



Phishing Detection with a Machine Learning Net

Ana Luís Carvalho Matos Bezerra

Mestrado em Estatística Computacional e Análise de Dados

Departamento de Matemática

2022

Orientador

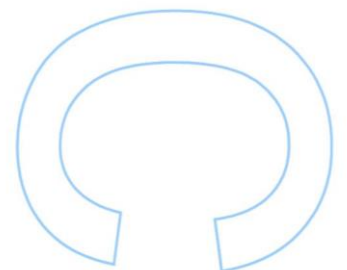
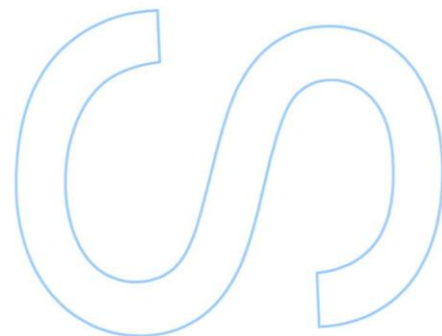
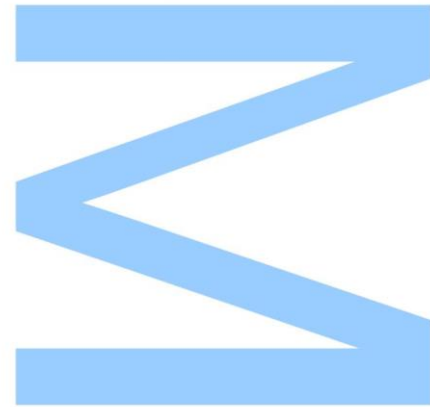
Ricardo Cruz, Investigador, Faculdade de Engenharia

Coorientador

Joaquim Costa, Professor Auxiliar, Faculdade de Ciências

Supervisor

Ivo Pereira, Professor Assistente, E-goí

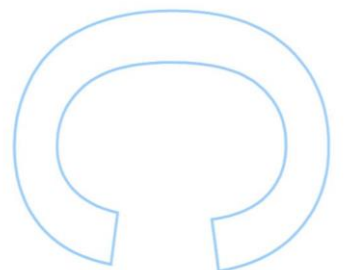
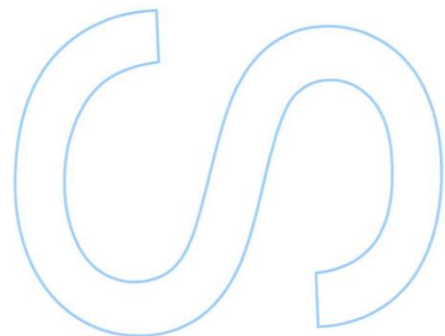
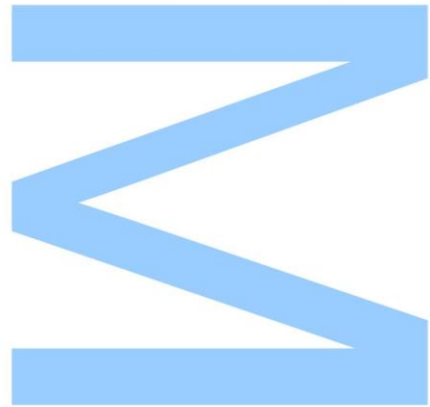




Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, ____ / ____ / ____



UNIVERSIDADE DO PORTO

MASTERS THESIS

Phishing Detection with a Machine Learning
Net

Author:

Ana BEZERRA

Supervisor:

Ricardo CRUZ

E-Goi supervisor:

Ivo PEREIRA

Co-supervisor:

Joaquim COSTA

*A thesis submitted in fulfilment of the requirements
for the degree of MSc. Computational Statistics and Data Analysis*

at the

Faculdade de Ciências da Universidade do Porto
Departamento de Matemática

December 23, 2022

Sworn Statement

I, Ana Luís Carvalho Matos Bezerra, enrolled in the Master Degree in Computational Statistics and Data Analysis at the Faculty of Sciences of the University of Porto hereby declare, in accordance with the provisions of paragraph a) of Article 14 of the Code of Ethical Conduct of the University of Porto, that the content of this dissertation reflects perspectives, research work and my own interpretations at the time of its submission.

By submitting this dissertation, I also declare that it contains the results of my own research work and contributions that have not been previously submitted to this or any other institution.

I further declare that all references to other authors fully comply with the rules of attribution and are referenced in the text by citation and identified in the bibliographic references section. This dissertation does not include any content whose reproduction is protected by copyright laws.

I am aware that the practice of plagiarism and self-plagiarism constitute a form of academic offense.

Ana Luís Carvalho Matos Bezerra

September 30th, 2022

Agradecimentos

Em primeiro lugar queria agradecer à minha família, particularmente aos meus pais, Regina e Luís, que sempre apoiaram as minhas decisões independentemente do quão arriscadas são e me tentam ajudar no que for preciso. Aos meus amigos que me incentivaram a arriscar no que verdadeiramente quero fazer. Agradeço também ao meu namorado, Edgar Santos, por tudo o que fez por mim durante esta fase. Obrigada por me trazeres “de volta à Terra” quando senti que tudo estava a correr mal, obrigada por debateres comigo as minhas ideias quando eu precisava de as assentar e obrigada por celebrares comigo todas as pequenas vitórias.

Aos professores doutores e orientadores Ricardo Cruz e Joaquim Costa, obrigada por me incentivaram a realizar as minhas ideias, por me deixarem explorar e por todos os conselhos e correções dados durante este processo. Ao Ivo, o meu orientador dentro da empresa, que sempre ajudou no que foi necessário também com os seus conselhos e me deu motivação para alcançar mais do que tinha em mente.

A todos do departamento de AI da E-goi quero agradecer o apoio e a colaboração durante o meu projeto, todas as dicas, todas as conversas e brincadeiras que me ajudaram a desenvolver as minhas capacidades técnicas mas acima de tudo fizeram com que este capítulo tenha sido bastante especial e positivo. Um apreço especial ao Daniel que sempre fez questão de me incluir em todas as atividades do departamento e me fez sentir verdadeiramente como membro da equipa. Às “Goi-Ninas”, Inês, Leonor e Carolina, obrigada por toda a boa disposição e todas as conversas de motivação. Também não posso deixar de agradecer ao Miguel Rebelo, que para além de ser um colega da equipa onde entreguei, tenho também o prazer de chamar de amigo. Obrigada por todo o esforço que me foi dedicado, por todo os conselhos e incentivo dado durante o projeto que foram sem dúvida fundamentais para que este tenha corrido bem.

UNIVERSIDADE DO PORTO

Abstract

Faculdade de Ciências da Universidade do Porto
Departamento de Matemática

MSc. Computational Statistics and Data Analysis

Phishing Detection with a Machine Learning Net

by [Ana BEZERRA](#)

Phishing attacks are becoming a common practice on the web that aims to steal sensitive information. E-mail *phishing* is one of the most common types of attacks on the web and can have a big impact on individuals and enterprises. There is still a gap in prevention when it comes to detecting *phishing* emails as new attacks are usually not detected. The goal of this master thesis was to develop a model capable of identifying *phishing* emails based on machine learning approaches. The work was performed in collaboration with E-goi, a multichannel marketing automation company. The data consisted of emails collected from the E-goi servers in the electronic mail format. The problem consisted of a classification problem with an unbalance of classes with the minority class corresponding to the *phishing* emails and having less than 1% of the total amount of emails.

The project was divided into several phases according to standard methodologies for Machine Learning Projects. Throughout these phases, it was selected and extracted data features based on the e-mail content and the literature regarding these types of problems. Due to the unbalance present in the data, several sampling methods based on under-sampling techniques were tested to see their impact on the model's ability to detect *phishing* emails. Additionally, different machine learning models for classification problems were tried and their performance evaluated. The final model consisted of a neuronal network being able to detect more than 80% of the *phishing* emails without compromising the remaining emails sent by E-goi users.

Keywords: *phishing*, cyber-security, machine learning, neural network, class imbalance, anomaly detection;

UNIVERSIDADE DO PORTO

Resumo

Faculdade de Ciências da Universidade do Porto

Departamento de Matemática

Mestrado em Estatística Computacional e Análise de Dados

Phishing Detection with a Machine Learning Net

por [Ana BEZERRA](#)

Ataques de "*phishing*" são uma prática online comum cujo objectivo é extrair informação sensível. E-mails de "*phishing*" é um do tipo de ataques mais comum e pode ter um grande impacto tanto em indivíduos como em empresas. Ainda existe uma lacuna ligada à prevenção destes ataques, nomeadamente à sua detetação, sendo que novos ataques não são geralmente detetados. O objetivo desta tese de mestrado foi desenvolver uma modelo capaz de identificar e-mails de "*phishing*" com base em métodos de aprendizagem automática. O trabalho foi realizado em colaboração com a E-goi, uma companhia de marketing digital multicanal. Os dados consistiram em e-mails recolhidos dos servidores da E-goi em formato e-mail digital. O projeto consistiu em um problema de classificação com a falta de balanço entre classes, sendo que a classe minoritária corresponde a e-mails de "*phishing*" contendo menos de 1% da quantidade total de e-mails.

O projeto foi dividido em várias fases de acordo com as metodologias usadas em problemas de aprendizagem automática. Ao longo destas fases foram selecionadas e extraídas as variáveis do problema de acordo com conteúdo dos e-mails e a literaturia correspondente a este tipo de problemas. Devido à falta de balanço entre classes, diferentes metodologias baseadas em subamostragem foram testadas e verificado o seu impacto na capacidade do modelo de detetar e-mails de "*phishing*". Adicionalmente foram testados diferentes métodos de aprendizagem automática para problemas de classificação e o seu desempenho foi avaliado. O modelo final consistiu em uma rede neuronal capaz de detetar mais de 80% dos e-mails de "*phishing*" sem comprometer os restantes e-mails enviados

pelos utilizadores da plataforma E-goi.

Palavras-Chaves: *phishing*, cibersegurança, aprendizagem automática, rede neuronal, desequilíbrio de classes, deteção de anomalias;

Contents

Acknowledgements	iii
Acknowledgements	v
Abstract	vii
Resumo	ix
Contents	xi
List of Figures	xiii
List of Tables	xv
Glossary	xvii
1 Introduction	1
1.1 Framing and presentation of the work	1
1.2 Contributions of the Work	2
1.3 Organization of the project	3
2 Background Knowledge	5
2.1 <i>Machine Learning</i>	5
2.1.1 Unsupervised Learning	6
2.1.1.1 Principal Component Analysis	6
2.1.1.2 K-means Clustering	6
2.1.1.3 Hierarchical Clustering	8
2.1.2 Supervised Learning	9
2.1.2.1 Classification Problems	9
2.1.2.2 Metrics	10
2.1.3 Deep Learning	12
2.1.3.1 DL model's Architecture	13
2.1.3.2 Optimization Algorithms	16
2.1.3.3 DL Models	18
2.2 Unbalanced Data Problems	20
2.2.1 Re-sampling methods	21

2.2.1.1	Over-sampling methods	21
2.2.1.2	Under-sampling methods	21
2.2.2	Ensemble methods	23
2.3	Natural Language Processing	25
2.3.1	NLP Components	26
2.3.2	NLU <i>Transformers</i>	26
2.3.2.1	BERT <i>Transformers</i>	28
3	State of the Art	29
3.1	phishing Description	29
3.2	Types of Phishing Attacks	31
3.3	Countermeasures	32
3.3.1	<i>Blacklisting</i>	32
3.3.2	<i>Machine Learning</i> for phishing detection	33
4	Methodology	35
4.1	Data Understanding	37
4.2	Data Preparation	42
4.2.1	Variables Preparation	42
4.2.2	Sampling Methods	43
4.3	Modeling	45
4.4	Evaluation	47
5	Results	50
5.1	Initial Analysis and Sampling Methods	50
5.2	Models Results	62
5.3	ANOVA Results	66
6	Conclusions	70
6.1	Limitations and Future Work	71
A	Appendix	73
	Bibliography	86

List of Figures

2.1	Example of an Hierarchical Clustering Dendrogram.	8
2.2	Confusion Matrix.	11
2.3	Example of a ROC curve (adapted from Gonçalves et al. [11]).	12
2.4	Representation of DL model versus tradition ML models Stevens et al. [12].	13
2.5	Representation of a neural network (adapted from LeCun et al. [13]).	14
2.6	Representation of backpropagation (adapted from Cristina [14]).	15
2.7	Representation of a convolutional neural network (adapted from Bhardwaj et al. [15]).	19
2.8	Representation of a recurrent neural network (adapted from Lipton et al. [18]).	19
2.9	Explanatory example of RUS problem.	22
2.10	Example: The diagnostic procedure followed by a doctor when he gets a new case. His residents offer their diagnoses, then the doctor selects the majority answer as the final diagnosis (adapted from Kunapuli [26]).	23
2.11	Desing of sequential ensembles (adapted from Kunapuli [26]).	25
2.12	Scaled Dot-Product Attention (left). Multi-Head Attention (right) (adapted from Vaswani et al. [32]).	27
3.1	Phishing Attacks Steps Scheme.	29
4.1	Project’s Methodology and Data flow.	36
4.2	Example of a newsletter (adapted from LookFantastic [54]).	38
4.3	E-mail’s body example.	39
4.4	Example of vector for color type variable if the e-mail contains text with red colors and black colors.	41
4.5	Explanatory example of Cluster Under-Sampling technique.	44
4.6	K-fold Cross-Validation Methodology.	46
5.1	Hierarchical Clustering Dendrogram to text data.	51
5.2	Biplot for all variables with exception of the text array.	52
5.3	Majority Sample clusters with K=4.	54
5.4	Metric Values for chosen Models.	57
5.5	Metric Values for different phishing ratios	59
5.6	Metric Values for different sampling methodologies.	61
5.7	Confusion Matrix for the training and testing for the NN(135,81,11,2) model.	64
A.1	Trasformer Architecture (adapted from Vaswani et al. [32]).	74
A.2	E-mail’s header example.	75
A.3	E-mail’s format example.	75

A.4	Correlation Diagram for all variables except text array.	76
A.5	Elbow and Silhouette Graphs for data with text sorted by Hierarchical Clustering.	77
A.6	Elbow and Silhouette Graphs for data with text sorted by PCA.	78
A.7	Different Sampling Methodology's Samples Location.	79
A.8	Confusion Matrix for the training and testing for the Random Forest Model.	80
A.9	ROC curves for the training and testing for the NN(135,81,11,2) model.	80
A.10	ROC curves for the training and testing for the Random Forest model.	81

List of Tables

3.1	phishing features and its descriptions.	34
4.1	Phishing E-mails description and amount.	39
4.2	Extracted variables from <i>.eml</i> files.	40
4.3	Phishing E-mails percentages	44
4.4	Models' parameters to be tested.	47
5.1	Type of features present in the dataset.	51
5.2	Sets Names and methodology description.	55
5.3	Results of F1 according to different text array sizes.	63
5.4	Final Models' results.	65
5.5	F1 average results for different phishing ratios and sampling methods.	67
5.6	Results of F1 according to different text array sizes.	69
A.2	Accuracy, Precision and Recall Values for different models, <i>phishing</i> ratios and sampling methods.	82
A.3	Grid search results for Random Forest.	83
A.4	Grid search results for Neuronal Network.	84
A.1	Activation functions	85

Glossary

AI	Artificial Intelligence
Adam	Adaptive Moment Estimation
BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolution neural networks
CRISP-DM	Cross Industry Standard Process for Data Mining
DNN	Deep Neural Networks
DL	Deep Learning
IP	Internet Protocol
FN	False Negatives
FP	False Positives
ML	Machine Learning
NLG	Natural Language Generation
NLP	Natural Language Processing
NLU	Natural Language Understanding
PCA	Principal Component Analysis
RNN	Recurrent Neural Network
ReLU	Rectified Linear Unit
RUS	Random Under-Sampling
SBC	Under-Sampling Based in Clustering
SGD	Stochastic Gradient Descent

SMOTE	Synthetic Minority Oversampling Technique
TN	True Negatives
TP	True Positives
URL	Uniform Resource Locator

Chapter 1

Introduction

1.1 Framing and presentation of the work

The Internet and the Web have changed humanity in many ways, some for the better and some definitively for the worse. Its use can often lead to bad behaviors and nowadays, with individuals being able to stay behind a computer screen and the anonymity that this type of environment provides, online criminal offenses are increasing at an accelerating rate. Security breaches are happening across all types of companies. Thus, in today's society, a good cyber-security system is paramount to all industries and not just online-based companies. Putting facts into numbers, RiskBased SECURITY [1] has found that in 2020, just from January to September, 36 billion records with sensitive information were released, making 2020 the "*worst year on record*" in manners of records releasing.

Phishing is a term used to describe a type of cyber-attack to steal sensitive information from its victims usually via a forged e-mail, text message, websites, among others. There are different types of phishing, some more easily targeted and recognizable to the user's eyes, and others more elaborated and difficult to track. In 2006, Dhamija et al. [2] found that even those with experience in internet security can be fooled by a "good" phishing website.

E-goï is a Portuguese company based in Matosinhos that provides a software as a

service (Saas) platform for multi-channel marketing, acting as E-mail Service Provider for one of their features. This means that an individual can use the E-goi platform to send mass e-mails to a list of contacts. Because E-goi has several package options having one free to the public where a user can send e-mails to up to 100 persons, it also suffers from criminal events such as phishing. In this case, the criminal uses the platform to send mass phishing e-mails, and because these types of platforms are associated with a secure image, the recipients oftentimes trust these e-mails and are more prone to trust in the received content. If criminal acts happen on the platform, such as phishing e-mails, a block will be issued and can lead to a disruption of service. Additionally, if the platform allows criminal activities as such it can lead to a breach of trust between the company and the customers.

1.2 Contributions of the Work

Phishing is a theme that has been increasingly researched throughout the years, as the number of attacks keeps rising. Still, there is a necessity to understand how this type of act happens and its characteristics of it, especially in regard to phishing e-mails. Thus this work aims to contribute knowledge regarding mass spread phishing attacks.

In regards to Machine Learning, this project also uses state of the art techniques and some processes that are not yet being frequently used for data processing such as sampling techniques based on under-sampling.

In regards to E-goi, this project can be quite useful by reducing the time necessary to evaluate and detect the phishing e-mails and by improving the traffic quality that goes through the company's servers.

1.3 Organization of the project

This document is organized into seven chapters describing in detail all the work carried out throughout the project as well as the reasons behind the decisions that had to be taken.

- **First Chapter - Introduction:** Presents the project's context and goals as well as its contributions.
- **Second Chapter - Background Knowledge:** Describes concepts regarding Machine Learning and its processes in depth to enlighten the reader with some concepts that will be used in the project.
- **Third Chapter - State of the Art:** Describes the concept of phishing and the overview of the related works regarding this theme.
- **Fourth Chapter - Problem and Data Description:** Explains better the project's necessity and the type of data that was provided for the problem.
- **Fifth Chapter - Methodology:** As the Chapter's name suggests, this chapter describes the methodology carried throughout the work and the used techniques from the beginning stages to the project's completion.
- **Sixth Chapter - Results:** Display the results for all the processes and techniques carried out in this project. It also contains the result's interpretations and possible explanations for some phenomena.
- **Seven Chapter - Conclusions:** Presents the project's conclusions, the achieved goals, as well as limitations and suggestions for future work.

Chapter 2

Background Knowledge

2.1 *Machine Learning*

Machine Learning (ML) and Artificial Intelligence (AI) are the buzzwords of the decade playing on the front stage of most industries. Having mathematics and statistics as their models' foundations, applications range in a variety of industries, from economics to informatics, health, retail, etc. ML models themselves can vary in complexity, from simple ones, such as linear regression, to more complex models like Recurrent Neural Networks (RNNs).

ML tasks can be divided into two subcategories: **supervised learning**, **unsupervised learning**. **Supervised learning** tasks are the most common in Machine Learning. The essence behind it is to create mechanisms capable of producing generalizations by looking at examples. To achieve such, the model trains with a set of labeled samples so, afterward, it can correctly identify new unlabeled instances. **Unsupervised learning** goal is to find something useful in the data when given unlabeled data, for example, "*these samples are e-mails, categorize them*". It can identify patterns in data, most of the time not visible to the human eye. It is also often used as a pre-processing step before applying a supervised learning model to achieve better data quality. [3] [4].

2.1.1 Unsupervised Learning

2.1.1.1 Principal Component Analysis

Many ML problems involve a great number of features making it harder for a model to train and find a good solution. Most times, it is beneficial to implement techniques capable of reducing the data dimension while maintaining most of the information. The most common reduction technique is Principal Component Analysis (PCA). The goal is to reduce the data into sequences of data projections – principal components. These are obtained by searching for orthogonal directions capable of explaining as much data variance as possible [5].

Let the data matrix X be of $n \times p$ size, where n is the number of samples and p is the number of variables. The data matrix can be transformed by subtracting each sample with the sample average vector. The covariance matrix, $C = X^T X / (n - 1)$, with size $p \times p$, is symmetric thereby the principal components can be obtained by eigen-decomposition of it:

$$CV = VD, \quad (2.1)$$

where V is a matrix of eigenvectors and D a diagonal matrix containing the eigenvalues λ_i organized in a decreasing order. The eigenvectors are the principal axes and the principal components are obtained by multiplying the matrix X by the eigenvalues V . These are organized in a decreasing order of variance, meaning that the first components encompass the majority of variance [5] [6].

2.1.1.2 K-means Clustering

Cluster analysis is one of the most common unsupervised learning methodologies. The aim is to ascertain if a given dataset is composed of distinct subgroups with different characteristics by measuring the degree of difference between the samples assigned to

each group. Given a certain dataset X and a certain number k of clusters, the K-means clustering goal is to minimize the sum of distances of the points in each cluster to their centroid. The centroid is an imaginary or real point representing the center of each cluster [6]. K-means it is an iterative method that can be summed into five steps:

1. **Initial attribution of K centroids:** K random points from the dataset are chosen as the initial centroids.
2. **Calculation of Euclidean distances:** For each cluster the euclidean distance between each dataset sample and the K centroids.
3. **Assignment of samples:** Each data sample is assigned to the cluster with the center closest to it.
4. **New K centroids:** The new centroids are assigned by taking the average of the points in each cluster. Additionally they are updated once a sample changes to another group.
5. **Repetition:** Steps 2 to 4 are repeated until the centroids stop changing or the number of iterations is reached [7].

An important step in the K-means algorithm is the assignment of the number of clusters k . If there is no prior knowledge of the data, it is necessary to estimate this parameter. The most known method is the **elbow method** where the sum of squared errors within the clusters is calculated for different values of k . This value decreases with the increase of k , the optimum number occurs when the sum of errors starts to stabilize. Another common method is the **silhouette method**, where the silhouette value (see Equation 2.2) measures how similar a point is to its own cluster, compared to other clusters. In Equation 2.2 the argument $a(i)$ is named average intra-cluster and represents the average distance between each point within a cluster. The $b(i)$ argument is called the average inter-cluster and represents the average distance between all clusters. The value of a Silhouette score can range between $[-1,1]$ and this score reach its maximum at the optimal k [8].

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad (2.2)$$

2.1.1.3 Hierarchical Clustering

As a clustering method, the main goal of this procedure is to aggregate the given data into clusters based on the data characteristics. This technique can be divided in two types: **Agglomerative** or **Divisive**. The first type starts by treating every observation as an individual cluster. Through an interactive process, these are merged based on similarity until a single unique cluster is achieved. The second type works in a reverse manner. All the observations start by being grouped into one cluster. Later, in each iteration, these are separated until each cluster is composed of a single observation. The Dendrogram is a visual aid for Hierarchical Clustering. This plot, similar to a tree-like diagram, presents all the sequences of merges that happen during the process – see Figure 2.1 [9].

