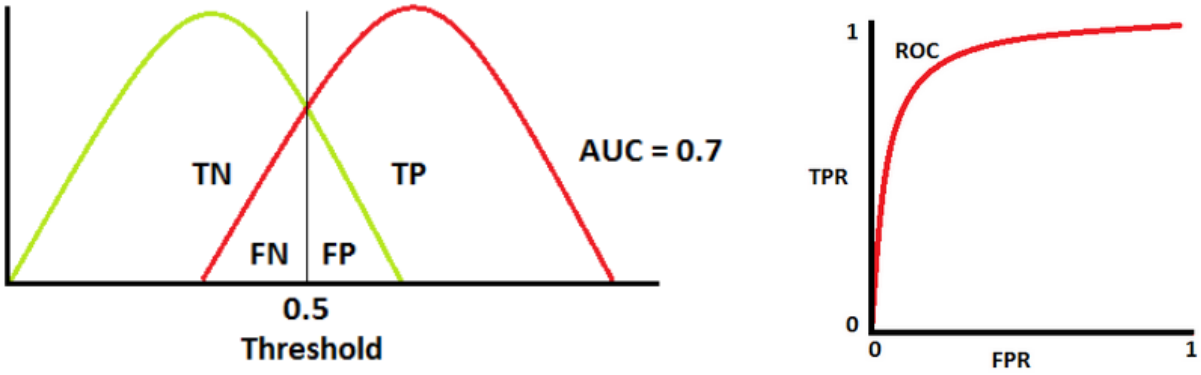

Fig. 5.4. Visualisation of an AUC score of 0.7. Most models demonstrate a similar ROC curve, with varying accuracy.
Source: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

	Text	Class
0	supply quality chinas exclusive dimensions unb...	1
1	sidlet know thx	0
2	dear friendgreetings youi wish accost request ...	1
3	mr cheung puihang seng bank ltddes voeux rd br...	1
4	surprising assessment embassy	0

Table 5.1. Visualisation of the dataset. 5 randomly extracted examples are displayed along with their class ID. Class 1 contains phishing emails, while Class 0 contains non-phishing emails

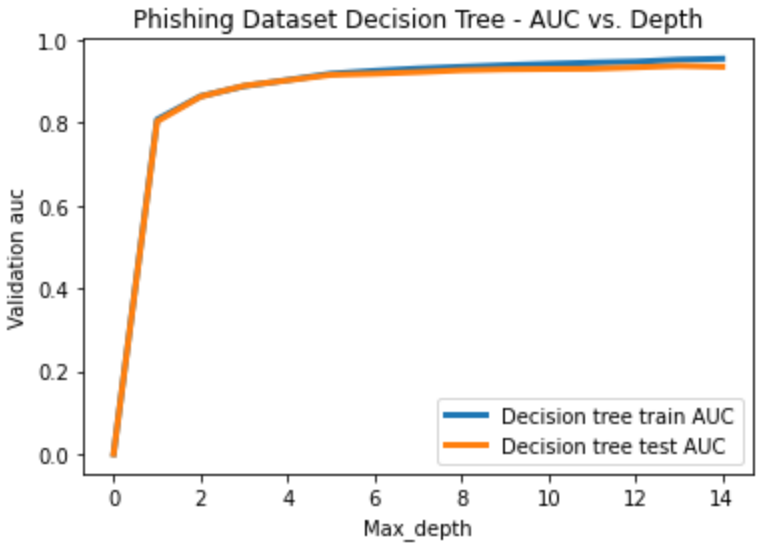

Fig. 5.5. Variation of decision tree model accuracy with increase in model depth. Optimum depth of the decision tree is found to be 14. However, decision tree underfits the data, as can be seen in the graph; the training accuracy remains high, but is noticeably higher than the testing accuracy.

