FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA

Active Learning for Cyber Attack Detection on Unlabeled URLs

Master Thesis im Fach Informatik

vorgelegt von

Max Möbius

angefertigt am

Lehrstuhl für Digitale Bildverarbeitung
Fakultät für Mathematik und Informatik
Friedrich-Schiller-Universität Jena

in Zusammenarbeit mit

DLR Institute for Data Science
07743 Jena
Germany

Gutachter:	Prof. Dr.-Ing. Joachim Denzler
Betreuer:	Badr-Eddine Bouhlal M.Sc, Dr. Clemens-Alexander Brust
Beginn der Arbeit:	17.07.2023
Ende der Arbeit:	31.01.2024

Kurzzusammenfassung

Für Techniken des maschinellen Lernens ist eine große Menge an gekennzeichneten Daten für Trainingszwecke erforderlich. Die vorhandenen markierten Datensätze sind jedoch veraltet, und die Cyberangriffe werden immer komplexer und ausgefeilter. Dies erhöht den Bedarf an Active Learning, das eine ständige Kennzeichnung neuer unbekannter Datenmuster ermöglicht, die Signaturen von Zero-Day-Attacks (unbekannte neue Angriffe ohne vorherige Signaturen) enthalten könnten. Diese Aufgabe kann mit verschiedenen Techniken erfüllt werden, darunter Blacklisting, regelbasierte oder maschinelle Lernverfahren. Diese Arbeit konzentriert sich auf die Erforschung der wesentlichen Schritte eines maschinellen Lernansatzes, insbesondere im Bereich des aktiven Lernens. Der Schritt der Datenvorverarbeitung kombiniert verschiedene Ansätze aus verwandten Arbeiten. Für die Merkmalsextraktion werden sieben Methoden verglichen, die von heuristischen Ansätzen bis zur natürlichen Sprachverarbeitung reichen. Die K-fache Kreuzvalidierung wird zur Validierung der extrahierten Merkmale und des ausgewählten Klassifikators verwendet. Es werden die maschinellen Lernklassifikatoren Random Forest und Stochastic Gradient Descent verwendet. Incremental Active Learning und Lifelong learning werden eingesetzt, und vier verschiedene Ansätze werden verwendet, darunter Pool-basiertes und Stream-basiertes Sampling, Query-by-Committee und Cluster Sampling. In dieser Arbeit werden hauptsächlich probabilistische Query Strategien verwendet. Die experimentellen Ergebnisse der verschiedenen Ansätze werden vorgestellt und in ihrer Verwendbarkeit mit Active Learning diskutiert. Abschließend wird die zukünftige Arbeit vorgestellt, um mögliche Wege für weitere Forschung aufzuzeigen.

Abstract

Machine learning techniques require a large amount of labeled data for training purposes. However, existing labeled datasets are outdated, and cyber-attacks are becoming more complex and sophisticated. This increases the need to use active learning techniques that constantly label new unknown data patterns, which might contain signatures of zero-day attacks (unknown new attacks with no preliminary signatures). Various techniques can accomplish this task, including blacklisting, rule-based, or machine-learning approaches. This thesis focuses on exploring the essential steps of a machine-learning approach, particularly in the domain of Active Learning. The Data Preprocessing step combines various approaches from related work. Seven methods are compared for Feature Extraction, ranging from heuristic approaches to Natural Language Processing. K-fold cross-validation is used to validate the extracted features and the selected Classifier. The machine learning classifier Random Forest and Stochastic Gradient Descent are utilized. Incremental Active Learning and Lifelong learning are utilized, and four different approaches are employed, including pool-based and stream-based sampling, Query-by-Committee, and Cluster Sampling. The thesis mainly employs probabilistic query strategies. Experimental results of different approaches are presented and discussed in their usability with Active Learning. Finally, future work is presented to highlight any potential avenues for further research.

Contents

Acknowledgment

Firstly, I would like to express my gratitude to Prof. Dr.-Ing. Joachim Denzler for providing me with the opportunity to write this thesis. I would also like to extend my thanks to my advisors at the German Aerospace Center (DLR) Jena, namely Badr-Eddine Bouhlal M.Sc and Dr. Clemens-Alexander Brust. Their guidance and feedback were instrumental in the successful completion of this project.

Furthermore, I am grateful to my family and friends for supporting me throughout the process and for their patience in reading over my thesis.

I want to mention that I utilized Grammarly to enhance the quality of my thesis. It was not used for any other purpose, such as text generation.

Chapter 1

Introduction

The use of web technology has grown considerably over the years, offering many possibilities with the fast-growing number of web pages available. However, this has also led to an increase in malicious pages that aim to extract information from a user's system or even provide access to the user's system itself. These pages mainly target personal information or account credentials and can even use unsuspecting users to gain access to sensitive data within their company. [WWXZ22, HY23, PP22]

With the increasing popularity of social networking, online shopping, and other parts of the internet, more people are using the web, making it a prime target for hackers. Therefore, cybersecurity plays a crucial role in ensuring the safety of users. Implementing systems that can identify malicious URLs with certainty and inform users before accessing bad web pages is essential. By doing so, everyone can surf the internet safely and enjoy its benefits without fearing cyber threats. [WWXZ22, HY23, PP22]

1.1 Motivation

There are different types of harmful URLs, such as Defacement, Malware, Phishing, Spam, and others. In the past, a technique called "blacklisting" was commonly used to detect malicious URLs. This technique involves creating lists of malicious URLs and triggers a warning signal if a user clicks on any of them. [WWXZ22, HY23]

Machine learning has become increasingly popular in recent times for identifying malicious URLs. In this process, URLs are first preprocessed and transformed into features. Subsequently, the machine learning model learns to recognize the features and characteristics of malicious URLs. Based on this learning, the model assigns labels of either "malicious" or "benign" to URLs. [WWXZ22]

Hackers are now changing the structure of URLs to avoid detection, which can make it difficult for machine learning models trained on URLs to detect newly appearing malicious URLs. However, "Active Learning" can help prevent this issue. It has two goals: first, to locate the most uncertain instances from the unlabeled dataset for a machine learning model, which reduces the number of instances that need to be labeled. Second, these instances can be used to improve the model for malicious URL detection. An algorithm searches the data to find the URLs that are most uncertain to label for the model, which are considered the "best" URLs. The model is then additionally fitted with this selected data. [BBK+22]

"Incremental Active Learning" is a process that enables continuous learning from a data stream. The process involves a cycle of data selection and data fitting. With each cycle, the model becomes more fitted to the new data and the next data selection can be more specific to the already improved model. This allows the model to continuously improve as new data is provided. [BBK+22]

1.2 Active Learning on unlabeled URLs

Our main goal is to create active learning algorithms that enhance machine learning models with limited data to achieve high-accuracy rates. Such a model and active learning algorithm can be utilized to identify the most uncertain URLs from a large pool of unlabeled data to improve the model. By doing so, the pool of unlabeled URLs is reduced to the most significant ones, thereby reducing the workload of labeling the URLs by a human expert.