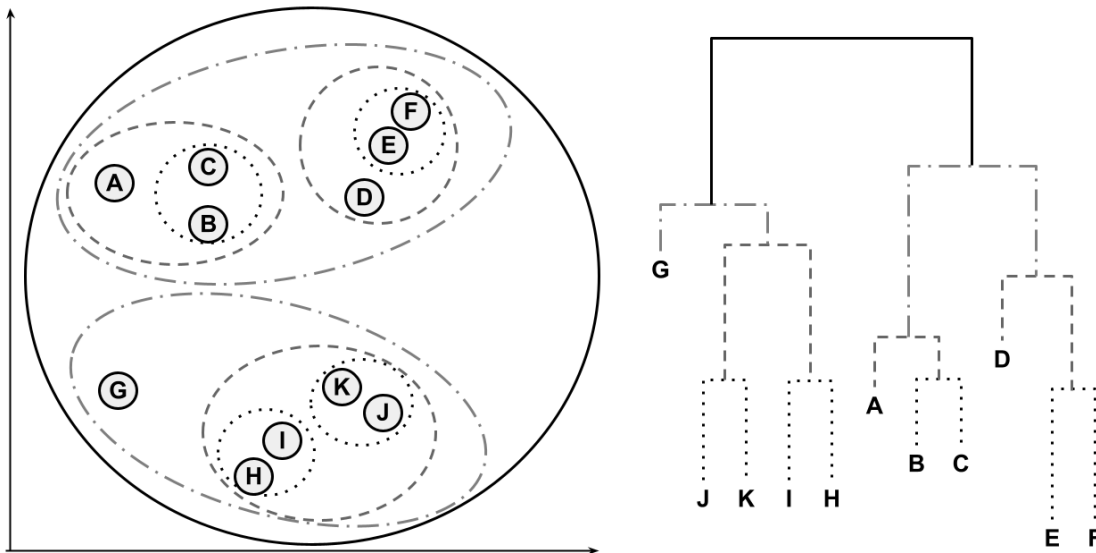


FIGURE 2.1: Example of an Hierarchical Clustering Dendrogram.

An important step in Hierarchical Clustering, regardless of the type, is calculating the similarity between clusters. There are several algorithms that can be used to calculate these distances.

- **Single linkage:** Uses the minimum of the similarity between points i and j such that i belongs to C_i and j belongs to C_j – Equation 2.3. The main advantage is to separate clusters with non-elliptical forms. Still, it can not separate well clusters if there is a

lot of noise between them.

$$D(c_i, c_j) = \min_{i \in C_i, j \in C_j} D(x_i, x_j) \quad (2.3)$$

- **Complete linkage:** This is the opposite of the single linkage. The algorithm uses the maximum similarity between the cluster's points – Equation 2.4. It has the advantage of separating well clusters with noise between them, though is biased towards spherical clusters.

$$D(c_i, c_j) = \max_{i \in C_i, j \in C_j} D(x_i, x_j) \quad (2.4)$$

- **Average:** Calculates the similarities between all pairs of observations and uses the average of these similarities – Equation 2.5. Similar to complete linkage this method is good at separating clusters even with noise. However, is also biased towards spherical clusters.

$$D(c_i, c_j) = \frac{1}{|C_i|} \frac{1}{|C_j|} \sum_{i \in C_i} \sum_{j \in C_j} D(x_i, x_j) \quad (2.5)$$

- **Ward:** This approach is similar to the average. Instead of using the average of similarities, it calculates the sum of the square distances. Likewise, it is good at separating clusters with a lot of noise but also biased towards spherical clusters.

2.1.2 Supervised Learning

ML supervised problems are classified according to their nature into two types: regression and classification. Regression problems, such as ordinary least squares, have as an output a continuous variable. In this method, the algorithm fits a linear model to minimize the residual sum of squares between the predicted outputs and the real values (targets). Algorithms for regression problems include Linear Regression, Support Vector Machines, Stochastic Gradient Descent, and K Nearest Neighbors, among others [4].

2.1.2.1 Classification Problems

Classification problems can be binary or multi-class classification. The first concerns problems where the output is a binary variable and the latter corresponds to cases having the

output as a group of classes. Common algorithms for these problems are Logistic Regression, Decision Tree, and Support Vector Machines [10].

- **Logistic Regression:** This algorithm predicts outcomes based on the *Sigmoid* function and uses maximum likelihood estimation to estimate the parameters of the probability distribution. The output is a probability value there is later compared with the threshold to identify a sample as 0 or 1.
- **Decision Tree:** Decision Trees can be used for both classification and regression problems. This method builds tree branches in a hierarchical approach and each node prior to the branch can be thought of as an if-else statement. The output of a branch can be another branch or leaves at the end of the tree. The branches are constructed based on the most important features and the final classification happens at the leaves of the decision tree.
- **Support Vector Machines:** This method is capable of performing linear or non-linear classification, regression, and even outlier detection. This method classifies the samples based on the position in relation to a border between the positive class and the negative class. The border maximizes the distance between the data points from both classes and is called a hyperplane [6].

2.1.2.2 Metrics

To access a model performance there are different types of metrics according to the algorithm type. For regression algorithms, the better known is the regression coefficient score R^2 . This corresponds to the proportion of variance explained by the variables thereby representing how well the model can predict unseen variables. Some other relevant metrics include the Mean Absolute Error, which, as the name suggests, corresponds to the expected value of the absolute error loss. The Mean Squared Error corresponds to the expected value of the squared quadratic error [4].

Most metrics used for classification problems relate to the confusion matrix. A confusion matrix plots the real labels from the samples against the predicted ones - see Figure 2.2. In this matrix, the diagonal corresponds to the match between the positive and

negative values from both classes. The positive samples identified correctly are called true positives (TP) and the negative samples identified correctly are named true negatives (TN). The negative samples predicted as positive are called false positives (FP) and the positive samples predicted as negative are called false negatives (FN) [4].

		Real Classes	
		Positive	Negative
Predicted Classes	Positive	TP	FP
	Negative	FN	TN

FIGURE 2.2: Confusion Matrix.

By the confusion matrix it is possible to calculate the accuracy (Equation 2.6), precision (Equation 2.7), recall (Equation 2.8), and F1-score (Equation 2.9) for the model:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.6)$$

$$Precision = \frac{TP}{TP + FP} \quad (2.7)$$

$$Recall = \frac{TP}{TP + FN} \quad (2.8)$$

$$F1 - score = \frac{2TP}{TP + 1/2 \times (FN + FP)} \quad (2.9)$$

An important step is to evaluate the threshold values' impact on the true positive and false negative samples. Most times, the value is set at 0.5. Intuitively thinking, if a threshold has a very low value most of the samples would be classified as positives, even the negative ones. Thereby, a higher number of false positives would also be in place. Similarly, if the threshold value is too high, it would lead to a low number of false positives. However, the number of false negatives would increase. To access how this ratio between the true positives and false negatives behaves for different thresholds, one can plot a **ROC** curve - see Figure 2.3. In this graph, the x-axis corresponds to the False Positive ratio and the y-axis to the True Positive ratio [11].

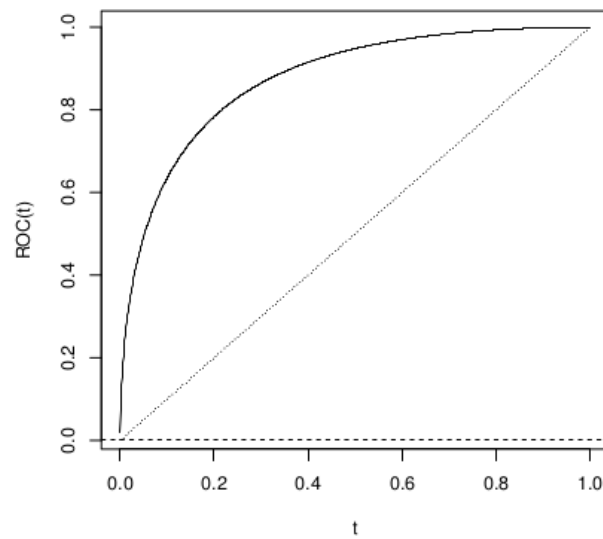


FIGURE 2.3: Example of a ROC curve (adapted from Gonçalves et al. [11]).

2.1.3 Deep Learning

There is a certain type of problems easily solved by humans that requires a great effort from machines, such as problems involving image recognition - i.e. the old google verification pop-ups asking to identify traffic lights in the pictures. On the other hand, very complex problems with an enormous amount of features can be nearly impossible for a human to solve and quite difficult for a machine too. Usually, this type of problem requires more complex models than the traditional ML algorithms can solve. Deep Learning (DL) is a branch of *Machine Learning* composed of a class of algorithms capable of approximating complicated non-linear processes making it possible to automate tasks that were previously limited to humans [12].

A DL model comprises a multi-layer stack of simple modules subject to learning. Many can compute non-linear input or output, therefore capable of understanding the levels of abstraction present in complex data and their complex structure. These types of models also include learning algorithms such as *back-propagation*, able to adjust the parameters of each layer accordingly to the representation from the previous layer. Due to the nature of this type of algorithm DL models are capable of being fed raw data - see Figure 2.4, instead of the traditional ML algorithms that required a feature extraction to have a good performance [13].

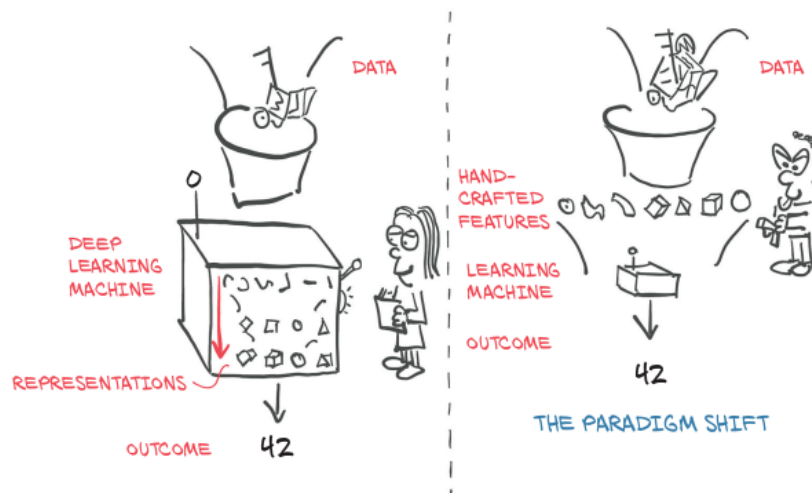


FIGURE 2.4: Representation of DL model versus tradition ML models
Stevens et al. [12].

2.1.3.1 DL model's Architecture

The Deep Learning models are called neural networks. These structures resemble the human nervous system with input, hidden and output layers – see Figure 2.5. Deep Neural Networks (DNN) consists of a neural network with several layers of nodes; depending on the task they can have different architectures. In a neural network, each layer has nodes connected to adjacent layers' nodes. Each connection has a weight value so the inputs are multiplied by the respective weights and summed in the node. The weight vector is adjusted by computing a gradient vector. This calculates the amount of change to the error value if a small increment would be added to the weights. Later, the weight vector is adjusted in the opposite direction to the gradient vector [13].

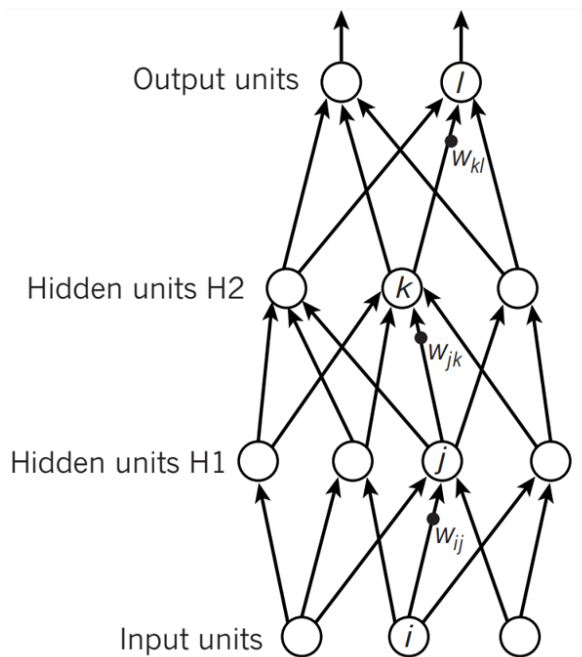


FIGURE 2.5: Representation of a neural network (adapted from Le-Cun et al. [13]).

As previously mentioned, in each neuron the inputs (x_1, \dots, x_n) are multiplied by weights (w_1, \dots, w_n) of the connections. The output corresponds to the overall sum of weights and inputs with an addition of a bias term (b), that can be integrated to the neuron as another weight (w_0). However, to ensure that the neural network is not restricted to learning exclusively in a linear weight, a non-linear activation function ($a(z)$) must be used – see Equation 2.10 [14] [15].

$$\text{output} = a(z) = a\left(\sum_{i=0}^n x_i + w_i\right) \quad (2.10)$$

The **training** of the neural network aims to search the set of weights to create a model that best represents the data patterns. To do so, neural networks rely on forward propagation and backpropagation. **Forward propagation** computes the outputs of each neuron through Equation 2.10 in the rightward direction. The initial weights can be randomly sourced or can be initialized by using a pre-trained model. **Backpropagation** serves to correct the weights attributed to the model. For this, first, it is necessary to calculate the errors from forward propagation by computing the difference between the attained results and the target value. Then, the algorithm calculates the gradient of this error through the

use of the chain rule and partial derivatives – see Equation 2.11 and Equation 2.12. This process is applied repeatedly in a backward manner, from the output layer to the input layer. Once these gradients are calculated, the weights of each module is re-iterated to a learning rate η – see Equation 2.13 [13] [15].

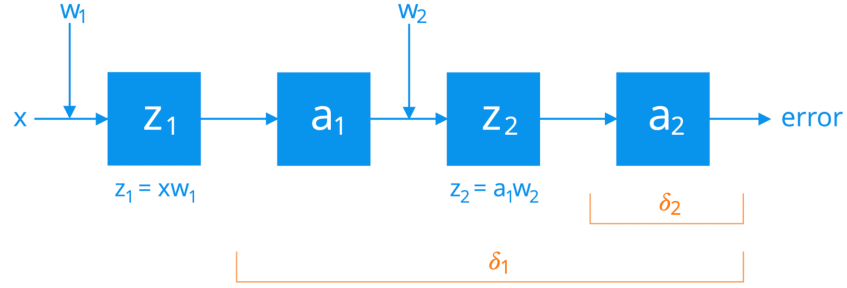


FIGURE 2.6: Representation of backpropagation (adapted from Cristina [14]).

$$\frac{\partial \text{error}}{\partial w_2} = \frac{\partial \text{error}}{\partial a_2} \times \frac{\partial a_2}{\partial z_2} \times \frac{\partial z_2}{\partial w_2} = \delta_2 \times \frac{\partial z_2}{\partial w_2} \quad (2.11)$$

$$\frac{\partial \text{error}}{\partial w_1} = \frac{\partial \text{error}}{\partial a_2} \times \frac{\partial a_2}{\partial z_2} \times \frac{\partial z_2}{\partial a_1} \times \frac{\partial a_1}{\partial z_1} \times \frac{\partial z_1}{\partial w_1} = \delta_1 \times \frac{\partial z_1}{\partial w_1} \quad (2.12)$$

$$w_1^{t+1} = w_1^t + (\eta \times \delta_1 \times \frac{\partial z_1}{\partial w_1}) \quad (2.13)$$

Several activation functions can be used in the artificial neuron. The function decides whether the incoming signals have reached the threshold and should output signals for the next level. In Table A.1, it is possible to see some different activation functions. Generally speaking, the most used activation function is the **Rectified Linear Unit (ReLU)** – see Equation 2.14. Not only is simpler than most activation functions, but also more efficient due to its linear and non-saturation form - for example, *sigmoid* and *tanh* both involve the exponential operation. However, this function can only be used in the hidden layers as its output is not in the probability space.

$$\sigma(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \quad (2.14)$$

There are some issues that can arise during the backpropagation stage and this can be due to the choice of activation functions. *Vanishing gradient* happens when the gradients are too small leading to a slow learning process or even a stop of the updating process. On the other hand, *exploding gradients* occur when the gradients are too large causing the learning process to diverge. This type of issue can happen if the activation function is not bounded or the learning rate is too big [15].

2.1.3.2 Optimization Algorithms

Optimization is a process that minimizes the errors and cost of the neural network providing the most accurate results possible. This process relies on minimizing the loss function. A usual loss function is the cross-entropy that builds upon the entropy by calculating how far an estimated value (p_i) is from its true value (t_i) according to the number of classes (n) – see Equation 2.15. From the loss function we can derive the gradients which are used to update the weights. The average overall losses constitute the cost [3].

$$\mathcal{L}_{CE} = - \sum_{i=1}^n t_i \log(p_i) \quad (2.15)$$

Several types of optimizers can be used depending on the necessity. **Gradient Descent** is one of the most basic and calculates the direction in which the weights should be altered. It does so by computing the first order derivative of the loss function and using the parameter θ – see Equation 2.16. In backpropagation, the loss is transferred through layers. The weights are modified to minimize the overall loss and depend on the losses. In the **Stochastic Gradient Descent (SGD)** the model's parameters are updated more frequently. This leads to faster conversion and decreases the risk of the optimizer being stuck in a local minimum. **Mini-batch Gradient Descent** is another variant for which the dataset is divided into batches. The parameters are updated after every batch.

By frequently updating the model parameters it works faster and can reduce the variance [16].

$$\theta = \theta - \eta \times \nabla_{\theta} J(\theta) \quad (2.16)$$

Momentum is a method to accelerate SGD by softening oscillations and accelerating the SGD process by adding a new parameter γ – see Equation 2.17. This term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions, leading to faster convergence. However, if the momentum is too high, the algorithm may miss the local minima and continue to rise up. To solve this issue **Nesterov Accelerated Gradient** was developed. The momentum term is also used to modify the parameters – see Equation 2.18, therefore giving an approximation of the next position of the parameters [16].

$$\begin{aligned} \mathcal{V}(t) &= \gamma \mathcal{V}(t-1) + \eta \times \nabla_{\theta} J(\theta) \\ \theta &= \theta - \mathcal{V}(t) \end{aligned} \quad (2.17)$$

$$\begin{aligned} \mathcal{V}(t) &= \gamma \mathcal{V}(t-1) + \eta \times \nabla_{\theta} J(\theta - \gamma \mathcal{V}(t-1)) \\ \theta &= \theta - \mathcal{V}(t) \end{aligned} \quad (2.18)$$

AdaGrad is a gradient based optimizer in which the learning rate is adjusted to the parameters. It performs larger updates for infrequent parameters eliminating the necessity to manually tune the learning rate. The sum of squares of the previous gradients is stored in a diagonal matrix G_t up to time t – see Equation 2.19. ϵ is a smoothing term avoiding the division by zero. The main drawback of this approach is the decrease in the learning rate which leads to slower training. **AdaDelta** is an extension of AdaGrad that tackles the decrease of the learning rate by limiting the array of accumulated past gradients instead of all previously squared gradients. In this algorithm, the sum of gradients is defined as a moving average. **Adaptive Moment Estimation (Adam)**, as AdaGrad, stores the exponential decaying average of the past squares gradients ($\mathcal{V}(t)$) but also keeps an

exponential decay average of the past gradients ($\mathcal{M}(t)$) – see Equation 2.20. The main advantage of Adam is its rapid speed [16].

$$\begin{aligned}\theta_{t+1,i} &= \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} g_{t,i} \\ g_{t,i} &= \nabla_{\theta} J(\theta_{t,i})\end{aligned}\tag{2.19}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\mathcal{V}(t) + \epsilon}} \mathcal{M}(t)\tag{2.20}$$

2.1.3.3 DL Models

Convolution Neural Networks (CNN) are mostly used for image recognition and video recognition tasks, as they are designed to process data in the form of multiple arrays or grid-like topologies. This type of network has two main properties: they can connect neurons forcing this to only take input from other neurons close to it. In that way, they reduce the number of weights since not all neurons are connected. Additionally, this network type uses parameter sharing, meaning that a limited number of weights are shared among all neurons in a layer. This lead to a reduction of the number of weights thus minimizing the chances of over-fitting [17].

The most important component of a CNN is the convolutional layer. It comprises a set of filters (also called convolutional kernels) convolved with a given input to generate an output feature map. The filter is defined by a set of weights and is applied across the areas of the input data. The purpose is to highlight a specific feature from the image, such as an edge or a line. The nearby neurons that participate in the input are called receptive field. The filter's output is a weighted sum of its inputs and represents the activation value of a neuron in the next layer. The neuron will be active if the feature is present at the spatial location. Between the convolutional layers, a sub-sampling or pooling layer is placed to reduce the size of the network and to reduce the network's susceptibility to shifts – see Figure 2.7. Max/mean pooling or local averaging filters are used often to

achieve sub-sampling. In the final layers, the neurons between them are fully connected and the classification process occurs [17] [15].

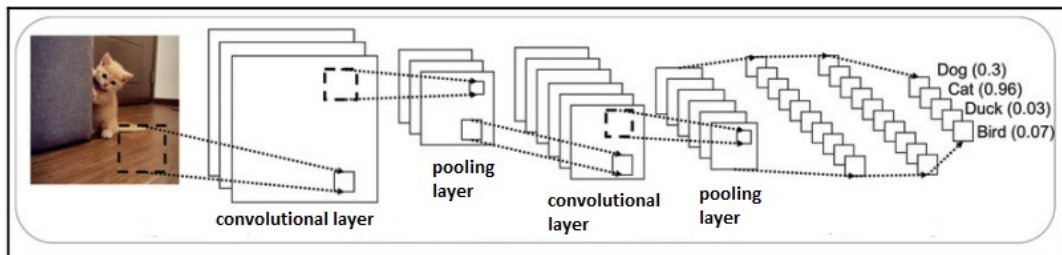


FIGURE 2.7: Representation of a convolutional neural network (adapted from Bhardwaj et al. [15]).

Recurrent Neural Networks are networks designed for taking into account sequential data, so any configuration/states of the network are impacted not only by the current input but also by their recent past. The architecture is similar to a feed-forward neural network with the addition of a looping mechanism between steps. At time t nodes with recurrent edges receive input from the current data point x_t and from the hidden node values h_{t-1} in the network's previous state – see Figure 2.8. The output at a certain time is calculated by the values in the hidden node h_t and can be influenced by the previous inputs from the previous state x_{t-1} . Each of the time steps in an RNN can be thought of as a layer, during backpropagation, errors go from one time step to the previous one, so the network is as deep as the number of time steps. The bigger the gradient, the bigger the adjustments will be. However, if the adjustments to a layer are too small it can lead to the *vanishing gradient* phenomenon and the earlier layers fail to do any learning[18] [15].