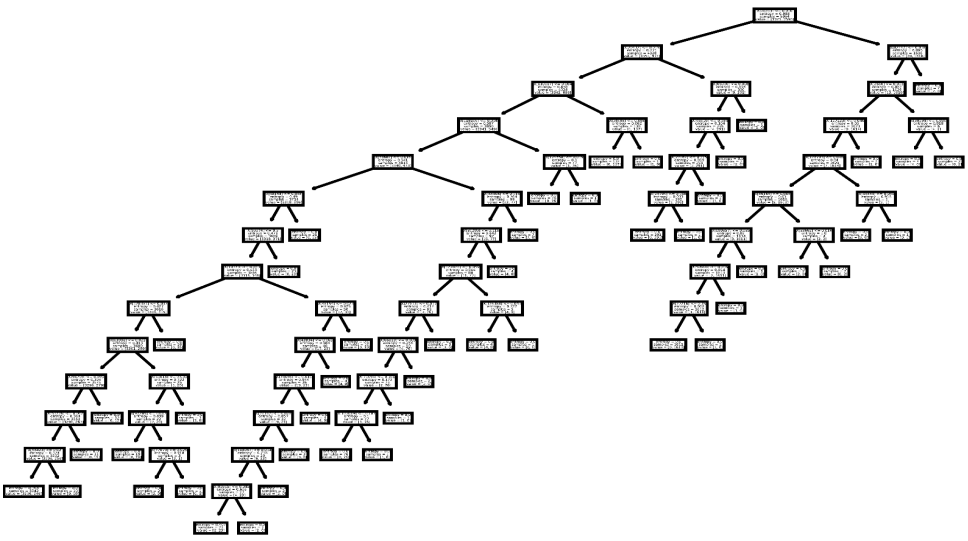

Fig 5.6. Resulting Decision Tree with a model depth of 14. The maximum depth specified was 15. The decision trees are iteratively trained upto the maximum depth, and the decision tree model with highest accuracy is selected.

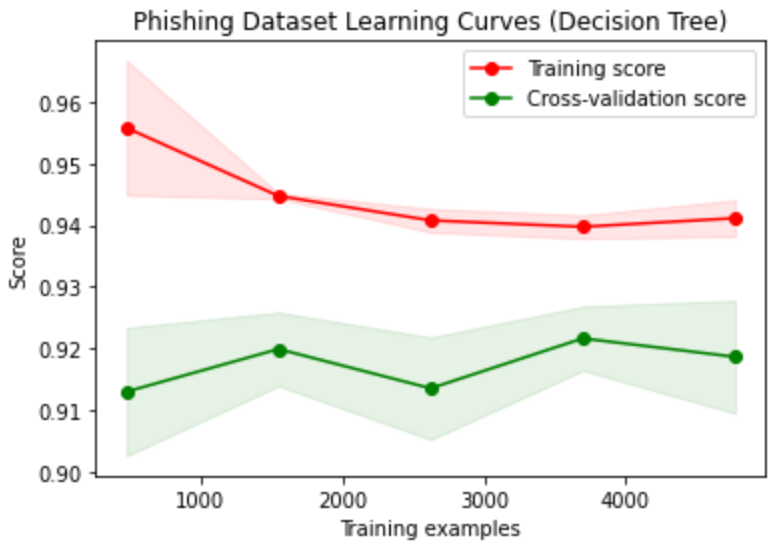

Fig. 5.7. Variation of training and testing accuracy with increase in number of training examples for pruned decision tree. Training accuracy decreases with an increase in number of training samples, demonstrating that the pruned decision tree is insufficient to properly represent the training data; i.e., underfitting occurs. Testing accuracy increases only marginally, and is similarly lower than training accuracy.