To achieve this, we need to establish an optimal pipeline that involves preprocessing the data, extracting the relevant features from the URLs, using a machine learning model to classify the data, and then using active learning to improve the model's performance.

The used datasets consist of labeled URLs that are preprocessed into features. It is important for the model to perform well even when working with unlabeled data. We test datasets with binary and multiple classes/labels and do not preprocess any class imbalance in the datasets.

1.3 Related work

This section reviews work that detects malicious URLs with machine learning and active learning. We will also mention similar methods used for a different task.

There are various papers about malicious URL detection with machine learning. They typically cover data preprocessing, feature extraction (or embedding creation), and machine learning. However, some parts are not included in every paper or explained further. Some also include active learning, which extends the machine learning process.

Preprocessing data or, especially for this task URLs, can be an important basis for optimal machine learning. The substrings "http://" and "https://" can be removed because they are a part of the URL protocol and therefore have only a little impact on detecting malicious URLs. The corresponding paper also mentions word segmentation. By using statistically more frequently appearing symbols in the context of URLs, we can segment the words of a URL from each other. The URLs are then truncated to a maximal length of 30 words, shorter URLs are filled up with 0s. [WWXZ22]

In [GYR+21], it is mentioned that effective preprocessing and the right classifier selection highly affect the accuracy of a machine learning model. The data preprocessing cleans the raw data and delivers more relevant data. Raw data may contain missing values, outliers, and unnecessary features which can lead to inaccurate results. Therefore, these entries are removed. Additionally, entries with infinite or NaN values are replaced with the mean value to ensure accuracy in the data.

The authors of [VBB23] discuss other preprocessing steps. Removing the string after the "?" character can be important. It contains the query parameter and can be noisy and without meaningful information. The URLs are additionally truncated

to 128 characters because there is no actual improvement in machine learning with longer URLs.

All three papers discuss different steps for preprocessing the URLs. They share the same target, reducing URLs to the important features, to reduce the impact of inaccurate assumptions.

There are multiple ways to detect malicious URLs, but this thesis focuses more on intrusion detection. There, a malicious URL is only detected, and another system or a human needs to take action. Two different types of techniques can be employed, blacklisting and heuristic approaches or machine learning approaches. In the context of this thesis, machine learning approaches are more suitable and commonly used today. They involve extracting features from the URLs, which are then used to train models to detect malicious URLs. [CYRR21, MRL+16a, LBK21]

[PKKG10] presents the method "PhishNet". It combines a heuristic approach with blacklisting. The first component proposes five heuristics to enumerate simple combinations of known phishing sites to discover new phishing URLs. The second does malicious URL detection with a blacklisting dataset of nearly 18.000 phishing URLs.

For machine learning, different types of feature extraction approaches are utilized. One approach captures different heuristics, such as URL length, hostname length, or special character count. More advanced techniques, like natural language processing (NLP), are sometimes also utilized. The authors of [CYRR21] have decided to extract lexical and host features using heuristics. The URL length, hostname length, number of digits or letters, IP address presence, and URL abbreviation are extracted there. This data preprocessing and active learning steps are also used in our thesis.

[MRL+16a] utilized lexical features for detection purposes. CFSSubsetEval and Infogain were employed as feature selection algorithms. From 79 features, the best ones for different labels are calculated (Classes: Spam, Phishing, Malware, Defacement and All). There, machine learning without active learning is used.

The authors of [LBK21] implemented malicious URL detection using NLP and machine learning. TF-IDF-, Count- and Hash Vectorization is used for feature extraction. The corresponding features are then used for training on standard models (k-Nearest-Neighbors, Decision Tree, Random Forest and Logistic Regression).

[VV19] addresses the same subject of detecting malicious URLs using TF-IDF, Count, and Hash Vectorization for feature extraction and machine learning with several models (Random Forest, Naive Bayes, and Logistic Regression).

Another paper ([JA23]) performs phishing URL detection with heuristical feature extraction, followed by processing them through a BERT transformer. First, they extracted some lexical and host features from the URLs, and then they processed the URLs for a second time using a BERT transformer, which is based on a pre-trained model called "bert-base-uncased". The transformer tokenizes the URLs and converts them to BERT embeddings, that contain more information than usual feature vectors. The researchers then combined the URL features and the embeddings and trained different models (Long-term Recurrent Convolutional Network, Character level Convolution Neural Network, and BERT).

Related work with blacklisting and heuristic approaches appears less often today, maybe only in combination with machine learning. The machine learning approaches that were explained cover nearly every common approach used. The papers shown are only examples because there is more work on the same task with similar methods.

After a model has been trained, it can be utilized for detection purposes. Active learning can then be applied to recognize the most relevant instances from an unlabeled pool, which can be used to improve the pre-trained model. [Set09]

A commonly cited paper for active learning and further techniques is [Set09]. There, the different types of query strategies, like pool-based and stream-based sampling, are shown, as well as query strategies, like Uncertainty Sampling, Query-By-Committee, or Expected Model Change.

The paper referred to as [CYRR21] also employs active learning. First, several models (Adaboost, Decision Tree, Gradient Boosting, Logistic Regression, and Random Forest) are trained and tested. To enhance the Logistic Regression model through active learning, they implemented Query-by-Committee (QBC). However, they did not define any models for the committee in QBC.

The authors of [PP22] work with randomized active learning on URL data. The utilized machine learning model is not mentioned, but different Sample sizes are compared.

In the article [SACAF21], different query strategies are presented. These include Uncertainty- (Least Confident), Margin- and Entropy Sampling, as well as Vote Entropy-, Consensus Entropy- and Max Disagreement Sampling for Query-by-Committee. Experiments were accomplished on image data. "Res-Net-50" was used for feature extraction and an SVM model was trained and improved with active learning. All six previously shown sampling methods are tested. The model accuracy results started by 20% to 60% and were improved up to 70%.

[APH22]), which works with image data too, is comparing Random Sampling, Margin Sampling, Coreset, and a more complex strategy called Active Learning with Asynchronous Model Predictions (ALAMP) or ALAMP-div. The study utilized a ResNet-18 model as "FT" and an SVM classifier.

[DW09] utilizes TF-IDF for feature extraction on spam message (spam e-mails). They tested different machine learning models and query strategies (Random-, Uncertainty- and Cluster Sampling (K-Medoids)).

More authors use clustering, but not typically in the context of active learning or URL data. In for example [SR21] k-Medoids clustering plays a significant role. There, the k-medoids algorithms Alternate, Partition Around Medoids (PAM), and Clustering Large Applications (CLARA) are utilized. PAM is the most commonly utilized method, which has led to the development of similar methods like Faster-PAM or FastPAM1. Because of the bad performance of PAM on large datasets, CLARA is implemented and utilized. All of the previously described methods are tested in various settings.