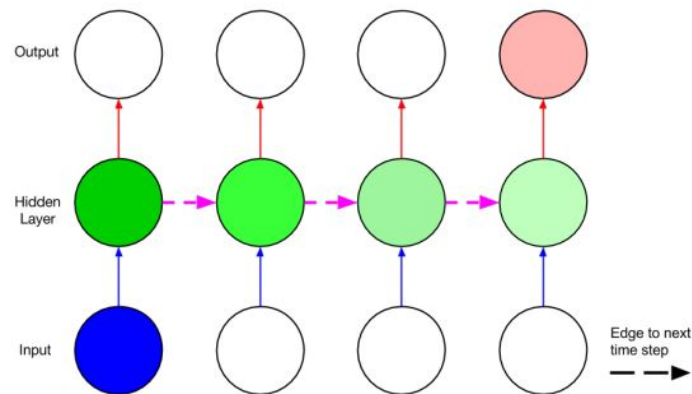


FIGURE 2.8: Representation of a recurrent neural network (adapted from Lipton et al. [18]).

2.2 Unbalanced Data Problems

Rare events and abnormal behavior are types of occurrences difficult to detect, yet, most of the time, their detection is crucial and requires a fast response [19]. Events such as e-mail phishing are considered rare events due to their low amount when compared to the amount of non-phishing e-mails. At E-goi, *black-lists* are used for e-mail fraud detection. Still, some e-mails containing new IP addresses, that have not been reported yet, can pass through the blocking system. These are scarce, especially when compared to not fraudulent e-mails, leading to an extremely uneven dataset.

Unbalanced datasets pose a problem to ML algorithms since the algorithms behind them are based on the assumption of an even dataset. In these situations, the most prevalent class is called the majority class, while the rarest class is called the minority class. In unbalanced datasets, ML models tend to have a poor performance when detecting the minority class and a good one for the majority class [20]. Following the example given by Liu et al. [21], for binary classification problems, a set with 99% samples belonging to class A and the remaining 1% belonging to class B would still have a 99% accuracy. Such occurrence is due to the low number of B samples during the model training resulting in a bias of the algorithm towards class A. Since class A composes 99% of the dataset the accuracy will be extremely high. However, if the goal is to find the samples within class B the model would have mediocre performance. In some problems, the misclassification of the minority classes can have severe impacts. For example, in ML models to detect the presence or absence of certain diseases a misclassification of a patient with a disease as disease-free can lead to patient health damage [22]. In this project, the misclassification of a phishing e-mail as a regular e-mail can lead to a security breach compromising the company's infrastructure and, for example, the release of private information.

Several approaches can be used to handle imbalanced problems; some can be focused on sample modeling, such as re-sampling techniques that treat the data before the use of an ML model. There are classification algorithms for imbalanced learning, such as ensemble methods and algorithmic classifier modifications[19].

2.2.1 Re-sampling methods

For re-sampling approaches, these are divided in three categories: **over-sampling methods**, **under-sampling methods**, and **hybrid methods**. The first consist of increasing the weight of the minority class by creating new minority class samples, the second use a sub-sample from the majority class by discarding intrinsic samples, and the latter is a combination of both under-sampling and over-sampling methods [19].

2.2.1.1 Over-sampling methods

With almost 20,000 citations, the most common techniques in manners of oversampling are duplicating the minority samples and Synthetic Minority Oversampling Technique - SMOTE. As the name states, the first technique is quite simple and consists in creating a copy of each sample of the minority class with some perturbation. SMOTE was first presented by Chawla et al. [23] in 2002 and produces synthetic samples from the minority class without changing the number of samples of the majority class. In this technique, the data is plotted into a feature space and for each minority class sample, a synthetic example is generated by the k nearest neighbor. Depending on the ratio between the majority class and the minority class the number of neighbors from the k nearest neighbors changes. By generating more synthetic samples, the dataset becomes balanced artificially. The main drawback of over-sampling methods as such is the increase of bias for the minority class [19].

2.2.1.2 Under-sampling methods

The simplest and most used method of under-sampling is Random under-sampling (RUS), which involves a random selection of majority class examples. The main issue with under-sampling techniques is the loss of information, especially with techniques such as RUS. If the selected samples happen to be in a localized selection this will lead to a biased sample thus having an inaccurate representation of the majority class, as shown in Figure 2.9.

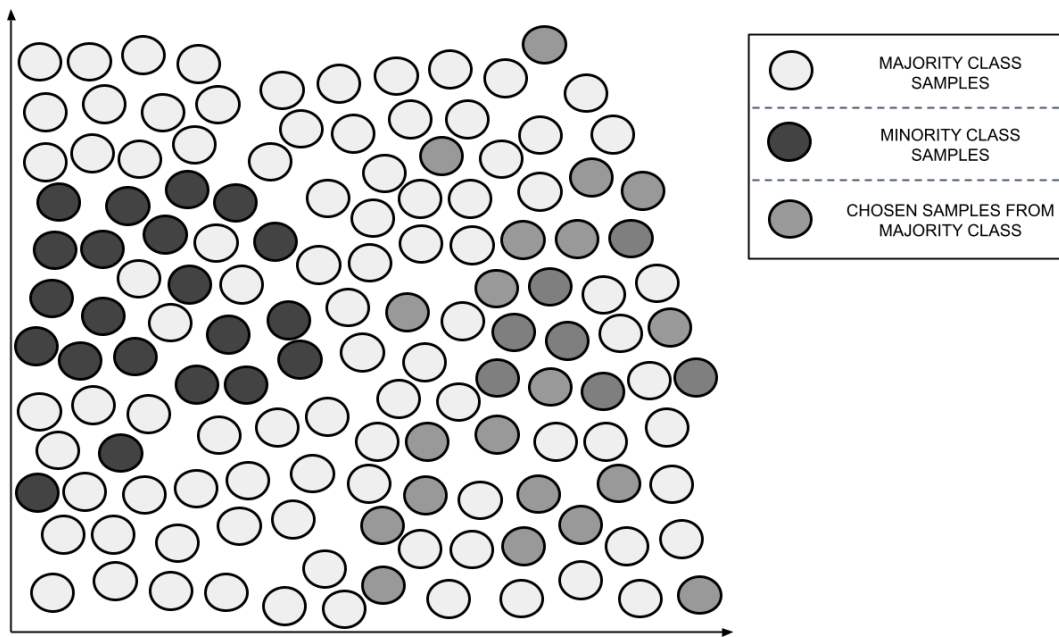


FIGURE 2.9: Explanatory example of RUS problem.

To overcome the loss of information issue, many researchers proposed other types of under-sampling techniques such as clustering techniques. Within this type of approach several methodologies can be used to address this issue, some of them are:

- Yen and Lee [24] proposed a clustering technique denominated as SBC (under-sampling based on clustering). The data-set is divided into K clusters and new samples come from clusters that contain a higher ratio of samples from the majority class. Their study found SBC to be more accurate and stable when compared to other methods such as RSU.
- Rahman and Davis [22] proposed a cluster under-sampling method before a classification model to identify patients with heart disease. Their sampling technique consisted of clustering only the samples from the majority class into K clusters. Afterward, K training subsets were built by combining samples from each K cluster with the samples from the minority class and the model is trained with each set. In the final training of the model, to build the decision support only the sub-datasets that gave a higher accuracy are used.
- Lastly, similar to the last approach, Lin et al. [20] proposed a cluster under-sampling method where the clustering is made only in majority class samples. However, in

their method, the K number of clusters is equal to the number of data samples in the minority class. In their method for each cluster, it is used the center value or the nearest neighbor to each cluster center ensuring an equal number of samples of each class.

2.2.2 Ensemble methods

As mentioned, when dealing with imbalanced problems one can use models designed to deal with this type of problem. Ensemble methods' goal is to achieve better predictions by training several ML algorithms, communally named estimators. Individually, these algorithms have poor performance due to the existence of a high bias or the existence of too much variance to achieve a robust solution. However, in an ensemble design, these poor predictions can be fused and through voting mechanisms, high performance can be achieved [25]. This way of thinking is not far from our daily decisions processes, an example given by Kunapuli [26] is to imagine a doctor diagnosing a complicated medical case. A usual approach is to consult with several other doctors from different specialties (cardiology, neurology, etc) and take into account their opinions to form the final decision regarding the patient's diagnosis – see Figure 2.10.

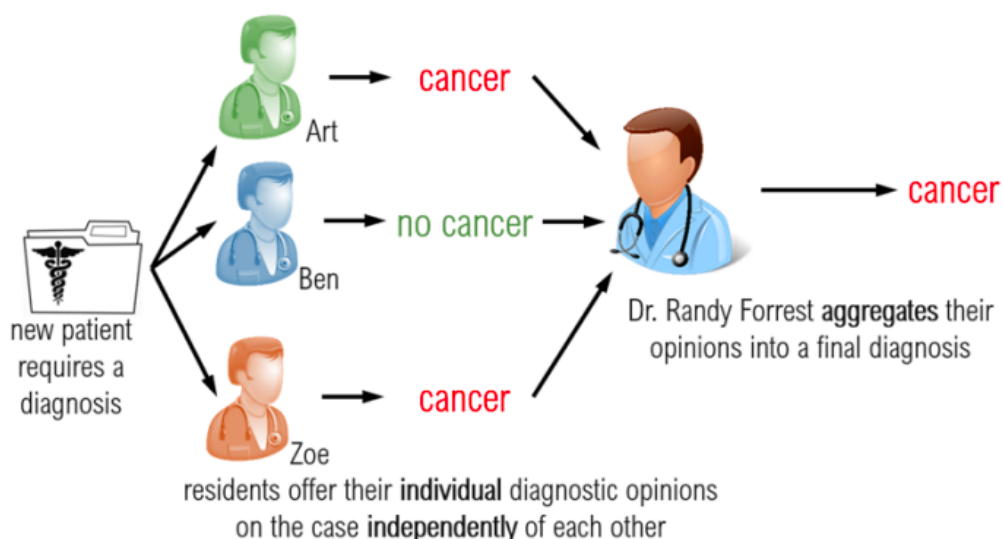


FIGURE 2.10: Example: The diagnostic procedure followed by a doctor when he gets a new case. His residents offer their diagnoses, then the doctor selects the majority answer as the final diagnosis (adapted from Kunapuli [26]).

Ensemble methods can be designed by having the models train in parallel or in a sequential manner. The most widely-used ensemble classification methods with a parallel design include bagging and random forests, and, in a sequential ensembling, boosting (such as AdaBoost) or gradient boosting [25].

Parallel ensembles, as the name suggests, trains each estimator individually thus allowing them to train in parallel. The ensemble can be homogeneous or heterogeneous. **Homogeneous** designs such as bagging (also named *bootstrap aggregation*) and *random forest* uses the same learning algorithms. To achieve diversity, usually, the data used to train those models is randomly sampled from the original training pool (or replicates of the training data with or without replacement as *bagging* does). Additionally, random sampling can be applied to features or both features and training data, *random forest* does the latter. Since the data and/or features fed to each estimator are different, the outcomes give different results. Lastly, the predictions are combined into one ensemble prediction leading to better performances. **Heterogeneous** ensemble, as the name implies, uses different learning algorithms. Depending on how the individual base estimator predictions are combined they can be parted into two different types. The first type assigns weights to the predictions from each estimator, where better estimators are assigned with higher weights. The second uses a learning algorithm where the predictions of individual base estimators are used as meta-data and trained to make the final predictions [25][26].

Sequential ensembles, such as *AdaBoost* or *gradient boosting*, exploit the dependence of base estimators by performing re-iterations where a estimator is trained to improve upon the predictions of the previous one – see Figure 2.11. In *AdaBoost*, the data samples are trained several times by the estimators. In each successive iteration, based on misclassified samples, the sample's weights are modified, therefore 'fixing' the previous estimator mistakes [27]. *Gradient boosting* randomly samples the data to get subsets for each estimator train on a subset. The residuals from each model are also used to train the next model. The goal is to minimize the sum of the residuals forcing the prediction close to the actual value [25].

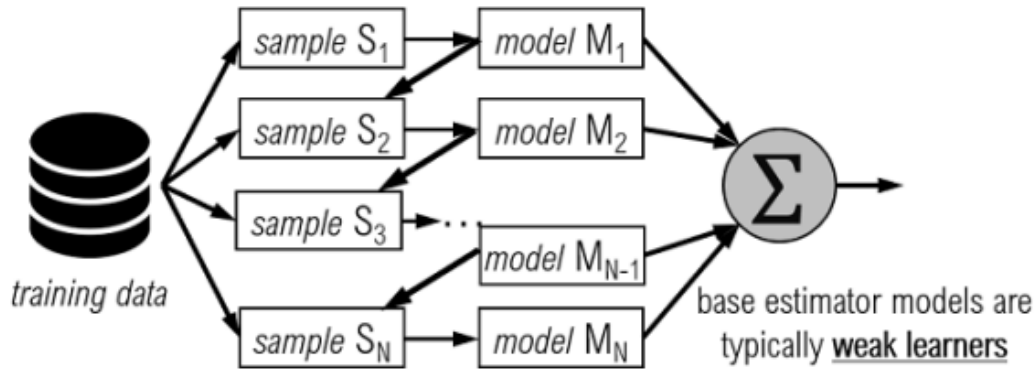


FIGURE 2.11: Design of sequential ensembles (adapted from Kunapuli [26]).

2.3 Natural Language Processing

Natural Language Processing (NLP) is a section of ML and Linguistics, having roots since the late 1940s with Machine Translation methods implemented to translate languages such as English and Russian. However, the term “Natural Language Processing” was first used in 1980 after the surge of AI processes, when computational grammar theory became highly researched [28]. Until today, NLP is a highly researched area used across a broad set of industries and applications like Machine Translation, e-mail Spam detection, and Information Extraction. It encompasses two major areas: Natural Language Understanding (NLU) and Natural language Generation (NLG) – see Figure 3.1. With these, NLP seeks to convert unstructured language data into a structured data format to understand speech and text and formulate relevant and contextual responses [29].

Natural Language Understanding is based on Linguistics, which is a science involving language’s meaning, context, and other forms. The five main pillars of NLU are Phonology, Morphology, Syntax, Semantics, and Pragmatics. These are responsible for understanding the systematic arrangement of sound; word formation and the meaning behind its smallest units; the structural dependency relationships between the words; and their possible meanings even without this being encoded into them [29][30].

Natural Language Generation produces overall text like phrases, sentences, etc. Is the opposite of NLU and involves four tasks to be able to generate overall text: Content

selection, Textual Organization, Linguistic Resources and Realization. These are responsible for selecting the information and transforming it if necessary; organizing it according to grammar; choosing the idioms and syntactic constructs; and deploying the output in a text or voice format [29].

2.3.1 NLP Components

Linguistics encompasses two areas – computational linguistics and theoretical linguistics. The first is mainly concerned with the study of natural language analysis by developing models to handle natural language as input. The last focuses on language performance and grammatical competence. Sentence analysis can be divided into two branches: syntax analysis and semantic analysis [31].

- **Syntax analysis:** Encompasses two tasks: the first uses a process named *parsing* that can determine the sentence structure by identifying not only the subject and object of each verb but each modifying word. The second task regularizes the syntax structure by simplifying a large number of possible input sentences into a small number of structures, i.e. “*John ate cake and Mary [ate] Cookies*” (adapted from Chowdhary [31]).
- **Semantic analysis:** Tries to understand the sentence’s meaning by seeking the conditions under which it is true. So there is a meaning assigned to the structures resulting from this step. It is also the branch that presents the biggest struggles [31].

2.3.2 NLU Transformers

Several model architectures can be used for language modeling. In 2017, Vaswani et al. [32] presented the Transformer – see Figure A.1. This is a model architecture that presents a simple network based on attention mechanisms with an encoder and a decoder stack. Later on, in 2018, Devlin et al. [33] presented BERT (Bidirectional Encoder Representations), a model based on the Transformer that uses only a selection of blocks of the encoders [34].

In the Transformer, the encoder stack is composed of six identical layers, each with two sub-layers. The first sub-layer is a multi-head self-attention mechanism and the second layer is a position-wise connected feed-forward network. The decoder is similar to the encoder, however, it has an additional sub-layer at the beginning to perform a multi-head attention mechanism. To achieve higher efficiency, the output of every sub-layer has a constant dimension that can be changed according to one's goals. Attention mechanisms are a "word-to-word" operation. An attention function goal is to find the relationship between each word in a sentence including the word being analyzed with itself. The base function in the Transformer is the Scaled Dot-Product Attention mechanism – see Figure 2.12, which maps a query and a set of key-value pairs. The output is a vector with a weighted sum of the values determined by a compatibility function of the query with the corresponding key. In each attention sub-layer, by linearly projecting the queries and key values, the model can run a defined number of attention mechanisms at the same time, as shown in Figure 2.12. This set-up allows the model to jointly attend to information from different representation sub-spaces at different positions allowing a higher efficiency and reducing computing time [29][34].

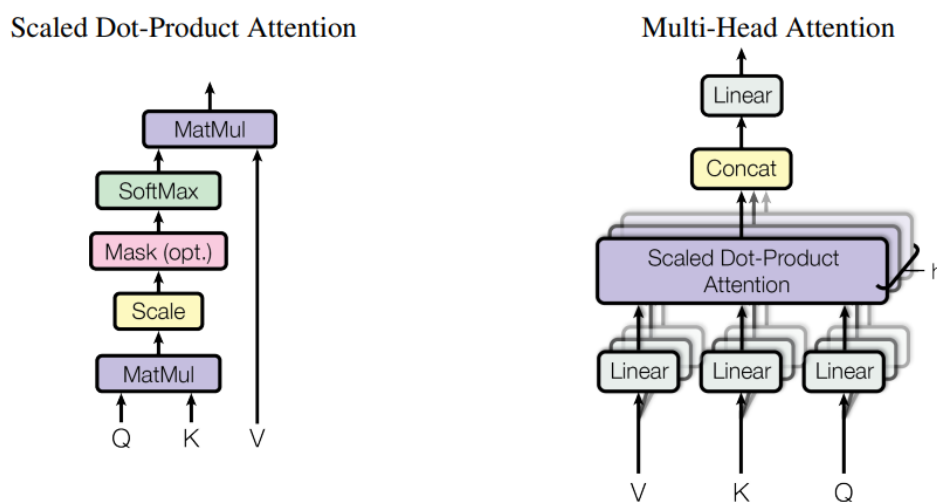


FIGURE 2.12: Scaled Dot-Product Attention (left). Multi-Head Attention (right) (adapted from Vaswani et al. [32]).

2.3.2.1 BERT Transformers

The BERT model is based on the original Transformer, but its architecture is simpler and empirically more powerful. It is designed to pre-train bidirectional representations from unlabeled text using Masked Language Models. When constructing the BERT model, Devlin et al. [33] divided its work in two major steps: *pre-training* and *fine-tuning*.

In the first step, the model was trained on sets of data such as the *BooksCorpus* and *English Wikipedia* that contains 800 million words and 2500 million words, respectively. The training was performed by the use of two unsupervised tasks: Masked Language Models and Next Sentence Prediction. In the first task, the bidirectional representations were trained by masking some percentage of the input tokens at random and then predicting those masked tokens – in Devlin et al. [33] research about 15% of the tokens were masked. Next Sentence Prediction aims to comprehend the relationships between two sentences that can not be accomplished with just a language model. In this task, the model was pre-trained with different pairs of sentences. In these pre-training examples, only 50% of the time the second sentence corresponded to the next sentence from the original text. In the second step of the model – *fine-tuning* – the model was initialized with the pre-trained parameters that were later fine-tuned by using labeled data [33].

Nowadays, BERT models are modified and updated to accomplish different tasks. A lot of these models can be found in the *Hugging Face* libraries, as this platform provides open-source NLP technologies [35]. Sentence-BERT models, a modification of BERT models using siamese and triplet networks are better at capturing sentiment information and also have higher computational efficiency when compared to the original BERT model [36].

Chapter 3

State of the Art

3.1 phishing Description

A phishing attack involves three steps – see Figure 3.1. The first is the victims receiving the content, the second is the victim following the instructions in the message – for example, opening the link to a fake website – and the last is the criminal profiting from the stolen information [37]. Since this type of attack is usually linked to financial gain, the internet provides an ideal scenario for the perpetrators, as there are multiple ways to hide their true identity. It can be also led to significant monetary gain. In 2011, Kim et al. [38] estimated that the losses of phishing attacks in the USA could go up to US\$61 million.

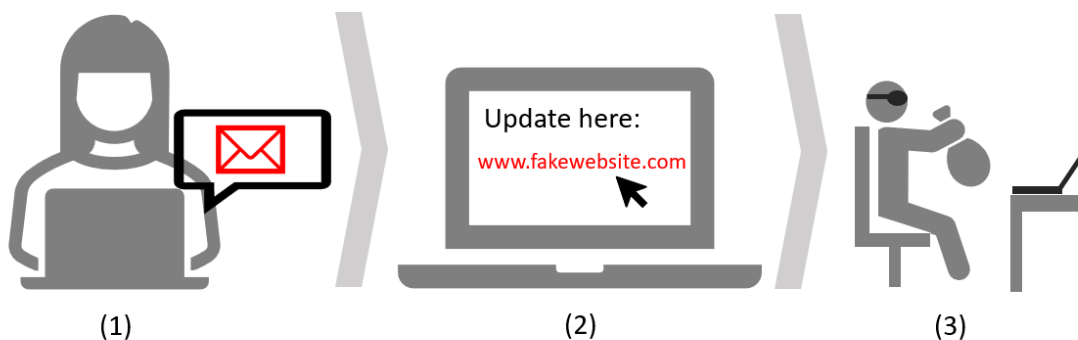


FIGURE 3.1: Phishing Attacks Steps Scheme.

A successful phishing attack relies upon a key factor: the ability to convey a sense of trust to the victim. Even before the technology revolution, con artists existed and the greatest ones were experts at persuading people to believe in their schemes through emotional manipulation and gaining the victims' trust. Thus the understanding of human behavior and its psychology is fundamental to be able to persuade a person through a message [39]. According to Ferreira and Teles [40], there are persuading principles that can be used on individuals to convince them to give out information and make them believe in a fraudulent e-mail message. Some examples are "*Authority*" – for example, an e-mail perpetrating a known figure of authority such as a re-noun establishment, "*Liking, Similarity & Deception*" – for example an e-mail that perpetrates a person familiar to the victim, and "*Need and Greed*" – this includes great opportunities for the victims and also involves the component of a limit time amount suggesting a sense of urgency.

Phishing attacks are evolving and becoming more focused on their targets. Nowadays, instead of a mass spread attack, a phishing e-mail can be sent to a certain group of individuals with similar characteristics, for example, individuals working at a certain company or with similar job descriptions [38]. On top of that, the type of attackers is also evolving, as a targeted e-mail phishing is more likely to be from a criminal organization with business and contingency plans and not just "*your teenage hacker like in the old days*" as Sheng et al. [41] describes. Aggravating these factors, the level of knowledge does not appear to be increasing within the newer generations. In 2019, profpoint. [42] reported that older generations were more aware of phishing. In their study, the older group – consisting of individuals with ages higher than 54 years old – had the best performance at explaining what phishing was, with 74% of interviewed individuals answering correctly. On the other hand, the group of individuals with younger ages – 18 to 21 years old – showed a worse performance with only 47% of the individuals knowing what the concept was. This can oppose quite a challenge to the dissemination of phishing attacks since the consumption of technology and online presence is increasing especially in younger generations [43].

3.2 Types of Phishing Attacks

In manners of phishing, as mentioned previously, there are different types of attacks. Not every type uses e-mail as the main tool, for example, *vishing* or *smishing* uses voice calls and SMS text. These types of attacks have also been increasing through the years. Just in 2019, out of the companies that reported phishing attacks, 84% also suffered from *smishing* and 83% from *vishing* [44]. Other types of phishing are much more complex and harder to perceive such as *pharming* in which the attackers hijack the Domain Name Server (DNS) from a certain enterprise or website so when the user types the real name the server redirects to a malicious website with a very similar IP address [45].