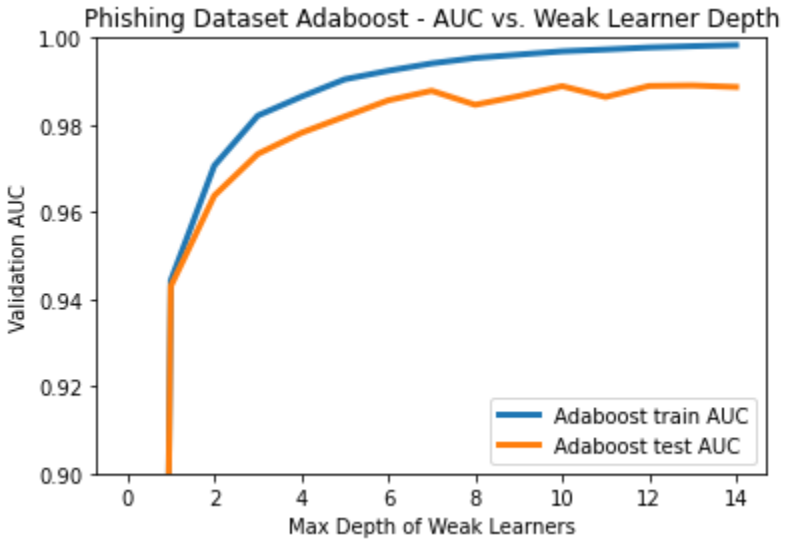

Fig. 5.8. Variation of boosting (Adaboost) model accuracy with increase in weak learner (Decision Tree) model depth. The visible difference between training and testing accuracies demonstrates the underfitting of the model on the training data, even with an increase in model depth.

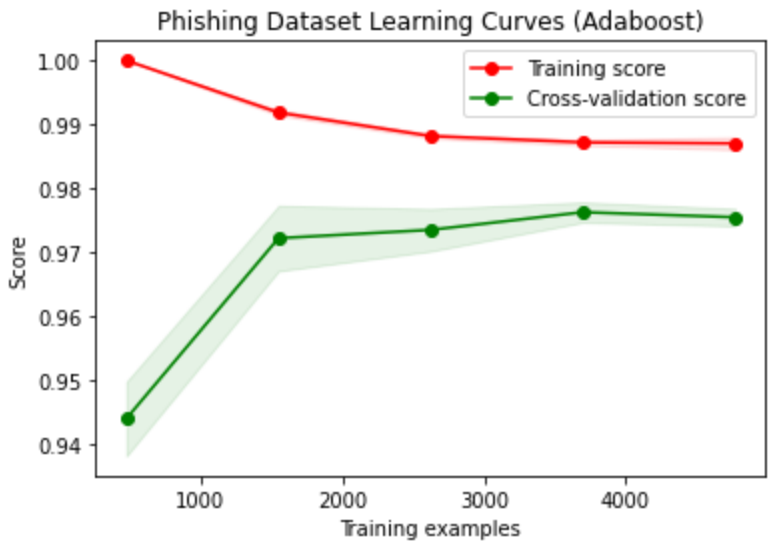

Fig. 5.9. Variation of training and testing accuracy with increase in number of training examples for Adaboost model (with Decision Tree as weak learner). A weak learner, the decision tree underfits the data even with aid of Adaboost.

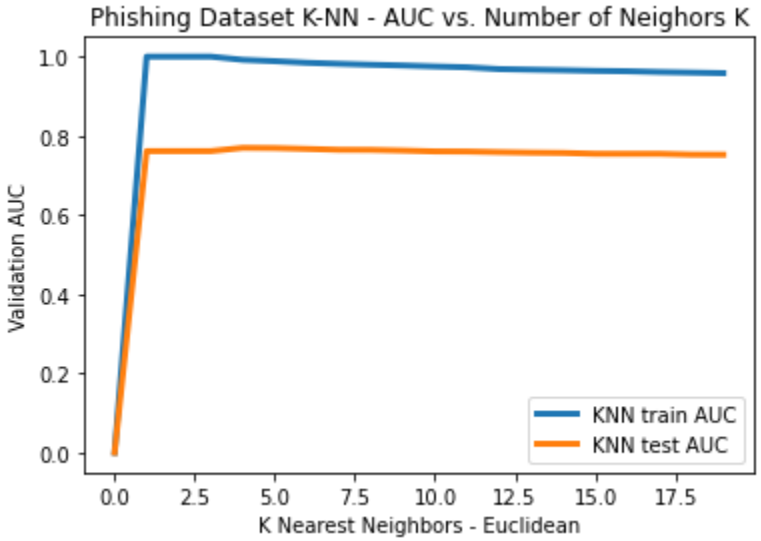

Fig. 5.10. Variation of k-Nearest Neighbours model accuracy with increase in k (number of neighbours used for training). Severe overfitting occurs, with a significant drop in accuracy between training and testing phases. Thus, the k-NN model is also not able to represent the data sufficiently well.