Expected Model Output Changes is another active learning strategy, that measures the expected change of model outputs and is created by the authors of [FRD14]. It's a pool-based query strategy, that can be used on classification or regression tasks. The EMOC technique was tested on image data utilizing Gaussian process regression models, as reported in the research paper.

There are various query strategies used in related work. Simple methods (like Uncertainty Sampling) are commonly used in the context of malicious URL detection, while more complex or self-implemented methods and clustering are often utilized for other tasks, such as image classification.

Incremental active learning and lifelong learning are techniques that extend active learning. Both are further explained and tested in [BBK+22]. The study demonstrates that these techniques can be used to enhance the performance of an initial model through a lifelong learning cycle. This cycle involves active learning, where instances are queried, followed by annotation, and lastly, incremental learning to improve the model. [KRFD16] discuss a similar lifelong learning cycle for visual recognition with image or video data.

There are many studies focused on detecting malicious URLs with the involved steps of data preprocessing, feature extraction, model training, and active learning. Some of them actually do not explain steps like data preprocessing and feature extraction. Additionally, active learning is often only utilized with simple query strategies, such as random or uncertainty sampling. Advanced methods are rarely taught and are mostly used for other tasks, such as image classification. Clustering, which can be used to filter the initial training data, is usually not applied to this task. Furthermore, incremental active- and lifelong learning is not widely used and even less in the context of malicious URL detection.

1.4 Overview

In this section, we will provide a brief overview of the thesis by listing upcoming chapters and explaining the information that can be found in each chapter.

The first chapter is called the Background (chapter 2), and it introduces the terms and techniques that will be used in the later chapters. The next chapter is the Methodology (chapter 3), which is divided into two parts. In the first part, we will explain the thesis's pipeline and the corresponding research questions. In the second part, we will describe the implementation of the ideas. The following chapter will show the experiments, along with some dataset descriptions and settings. The experiments will be further explained and the relevant results will be displayed. Finally, the last two chapters will provide the thesis's conclusion and some further work.

Chapter 2

Background

The background provides an explanation of terms that may not be familiar to all readers and are used in later chapters.

2.1 Uniform Resource Locators

"Uniform Resource Locators (URLs) are a standardized format for describing the location and access method of resources via the internet" [AVW20]

Figure 2.1: Uniform Resource Locator Structure [AVW20]

[AVW20] presents some information about URLs. URLs are an essential part of navigating the web and sharing online resources. They are made up of various components, as shown in Figure 2.1. The picture also demonstrates that different combinations of the components can lead to the same page. The **<user>:<password>** and **<port>** components are not commonly used in the user-facing. That means if you are only using and not working with URLs, you don't see them usually. The **<host>** itself can be split into three components, **<subdomain>.<domain>.<top level domain>**. They are separated by dots.

For example, in the URL **www.bbc.com** shown in the picture, the top-level domain is **com**, the domain is **bbc**, and the subdomain is **www**. Another example is **facebook.mobile.com**; when the browser tries to visit this URL, the server associated with **com** is contacted to do a lookup for **mobile**. Then the same is done for **mobile** to lookup for **facebook**. So URLs are always read from back to front. [AVW20]

2.2 URL Attack Techniques

Cyberattack Techniques refer to the various methods used by hackers or attackers to gain unauthorized access to user's data or -systems or to cause damage to them. These attacks are typically carried out using malicious URLs, and can take the form of Defacement, Malware, Phishing, and Spam attacks. These cyberattacks only work because an unaware user clicks on URLs of malicious websites. Below, the attacks named by [AAA+22] are explained further.

2.2.1 Defacement URL Attacks

When the user is redirected to a malicious website, the site's visual appearance or content is altered without their knowledge or consent (same Website, but one or multiple visual parts are changed). This involves taking down the original website and modifying it, without requiring any authorization. This is also known as website penetration. [AAA+22]

> "*For example, changing the company logo or name, or inserting totally extraneous text, is normally considered a serious defacement, while the normal upgrade of Web site content and of the graphical evolution of the interface, as well as the changing advertising banners, should not generate continuous alarms.*" [BCCR19]

2.2.2 Malware URL Attacks

In this attack, users are directed to websites that install malware on the user's device. This malware is usually used for file corruption, keystroke logging, and identity theft. Such a method is named **Drive-by download**, where the user unintentionally downloads malware by visiting a malicious website, even if only for a brief moment. [AAA+22]

2.2.3 Phishing URL Attacks

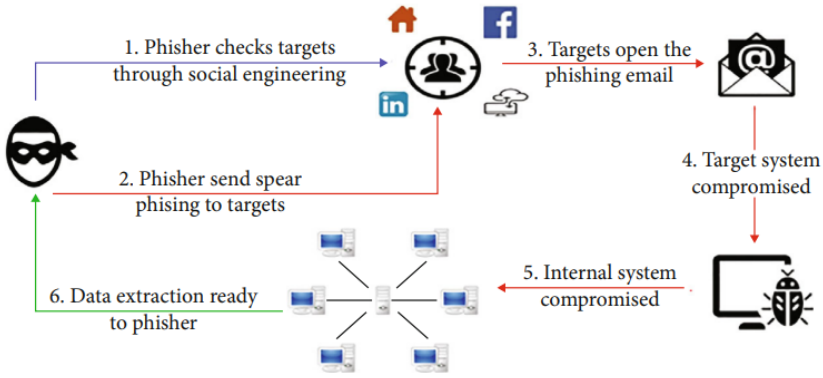

Figure 2.2: Phishing Attack [MBA+21]

Phishing is a type of attack that aims to steal valuable information, e.g. by tricking users into clicking on a link that takes them to a fake website. One way is email-to-email (Hackers send mail asking for personal data). Others are email-to-website (Hacker sends a mail with phishing URL inside), website-to-website (User clicks on phishing URL on a website or online advert), and browser-to-website (Use of incorrectly spelled URL that leads to phishing website). [AVW20, MBA+21]

URL manipulation is a method used by cybercriminals to deceive users into visiting a malicious website instead of the intended one. For instance, a user may want to log in to their bank account by clicking on a link, like `https://bankofthewest.foobar.com`, but the URL they click on is not the correct one. Instead of leading them to their bank's website (Original page: `https://bankofthewest.com`), the link takes them to a fake website that looks like the bank's login page. The user may then enter their login credentials on this fake page, which the cybercriminals

can use to steal their sensitive information. This type of attack can also infect the user's computer with malware. [AVW20]

2.2.4 Spam URL Attacks

Spammers use a technique called "cloaking" to create web pages that deceive the browser engine into thinking that the website is harmless. They do this by illegally boosting the website's search ranking. The higher a website is in the search ranking, the higher it is listed if a user searches for a related topic. The objective is to attract more users, who may then click on the corresponding URL. Alternatively, spammers may send spam emails that contain spam URLs. [AAA+22]