Still, the most common way to do phishing is via e-mail. According to Ghazi-Tehrani and Pontell [44], up to 96% of attacks are performed through this channel. In the beginning, these were poorly made and only had a success ratio of 0.5%, however, these days, due to the methods used to detect scams, e-mail phishing has evolved and comes in different types [44][46]:

- ***Deceptive phishing:*** Usually done through e-mail and portrays a message from a legitimate company asking for the user's information. A known example is the PayPal Scammers attacks in which the attackers asked the victims to resolve some problems with their accounts by clicking on the e-mail's link to the supposed PayPal website.
- ***Spear-phishing:*** Similar to deceptive phishing but provides more personal information such as the user's name, their company phone number, etc. Having this type of personal information shown rises the level of trust in the message making the users more susceptible to follow the steps and click on the hyperlinks or open the attachments.
- ***Whaling/CEO fraud:*** A previous analysis of the company is made to find the organization's CEO or other individuals with top positions. These individuals are targeted in order to collect sensitive information regarding the company's data.

- **Business E-mail Compromise:** A similar technique to *whaling* is used, however, instead of targeting the CEO, a lower-level employer is targeted and the subject of the e-mail pretends to be the CEO to get a money transfer or sensitive data [46].

3.3 Countermeasures

To ensure that a phishing attack is not accomplished there are measures that can be applied. In this manner, in 2009, Sheng et al. [41] interview 31 experts in different fields, from CERT (Computer Emergency Response Team) such as academic researchers, law enforcement experts, etc. The major recommendations involved the implementation of secure coding practice, investment in cyber-security, improving law enforcement and subsequent investment in specific types of technologies to better prioritize cases. Perhaps the most important was education of the overall public on these issues.

Most of the time the weakest link in a phishing chain is the human, even experts in the matter are not free from falling into a phishing attack since these have been more organized and targeted [41]. Thus, to complement human training, technical approaches are fundamental. In this subject, different methods can be applied, some are specific to certain types of phishing and others are more general.

3.3.1 Blacklisting

Realtime Blackhole Lists, *Domain Name System Black Lists* or *Blacklists* are lists of URLs, IP addresses, or keywords that had been previously detected as phishing. These lists can be made through the use of heuristics approaches or manual ones. The heuristic methods are often performed with Machine Learning (ML) to examine the URL, HTML, and server characteristics to classify the sites and addresses. Nonetheless, the best-known *blacklists* are operated by some of the biggest software companies such as Microsoft, Google, and PhishTank and each entry of the list is manually verified by these companies employees. This type of approach is good to screen the spread of attacks and has a relatively low False Positive (FP) rate, however, it fails in protecting against new attacks that use links that are

not disclosed on these lists. So, if the perpetrator creates a new website and IP address to release their attack, this type of security would fail at the moment of the attack and only be noticed afterward [37].

3.3.2 *Machine Learning for phishing detection*

In areas where fraud or anomaly detection have a high impact on the business, i.e. credit card fraud, the use of Machine Learning has become paramount to deal with threats in a real-time manner. In addition to the use of complex models, data processing is a key factor in having faster and lighter systems thus saving costs and computational resources [47].

Focusing solely on phishing attacks, in 2021 Ghazi-Tehrani and Pontell [44] conducted a study to 62 individuals from three different backgrounds: information technology/security professionals, *hackers* and researchers. When questioned about technological solutions to combat phishing 96.7% mentioned Artificial Intelligence (AI). Several studies have been made to detect phishing e-mails and websites with an accuracy of up to 97.3% using several types of ML methods [48]. To have a good detection system of websites and/or emails one of the critical steps is choosing which data features are going to be fed to the ML model. Some models utilize vision techniques, such as snapshots of the website, others use features from the website, e-mail, or text. Regarding e-mail phishing a lot of papers focus mainly on simpler e-mail characteristics without analyzing the e-mail text itself or only accounting for words with higher frequency in the phishing e-mails – see Table 3.1.

Feature	Description
Presence of URLs and IP-based links [49] [48]	The use of the visible URLs in the e-mail text, especially those without the “s” in “https” can be possibly linked to insecure fake websites.
Number of dots in URL links [49][50]	Commonly, the domain name of a legitimate organization should not have a large number of dots.
Number of URLs and IP-based links [49][50]	Usually, phishing e-mails contain a high number of links to fake-websites.
Length of URL links [51]	phishing e-mails typically contain long URLs to hide the doubtful part in the address bar.
Count of Images [52][40]	The high number of images can be associated with phishing e-mails as this are a way to distract the victims from the e-mail content and its discrepancies.
Word List Features [49][50][40]	A binary indicator of the presence of a select list of words associated with phishing.
Time stamp [47]	Certain hours and days of the week can be associated with phishing.

TABLE 3.1: phishing features and its descriptions.

Chapter 4

Methodology

Machine Learning projects often follow a certain methodology where the project is divided into several phases with specific timelines to facilitate the development of the ML solution. One of the most common methodologies is the Cross-Industry Standard Process for Data Mining (CRISP-DM) proposed in the 2000s by Wirth and Hipp [53]. This methodology has been altered over the years and adapted for different projects and companies, therefore the number of phases may vary. From an overall point of view, there are six phases one may consider in an ML project: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, Deployment. Each phase has sub-tasks associated with it, moreover one may relapse to the original stages since an ML project is fluid and the goals may change with the increase of knowledge.

A tailored version of this methodology with five phases was carried out throughout the project – see Figure 4.1. The first two phases were the most time-consuming to ensure data quality as it is one of the most critical aspects for a good performance in an ML model. Again, Figure 4.1 shows the data flow across the project's phases. As the data pool of phishing e-mails is quite low (only 0.1% of the entire data pool) one key goal is to create a system where the new data after being analyzed by the model is re-directed to the pool data and possibly re-train the model to achieve higher performances.

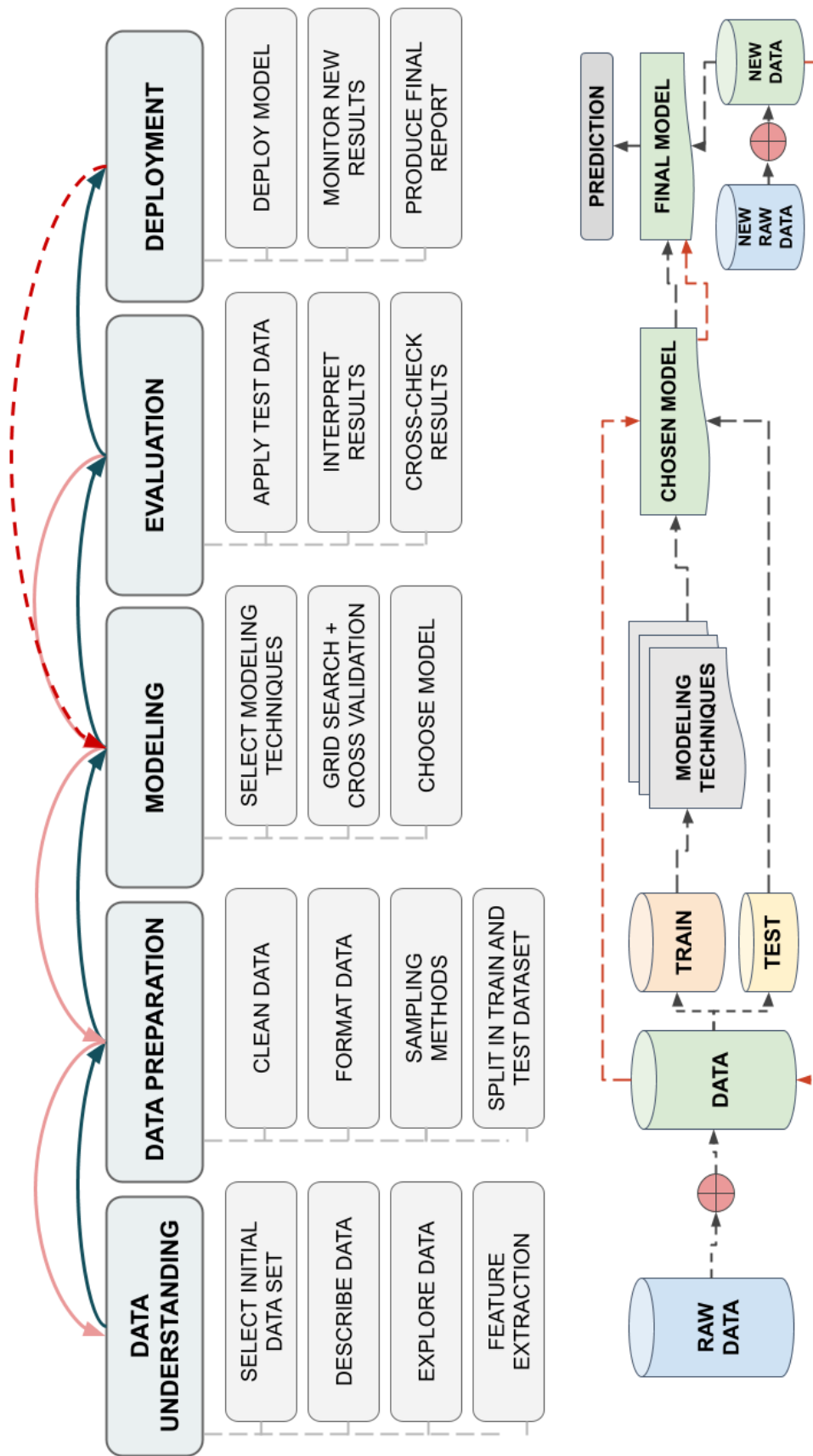


FIGURE 4.1: Project's Methodology and Data flow.

4.1 Data Understanding

As mentioned in Section 1.1, E-goi is a SaaS multi-channel marketing automation platform also operating as an E-mail Service Provider (ESP), meaning that a user can send mass e-mails, namely campaigns, and newsletters, among others. These types of companies rely on their reputation and are monitored by Inbox Service Providers, who enforce good faith practices. They are also strongly affected by criminal events such as phishing. An attacker uses the platform to mass distribute their fraudulent e-mail since the probability of rejection is decreased. If the company allows criminal activity, a block is issued which leads to the disruption of service. Thus it is important to ensure the user operates in good faith. Content monitorization is not a scalable task when performed by humans, especially for companies that work as an ESP. This type of task can be slow and tedious and even require many working hours, being unfeasible for a scaling possibility. Furthermore, since these types of companies are strongly affected by their reputation, the threat must be dealt with in an efficient manner. Given the complexity and variability of the attacks the tools used to minimize and diffuse them need to be adaptable and have to be quite precise. It is not desirable to block a user account of a real customer since this lead to a break in the customer's trust resulting in a bad relationship with them.

Companies such as E-goi, and E-goi itself, often use *blacklists* to block most of the phishing attacks. Unfortunately, as mentioned in Section 3.3.1, this type of security tool is not fully preventive since if the new attack uses e-mail addresses or URLs not present in these lists, the attack goes by undetected. As explained in Section 3.1, the rate of phishing attacks are increasing over the years, therefore a truly preventive tool is necessary as new attacks are created every day. For this purpose, an ML-based tool can be quite useful.

The gathered data is a sample of e-mail files with extension *EML*, with a total of 514 027 ($\approx 99.9\%$) *.eml* files that are classified as not phishing and 501 ($\approx 0.1\%$) classified as phishing e-mails. The type of e-mail is usually campaigns or newsletters, like the e-mails one receives every day accumulating in the 'Promotion' folder – Figure 4.2. On the E-goi platform, a customer can choose to upload its campaign files or use the base templates available on the website.

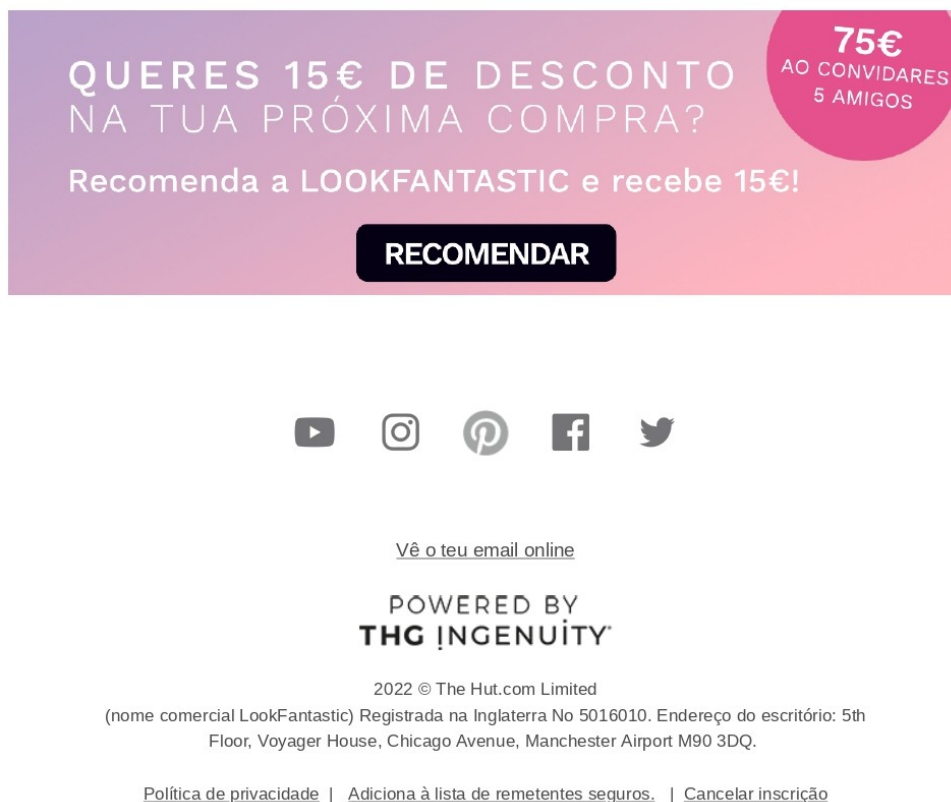


FIGURE 4.2: Example of a newsletter (adapted from LookFantastic [54]).

An e-mail file is composed of three parts: the headers, the body, and the formats. The e-mail headers contain information regarding the user, the date in which the e-mail was sent, and the subject/title of the e-mail, among others. The body of the e-mail contains the written message and the formats contain information regarding the type of font, sizes, number of images, etc. Due to the system complexity when opening a *.eml* file with a simple text editor, the text component may have some formatting errors across it. Some of these are displayed in Figure 4.3 inside the yellow margins. Additionally, other errors can occur, such as the absence of e-mail's content or the absence of the e-mail address. Lastly, a lot of users chose to send the same e-mail several times thus leading to a high amount of duplicated emails ($\approx 50.0\%$) that can not be used throughout the training of as it may led to an increase in the model's bias. Table 4.1 presents the types and amounts of these errors.

```

nn
n nnn
n nnn
n n
A CABECEIRAS URBAN RACE u00e9 uma prova BTT de 3 horas de
resistuu00eancia noturna que seruu00e disputada num circuito fechado com
base no Mosteiro de Su00e3o Miguel de Refojos, teruu00e uma
extensuu00e3o de cerca de 5 km e percorreruu00e o centro histuu00f3rico
de Cabeceiras de Basto, extensuu00edvel aos seus belos jardins,
prau00e7as, ruas, recantos e locais emblemuu00elticos ligados u00e0
riquuu00edssima histuu00f3ria e patrimuu00f3nio cabeceirenses que
seruu00e3o invadidos por cerca de 5 centenas de bicicletas, amantes e
adeptos da modalidade. n
n
n
A prova realiza-se dia 24 de agosto 2019 a partir das 20h00 e contempla
a participau00e7uu00e3o a solo, duplas e triplas, distribuu00eddos por
vu00elrias classes e categorias. n
Esta prova estuu00e inserida e pontua para o TROFu00c9U URBAN RACE.n
n
nn
nnnn
nnnn
nn

```

FIGURE 4.3: E-mail's body example.

Description	Number of E-mails	% of E-mails
Absence of e-mail content	3713	0.72
Absence of e-mail address	108855	21.17
Duplicated e-mail	246 732	48.00

TABLE 4.1: Phishing E-mails description and amount.

As explained the *.eml* file's structure can be divided into three different sections. Taking into account Section 3.3.2 several features were extracted with resource to *Python* language – see Table 4.2. The used libraries included *Numpy* (version 1.22.4) and *Pandas* (version 1.4.2). Some of the features such as cyclic features (hour and day of the week) and text related variables were later processed.

Section	Variable Name	Variable Description
Header	Hour	Numerical variable containing hour of the day at which the e-mail was sent. Varies between 0 and 23.
	Day	Numerical variable containing the day of the week. Varies between 1 and 7, 1 corresponding to Monday and 7 to Sunday.
	First part address	Numerical variable containing the size of the first part of the e-mail address (Ex: example@gmail.com has length 7)
	Last part address	Numerical variable containing the size of the last part of the e-mail address (Ex: example@gmail.com has length 9)
	Dot presence	Binary variable indicating the presence of more than one dot in the second part of the e-mail address (Ex: example@fake.website.com has value 1)
Body	Text	Array containing the e-mail text content.
	Body Length	Numerical variable containing the length of the text array.
	Visible HTTPS	Binary variable indicating the presence of secure visible links in the text.
	Visible HTTP	Binary variable indicating the presence of insecure visible links in the text.
	Emoji	Binary variable indicating the presence of emojis in the text
Format	Images	Categorical variable according to number of images. Three categories.
	Invisible HTTPS	Binary variable indicating the presence of secure invisible links in the text.
	Invisible HTTP	Binary variable indicating the presence of insecure invisible links in the text.
	Font type	Vector containing the presence of different types of fonts throughout the text.
	Font size	Vector containing the presence of different sizes of font throughout the text.
	Font color	Vector containing the presence of different colors of font throughout the text.

TABLE 4.2: Extracted variables from *.eml* files.

The aesthetic variables: 'Font type', 'Font size', and 'Font color' – were extracted differently than most binary variables. Since an e-mail can have different types of fonts, sizes or colors, these features were each kept in a vector form. These vectors had different dimensions according to the type of variable, for example, the color type variable had a dimension of nine corresponding to the colors of a rainbow. Figure 4.4 shows an example of how the vectors are constructed; each position of the vector corresponds to a color and can vary between 0 and 1 where the number "1" indicates the presence of the color and "0" the absence of it. In this example the e-mail text had red and black, therefore the first and seven-position have a positive value, while the remaining positions remain at zero.

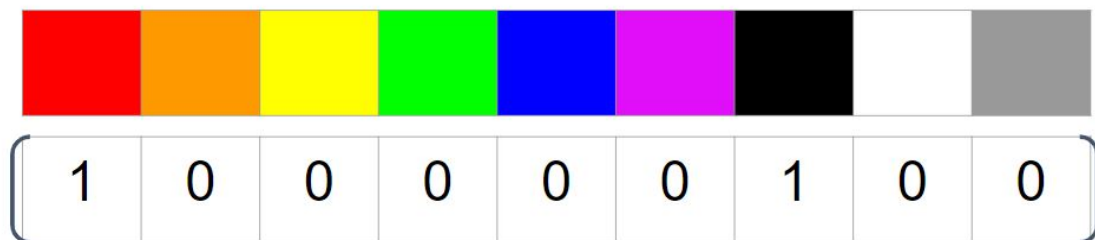


FIGURE 4.4: Example of vector for color type variable if the e-mail contains text with red colors and black colors.

For each aesthetic variable the vectors were organized in the following manner:

- **Font type** – [*neo-grotesque, geometric, humanist, sans-serif, other, transitional*]: Since there were a lot of different font types these were grouped into six categories corresponding to the font's family. For example, font types such as *arial* and *calibri* are included in the sans-serif family, while others such as *verdana* are considered humanist.
- **Font size** – [*<10px, [10,12[px, 12px, 13px,]13-20]px, >20px*]: The categories were chosen according to the sample's distribution for the font size.
- **Font color** – [*red, orange, yellow, green, blue, purple, black, white, grey*]: The colors were chosen according to the RGB coordinates.

4.2 Data Preparation

4.2.1 Variables Preparation

Following the methodology presented in Figure 4.1 after extracting the potential variables for the final model is necessary to clean and format the extracted data. For this purpose some steps were carried out:

- **Continuous** variables except for cyclic variables were standardized according to Equation 4.1. This is important, especially for some simpler models such as logistic regression to ensure that higher weight is not given to variables that have large scales.

$$\hat{z}_i = \frac{x_i - \hat{x}}{s} \quad (4.1)$$

- **Cyclic** variables such as hour are a complex case to have in a model due to their cyclic nature. Humans understand that the '23h00' is closer to '00h00' than '02h00', however, if we fed the integer of the hour to a model this would assume that '23' is the farthest number apart from the '00' hour. Therefore is important to encode these cyclic variables in a manner that their nature is still carried. Due to their nature these can be transformed through the equations that containing the cosine and sine functions – Equation 4.2 and Equation 4.3 where h_i is the original variable (ex: hour) and max_{h_i} is the maximum value it can achieve (ex: for hour the max_{h_i} has a value of 23).

$$x_i = \sin \frac{2 \times h_i}{max_{h_i}} \quad (4.2)$$

$$x_i = \cos \frac{2 \times h_i}{max_{h_i}} \quad (4.3)$$

- For **Categorical** variables the *dummies* were computed. *Dummies* are attributes created according to the number of classes present in the categorical variable. Therefore, if a variable has N classes, N attributes are created where each attribute is a binary variable indicating the presence or absence of a class.
- The **Text** array was processed by NLP techniques discussed in 2.3. For that purpose, a pre-trained multilingual model from the *Hugging Face* library was used. For this

project, the '*paraphrase-multilingual-MiniLM-L12-v2*' was used that transforms the data array into a vector with a dimension value of 384 [36]. Afterward, to decrease the matrix dimension, an agglomerating clustering was performed via hierarchical clustering using a bottom up approach. The new dimension size was studied considering the loss of information.

4.2.2 Sampling Methods

At the chapter's beginning it was mentioned that only $\approx 0.1\%$ of the data was phishing e-mails corresponding to a total of 413 samples after the data cleansing step. Furthermore, there were a large amount of duplicated e-mails, meaning that the same user send an e-mail with the same content more than once. To ensure that the data bias was minimal, it was decided to kept only one copy of the duplicated e-mails thus leading to a total amount of 214 phishing e-mails and 214 000 "normal" e-mails. Taking into consideration those constraints and the amount of phishing e-mails it was opted to use an under-sampling method. Different approaches were tried, some of them based on Rahman and Davis [22] and Lin et al. [20] works. Similar to Rahman and Davis [22] work, different sets were created based on the different approaches to be trained by the model:

- **RUS** – Collect majority class samples at random.
- **Cosine Similarity** – Sub-sample majority class by collecting samples closest to the minority class with resource to the cosine similarity. According to Huang et al. [55] is one of the most popular similarity measures applied to text documents.
- **Clustering technique 1** – Sub-sample majority class by collecting samples from different clusters. These clusters are obtained by performing the K-means clustering technique to the majority class. The K number of clusters is attained through empirical methods such as elbow method, *gap-statistic* method, and *silhouette* method [8] [56]. Instead of collecting samples at random from each cluster, it was collected the samples closest to the clusters' centers and samples from each cluster closest to the minority class – see Figure 4.5.

- **Clustering technique 2** – Sub-sample majority class by collecting samples from different clusters with a K number of clusters equal to the number of phishing samples [20].