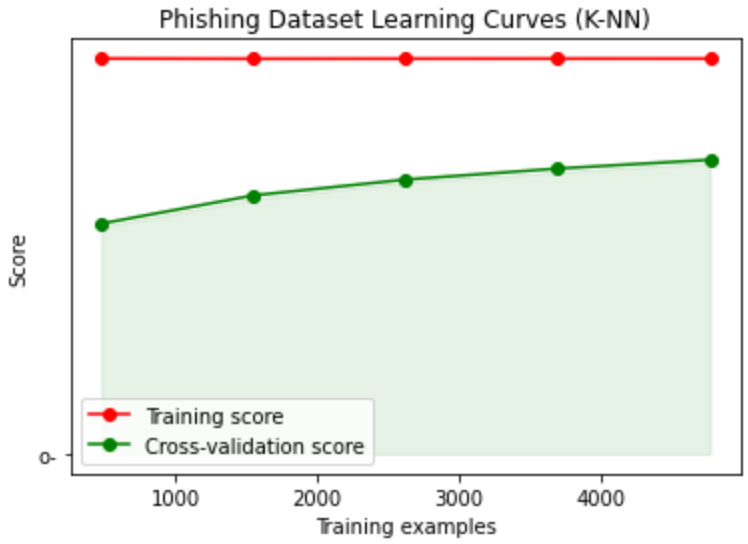

Fig. 5.11. Variation of training and testing accuracy with increase in number of training examples for k-Nearest Neighbours model. Overfitting causes the model to learn the training data very well, however, it causes a performance drop on unseen test data.

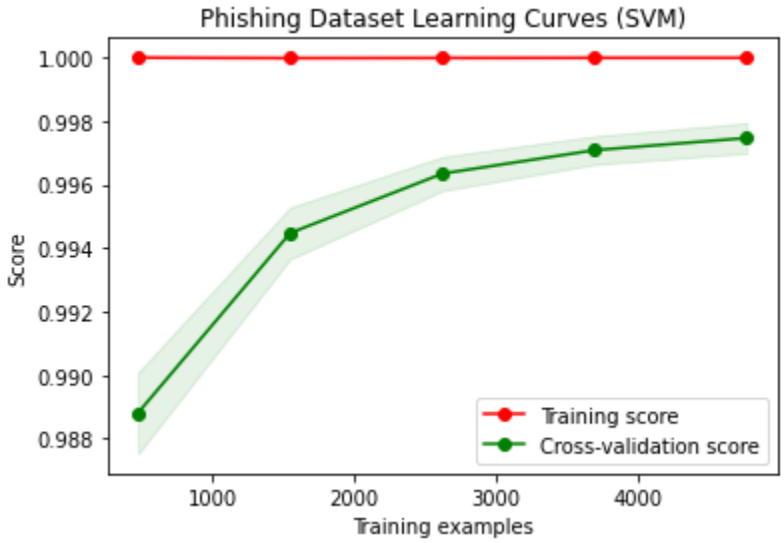

Fig. 5.12. Variation of training and testing accuracy with increase in number of training examples for Support Vector Machine: a deep learning architecture. The deep learning nature of the model enables a much better feature representation and the model provides a good fit for the data.

Model	Training Accuracy	Testing Accuracy
Decision Tree	0.954856	0.937453
Pruned Decision Tree	0.958897	0.939421
Boosting with Decision Tree Weak Learner	0.998333	0.989066
k-Nearest Neighbours	0.999807	0.769443
Support Vector Machine (linear kernel)	0.999995	0.997513
Support Vector Machine (poly kernel)	0.999997	0.99792

Table 5.2. Comparison of the accuracies of the trained models. It is evident that pruning the decision tree increases accuracy by curbing overfitting. The k-Nearest Neighbours model suffers from gross overfitting, however, showing a much reduced accuracy over (previously unseen) test data. The Support Vector Machine provides a good fit for the data, and additionally, is a light network. We find that the linear kernel SVM is sufficient to represent the data, as the poly kernel increases model complexity without a significant increase in accuracy.

VI. Conclusion

We can conclude from the above results and plots that Support Vector Machine performs the best and the algorithm with lowest metrics on testing is k-NN, as it is the least effective algorithm for solving complex classification problems. Pruning does not significantly increase the performance of the Decision Tree, however, it greatly reduces model complexity. Boosting performs better than standard Decision Trees, but remains inferior to pruning and this complicates the model without adding significant benefit.