2.3 Malicious URL Detection

Malicious URL Detection is a crucial process in cybersecurity. It involves examining website links to determine whether they are malicious or not. This is done to prevent web users from visiting harmful websites. Different systems have varying functionalities and detect malicious URLs at different points in the network. [Cha13, SLH17]

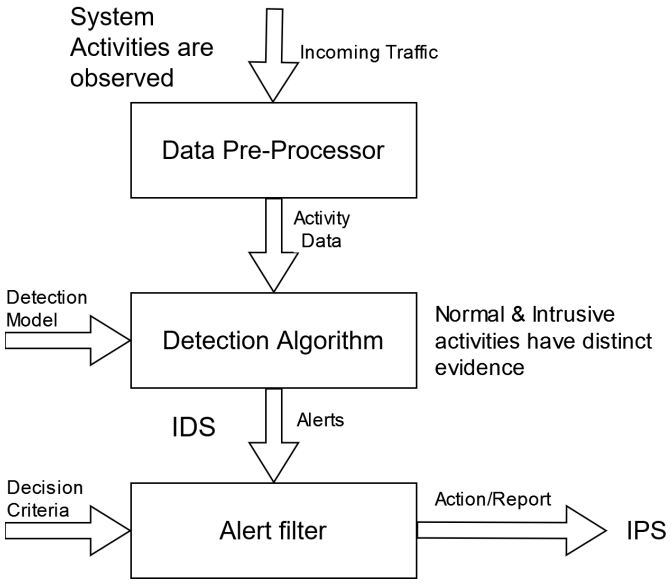

Figure 2.3: IDS and IPS in network security, reworked from [Cha13]

There are two systems that can be used for this process, **Intrusion Prevention Systems (IPS)** and **Intrusion Detection Systems (IDS)** (as shown in Figure 2.3).

IPS are like a firewall. They directly sit between two networks and control the traffic going through them. The only difference is that firewalls deny access for requests that do not match the safety definitions, and IPS only accepts requests that do not seem malicious to the system. [Cha13]

IDS are installed software or physical appliances that monitor network traffic to detect unwanted malicious traffic. IDS tools can also store the detected events in a log file for later review. [Cha13]

The difference between IPS and IDS is illustrated in Figure 2.3. IDS only detects the problem and requires further assistance from a human or another system to interpret the results and take action. On the other hand, IPS goes a step further by detecting the problem and rectifying it if necessary. [Cha13]

IDS are the more fitting technique for the thesis. With the development of various approaches over time, URL attacks can be detected. Additional systems are used to review the data, and decisions are made based on that information, either granting or denying access. Down below are popular approaches listed from the past to today. [SLH17]

2.3.1 Blacklisting and Heuristic/Rule-based Approach

Blacklisting is a common and classical technique that consists of a list of URLs that are well-known for being malicious. Whenever a user visits a new URL, the list is checked to see if the URL is present in it. If the URL is found, it is classified as malicious, otherwise it is marked as benign. However, the challenge with this approach is that new malicious URLs are created every day, and if they are not present on the list, they cannot be detected. This means that new threats may go undetected or it may require a significant effort to update the list frequently. [SLH17]

The **Heuristic/Rule-based** approach is basically an extension of the blacklist method. Attacks that appear more than once are identified and get signatures.

Intrusion detection systems are then used to scan the website for these signatures. These threats can also be detected in new URLs. This approach is also named "blacklist of signatures" because the signatures are added as rules to a list, which is then used as a blacklist. The used methods are only functional for a limited number of common threats, so not for all types of attacks. Additionally, obfuscation techniques can be used to bypass the systems check. Another problem is that if the website isn't launching the attack directly after visiting, a detection system can't detect the attack or not immediately. [SLH17]

2.3.2 Machine Learning Approaches

The analysis of URL information involves a process called **Feature Extraction**, which is important for associating each URL with certain features. This process is important not only for malicious URLs but also for benign ones. Enhancing the features that reflect a URL makes it easier to distinguish between benign and malicious URLs. Machine learning algorithms are used for the detection. Firstly, features are extracted, and then a suitable model is trained using this data. The model can then be employed to label new URLs with different labels/classes, such as benign or malicious (More different labels/classes are possible). [SLH17, RVK21, RPA22]

For feature extraction, there are different approaches that use either fixed features of URLs, like **Classical Feature Extraction** or Natural Language Processing (NLP) methods that extract the features in a more complex way, like by **Vectorization** or by **Transformer**. In terms of machine learning, the success of the algorithm depends on how well it fits the data used. [SLH17, RVK21, RPA22]

The feature extraction is an essential part of the thesis and is further explained in the next sections.

2.4 Classical Feature Extraction

2.4.1 URL Feature Categories

The most common ones are the **Lexical Features**. These features are directly extracted from the URL string and include aspects such as the length of the URL, certain parts of the URL, and the count of specific characters (., /, @, ...). [RPA22]

Host or Web Server based Features are another type of feature that can provide information about a website's location, identity, management style, and properties of the host. These features include the IP address, domain registration, and location. [RPA22]

Page based Features are extracted from the page contents (Issue: malicious website need to be accessed for that). Like HTML features, JavaScript features or Visual similarity features. [RPA22]

The last are **Popularity Features**, which are obtained from third party servers, like page rank, link popularity score or social reputation features. These features are e.g. Public share count on Facebook and Twitter or Google page rank. [RPA22]

The last two types are less commonly used.

2.4.2 Extraction Technique

URL Feature Extraction is a more simple technique. Here, specific features from the previously named categories are selected. The selected features are extracted for every URL and then used in machine learning to label them accordingly.

2.5 Natural Language Feature Extraction

2.5.1 Natural Language Processing (NLP)

NLP is a computerized method used for analyzing text. The goal is to process text to "human-like language", which is also referred to as Natural Language Understanding

(NLU). Human-like language means allowing the computer to understand the text in the same way as a human can. So, the true goal is for an NLP system to be a true NLU. For that, it needs to paraphrase an input text, then translate it into another language, answer questions about the content of the text, and in the end, draw interferences from it. However, drawing inferences is still a task that NLP needs to improve upon to achieve true NLU. [Lid01]

In the thesis, URLs instead of text are analyzed and processed. The following techniques demonstrate how features are extracted out of URLs using NLP.