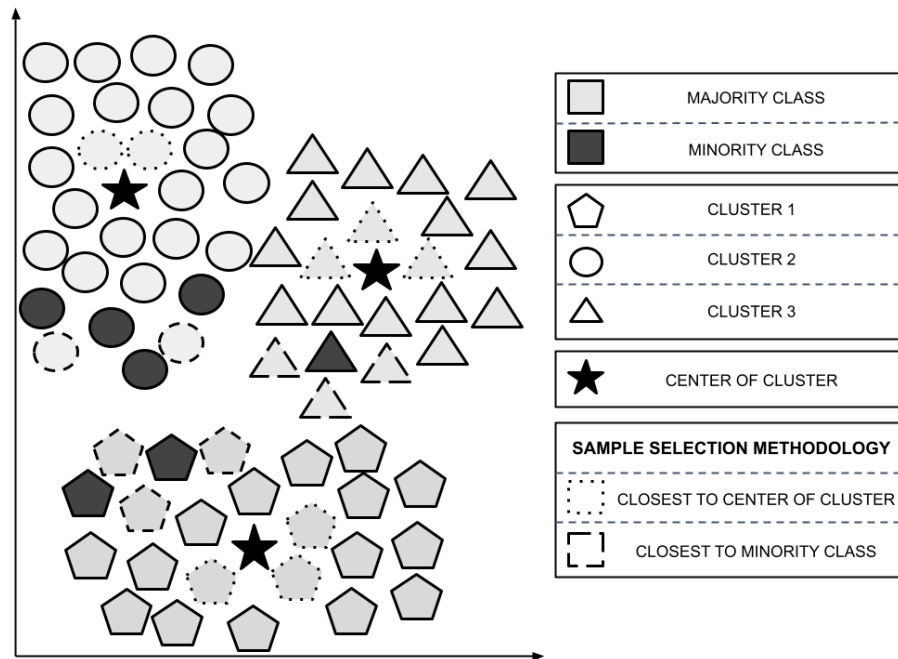


FIGURE 4.5: Explanatory example of Cluster Under-Sampling technique.

In addition to the different sampling techniques, it was also created sets with different **percentages** of phishing samples, presented in Table 4.3, as some of the models to be trained are more suitable to work with unbalanced data.

Ratio: N minority class : N majority class	% of phishing
1:1	50
1:2	33
1:4	20
1:19	5
1:99	1

TABLE 4.3: Phishing E-mails percentages .

To compare the difference between the different approaches and ratios it was performed a **One-way ANOVA** was performed for each factor. ANOVA stands for Analysis of Variance and studies the effect of a factor with several categories (for this case sampling methods or phishing ratio) on a dependent variable. It is based on hypothesis tests where the null hypothesis states that the means (μ) of each category are equal – see Equation 4.4.

$$\begin{aligned} H_0 : \mu_1 = \mu_2 = \dots = \mu_N \\ H_1 : \text{Not all } \mu_j \text{ are equal } (j = 1:N) \end{aligned} \quad (4.4)$$

In this study, the chosen dependent variable was the F1-score for a chosen model. The main assumptions for this technique are sample independence, variance equality, and normality. To ensure this conditions the used model was trained several times with different seeds and the histogram and QQ-Plot of the new overall sample were studied [57].

4.3 Modeling

As stated, for each sampling method and each phishing ratio a set was created to be trained by different ML models. Taking into account previous works on this subject, several models were chosen to determine the best results by training them using a k-fold cross-validation technique. Cross-validation techniques are often used in Machine Learning to assess the model stability and performance and ensure that this is not overfitting the parameters to the trained data. In **k-fold cross validation** the training data is split into smaller sets called folders, for which, one of the folders is used as a validation set while the other folders are used to train the model. This process is iterative, for each iteration the training set is split into k folders and the chosen evaluation parameter is determined by testing the data in the validation folder; this folder is different in every iteration – see Figure 4.6. At the end of the loop, the final evaluation parameter score is achieved by computing the mean of the iterations. Still, it is important to have different a testing set to confirm the attained results by the cross-validation [58].

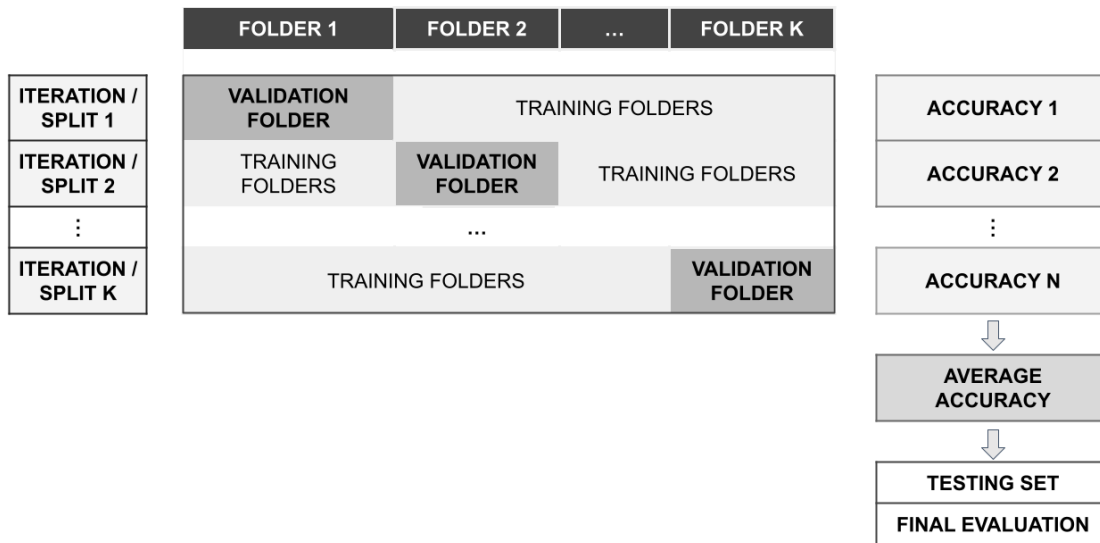


FIGURE 4.6: K-fold Cross-Validation Methodology.

The initial models to be trained were Logistic Regression, Support Vector Classification (SVC), Decision Tree, Random Forest, AdaBoost Classifier, and Gradient Boosting Classifier from the *scikit-learn* library. For each model, the results for accuracy, precision, and recall were attained and compared among themselves. Out of this model, it was decided to proceed with a Random Forest Classifier and a Neural Network (named in the *scikit-learn* library as “MLPClassifier”). For both models, an **exhaustive grid search** was computed to find the best hyperparameters. These are the model’s arguments previously defined by the user. These are parameters such as the number of hidden layers for the neural network. In an exhaustive grid search, several parameter values are considered and different combinations are tried to achieve a higher value for a scoring metric; this metric is defined by the user. In this work the researched hyper-parameters are presented in Table 4.4. The scoring metric was the F1-score, since it depends on both the precision and recall of the model.

Model	Parameter's Name	Parameter's Description
Random Forest	Criterion	Measures the quality of split for the decision trees. Three possibilities: "gini", "entropy" an "log_loss".
	"min_samples_leaf"	Integer that represents the minimum number of samples required to be at a leaf node.
	"max_features"	Number of features to consider when looking for the best split. Possibilities: "sqrt", "log2" or a ratio between 0 and 1.
	oob_score	Whether to use out-of-bag samples to estimate the generalization score. Set to be True or False.
Neural Network	"hidden_layer_sizes"	Inner layers configuration and number of neurons in each layer.
	"activation"	Activation function for the inner layers. Four possibilities: "identity", "logistic", "tanh", "relu".
	"solver"	The solver for weight optimization. Three possibilities: "lbfgs", "sgd", "adam".
	max_iter	Maximum number of iterations, for stochastic solvers this parameter determines the number of epochs.

TABLE 4.4: Models' parameters to be tested.

4.4 Evaluation

In manners of **evaluating a model**, when it comes to classification problems, there are several metrics that can be used – explained in 2.1.2. For each final model, the values for accuracy, precision, and recall were calculated as confidence intervals for these metrics. The confidence intervals were attained using a bootstrap methodology based on samples with a size of 200. These samples were attained by running the final model several times with different seeds. Other evaluation methods include computing and plotting the confusion matrix and ROC curve for both the training and testing set. Additionally, since

under-sampling methods are being used to collect training and test sets and, even though some of the methods used are trying to include as much data variance as possible, is plausible to assume that information is being lost. If a *non-phishing* e-mail is classified as phishing it will lead to a block of the customer's account. Thus is important to see how many e-mails samples are being classified as phishing from the samples of the majority class that were not selected to be part of the training or testing set. As the processing time of the model's prediction function increases with the number of samples, it was decided to create an iterative process to verify the percentage of blocked accounts.

- Randomly collects 10 000 samples from the majority class that are not present in the training or testing set.
- Predict the binary result and count the number of positives predictions (false positives), compute the percentage of it ($\frac{N^{\circ} \text{ of positives}}{10000}$) and store the value;
- Repeat the process 10 times, for each time storing the percentages;
- Lastly compute the average of the percentages: % of Blocked Accounts – Equation 4.5.

$$\%Blocked\ Accounts = \frac{\sum_{i=1}^{10} Percentage_i}{10} \quad (4.5)$$

Nevertheless, these metrics are only in regard to the model's performance. From a research stand of view, and in some industries such as health, it can be also important to understand how much each feature contributes to the overall prediction capability of the model. In some models such as Logistic Regression and Decision Trees, it is possible to estimate the coefficients associated with each variable. Nevertheless, in models such as Neural Networks, understating the impact of each feature on the overall performance can be quite complicated as a lot of transformations occur in the hidden layers. To access this problem there are some methods such as **Permutation Importance** (PIMP), proposed by Altmann et al. [59]. In this procedure, each feature is randomly shuffled N times, and for each time the model score is determined, leading to a vector with N importance measures for every feature. Afterward, the PIMP algorithm normalizes the biased measures based on permutation tests and returns significant P-values for each feature. Another technique to access feature importance is **Local Interpretable Model-Agnostic Explanations** (LIME) proposed by Ribeiro et al. [60] in 2016. For this the algorithm explains the predictions of

any classifier or regressor by identifying an interpretable model closest to it. So the goal is to minimize a measure of how unfaithful an interpretable model is to the actual model. Nonetheless, both techniques still have their disadvantages. PIMP does not perform well for features with high correlation, while LIME can be quite sensitive to a small change in input features. As a complement, it was decided to do a simple procedure where, similar to permutation importance, the model was run without one of the features at the time and the values were compared with the original ones. Once more, this type of procedure was its own drawback as it can not access the importance of features with high correlation.

Chapter 5

Results

5.1 Initial Analysis and Sampling Methods

As shown in Table 4.2 from section 4.1, one of the features is an array containing the text information. This is attained through the use of a transformer capable of analyzing text information. The initial array has a dimension of 384 column-wise, thus, it was necessary to further reduce this array by using reduction techniques such as PCA and Hierarchical Clustering. For both techniques, the final number of dimensions was achieved empirically. For PCA it was chosen a number of components that represented at least 50% of the total amount of variance in the text data. While in hierarchical clustering two alternatives were considered, the first based on the data's dendrogram – see Figure 5.1 and the latter was attained later by iteration in the final model.

- **PCA** – 20 principal components were considered, which represented about 50% of the total variance.
- **Hierarchical Clustering** – In Figure 5.1 it is possible to see that the cut associated with the largest gap “runs” to three vertical lines, therefore the first alternative considered 3 clusters, while the latter considered 100 clusters.

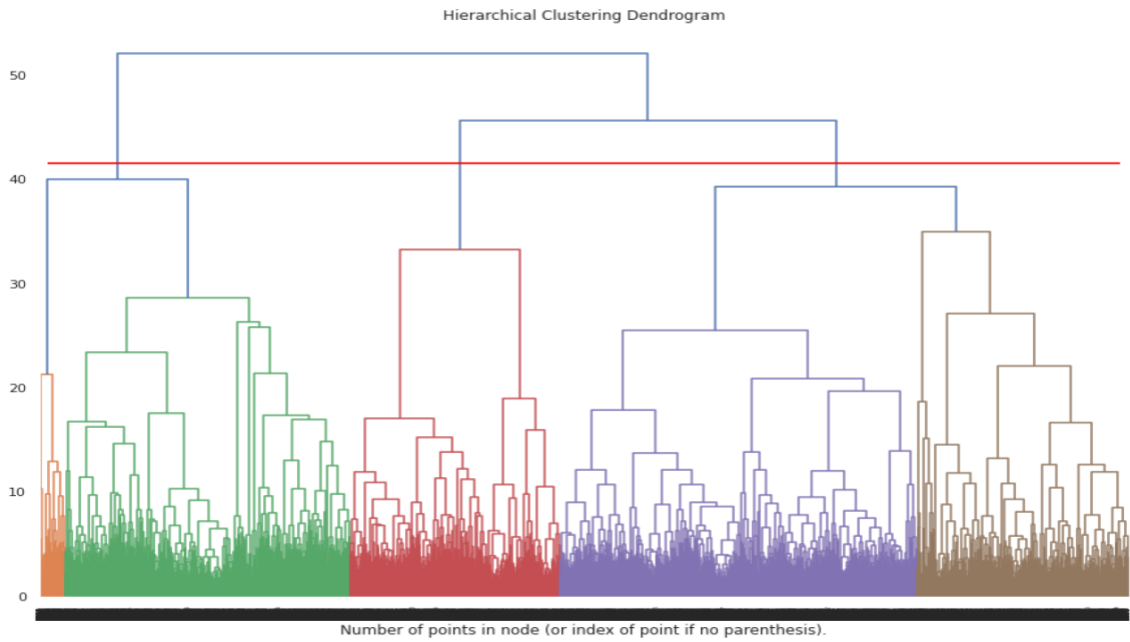


FIGURE 5.1: Hierarchical Clustering Dendrogram to text data.

Thus, taking into account the transformations carried on the unorganized data the final dataset is a mix of continuous and categorical variables, the amount of each type of feature is shown in Table 5.1.

Type of Feature	Number of Features
Continuous	7 + [3 or 20 or 100]
Binary	4
Categorical (More than two categories)	24

TABLE 5.1: Type of features present in the dataset. .

To better understand how the features are related between themselves and how the data is dispersed, correlation matrix was computed for all variables except for the text variable. It was found that the correlation between variables is very low in most variables thus the features appear to be independent. More information can be seen in Figure A.4. Additionally, a PCA analysis was carried out on the same features. From the biplot represented in Figure 5.2, it is clearly visible the impact on some variables such as the presence of certain fonts' families and the number of images in both the first and second component.

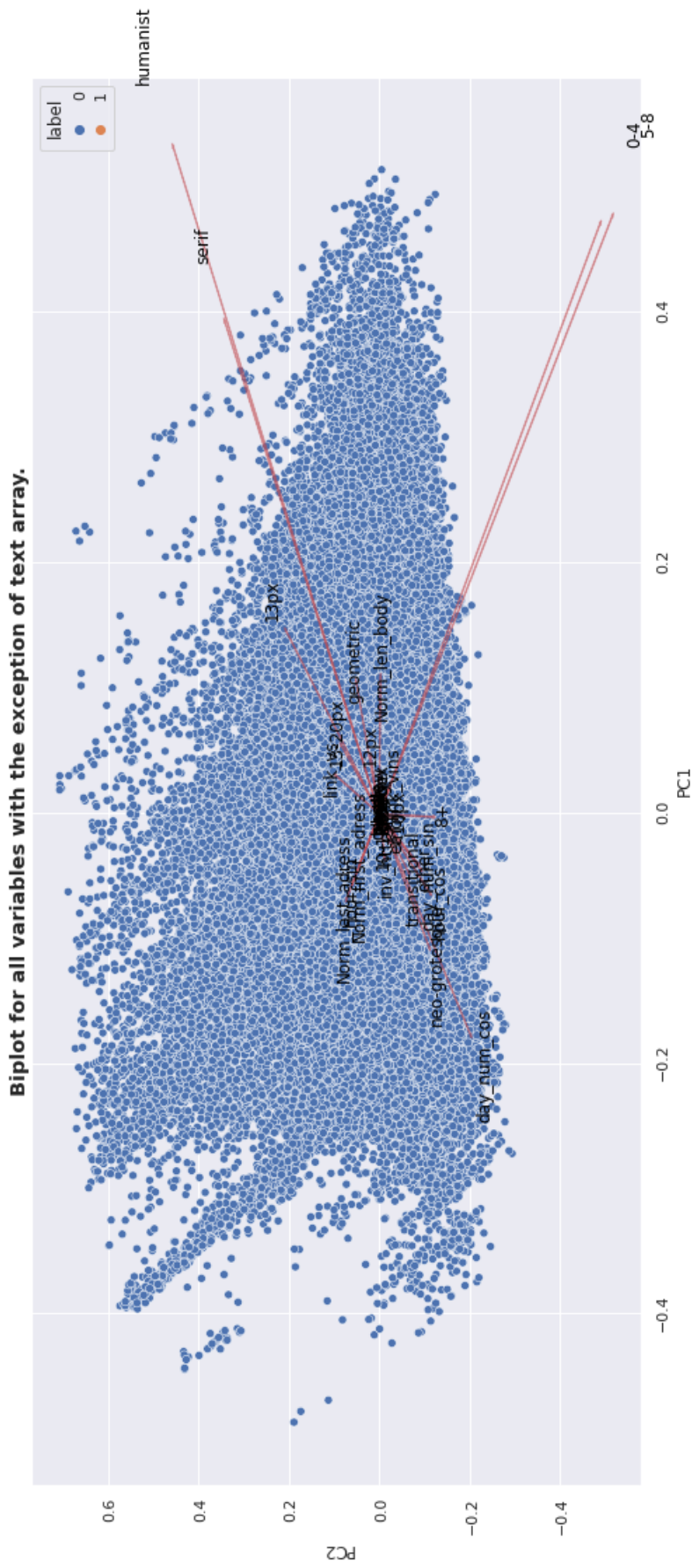


FIGURE 5.2: Biplot for all variables with exception of the text array.

As described in section 4.2, several sampling methods were conducted to create different training and testing sets, one of which was conducting a K-means analysis on the majority sample and retrieving samples from each cluster. One of the main steps of the K-means methodology is to choose the number of clusters – K . This can be previously determined (for example choosing the number of clusters to be equal to the number of phishing samples) or it can be achieved through empirical methods such as the elbow method or the silhouette method. Both methods were performed on data containing the text array sorted by the PCA technique and the hierarchical clustering technique. In both variants for the text sorting method the number K clusters determined was 4 by both the silhouette and elbow technique. In Figure 5.3 shows the attained clusters, with the first cluster having 37.6% of the total amount of observations, the second cluster having 8.4% of the total amount of observations, while the third and fourth clusters have 41.3% and 12.7% of the total amount of observations, respectively.

Additionally, following Rahman and Davis [22] approach, for each cluster two different sets were created, one containing the samples closest to the cluster's centroids and the other with samples closest to the phishing samples average from each cluster. Table 5.2 shows the different sets created. These eleven sets were combined with the different phishing ratios, so a total of five data-frames were created each containing eleven columns with the sample indexes. According to the phishing ratio the length of the data-frame changed: 428 samples for a set with 50% of phishing samples, 642 samples for a set with 33% of phishing samples, 1070 samples for a set with 20% of phishing samples, 4280 samples for a set with 5% of phishing samples, and 21400 samples for a set with 1% of phishing samples.



FIGURE 5.3: Majority Sample clusters with K=4.

Set's Name	Description
"random"	Samples retrieved in a random way.
"ind_agglo"	Samples closest to the phishing trough cosine similarity for data with text sorted by PCA.
"ind_pca"	Samples closest to the phishing trough cosine similarity for data with text sorted by Hierarchical Clustering.
"agglo_centroids_a"	Samples closest to the clusters' centroids for 4 clusters and data with text sorted by Hierarchical Clustering.
"agglo_closephish_a"	Samples closest to the phishing samples average for 4 clusters and data with text sorted by Hierarchical Clustering.
"agglo_centroids_phish"	Samples closest to the clusters' centroids for the number of clusters equal to the number of phishing samples and data with text sorted by Hierarchical Clustering.
"agglo_closephish_phish"	Samples closest to the phishing samples average for the number of clusters equal to the number of phishing samples and data with text sorted by Hierarchical Clustering.
"pca_centroids_a"	Samples closest to the clusters' centroids for 4 clusters and data with text sorted by PCA.
"pca_closephish_a"	Samples closest to the phishing samples average for 4 clusters and data with text sorted by PCA.
"pca_centroids_phish"	Samples closest to the clusters' centroids for the number of clusters equal to the number of phishing samples and data with text sorted by PCA.
"pca_closephish_phish"	Samples closest to the phishing samples average for the number of clusters equal to the number of phishing samples and data with text sorted by PCA.

TABLE 5.2: Sets Names and methodology description.

It was assured that both the test and training sets contained a 5% ratio of phishing samples. As stated in 4.3 an exhaustive grid search was performed with each one of the chosen models. The number of folders varied between two and three according to the training samples length, so training samples with less than 1000 samples the number of folders was two, while for the other cases the number of folders was three. Each training set had a size corresponding to 75% of the total amount of samples, while the testing set had 25% of the overall samples. To access each model's performance the accuracy, precision, and recall values were recorded and the boxplots for these metrics were plotted – Figure 5.4. Moreover, in Table A.2 it is possible to see the average values of each metric according to the model, phishing ratio, and sampling method.

The three ensemble models (Random Forest, AdaBoost, and Gradient Boost) had better values for all metrics, with the Random Forest model displaying the best values when it comes to precision with an average of about 94.25%. This is quite important as the precision is linked to the number of false positives inversely – see Equation 2.7. Thus higher values of precision indicate that the model has low false positives which in the problem's context is very important since is not desirable to block e-mails and accounts from real customers. When it comes to recall, the three ensemble models appear to have similar average values and distributions, while the decision tree appears to give the best results having the highest value average of 65.29%. Still, the average values for this metric appear to be lower than the desired values. The recall metric is linked to the number of false negatives – see Equation 2.8. Low recall values means that the model is failing in detecting the phishing samples. Therefore is important to have a model that can achieve a higher recall value without compromising the precision value.