References

[1] Ozcan, A., Catal, C., Donmez, E., & Senturk, B. (2021). A hybrid DNN–LSTM model for detecting phishing URLs. In Neural Computing and Applications. Springer Science and Business Media LLC. https://doi.org/10.1007/s00521-021-06401-z

[2] Salloum, S., Gaber, T., Vadera, S., & Shaalan, K. (2022). A Systematic Literature Review on Phishing Email Detection Using Natural Language Processing Techniques. In IEEE Access (Vol. 10, pp. 65703–65727). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/access.2022.3183083

[3] Do, N. Q., Selamat, A., Krejcar, O., Herrera-Viedma, E., & Fujita, H. (2022). Deep Learning for Phishing Detection: Taxonomy, Current Challenges and Future Directions. In IEEE Access (Vol. 10, pp. 36429–36463). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/access.2022.3151903

[4] AbdulNabi, I., & Yaseen, Q. (2021). Spam Email Detection Using Deep Learning Techniques. In Procedia Computer Science (Vol. 184, pp. 853–858). Elsevier BV. https://doi.org/10.1016/j.procs.2021.03.107

[5] Salloum, S., Gaber, T., Vadera, S., & Shaalan, K. (2021). Phishing Email Detection Using Natural Language Processing Techniques: A Literature Survey. In Procedia Computer Science (Vol. 189, pp. 19–28). Elsevier BV. https://doi.org/10.1016/j.procs.2021.05.077

[6] et al., A. (2020). Detecting phishing attacks using a combined model of LSTM and CNN. In International Journal of ADVANCED AND APPLIED SCIENCES (Vol. 7, Issue 7, pp. 56–67). International Journal of Advanced and Applied Sciences. https://doi.org/10.21833/ijaas.2020.07.007

[7] Grant Ho, Aashish Sharma, Mobin Javed, Vern Paxson, and David Wagner. 2017. Detecting credential spearphishing attacks in enterprise settings. In Proceedings of the 26th USENIX Conference on Security Symposium (SEC'17). USENIX Association, USA, 469–485.

[8] Hakim, Z. M., Ebner, N. C., Oliveira, D. S., Getz, S. J., Levin, B. E., Lin, T., Lloyd, K., Lai, V. T., Grilli, M. D., & Wilson, R. C. (2020). The Phishing Email Suspicion Test (PEST) a lab-based task for evaluating the cognitive mechanisms of phishing detection. In Behavior Research Methods (Vol. 53, Issue 3, pp. 1342–1352). Springer Science and Business Media LLC. https://doi.org/10.3758/s13428-020-01495-0

[9] Fitzpatrick, B., Liang, X. "Sherwin," & Straub, J. (2021). Fake News and Phishing Detection Using a Machine Learning Trained Expert System (Version 1). arXiv. https://doi.org/10.48550/ARXIV.2108.08264

[10] Sun, Y., Chong, N., & Ochiai, H. (2021). Federated Phish Bowl: LSTM-Based Decentralized Phishing Email Detection (Version 2). arXiv. https://doi.org/10.48550/ARXIV.2110.06025

[11] Hamza, A., & Moetque, H. (2017). Feature Weight Optimization Mechanism for Email Spam Detection based on Two-step Clustering Algorithm and Logistic Regression Method. In International Journal of Advanced Computer Science and Applications (Vol. 8, Issue 10). The Science and Information Organization. https://doi.org/10.14569/ijacsa.2017.081054