2.5.2 Word Vectorization / Word Embedding

The terms Word Vectorization and Word Embedding are equally used. [BAK+19]

The vectorization is a text processing technique working with NLP. Various methods such as TF-IDF, Count or Hashing Vectorization are employed to convert URLs into feature vectors, which are lists of numerical values. For that, the processed data must be split into trainings-data and remaining-data. The vectorizer is then fitted on the trainings-data and the whole dataset can be converted to vectors. [LBK21]

For that, the input text or URLs are converted into numerical values and saved in the feature vectors. These numerical values should represent the most important information of the input. Machine learning models can now easily analyze or understand the given input. [SGP+23]

TF-IDF Vectorization

> "*TF-IDF stands for "term frequency-inverse document frequency", which means the weight allocated to each token not only depends on its frequency in a document but also how persistent that term is in the entire corpora*" [VV19].

$$\text{TFIDF}(t, d, D) = \text{TF}(t, d) \cdot \text{IDF}(t, D) \tag{2.1}$$

t: Term/Word

d: Document (consists of multiple terms)

D: Collection of documents

This method basically calculates the term frequency $\text{TF}(t, d)$ and inverse document frequency $\text{IDF}(t, D)$ (see Equation 2.1). The term frequency is the count of one term/word in one document, and inverse document frequency is the count of documents in which one term appears. [Gau20, VV19]

Count Vectorization

Count-Vectorization is the most straightforward vectorization, it counts the number of times a token shows up in a document and uses this value as its weight [VV19].

2.5.3 Transformer

Transformer, also named transformer-based pretrained language models (T-PTLM). Big existing ones are GPT-1, BERT, XLNet, RoBERTa, ELECTRA, T5, ALBERT, BART and PEGASUS. Such models are useful because they can learn universal language representations from a large volume of data. [KRS21]

Deep learning models such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) encounter difficulties in learning word representations with locality bias and modeling long-term contexts. Transformers do not face this problem as they rely on self-attention, which assigns weights to words, allowing the model to determine the significance of each word. Self-attention is more parallelized than RNNs, and the transformer can easily model long-term contexts as every token attends to all the tokens on the input sequence. Transformers also use a large amount of encoder and decoder layers, enabling them to learn complex language information. This can also lead to an expensive and time-consuming process. [KRS21]

Transformers are machine learning models trained on a large amount of data in self-supervised learning. This method enables the model to learn the language it has been trained on. The more data is used and the bigger the model size, the better the performance of the transformer model. These models can be trained on general data or specific data that is used for the experiments. [KRS21]

Transformers can be utilized for both feature extraction and classification. These two processes require different types of models. For the classification, a matrix of numbers is usually needed. For that, the input text needs to be mapped to a sequence of dense, low-dimensional vectors commonly named embeddings. Sub-word and character embeddings are commonly used instead of word embeddings as they have a smaller vocabulary size. Tokenizers are often used to create the sub-word vocabulary. [KRS21]

In the case of the thesis, the transformers are only needed for the feature extraction. That's the reason, why further explanation for the classification use is not needed.

Bidirectional Encoder Representations from Transformers (BERT)

BERT interprets textual material bidirectionally and collects contextual information.

There are two steps in the process of the BERT tokenizer: **Canonicalization** and **Tokenization**. In the first one, numerical punctuation marks, and special characters are ignored, and upper-case letters are converted to lowercase. In the tokenization (uses a glossary of 30,522 words), the input is segregated into a certain format of a list of tokens as in the glossary. WordPiece tokenization is used to handle terms that are not present in the glossary. It splits the word text into subwords or root words by removing prefixes and suffixes. Example: "looking" is represented as "look + ##ing". [SMSX19]

The BERT model converts the tokenized URL from the BERT tokenizer into BERT embeddings, which reflect the contextual knowledge of each token in the URL sequence. The embeddings capture the contextual links between words and subwords by encoding the semantic and syntactic information of the tokens. In general, the embeddings help models to understand the meaning and context of the URLs. [JA23]

2.6 Evaluation

2.6.1 Dataset Splitting

To ensure reliable and accurate machine learning model training, it is crucial to split a dataset into independent sets before using it for testing or training. It is important that no set contains any data from another set. Otherwise, the model already knows data from the test set, and the test data no longer provides a reliable measure of generality. [Lon21]

Only a training- and test-set is needed for general model training. A Split of 80:20 or 70:30 for training- and test-data are usual (means 80% to 20%). The training-set is used to train the model, and the test-set is utilized to calculate the model's accuracy. [Lon21]

An effective way to split a dataset, especially in active learning, is by dividing it into three sets: initial, test, and unlabeled training set. You can split it in a ratio of like 80:10:10 or 10:80:10. The initial training set is used for the initial model training, the test set is used to calculate accuracy, and the unlabeled training set is used for active learning, which is further explained in section 2.7.

2.6.2 Cross-Validation

Cross-validation (CV) is a data resampling method and is used to utilize the collected data better, reduce overfitting, and generalize the model predictions. [Ber19, NCB23]

k-fold Cross-Validation

The whole dataset is partitioned into k parts (all equal size). Every partition is called a fold. One fold is used as a test-set, and k-1 folds as train-sets. To use every fold as a test-set, the process is done k times. The average overall k performance measurement is the cross-validated performance. The standard-deviation can also be calculated and shows the variation of the k result values. [Ber19]

Leave-one-out cross-validation

Please keep in mind that this process involves using only one sample data as a test-set and n-1 samples for train-set, which is a more specialized version of k-fold cross-validation. Like in k-fold cross-validation the process is repeated n times, so the computational costs are very high, so it shouldn't be used with large datasets. [Ber19]

It's worth noting that cross-validation is a process in machine learning used to validate the data used, such as after preprocessing and feature selection, as well as in model hyperparameter tuning, such as finding the optimal number of nearest neighbors in the k-nearest-neighbor classifier.[Ber19, Lon21]

It is not recommended to use Cross-Validation in the later stages of the process because it becomes difficult to select a single model from multiple created models. Selecting the model with the highest accuracy leads to another problem, as the test-set used for evaluation may be too small and easy to predict, which can result in high accuracy. To overcome this issue, it is suggested to use a different test-set with more data or keeping some data aside before performing Cross-Validation. [Lon21]

2.7 Active Learning

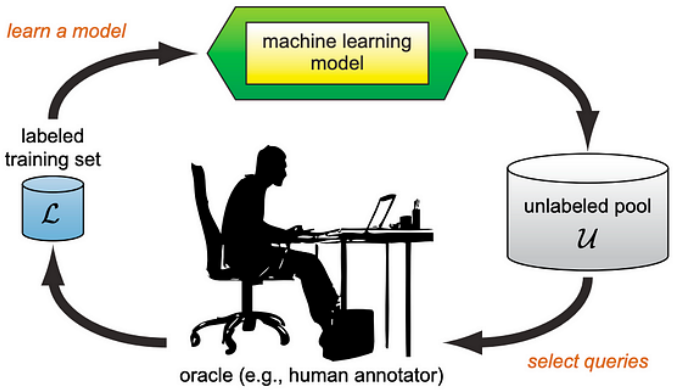

Figure 2.4: Active Learning Cycle [Set09]

Active learning is a process that is applied if there is a big pool of unlabeled data and a machine learning model that is already trained on comparable data. These

techniques should find the most relevant or uncertain samples for the model to annotate. The benefit is that only a small part of the unlabeled pool needs to be annotated. Furthermore, the model can be improved with the found samples. The more the model improves, the better the algorithm for that particular model or use case. Stopping Criteria are used to determine when an active learning cycle should stop. These criteria can include a number of cycles or a maximum accuracy level. [BBK+22, Set09]