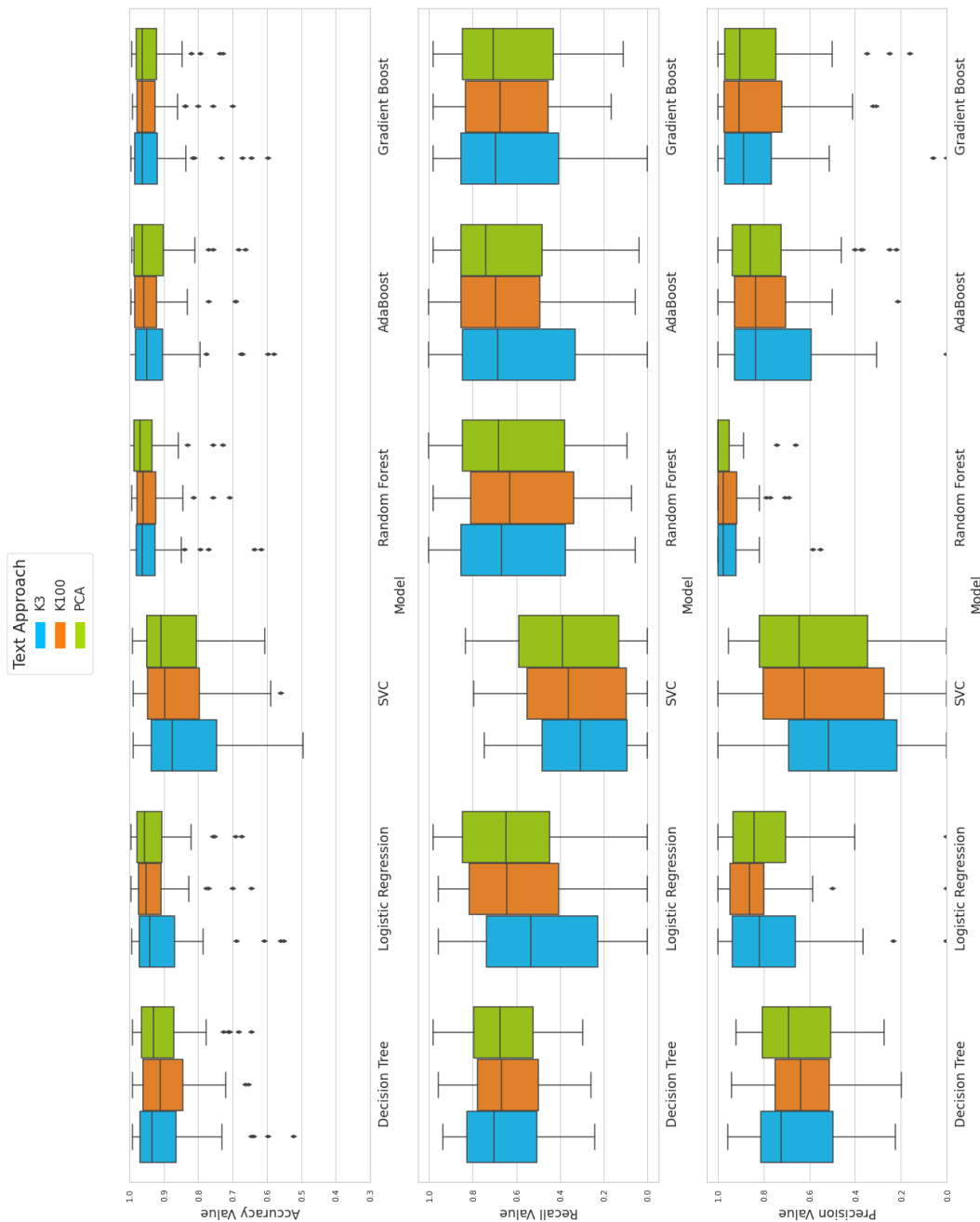


FIGURE 5.4: Metric Values for chosen Models.

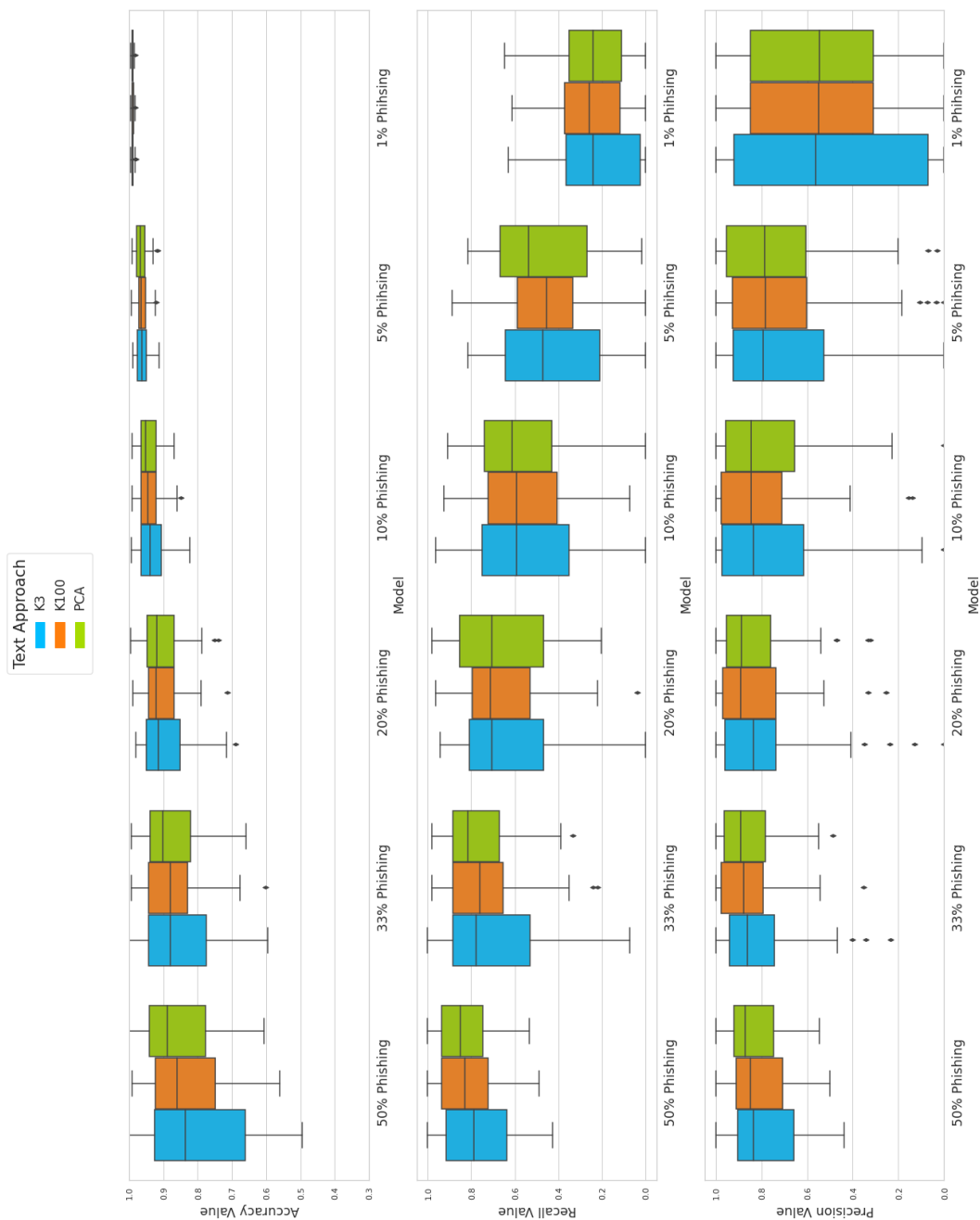


FIGURE 5.5: Metric Values for different phishing ratios

Taking the same approach for each **phishing ratio** the previously discussed metrics were recorded – see Figure 5.5 and Table A.2. Looking at accuracy and recall metrics the impact of different phishing percentages is quite visible, as more unbalanced models have significantly higher accuracy and lower recall. The probable explanation for this phenomenon is the increase of the model bias towards the negative labels (samples that are not phishing) since the true positives and true negatives are present in both parcels of the accuracy metric equation – see Equation 2.6). The higher the difference in ratio between the majority and minority class the hardest it will be to identify the minority class, thus the lower recall values. However, since most samples will be from the majority class the number of true negatives (samples that are not phishing evaluated as not phishing) will be quite high thus leading to higher values of accuracy. Regarding the precision metric, this suffers quite the impact when the phishing ratio decreases from 5% to 1%, with the average value ranging from 71.1% to 54.1%.

In Figure 5.6 it is plotted the average values for the chosen metrics according to the different **sampling methods**. For these approaches is visible that the sampling methods using the K-means methodology tend to display higher values for all metrics, especially the version that retrieves samples closest to the clusters' centroids. Interestingly the results from the methods that encompass the cosine similarity gave the worse results. The model appears to have difficulty in identifying the phishing samples but also is classifying a lot of "normal" samples as phishing, as the precision values are lower than the others with an overall average of about 51%. When comparing the results from the K-means methodologies with the number of clusters attained in an empirical manner (K=4) and the number of clusters equal to the number of phishing samples, the average values for all metrics are very similar having less than 2% difference for both the precision and recall.

Nevertheless, to confirm the hypothesis suggested by the initial analysis, it was important to access the actual differences between phishing ratios and sampling methods. Thus the final model was run fifty times with the different sets and seeds. The F1-scored for each iteration was saved to be later analyzed in One-way ANOVA, as explained in 4.2.2. The results are detailed in 5.3.

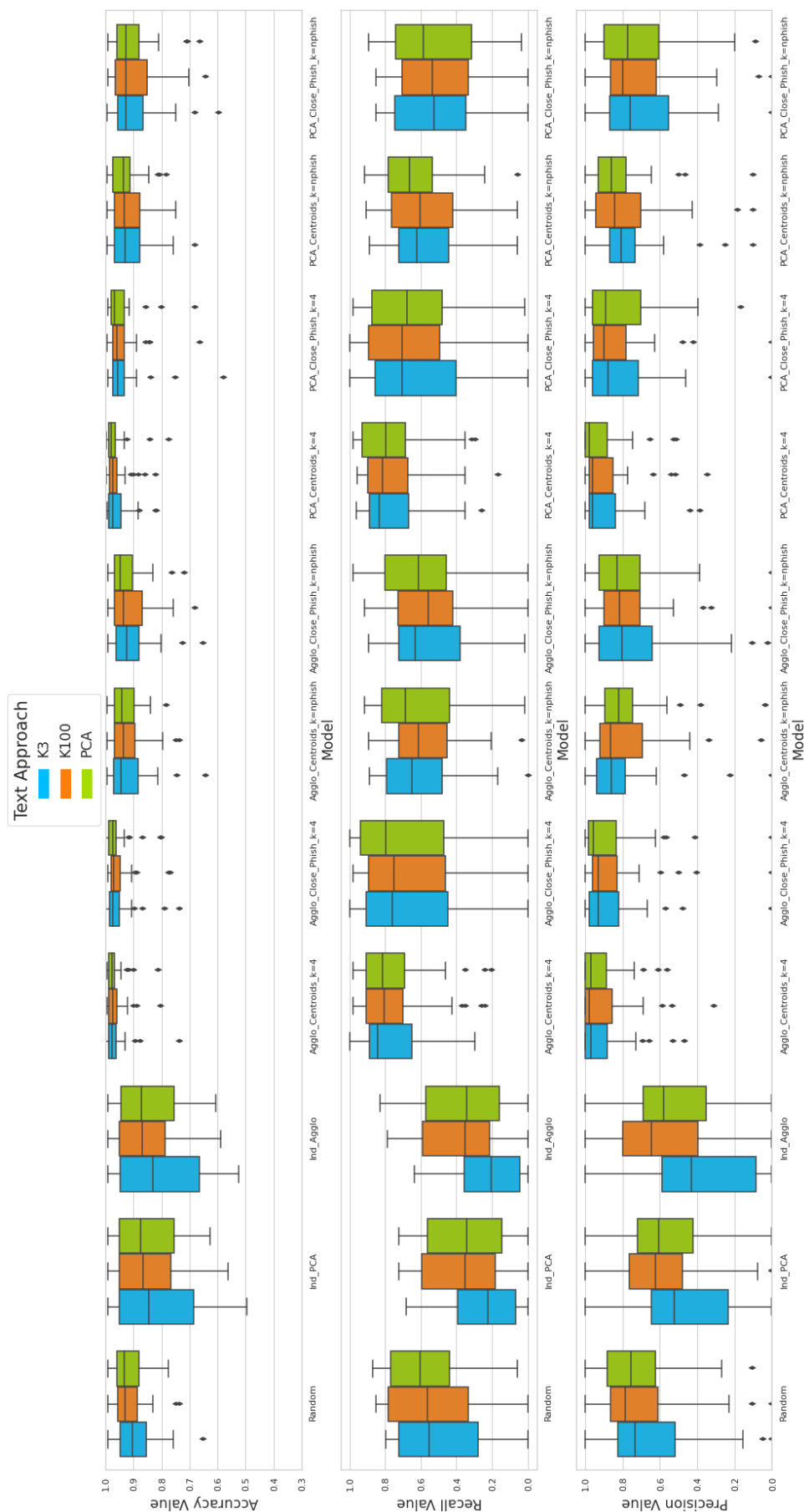


FIGURE 5.6: Metric Values for different sampling methodologies.

5.2 Models Results

Taking into account the initial models results, the Random Forest model displayed the best scores. However, is important to notice that this value is an average of all sets with different phishing percentages and sampling methods. Based on the results displayed in Figure 5.6 and Figure 5.5 it was chosen two different types of methodology and two different phishing ratios to perform an exhaustive grid search: "pca.centroids.phish" and "pca.centroids.a", 5% phishing ratio and 1% phishing ratio. Two exhaustive grid searches were executed one in order to the F1 metric and the other in other to the recall metric. According to the parameters list displayed in Table 4.4, for the random forest model the following values were considered for the grid search:

- **criterion:** *gini* and *entropy*, default = *gini*;
- **oob_score:** *True* and *False*, default = *False*;
- **min_samples_leaf:** 1, 8 and 10, default = 1;
- **max_features:** "sqrt", "log2", 0.1, 0.3, 0.5 and 0.7, default = "sqrt" ;

The best results for recall were achieved for the "pca.centroids.a" with a 5% phishing ratio, with the **criterion** parameter set to *gini*, the **oob_score** set to *True*, the **min_samples_leaf** with value 1 and the **max_features** set to the value 0.7 – see Table A.3. Still, this set of conditions also led to the highest amount of % of Blocked Accounts with a value of 25.63 %, which is undesirable. The values attained with the sampling method "pca.centroids.phish" for the % of Blocked Accounts were significantly lower, with the highest being close to 6%, however, the recall values for this set only achieve 44.44%, also an undesirable value. Thus it was decided to proceed with a neural network to try to achieve a higher recall metric without compromising the % of Blocked Accounts. It was also computed a random forest model with the "pca.centroids.phish" set to later compare the results and differences.

For the neural network, the first several trials were made with different text array sizes to determine the best K number for the hierarchical clustering. Additionally, the text array was also concatenated by the PCA technique with 20 principal components. It was considered one hidden layer and the number of neurons was attained by dividing the size of the input layer by two – see Table 5.3. The results show that the higher the

size of the array, the better the results for the recall metric. As the number of clusters decreases the information loss increase, however, a high dimension array for the text data may overpower the other variables.

Text Array Size	NN Layer Configuration	F1 results
3	(38,16,2)	78.73 %
20	(55,28,2)	80.00 %
60	(95,48,2)	76.70 %
100	(135,68,2)	84.31 %

TABLE 5.3: Results of F1 according to different text array sizes.

Afterward, taking into account a text dimension array of 100 and similar to the approach used for the random forest, an exhaustive grid search was executed for the two different sampling methods and phishing ratios – see Table A.4. Based on the attained results, and taking into account the blocked account percentages, it was chosen to compute a neural network using the dataset 5% phishing ratio plus "pca_centroids_phish" for the training and testing sets. It is also important to keep in note that the grid search was performed with cross-validation having three folders, thus the final results are likely to be higher since the model will have more samples to train. Even though the best results pointed to a neural network configuration with three hidden layers, when computing the final model the difference between three hidden layers and two hidden layers was very low. To have a simpler and faster model it was chosen a layer configuration with two hidden layers – (135,81,11,2). The activation function for the hidden layers was the "relu", the solver was the "adam", and the maximum number of iterations was set up to 500. The training set was composed of had 3050 negative samples and 160 positive samples (phishing observations), while the testing set had 1016 negative samples and 54 positive samples.

Figure 5.7 displays both **confusion matrix** for the training (orange color) and testing (blue color) sets. The neural network was able to correctly identify every sample in the training pool. When an algorithm tends to achieve "perfect" results in the training phase

it may indicate an over-fitting of the model to training observations and later fail to predict correctly in the testing set, since these are observations that the model has not seen. The test confusion matrix shows that the model fails to predict some observations, with 6 positive samples (11.11%) being classified as negative and 3 negative samples (0.30%) being classified as positive samples. Comparing the results with the random forest – see Figure A.8, is possible to see that, like the neural network, the random forest correctly predicted all the samples in the training pool. However, in the testing pool it failed to predict more than half of the positive samples (53,70%). Thus the random forest model appears to be biased towards the negative samples since it also does not have false positives, while the neural network appears to avoid this type of bias.

Accounting for the results from the confusion matrix, the ROC curves for the testing set for both models act accordingly to what is expected – see Figure A.9 and Figure A.10. Since the AUC (Area under the curve) displays the ability for the model to predict correctly the samples for both classes and the dataset is unbalanced, its normal for the curves for both models to not be so different because the amount of negatives is very high overpowering the bad results of the forest when predicting positive labels.

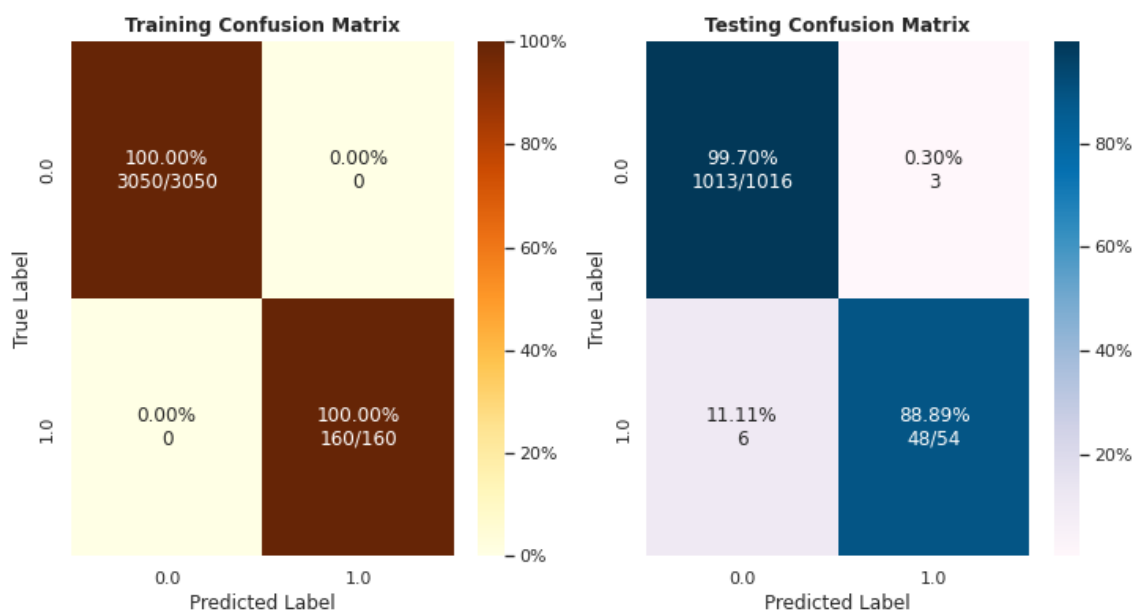


FIGURE 5.7: Confusion Matrix for the training and testing for the NN(135,81,11,2) model.

The **results for both models** are displayed in Table 5.4, the 95%CI were achieved with

a bootstrap technique explained on Section 4.4. At the end of the project, newer phishing samples were collected from the E-goi servers, these correspond to the “% New Right” and is a good way to see access the bias present on the sample used to generate the training and testing set. The results for the Neural Network are much better for detecting phishing samples. The Random Forest bias is quite visible as the value attained for the recall metric is quite different form the result given by the new phishing samples – only 6.66% of the new samples were classified as phishing. Taking into account all the constraints of the problem, from the unorganized data files to the small number of phishing observations, the **neural network** was able to achieve **good results**. Even though there is a small discrepancy between the results for the new phishing samples and the recall metric (difference of $\approx 5\%$), the model was able to detect more than 4/5 of the total phishing samples without blocking a high amount of customer’s accounts.

Model	Metric	Score
neural Network	Accuracy	[99.16 %, 99.20 %]
	Precision	[95.65 %, 96.25 %]
	Recall	[87.35 %, 87.75 %]
	F1	[91.38 %, 91.71 %]
	% Blocked Accounts	4.62 %
	% New Right	82.67 %
Random Forest	Accuracy	[97.42 %, 97.45 %]
	Precision	[99.07 %, 99.47 %]
	Recall	[49.26 %, 49.90 %]
	F1	[65.78 %, 66.37 %]
	% Blocked Accounts	0.35 %
	% New Right	6.67 %

TABLE 5.4: Final Models’ results.

Different techniques were tried to access the impact of each feature in the model. The results given by the permutation importance and the LIME approach suggested the text variables had the highest impact on the model. The hour for which the e-mail was sent and the length of the e-mail address also display a higher impact as well as the presence

of some font colors, namely black, yellow and white. Following the simpler methodology, the model was trained 50 times with different seeds without each feature at the time and the metric values were scored. Afterward, the results were compared with the base model (trained with all the features) to see if there were significant differences between both means at a significant level of 0.05. For ten features the p-value demonstrated a significant value, meaning that the average of the trained model without that feature is statistically different from the average of the model trained with all the features. These variables were regarding the presence of visible URL links throughout the text, the e-mail address length, the hour at which the e-mail was sent, the day of the week at which the e-mail was sent, and the information present in the e-mail (text features) and the presence of the font color green and the font size over 20 pixels.

5.3 ANOVA Results

To evaluate the difference among *phishing ratios* and sampling methods, for each set, the NN model was run fifty times and the F1-score was recorded. Afterward, a One-way ANOVA was carried out with a significance level (α) of 0.05. The null hypothesis states that the means of all groups are equal. The results for the p-values, averages, and standard deviations are shown in Table 4.4). Since the p-values were significantly lower than the significant level there is enough statistical evidence to reject the null hypothesis. Thus, for either variation, it appears to be at least a group performing differently than the others.

Variable	Name	F1-score	P-value
phishing Ratio	50% phishing	90.90 ± 2.90 %	1.42 e ⁻¹⁴²
	33% phishing	90.84 ± 2.85 %	
	20% phishing	86.47 ± 3.90 %	
	10% phishing	85.98 ± 3.36 %	
	5% phishing	84.93 ± 3.56 %	
	1% phishing	77.36 ± 10.3 %	
Sampling Methods	"random"	64.58 ± 5.60 %	0.0
	"ind_agglo"	55.27 ± 6.07 %	
	"ind_pca"	54.74 ± 5.99 %	
	"agglo_centroids_a"	83.05 ± 4.21 %	
	"agglo_closephish_a"	75.62 ± 5.30 %	
	"agglo_centroids_phish"	77.45 ± 4.61 %	
	"agglo_closephish_phish"	74.30 ± 4.68 %	
	"pca_centroids_a"	84.32 ± 3.97 %	
	"pca_closephish_a"	70.00 ± 4.89 %	
	"pca_centroids_phish"	78.60 ± 4.99 %	
	"pca_closephish_phish"	69.61 ± 5.12 %	

TABLE 5.5: F1 average results for different phishing ratios and sampling methods.

Taking into account the average value for the F1-score and the standard deviation presented in Table 5.5), in regards to the **phishing ratio**, the difference between the 50% ratio and the 30% ratio is not significant. In addition, the values for 20%, 10%, and 5% phishing ratios are quite similar. Only the 1% percentage displays a heavier drop in the F1-score, also presenting a higher standard deviation. Looking at the ratios of normal to phishing samples necessary to achieve each percentage, 20% corresponds to a ratio of 1:4; 10% to a ratio of 1:9; 5% to a ratio of 1:19; and 1% to a ratio of 1:99. Thus the increase of samples necessary in a case with 1% phishing is significantly higher which may difficult the model's learning job.