[12] Kim Soon, G., Kim On, C., Mohd Rusli, N., Soo Fun, T., Alfred, R., & Tse Guan, T. (2020). Comparison of simple feedforward neural network, recurrent neural network and ensemble neural networks in phishing detection. In Journal of Physics: Conference Series (Vol. 1502, Issue 1, p. 012033). IOP Publishing. https://doi.org/10.1088/1742-6596/1502/1/012033
[13] Bergholz, A., Chang, J.H., Paass, G., Reichartz, F., & Strobel, S. (2008). Improved Phishing Detection using Model-Based Features. CEAS.
[14] Kaytan, M. & Hanbay, D. (2017). Effective Classification of Phishing Web Pages Based on New Rules by Using Extreme Learning Machines . Computer Science , 2 (1) , 15-36 . Retrieved from https://dergipark.org.tr/en/pub/bbd/issue/30846/333818
[15] Rao, R. S., & Ali, S. T. (2015). A Computer Vision Technique to Detect Phishing Attacks. In 2015 Fifth International Conference on Communication Systems and Network Technologies. 2015 Fifth International Conference on Communication Systems and Network Technologies (CSNT). IEEE. https://doi.org/10.1109/csnt.2015.68
[16] Feng, F., Zhou, Q., Shen, Z., Yang, X., Han, L., & Wang, J. (2018). The application of a novel neural network in the detection of phishing websites. In Journal of Ambient Intelligence and Humanized Computing. Springer Science and Business Media LLC. https://doi.org/10.1007/s12652-018-0786-3
[17] Li, Q., Cheng, M., Wang, J., & Sun, B. (2022). LSTM Based Phishing Detection for Big Email Data. In IEEE Transactions on Big Data (Vol. 8, Issue 1, pp. 278–288). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/tbdata.2020.2978915
[18] Das, A., Baki, S., Aassal, A. E., Verma, R., & Dunbar, A. (2019). SOK: A Comprehensive Reexamination of Phishing Research from the Security Perspective (Version 1). arXiv. https://doi.org/10.48550/ARXIV.1911.00953
[19] Basit, A., Zafar, M., Liu, X., Javed, A. R., Jalil, Z., & Kifayat, K. (2020). A comprehensive survey of AI-enabled phishing attacks detection techniques. In Telecommunication Systems (Vol. 76, Issue 1, pp. 139–154). Springer Science and Business Media LLC. https://doi.org/10.1007/s11235-020-00733-2
[20] Zuraiq, A. A., & Alkasassbeh, M. (2019). Review: Phishing Detection Approaches. In 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS). 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS). IEEE. https://doi.org/10.1109/ictcs.2019.8923069
[21] Karim, A., Azam, S., Shanmugam, B., Kannoorpatti, K., & Alazab, M. (2019). A Comprehensive Survey for Intelligent Spam Email Detection. In IEEE Access (Vol. 7, pp. 168261–168295). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/access.2019.2954791
[22] Rao, R. S., & Pais, A. R. (2018). Detection of phishing websites using an efficient feature-based machine learning framework. In Neural Computing and Applications (Vol. 31, Issue 8, pp. 3851–3873). Springer Science and Business Media LLC. https://doi.org/10.1007/s00521-017-3305-0

[23] Jain, A. K., & Gupta, B. B. (2017). Towards detection of phishing websites on client-side using machine learning based approach. In Telecommunication Systems (Vol. 68, Issue 4, pp. 687–700). Springer Science and Business Media LLC. https://doi.org/10.1007/s11235-017-0414-0
[25] Fang, Y., Zhang, C., Huang, C., Liu, L., & Yang, Y. (2019). Phishing Email Detection Using Improved RCNN Model With Multilevel Vectors and Attention Mechanism. In IEEE Access (Vol. 7, pp. 56329–56340). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/access.2019.2913705
[26] Harikrishnan, N. B., Vinayakumar, R., & Soman, K. P. (2018, March). A machine learning approach towards phishing email detection. In Proceedings of the Anti-Phishing Pilot at ACM International Workshop on Security and Privacy Analytics (IWSPA AP) (Vol. 2013, pp. 455-468).
[27] Bountakas, P., Koutroumpouchos, K., & Xenakis, C. (2021, August). A Comparison of Natural Language Processing and Machine Learning Methods for Phishing Email Detection. In The 16th International Conference on Availability, Reliability and Security (pp. 1-12).
[28] Smadi, S., Aslam, N., & Zhang, L. (2018). Detection of online phishing email using dynamic evolving neural network based on reinforcement learning. Decision Support Systems, 107, 88-102.
[29] Qachfar, F. Z., Verma, R. M., & Mukherjee, A. (2022, April). Leveraging Synthetic Data and PU Learning For Phishing Email Detection. In Proceedings of the Twelveth ACM Conference on Data and Application Security and Privacy (pp. 29-40).
[30] Gangavarapu, T., Jaidhar, C. D., & Chanduka, B. (2020). Applicability of machine learning in spam and phishing email filtering: review and approaches. Artificial Intelligence Review, 53(7), 5019-5081.