The process of active learning involves an annotation process if the additional data is not labeled corresponding to the labels/classes of the model. Samples are annotated, usually by humans, when they are selected. [BBK+22, Set09]

Active learning can be used for different tasks, such as speech recognition, information extraction, or like in the case of the task, classification and filtering. [Set09]

There are different sampling strategies and corresponding sampling methods. Two popular strategies are **Stream-based Selective Sampling** and **Pool-based Active Learning**. The stream-based variant selects samples from a stream of data. Per epoch, one sample is given, and for that, it is chosen whether it should be queried or not. The pool-based method, on the other hand, selects samples from a large pool of instances (This strategy is shown in Figure 2.4). Pool-based sampling is also the most used variant, with methods like **Uncertainty Sampling** or **Query-by-Committee**. Uncertainty Sampling selects the samples that are the most uncertain for the current model. Query-by-Committee uses multiple models to select the best samples — those that are chosen by the most models. [Set09]

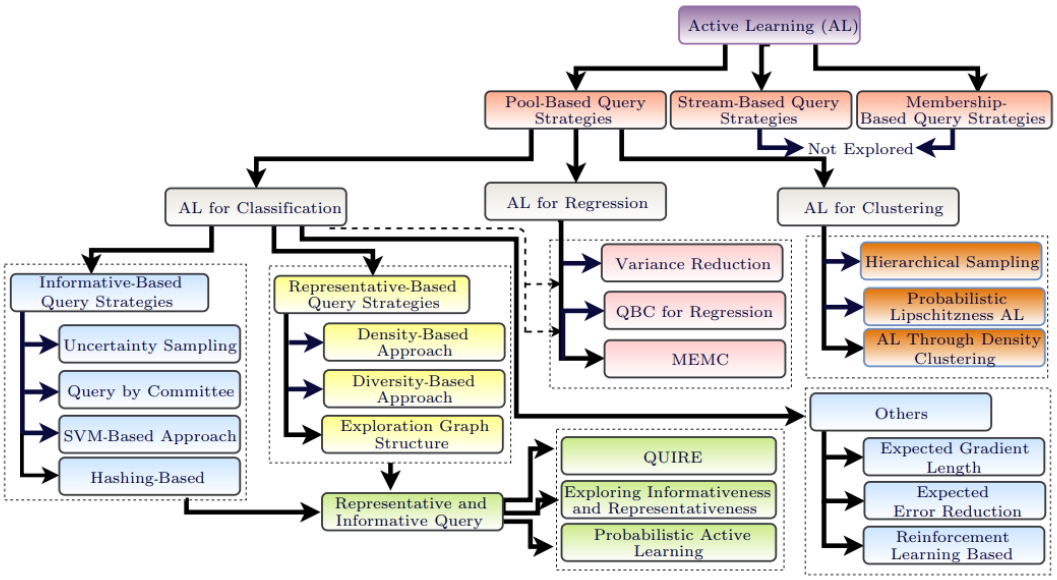

Figure 2.5: Sampling Strategies [KG20]

There are numerous strategies and methods, as shown in Figure 2.5 above. Over time, more complex strategies have been developed to improve a model to achieve the best possible results.

2.7.1 Uncertainty Sampling

Uncertainty Sampling is a popular and widely used active learning technique. In this method, the most uncertain instances are queried. In the instances, the learner is least certain how to label. The posterior probabilities are used to make final decisions. This method is often quite straightforward for probabilistic learning models. [SACAF21]

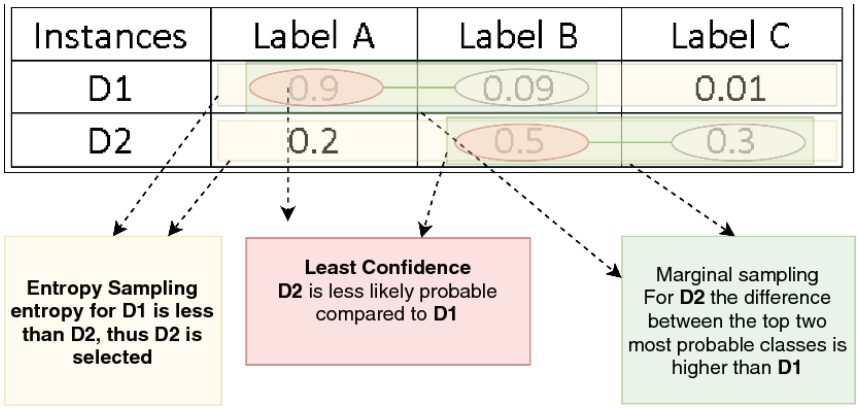

Figure 2.6: Working mechanism of the different sampling strategies used for uncertainty sampling [SACAF21]

Figure 2.6 shows two instances with the possible labels. For each instance, the probability of each label is displayed.

Least Confidence Sampling

Least Confidence Sampling chooses the instance from the unlabeled pool of samples that has the least confidence about its most likely label. In general, the strategy is working better for multiclass classification. The corresponding equation is [SACAF21]:

$$\mathrm{LC}(x) = \mathrm{argmax}_x(1 - p_\theta(y \mid x)) \tag{2.2}$$

x: Current sample

y: The most probable label

θ: The underlying model

$p_\theta()$: Posterior probability

In Figure 2.6 you can find how it works on an example.

Margin Sampling

Least Confidence does not consider the rest of the labels, which might be useful in the selection process. Margin Sampling incorporates the posterior probability of the second most likely label by selecting an instance having the least difference between the top two most probable labels. The equation for that is [SACAF21]:

$$\mathrm{MS}(x) = p_\theta(y_1 \mid x) - p_\theta(y_2 \mid x) \tag{2.3}$$

x: Current sample

y_1: The most probable label

y_2: The second most probable label

θ: The underlying model

$p_\theta()$: Posterior probability

An example is shown in the Figure 2.6.

Entropy Sampling

Like in Margin Sampling, taking the top two most probable labels to represent the probability distribution is not sufficient, if there is a high number of classes. Entropy Sampling utilizes the probability distribution by calculating the entropy of each instance. [SACAF21]

$$\mathrm{ES}(x) = -\sum_{y \in Y} p_\theta(y \mid x) \cdot \log_2 p_\theta(y \mid x) \tag{2.4}$$

x: Current sample

Y: Output class

y: The most probable label

θ: The underlying model

$p_\theta()$: Posterior probability

It is a more useful technique for probabilistic multi-class classification but works as well as Least Confident and Margin Sampling for binary classification. [SACAF21]

2.7.2 Query-by-Committee

Multiple trained classifier $C = (\theta_1, \theta_2, \ldots, \theta_n)$ are used. The queries are selected by measuring the disagreement between these models. The aim is to get a more precise search with as few labels as possible.