When analyzing the **Sampling Methods**, it is observed the difference between the sampling methods evolving clustering to the others, the first displaying higher F1-scores. When comparing the number of clusters used for the clustering methods, the ones in which the number was attained through an empirical analysis ($k=4$) registered higher values. Moreover, the samples collected near the centroids of each cluster also gave higher results: 83.05% compared with 75.462% and 77.45% compared with 74.30% for the method with the text array sorted by hierarchical clustering. For the method with the text array sorted by PCA: 84.32% compared with 70.00% and 78.60% compared with 69.61%.

Looking at Table 5.6, it is possible to see a significant difference in the % of Blocked Accounts between the clustering approaches with the different numbers of clusters (k). The approach considering four groups of clusters displayed higher values, which is not desirable. This may be linked to the clusters' density. Since the approaches are selecting either samples closest to the clusters' centroids or closest to the phishing samples, several observations far apart from these two points are not being considered. Thus a lack of variance is present in training and testing sets, explaining the increase in false positives. For the method with k clusters equal to the number of phishing samples, the K-means algorithm forces the centroids to disperse thus increasing variance and reducing bias. This explains why the variant with the higher amount of K clusters performs better.

Sampling Methods	% Blocked Accounts
"random"	0.88 ± 0.11 %
"ind_pca"	0.29 ± 0.10 %
"ind_agglo"	0.48 ± 0.15 %
"agglo_centroids_a"	35.12 ± 2.42 %
"agglo_closephish_a"	38.47 ± 3.29 %
"agglo_centroids_phish"	7.08 ± 0.73 %
"agglo_closephish_phish"	7.57 ± 0.15 %
"pca_centroids_a"	29.66 ± 1.76 %
"pca_closephish_a"	28.00 ± 4.05 %
"pca_centroids_phish"	4.65 ± 0.42 %
"pca_closephish_phish"	5.05 ± 0.11 %

TABLE 5.6: Results of F1 according to different text array sizes.

Chapter 6

Conclusions

This project proposed as the final model a neural network with two hidden layers that was able to achieve good results in detecting mass spread phishing e-mails ($\approx 88\%$) without blocking a high amount of customers' accounts (less than $\approx 5\%$). The features' impact on the model was also evaluated. The main factors contributing to an e-mail prediction as phishing are the message's content, the length of the e-mail address, and the hour at which the e-mail was sent. Still it is quite difficult to fully understand the true impact of the features in a neural network. Thus, this subject has been thoroughly studied and several methodologies are already being used to assess this issue.

A key aspect of a model with good performance is the **data quality**. Thus feature processing and data mining are crucial in a ML project, especially when dealing with **unbalanced data-sets**. For this thematic, ensemble models, as expected, perform much better than single models. However, it is necessary to see the "degree" of unbalance, as shown in 5.3. There can be a balance between the **ratio of majority class to minority class** that allows the model to exhibit a good performance and keep as much as information possible from the majority class. In this project the 1:19 ratio (5% phishing samples) yielded good results which were not so different from the 1:9 (10% phishing samples) and 1:4 (20% phishing samples) ratios. It is noted, however, that a higher ratio – 1:99 (1% phishing samples) – led to a significant drop in the model's performance. In manners of **under-sampling**, there is a significant difference between the different techniques

with techniques based on clustering demonstrating better results. However, these types of techniques can be prone to increase the bias present in the collected sample. This is specially true if the samples are being collected based on the distance between each observation and cluster centroids. A possibility to reduce this bias is to use a higher number of clusters, as demonstrated by the results attained for the techniques that used a number of clusters equal to the number of phishing samples. Another possibility could be to collect random samples for each cluster and not base the search on proximity to certain coordinates such as the centroids' coordinates, something that can be explored in future works.

6.1 Limitations and Future Work

- **Feature Extraction:** Extraction features from the raw *EML* files was one of the biggest limitations, as shown in 4.2. Since the file content can be generated in different platforms besides E-goi a lot of formatting errors can occur. The typical packages to work with regular *EML* files do not work for this particular type of e-mail, hindering the process of feature extraction;
- **Unbalanced Dataset:** Another main limitation is the data unbalance between the majority class (non phishing e-mails) and the minority class (phishing e-mails). Even though different under-sampling techniques were used to minimize the loss of information, this is a constraint to be recognized. Hence, it may be necessary to monitor the model's results after implementation and re-train it if necessary.

Based on the knowledge acquired throughout this work there are some future projects that may be useful for E-goi:

- **Fake user detection:** Similar to the work developed in this project, a model capable of detecting fake user profiles based on their e-mail address could be helpful. This could block accounts prior to their enrollment on the E-goi's platform, and so prevent the spread of phishing campaigns;

- **Anti-SPAM recommendation model:** One of the problems for a common user of the E-goi platform is their campaigns and flyers being marked as SPAM or potential phishing. Based on the knowledge acquired and some tools created throughout this work, such as the feature extraction for the *.eml* files. A future project could be the development of a model capable of analyzing potential campaigns, newsletters, etc., giving recommendations for these not to be so easily blocked by the automatic SPAM filters in most e-mail services.

Appendix A

Appendix

This chapter presents additional content such as schemes and tables that may be consulted for further information.

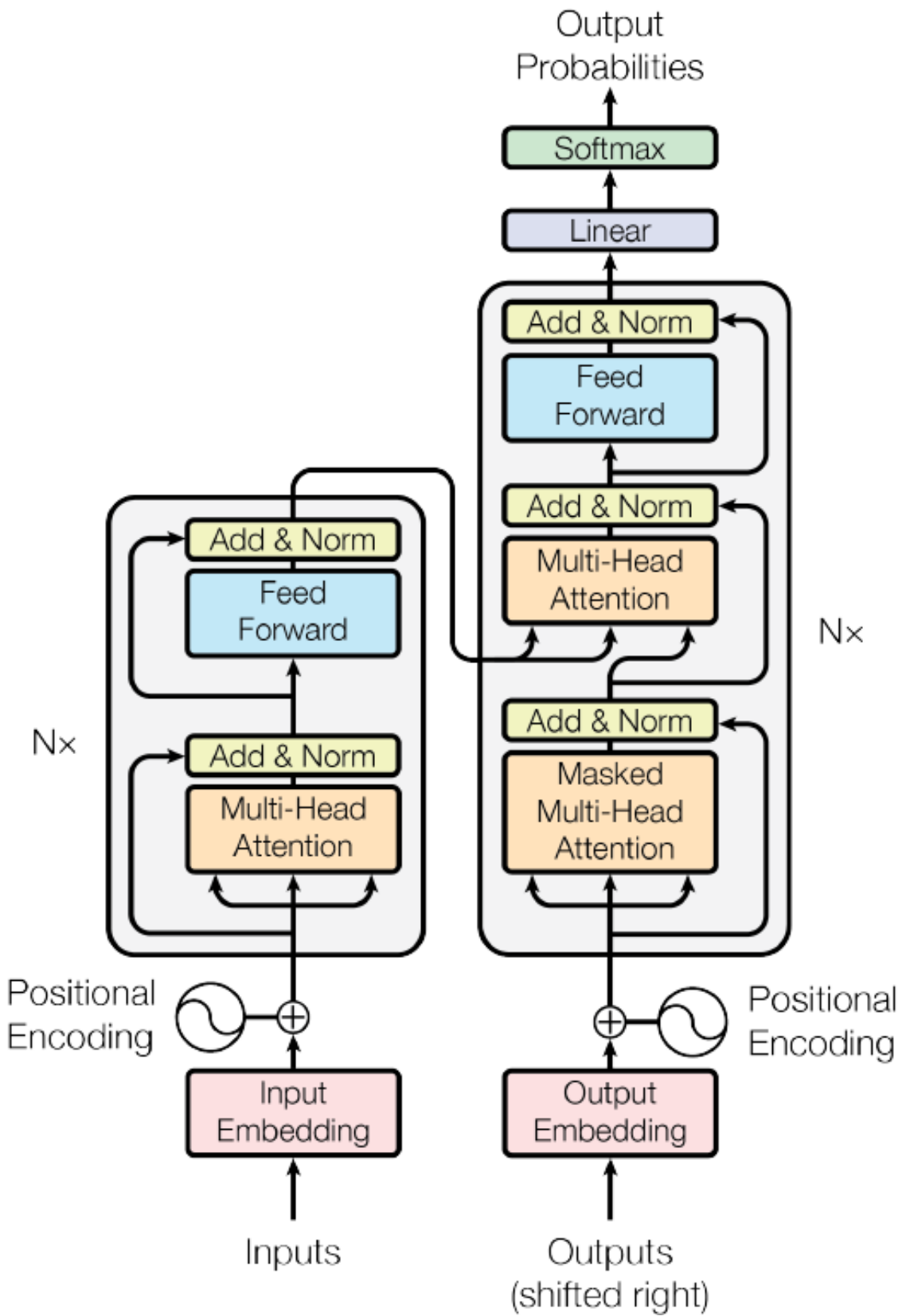


FIGURE A.1: Trasformer Architecture (adapted from Vaswani et al. [32]).

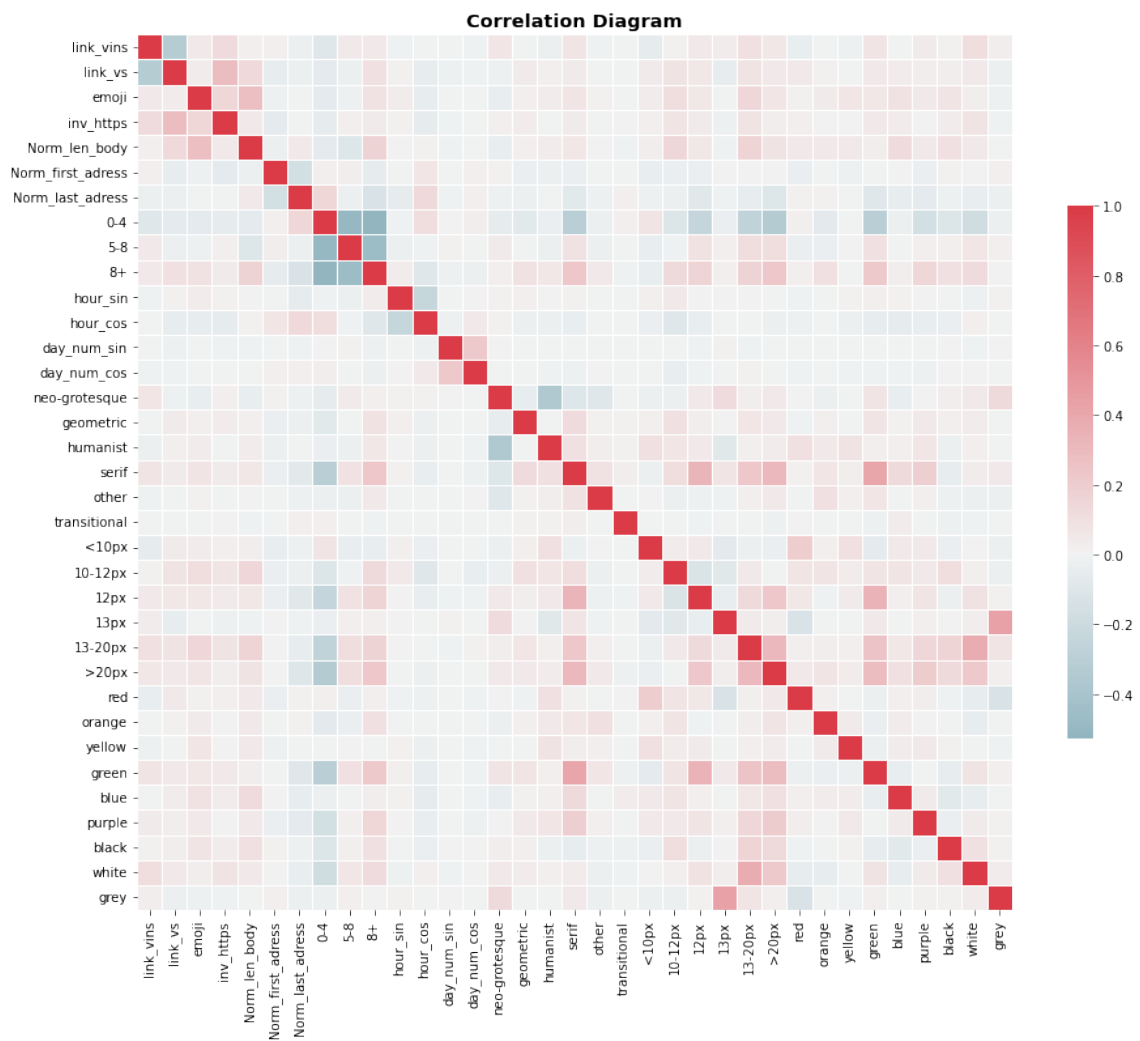


FIGURE A.4: Correlation Diagram for all variables except text array.

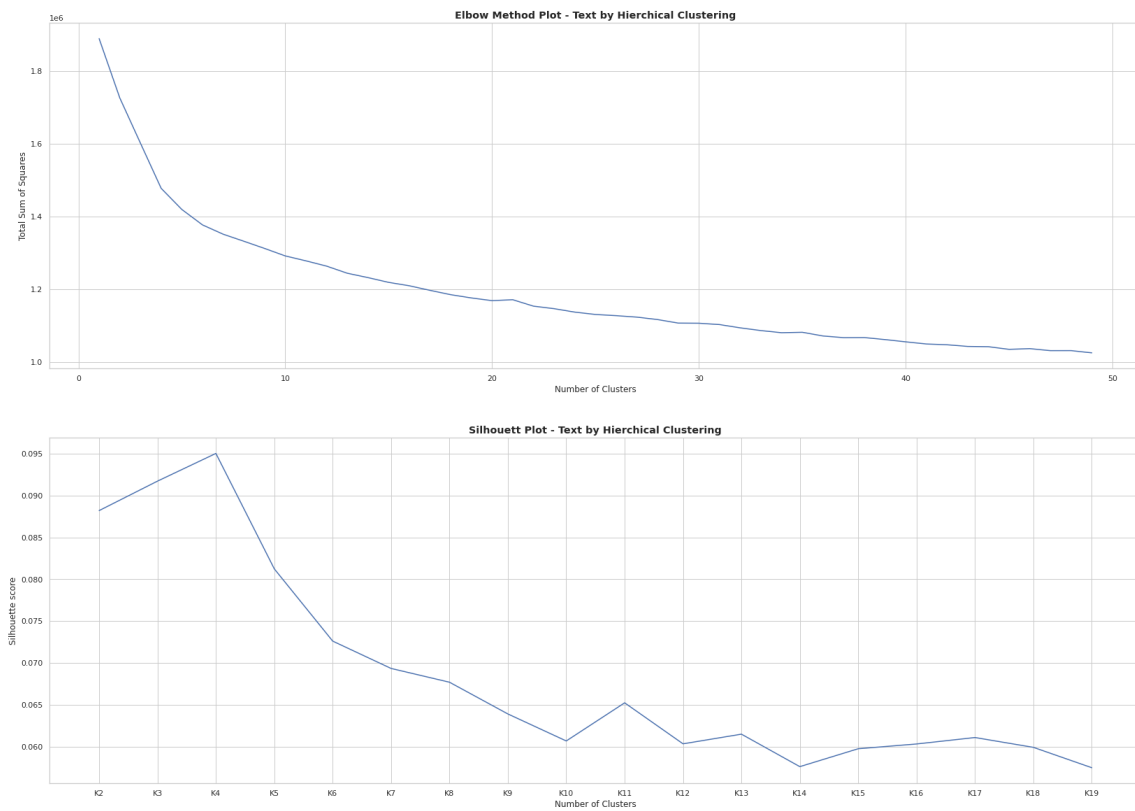


FIGURE A.5: Elbow and Silhouette Graphs for data with text sorted by Hierarchical Clustering.

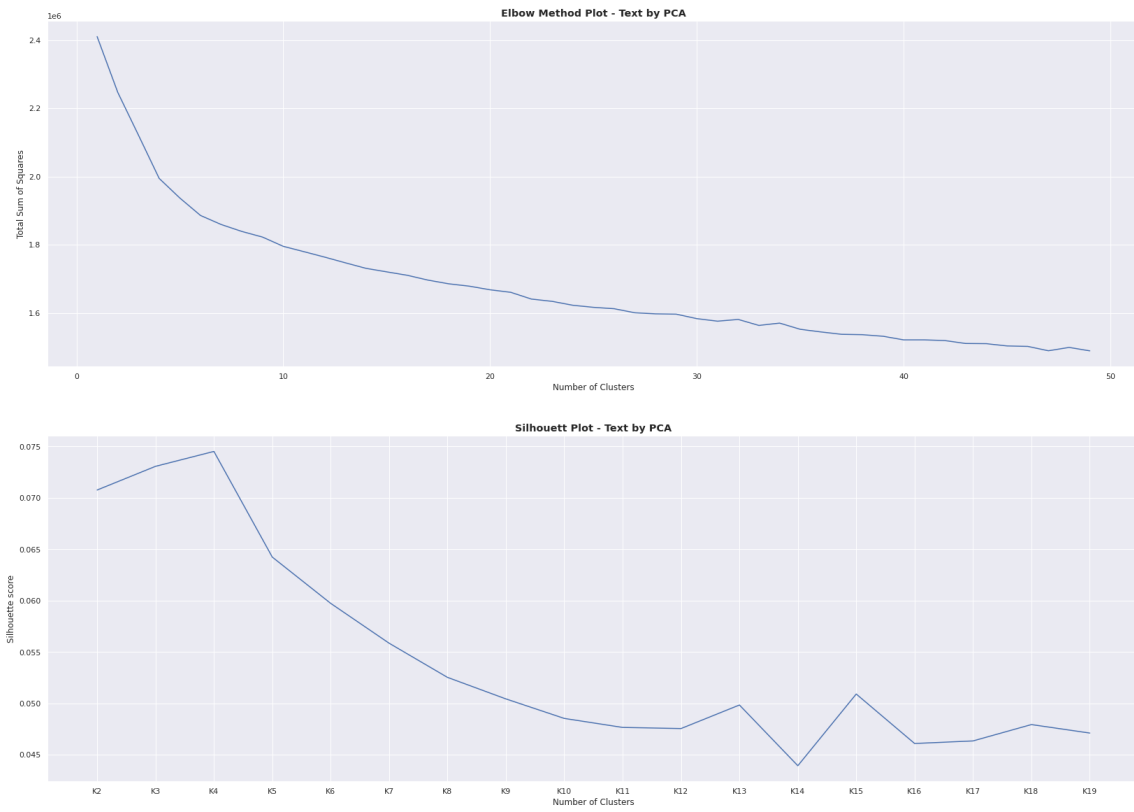


FIGURE A.6: Elbow and Silhouette Graphs for data with text sorted by PCA.

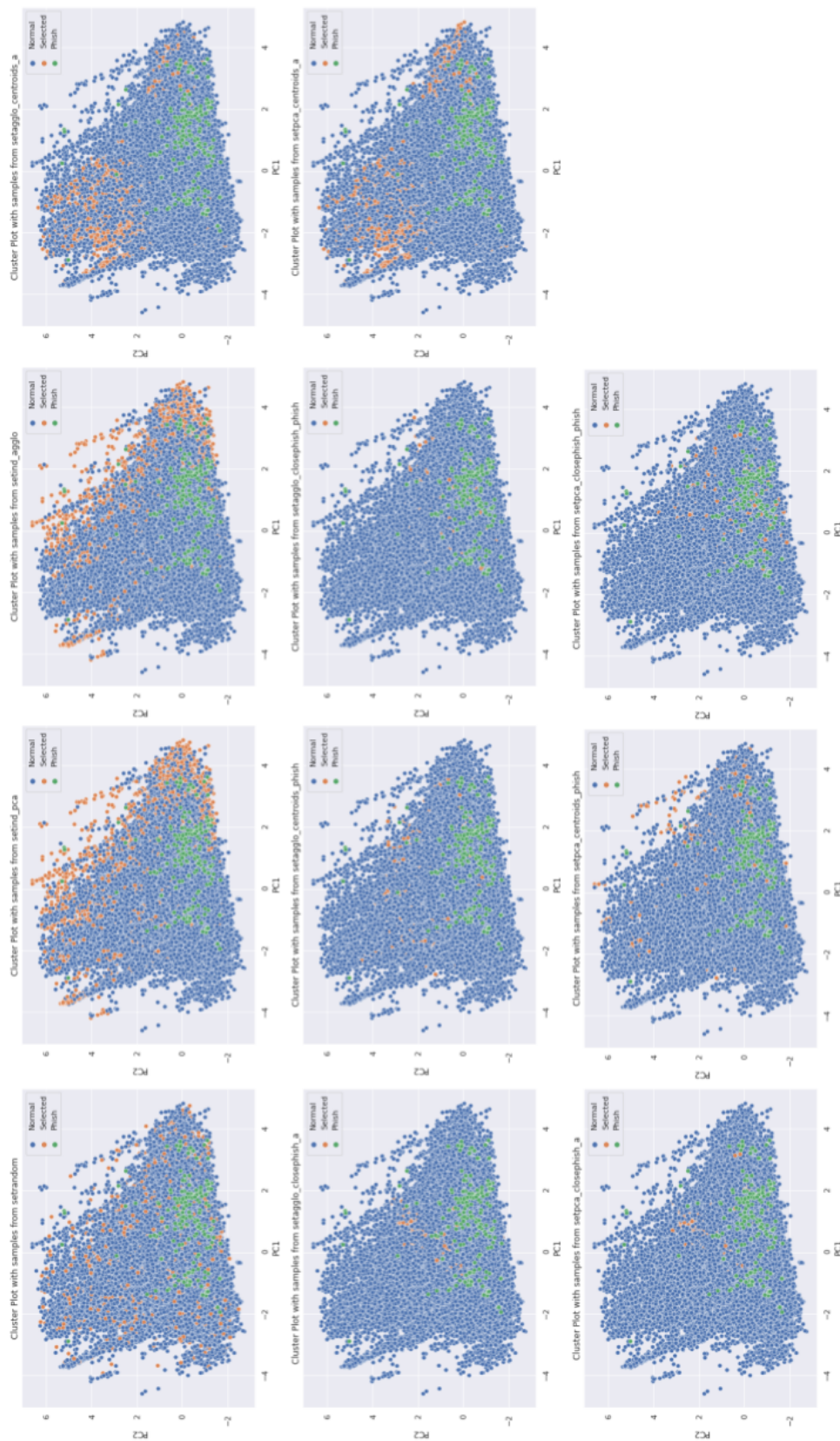


FIGURE A.7: Different Sampling Methodology's Samples Location.