Appendices

Source Code

```
nltk.download('stopwords')

from IPython.display import Image

##Load Dataset

data_df = pd.read_csv('fraud_email_.csv')

data_df = data_df.dropna()

data_df['Text']    =    data_df['Text'].apply(lambda    x:    "
".join(x.lower() for x in x.split()))
data_df['Text'] = data_df['Text'].str.replace('[^\w\s]','')

stop = stopwords.words('english')
data_df['Text'] = data_df['Text'].apply(lambda x: " ".join(x for
x in x.split() if x not in stop))


from sklearn.feature_extraction.text import TfidfVectorizer
corpus = data_d f['Text']
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(corpus)


print(X.shape)

data_df = data_df.dropna()

stopset = set(stopwords.words("english"))
vectorizer  =  TfidfVectorizer(stop_words  =  stopset,  norm='l2',
decode_error='ignore',binary=True)
features = vectorizer.fit_transform(data_df["Text"])
```

```
print(features.shape[:])
y = data_df["Class"]

train_X,   test_X,   train_y,   test_y   =   train_test_split(X,
data_df["Class"], test_size=0.5, stratify=data_df["Class"])

##EDA

data_df.head()

data_df.info()

data_df.describe()

##Decision Tree

max_depth = 15

tree_auc_train,     tree_auc_test     =     np.zeros(max_depth),
np.zeros(max_depth)

for i in range(1,max_depth):
     clf_decision_tree                                          =
tree.DecisionTreeClassifier(max_depth=i,
criterion='entropy',random_state=1)
     clf_decision_tree = clf_decision_tree.fit(train_X, train_y)

     tree_auc_train[i]          =          roc_auc_score(train_y,
clf_decision_tree.predict_proba(train_X)[:,1])
     tree_auc_test[i]           =           roc_auc_score(test_y,
clf_decision_tree.predict_proba(test_X)[:,1])

print("Best       tree       depth       training:       "       +
str(np.argmax(tree_auc_train, axis=0)))
print("Highest      AUC      score      training:      "      +
str(np.max(tree_auc_train, axis=0)))
```

```
print("Best tree depth testing: " + str(np.argmax(tree_auc_test,
axis=0)))
print("Highest AUC score testing: " +  str(np.max(tree_auc_test,
axis=0)))

##Decision Tree Pruning

from sklearn.tree._tree import TREE_LEAF

def is_leaf(inner_tree, index):

     return (inner_tree.children_left[index] == TREE_LEAF and
          inner_tree.children_right[index] == TREE_LEAF)

def prune_index(inner_tree, decisions, index=0):
     # Start pruning from the bottom
     if                  not                 is_leaf(inner_tree,
inner_tree.children_left[index]):
     prune_index(inner_tree,                          decisions,
inner_tree.children_left[index])
     if                  not                 is_leaf(inner_tree,
inner_tree.children_right[index]):
     prune_index(inner_tree,                          decisions,
inner_tree.children_right[index])

     # Prune children if both are leaves
     if   (is_leaf(inner_tree,   inner_tree.children_left[index])
and
     is_leaf(inner_tree, inner_tree.children_right[index]) and
     (decisions[index]                                          ==
decisions[inner_tree.children_left[index]]) and
     (decisions[index]                                          ==
decisions[inner_tree.children_right[index]])):
     # turn node into a leaf
     inner_tree.children_left[index] = TREE_LEAF
     inner_tree.children_right[index] = TREE_LEAF
     print("Pruned {}".format(index))
```

```
def prune_duplicate_leaves(mdl):

     decisions                                                    =
mdl.tree_.value.argmax(axis=2).flatten().tolist() # Decision for
each node
     prune_index(mdl.tree_, decisions)

clf_decision_tree   =   tree.DecisionTreeClassifier(max_depth=i,
criterion='entropy',random_state=1)
clf_decision_tree = clf_decision_tree.fit(train_X, train_y)
prune_duplicate_leaves(clf_decision_tree)

tree_auc_train_pruned          =          roc_auc_score(train_y,
clf_decision_tree.predict_proba(train_X)[:,1])
tree_auc_test_pruned           =           roc_auc_score(test_y,
clf_decision_tree.predict_proba(test_X)[:,1])

print("pruned     decision     tree     training:     "     +
str(tree_auc_train_pruned))
print("pruned     decision     tree     testing:     "     +
str(tree_auc_test_pruned))
```