Vote Entropy Sampling

Vote Entropy is the generalized entropy-based uncertainty sampling, with the equation [SACAF21]:

$$\mathrm{VE}(x) = \mathrm{argmax}_x - \sum_i \frac{V(y_i)}{C} \cdot \log \frac{V(y_i)}{C} \tag{2.5}$$

x: Current sample

y_i: Vector of all possible labels

C: Committee of classifier

$V(y_i)$: Total number of votes for a label

Every model votes for a sample, the votes are then summed up for every sample, and the entropy of every vote is calculated. The instance with the largest entropy of its votes is selected. [SACAF21]

Consensus Entropy Sampling

In Consensus Entropy Sampling, the class/label probabilities with each model from the committee are calculated. Then, the average is taken, which is the consensus probability, and the samples with the largest entropy are selected for labeling. The corresponding equation is [SACAF21]:

$$\mathrm{CE}(x) = \frac{1}{C} \sum_{c=1}^{C} p_\theta(y_i) \tag{2.6}$$

x: Current sample

y_i: Vector of all possible labels

C: Committee of classifier

θ: The underlying/current model

$p_\theta()$: Probability with the model θ

Max Disagreement Sampling

Here, the disagreement of each model is calculated by using consensus probability. Then, the samples with the largest disagreement are selected. In this way, the actual disagreement is taken into account. [SACAF21]

2.7.3 Incremental Active Learning

Incremental active learning is a variant of active learning. The difference is that the chosen samples are fitted to the model before the start of the next cycle. This creates a feedback loop, allowing the model to be updated directly with the newly selected data. As a result, the selection algorithm needs to adapt to determine which samples should be selected next for further improvement. Over time, the probability of selecting samples that don't lead to improvement can be reduced. [BBK+22]

2.7.4 Lifelong Learning

Lifelong learning extends incremental active learning further. Every epoch, models are typically built entirely from scratch and trained using the train-set and freshly queried samples. With this technique, the already trained model is only learned/fitted with the new queried samples. This process is more efficient since less data is required to fit each epoch, and the model can be maintained. The accuracy results are consistent with the results of incremental active learning. [BBK+22, KRFD16]

2.8 Clustering

Clustering is an unsupervised machine learning algorithm. It deals with the data structure partition in an unknown area, which means the technique identifies similar groups of data in a large dataset. These groups are then named clusters. [XT15]

There is no perfect definition, but some rules are important:

1. "*Instances in the same cluster, must be similar as much as possible*" [XT15]
2. "*Samples in different cluster, must be different as much as possible*" [XT15]
3. "*Measurement for similarity and dissimilarity must be clear and have practical meaning*" [XT15]

The process can also be divided into general steps:

1. "*Extract and select the most representative features from the original data set*" [XT15]
2. "*Design the clustering algorithm according to the characteristics of the problem*" [XT15]
3. "*Evaluate the clustering result and judge the validity of algorithm*" [XT15]
4. "*Give some practical explanation for the clustering result*" [XT15]

2.8.1 K-Medoids Clustering

This technique is similar to k-means clustering, which involves dividing a dataset into k clusters. Initially, k samples are randomly selected as initial medoids, where each medoid represents a cluster. A Medoid is the sample with the lowest distance to all other instances in the cluster. For the distance calculation, different measures can be used, but typically, Euclidean distance is used. All other instances are then assigned to the clusters that they best fit in, meaning they are assigned to the cluster with the lowest distance to its medoid. [Bha14, PJ09]

During the process of k-medoids, each instance is examined to determine whether it would be a better representation for a cluster than the current medoid. If that

is the case, a medoid is changed to the new one, and all instances are reallocated to their respective clusters. This cycle is repeated until no medoid changes occur. [Bha14, PJ09]

There are different algorithms of k-medoids, such as Alternate, Partitioning Around Medoids (PAM), and Medoid Silhouette Clustering (MSC). [SR21, Bha14, LS22]

Partitioning Around Medoids (PAM)

Partition Around Medoids is a clustering algorithm, that is a variation of the k-medoids algorithm. In the following, the algorithm is explained furthermore:

1. There k instances are randomly selected as medoids
2. For each pair of non-medoid instance x_j and selected medoid m_i, calculate the total swapping cost $S(x_j, m_i)$. If S ¡ 0, m_i is replaced by x_j
3. Repeat steps 2. and 3. until there is no change in medoids

$$\text{Cost} = \sum_{x_j} \text{dist}(x_j, m_i) \tag{2.7}$$

$$S = \text{Cost}(\text{new } C_i) - \text{Cost}(C_i) \tag{2.8}$$

C_i: A cluster that represents a feature with instances inside

x_j: An instance inside the cluster C_i from the dataset

m_i: The medoid of the cluster C_i

dist(): The distance function

Cost(): Calculates the sum of the distances between each instance x_j and the medoid m_i

S(): Calculates the swapping costs by subtracting the costs of the cluster C_i with x_j as medoid and the costs of the cluster C_i with m_i as medoid

Clustering Large Applications (CLARA)

CLARA is a clustering algorithm that applies sampling approaches to handle larger datasets. It is more efficient compared to PAM when working with larger datasets. The algorithm involves two main steps: Sampling and Clustering. **Sampling** divides the dataset into smaller subsets. In the **Clustering** step, the PAM algorithm is applied to each subset to find the optimal k medoids based on a chosen distance function. These two steps are repeated until a set of medoids with minimal cost is found. [Bha14, SR21]

Chapter 3

Methodology

In this chapter, the methodology is explained. This includes initial ideas and an overview of the project pipeline implementation.

3.1 Ideas

We conducted research on related work and came up with ideas on how to complete the task. The related work typically involves three to four steps: dataset preprocessing, feature extraction, machine learning, and possibly active learning. All of these steps should be included in this thesis, and we should explore testable methods from related work that have proven helpful in producing high-accuracy results.

To preprocess the data, we need to discover methods that make the data less noisy and have a positive impact.

For feature extraction, it is necessary to search for methods that extract the features in different forms. It can be done through heuristic feature extraction, such as using the length of URL parts or the count of chars, or through NLP, where Vectorization or Transformers are often used to produce word embeddings that can be converted to feature vectors.

In machine learning, we do not need classifiers that reach the highest accuracies. The primary focus should be on improving the models with active learning, not initial training. Therefore, it is advisable to test both simple models such as Random

Forest or SVM from related work, as well as complex models, to determine the level of complexity that is required to avoid overfitting

We should test different techniques for active learning, primarily incremental and lifelong learning. Therefore, we should test various query strategies.

Corresponding to the ideas, research questions were defined (they are marked with "RQ" if used):

1. Which preprocessing steps can be added to improve the Classification of malicious URLs?
2. Which feature extraction methods are most suitable for URL Labeling?
3. Is Natural Language Processing (NLP) usable for feature extraction with URL data?
4. Which machine learning classification models are appropriate for URL Classification?
5. Which query strategies perform the best for URL Labeling?
6. How can clustering be utilized for URL Labeling?
7. Should lifelong learning be integrated into the project?