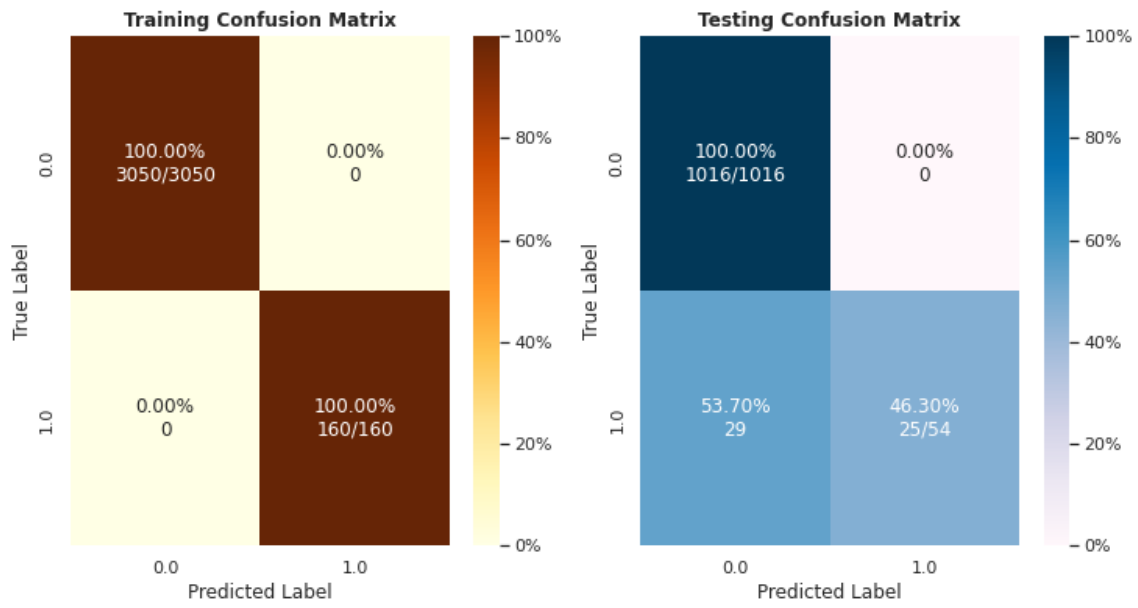


FIGURE A.8: Confusion Matrix for the training and testing for the Random Forest Model.

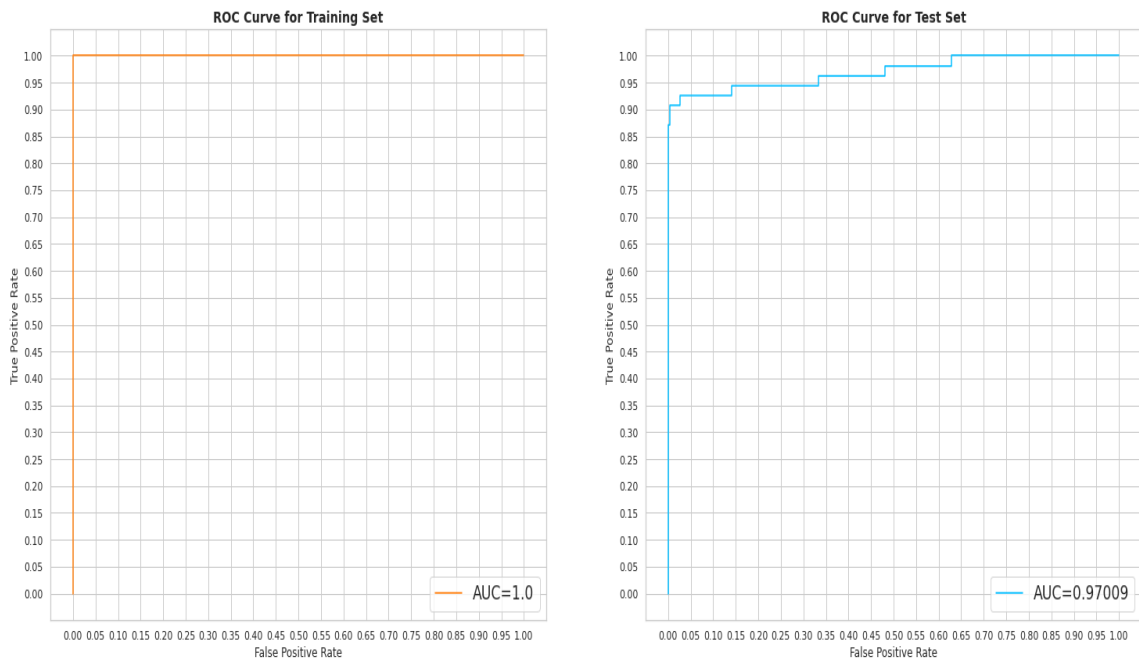


FIGURE A.9: ROC curves for the training and testing for the NN(135,81,11,2) model.

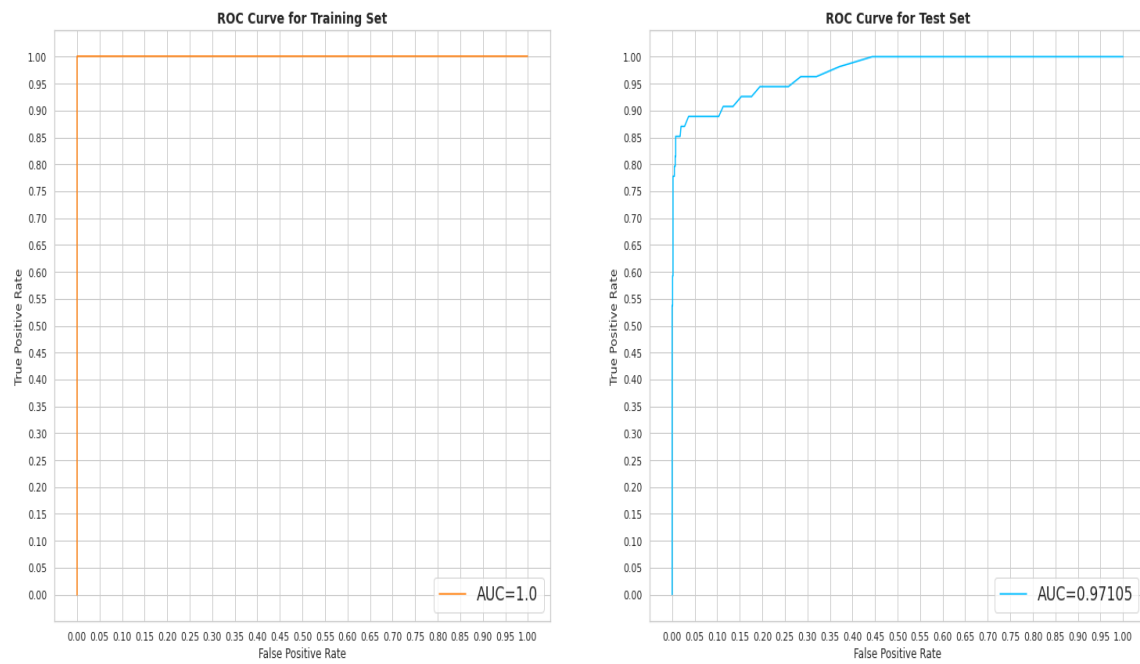


FIGURE A.10: ROC curves for the training and testing for the Random Forest model.

Name	Mean Accuracy	Mean Precision	Mean Recall
Decision Tree	89.91%	64.20%	65.29%
Logistic Regression	92.15%	77.45%	54.96%
SVC	85.72%	52.20%	34.32%
Random Forest	94.25%	95.27%	60.34%
AdaBoost	93.04%	78.46%	63.24%
Gradient Boost	93.96%	83.10%	63.66%
50% <i>Phishing</i>	82.79%	81.05%	80.28%
33% <i>Phishing</i>	86.67%	84.20%	73.20%
20% <i>Phishing</i>	90.28%	82.35%	64.77%
10% <i>Phishing</i>	93.97%	77.82%	55.13%
5% <i>Phishing</i>	96.30%	71.10%	44.44%
1% <i>Phishing</i>	98.98%	54.12%	23.97%
""random""	90.77%	69.15%	52.88%
""ind_agglo""	83.65%	50.44%	31.83%
""ind_pca""	83.66%	53.42%	32.39%
""agglo_centroids_a""	96.47%	90.63%	75.19%
""agglo_closephish_a""	95.58%	85.14%	67.73%
""agglo_centroids_phish""	92.33%	79.54%	60.74%
""agglo_closephish_phish""	91.57%	75.78%	55.66%
""pca_centroids_a""	96.19%	90.03%	75.52%
""pca_closephish_a""	94.12%	81.36%	63.62%
""pca_centroids_phish""	91.95%	78.83%	59.40%
""pca_closephish_phish""	90.18%	71.87%	51.65%

TABLE A.2: Accuracy, Precision and Recall Values for different models, *phishing* ratios and sampling methods.

Sampling Method	Phishing Ratio	Best Parameters	Metrics Results	% Blocked accounts
Grid Search in order to F1 metric				
"pca-centroids_a"	5%	max_features = 0.3, obb_score = True	Precision = 97.50 % Recall = 72.22 %	25.63 %
	1%	criterion = gini, max_features = "sqrt", obb_score = True, min_samples_leaf = 1	Precision = 96 % Recall = 44.44 %	23.35 %
"pca.centroids_phish"	5%	criterion = <i>entropy</i> , obb_score = True	Precision = 96 % Recall = 44.44 %	5.2 %
	1%	criterion = gini, max_features = "sqrt", obb_score = True, min_samples_leaf = 1	Precision = 96 % Recall = 44.44 %	0.32 %
Grid Search in order to recall metric				
"pca-centroids_a"	5%	max_features = 0.7, obb_score = True	Precision = 96.00 % Recall = 73.00 %	25.64 %
	1%	criterion = gini, max_features = "sqrt", obb_score = True, min_samples_leaf = 1	Precision = 96 % Recall = 44.44 %	23.35 %
"pca.centroids_phish"	5%	criterion = gini, max_features = "sqrt", obb_score = True, min_samples_leaf = 1	Precision = 96 % Recall = 44.44 %	6.67 %
	1%	criterion = gini, max_features = "sqrt", obb_score = True, min_samples_leaf = 1	Precision = 96 % Recall = 44.44 %	0.32 %

TABLE A.3: Grid search results for Random Forest.

Sampling Method	Phishing Ratio	Best Parameters	Metrics Results	% Blocked accounts
Grid Search in order to F1 metric				
"pca_ _centroids_a"	5%	hidden_layers_sizes = (135,81,35,11,2) max_iter = 500 activation = "tanh"	Precision = 92.16 % Recall = 87.03 %	38.34 %
	1%	hidden_layers_sizes = (135,68,2)	Precision = 97.37 % Recall = 68.51 %	11.065 %
"pca_centroids _phish"	5%	hidden_layers_sizes = (135,81,35,11,2) max_iter = 500	Precision = 91.11 % Recall = 75.93 %	5.00 %
	1%	hidden_layers_sizes = (135,68,2)	Precision = 84.09 % Recall = 68.52 %	1.43 %
Grid Search in order to recall metric				
"pca_ _centroids_a"	5%	hidden_layers_sizes = (135,81,35,11,2) max_iter = 500 activation = "tanh"	Precision = 92.16 % Precision = 87.04 %	38.25 %
	1%	hidden_layers_sizes = (135,68,2)	Precision = 97.37 % Recall = 68.52 %	11.14 %
"pca_centroids _phish"	5%	hidden_layers_sizes = (135,81,35,11,2) max_iter = 500 activation = "tanh"	Precision = 0.82 % Recall = 0.77 %	5.54 %
	1%	hidden_layers_sizes = (135,68,2)	Precision = 84.09 % Recall = 68.52 %	1.45 %

TABLE A.4: Grid search results for Neuronal Network.

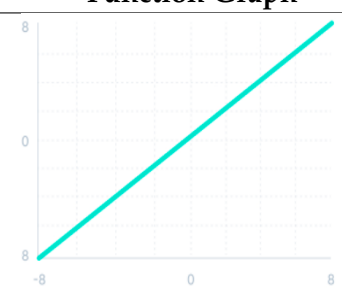
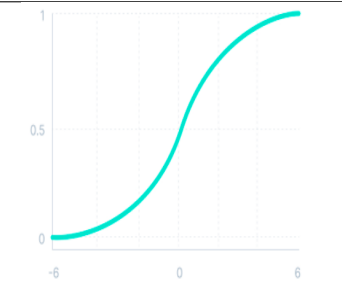
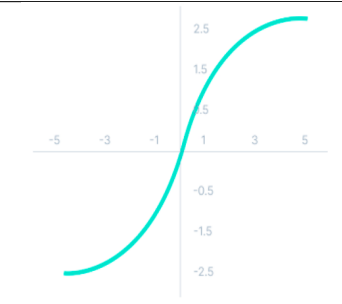
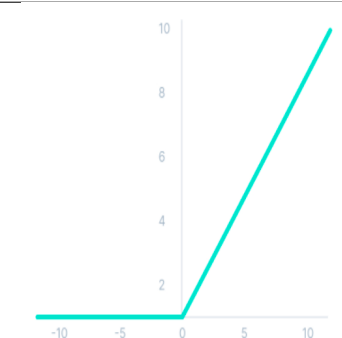
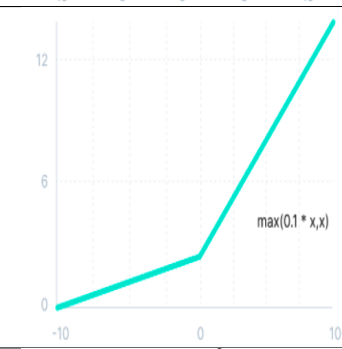
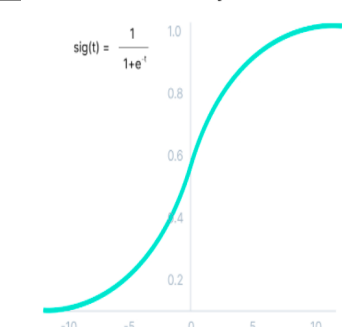
Function Name	Function Formula	Function Graph
Linear (Identity)	$\sigma(x) = x$	
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	
Tanh	$\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	
Rectified Linear ReLU	$\sigma(x) = \max(0, x)$	
Leaky ReLU	$\sigma(x) = \begin{cases} x, & x \geq 0 \\ ax, & x < 0 \end{cases}$	
Softmax	$s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$	

TABLE A.1: Activation functions

Bibliography

- [1] RiskBased SECURITY, “2020 Q3 Report Data Breach QuickView,” Tech. Rep., 2020. [Cited on page 1.]
- [2] R. Dhamija, J. D. Tygar, and M. Hearst, “Why phishing works,” in *Proceedings of the SIGCHI conference on Human Factors in computing systems*, 2006, pp. 581–590. [Cited on page 1.]
- [3] J. N. Tavares, “COMPUTATIONAL STATISTICS AND DATA ANALYSIS: Module 1: Overview. dynamic programming.” Faculty of Science of University of Porto, 2021. [Cited on pages 5 and 16.]
- [4] A. Gron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 1st ed. O’Reilly Media, Inc., 2017. [Cited on pages 5, 9, 10, and 11.]
- [5] C. O. S. Sorzano, J. Vargas, and A. P. Montano, “A survey of dimensionality reduction techniques,” *arXiv*, 2014. [Cited on page 6.]
- [6] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009, vol. 2. [Cited on pages 6, 7, and 10.]
- [7] A. Likas, N. Vlassis, and J. J. Verbeek, “The global k-means clustering algorithm,” *Pattern Recognition*, vol. 36, no. 2, pp. 451–461, 2003. [Cited on page 7.]
- [8] P. J. Rousseeuw, “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,” *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987. [Cited on pages 7 and 43.]

- [9] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967. [Cited on page 8.]
- [10] E. Carrizosa and D. R. Morales, "Supervised classification and mathematical optimization," *Computers & Operations Research*, vol. 40, no. 1, pp. 150–165, 2013. [Cited on page 10.]
- [11] L. Gonçalves, A. Subtil, M. R. Oliveira, and P. de Zea Bermudez, "ROC curve estimation: An overview," *REVSTAT-Statistical journal*, vol. 12, no. 1, pp. 1–20, 2014. [Cited on pages xiii, 11, and 12.]
- [12] E. Stevens, L. Antiga, and T. Viehmann, *Deep Learning with PyTorch*, 2020. [Online]. Available: <https://pytorch.org/assets/deep-learning/Deep-Learning-with-PyTorch.pdf> [Cited on pages xiii, 12, and 13.]
- [13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. [Cited on pages xiii, 12, 13, 14, and 15.]
- [14] S. Cristina. (2021) Calculus in Action: Neural Networks. [Online]. Available: <https://machinelearningmastery.com/calculus-in-action-neural-networks/> [Cited on pages xiii, 14, and 15.]
- [15] A. Bhardwaj, W. Di, and J. Wei, *Deep Learning Essentials: Your hands-on guide to the fundamentals of deep learning and neural network modeling*. Packt Publishing Ltd, 2018. [Cited on pages xiii, 14, 15, 16, and 19.]
- [16] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv*, 2016. [Cited on pages 17 and 18.]
- [17] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE access*, vol. 7, pp. 53 040–53 065, 2019. [Cited on pages 18 and 19.]
- [18] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv*, 2015. [Cited on pages xiii and 19.]
- [19] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert systems with applications*, vol. 73, pp. 220–239, 2017. [Cited on pages 20 and 21.]

- [20] W.-C. Lin, C.-F. Tsai, Y.-H. Hu, and J.-S. Jhang, "Clustering-based undersampling in class-imbalanced data," *Information Sciences*, vol. 409, pp. 17–26, 2017. [Cited on pages 20, 22, 43, and 44.]
- [21] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 539–550, 2008. [Cited on page 20.]
- [22] M. M. Rahman and D. Davis, "Cluster based under-sampling for unbalanced cardiovascular data," in *Proceedings of the World Congress on Engineering*, vol. 3, 2013, pp. 3–5. [Cited on pages 20, 22, 43, and 53.]
- [23] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002. [Cited on page 21.]
- [24] S.-J. Yen and Y.-S. Lee, "Cluster-based under-sampling approaches for imbalanced data distributions," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5718–5727, 2009. [Cited on page 22.]
- [25] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, no. 2, pp. 241–258, 2020. [Cited on pages 23 and 24.]
- [26] G. Kunapuli, *Ensemble Methods for Machine Learning*. Manning, 2022. [Online]. Available: <https://books.google.pt/books?id=wXGazgEACAAJ> [Cited on pages xiii, 23, 24, and 25.]
- [27] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009. [Cited on page 24.]
- [28] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, "Natural language processing: an introduction," *Journal of the American Medical Informatics Association*, vol. 18, no. 5, pp. 544–551, 09 2011. [Cited on page 25.]
- [29] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: State of the art, current trends and challenges," *Multimedia Tools and Applications*, 08 2017. [Cited on pages 25, 26, and 27.]
- [30] E. D. Liddy, "Natural language processing," 2001. [Cited on page 25.]

- [31] K. Chowdhary, "Natural language processing," *Fundamentals of artificial intelligence*, pp. 603–649, 2020. [Cited on page 26.]
- [32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *Advances in neural information processing systems*, 2017. [Online]. Available: <http://arxiv.org/abs/1706.03762> [Cited on pages xiii, 26, 27, and 74.]
- [33] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *ArXiv*, vol. abs/1810.04805, 2019. [Cited on pages 26 and 28.]
- [34] D. Rothman, *Transformers for Natural Language Processing: Build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more*. Packt Publishing, 2021. [Online]. Available: <https://books.google.pt/books?id=Cr0YEAAAQBAJ> [Cited on pages 26 and 27.]
- [35] "Transformers," <https://huggingface.co/docs/transformers/index>, accessed: 2022-03-08. [Cited on page 28.]
- [36] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," *CoRR*, vol. abs/1908.10084, 2019. [Online]. Available: <http://arxiv.org/abs/1908.10084> [Cited on pages 28 and 43.]
- [37] J. Hong, "The state of phishing attacks," *Communications of the ACM*, vol. 55, no. 1, pp. 74–81, 2012. [Cited on pages 29 and 33.]
- [38] W. Kim, O.-R. Jeong, C. Kim, and J. So, "The dark side of the internet: Attacks, costs and responses," *Information Systems*, vol. 36, no. 3, pp. 675–705, 2011. [Cited on pages 29 and 30.]
- [39] J. E. R. Greene, *The 48 laws of power*. Penguin Publishing Group, 2020. [Cited on page 30.]
- [40] A. Ferreira and S. Teles, "Persuasion: How phishing emails can influence users and bypass security measures," *International Journal of Human-Computer Studies*, vol. 125, pp. 19–31, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1071581918306827> [Cited on pages 30 and 34.]

- [41] S. Sheng, P. Kumaraguru, A. Acquisti, L. Cranor, and J. Hong, "Improving phishing countermeasures: An analysis of expert interviews," in *2009 eCrime Researchers Summit*. IEEE, 2009, pp. 1–15. [Cited on pages 30 and 32.]
- [42] profpoint. (2019) 2019 state of the phish report: Attack rates rise, account compromise soars. Accessed: 2022-02-08. [Online]. Available: <https://www.proofpoint.com/us/corporate-blog/post/2019-state-phish-report-attack-rates-rise-account-compromise-soars> [Cited on page 30.]
- [43] E. J. Helsper, "Gendered internet use across generations and life stages," *Communication Research*, vol. 37, no. 3, pp. 352–374, 2010. [Cited on page 30.]
- [44] A. K. Ghazi-Tehrani and H. N. Pontell, "Phishing evolves: Analyzing the enduring cybercrime," *Victims & Offenders*, vol. 16, no. 3, pp. 316–342, 2021. [Cited on pages 31 and 33.]
- [45] SecurityScordcard. (2021) 12 types of phishing attacks and how to identify them. Accessed: 2022-02-08. [Online]. Available: <https://securityscorecard.com/blog/types-of-phishing-attacks-and-how-to-identify-them> [Cited on page 31.]
- [46] R. Prasad and V. Rohokale, *Phishing*. Cham: Springer International Publishing, 2020, pp. 33–42. [Online]. Available: https://doi.org/10.1007/978-3-030-31703-4_3 [Cited on pages 31 and 32.]
- [47] B. Branco, P. Abreu, A. S. Gomes, M. S. Almeida, J. T. Ascensão, and P. Bizarro, "Interleaved sequence RNNs for fraud detection," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 3101–3109. [Cited on pages 33 and 34.]
- [48] V. Shahrivari, M. M. Darabi, and M. Izadi, "Phishing detection using machine learning techniques," *CoRR*, 2020. [Cited on pages 33 and 34.]
- [49] A. Akinyelu and A. Adewumi, "Classification of phishing email using random forest machine learning technique," *Journal of Applied Mathematics*, vol. 2014, 04 2014. [Cited on page 34.]
- [50] N. Zhang and Y. Yuan, "Phishing Detection using Neural Network," *CS229 lecture notes*, 2012. [Cited on page 34.]

- [51] V. Shahrivari, M. M. Darabi, and M. Izadi, "Phishing Detection using Machine Learning Techniques," *arXiv*, 2020. [Cited on page 34.]
- [52] S. Afroz and R. Greenstadt, "PhishZoo: Detecting phishing websites by looking at them," in *2011 IEEE Fifth International Conference on Semantic Computing*, 2011, pp. 368–375. [Cited on page 34.]
- [53] R. Wirth and J. Hipp, "CRISP-DM: Towards a standard process model for data mining," in *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, vol. 1. Manchester, 2000, pp. 29–40. [Cited on page 35.]
- [54] LookFantastic. (2022) Promotion Campaign LookFantastic. Accessed: 2022-05-19. [Online]. Available: <https://www.lookfantastic.pt/myreferrals.list> [Cited on pages xiii and 38.]
- [55] A. Huang *et al.*, "Similarity measures for text document clustering," in *Proceedings of the sixth New Zealand Computer science Research Student Conference*, vol. 4, 2008, pp. 9–56. [Cited on page 43.]
- [56] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the GAP statistic," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001. [Cited on page 43.]
- [57] R. M. Heiberger and E. Neuwirth, "One-Way anova," in *R through excel*. Springer, 2009, pp. 165–191. [Cited on page 45.]
- [58] "Cross-validation: evaluating estimator performance," https://scikit-learn.org/stable/modules/cross_validation.html, accessed: 2022-03-30. [Cited on page 45.]
- [59] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: a corrected feature importance measure," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, 2010. [Cited on page 48.]
- [60] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144. [Cited on page 48.]