3.2 Implementation

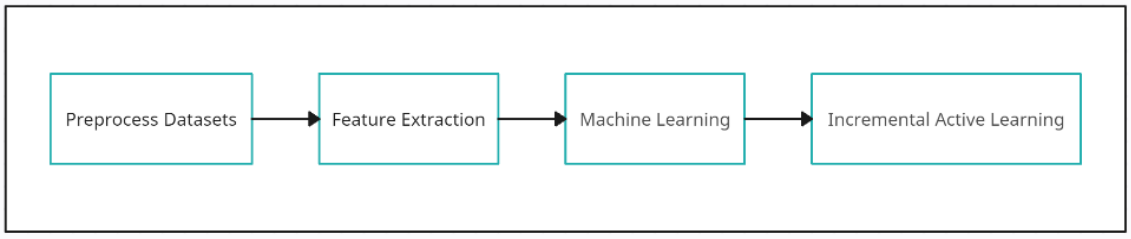

Figure 3.1: The whole pipeline

The Figure 3.1 shows the process steps utilized in this thesis. All of them are further explained in the following.

3.2.1 Preprocess datasets

The first step in the process is to clean the datasets by removing entries that have outstanding values such as missing values, wrong data types, NaN values, and infinite values. These entries are not useful and cannot be used in the analysis. [GYR+21]

Next, the URLs are preprocessed by removing the substrings "http://" and "https://" as they only identify the protocol and are not very useful in identifying malicious URLs (see section 2.1). [WWXZ22]

Additionally, the query parameters, which can be noisy and without meaningful information, are removed by eliminating the string after the first appearing "?". Longer URLs do not improve learning, so every URL is truncated to 128 characters (see section 2.1). [VBB23]

3.2.2 Feature Extraction

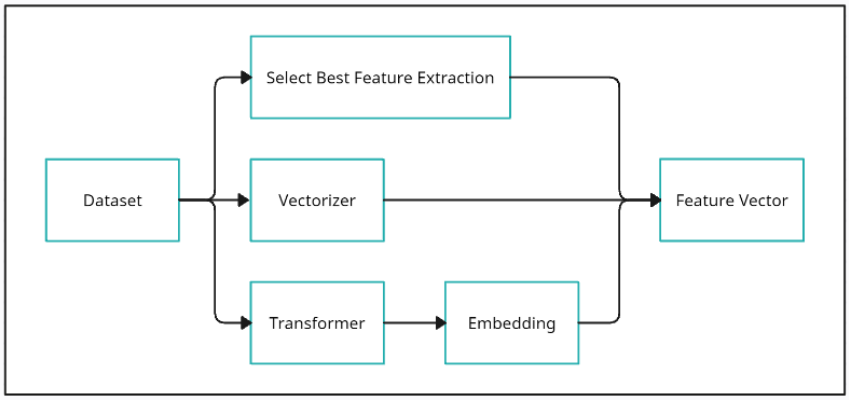

Figure 3.2: Feature Extraction Pipeline

Figure 3.2 depicts the feature extraction pipeline. This provides a general overview of the process that is carried out.

For the feature extraction different techniques are tested: **Select Best Feature Extraction**, **Vectorizer**, **Transformer**. Various methods are implemented with the named techniques to transform the URLs into feature vectors. The first four methods correspond to **Select Best Feature Extraction**, the next two to the

feature extraction by **Vectorizer** and the last one is a **Transformer**. They are further explained in the next part.

After the feature extraction, models based on the newly acquired data are tested with "k-fold Cross-Validation". With the cross-validation, machine learning models are trained, and the results can be used to validate whether the data can be further used.

In the following the extraction methods are explained more. Every method gets the name **feature_ex** + the method number, like **feature_ex1**.

Select Best Feature Extraction

First, lexical features and host features corresponding to the URLs are calculated. For other features, ratios and rates between the features are also computed. Every method selects specific features that build up the feature vectors. The following table lists, which features are used by which method:

Method	Features
feature_ex1	URL Length | Hostname Length | Count Digits | Count Letters | Use of IP | Short URL
feature ex2	Entropy Domain | Arg Path Ratio | Arg URL Ratio | Arg Hostname Ratio | Path URL Ratio | Char Cont Rate | Number Rate Filename | Hostname URL Ratio | Number Rate URL | Path Hostname Ratio | Number Rate Afterpath | Avg Path Token Length
feature_ex3	**feature_ex1** ∪ **feature_ex2**
feature_ex4	Best k features from **feature_ex3**

Table 3.1: Select best feature methods

The chosen lexical and host features of **feature_ex1** are selected because this combination would give the best prediction. Further details are available in [CYRR21].

The lexical features of **feature_ex2** are identical to the ones used in [MRL+16a]. Different algorithms were used to find best features for every possible label. Here the overall best features (not specific to one label) were used.

In the third feature extraction method (**feature_ex3**), both papers' common features were used since the two papers did not share any features.

For **feature_ex4** the best features of a list of different features are chosen.

Feature selection algorithms are used to select the most relevant features for classification tasks. There are three possible algorithms, that can be utilized for this purpose. One algorithm utilizes the ANOVA F-Value or F-Test [SW+89]. Another estimates the mutual information for a discrete target variable [PLD03]. The last algorithm calculates the chi-squared statistics between each non-negative feature and class [LS95]. It is possible to achieve high results with all three algorithms, depending on the machine learning algorithm, the selection of features, and the number of chosen features. Therefore, there is no best algorithm. [ASBAK+23]

We use the algorithm working with mutual information. It calculates the relevance of every feature to a target variable [DGG23]. In this case, the relevance to the labels is calculated, and the 10 best features are selected (There is no best number of features; the number depends on the model and the possible features [ASBAK+23]).

Feature description for **feature_ex1** [CYRR21]:

A short URL is a substantially shorter URL that redirects to the original longer URL. For that, shortening services like "bit.ly" or "tinyurl.com" are usually used. An URL, like `http://www.this.is.a.long.url.com/indeed.html` is converted into `http://bit.ly/dv82ka`, where "dv82ka" is the key that represents the original URL. [APK+11]

Table 3.2 provides a detailed description of the features and their corresponding types.

Feature	Description	Type
URL Length	Length of URL (Number of characters)	integer value
Hostname Length	Length of URL hostname component	integer value
Count Digits	Count of digits (0-9)	integer value
Count Letters	Count of symbols from the alphabet (A-Z, a-z)	integer value
Use of IP	Is IP used instead of domain name?	boolean value
Short URL	Is a known short URL string shown?	boolean value

Table 3.2: Feature description and type for feature_ex1

Feature description for **feature_ex2** [MRL+16a]:

The URL components that are mentioned in the following are further explained in the background at section 2.1.

"Afterpath" is equivalent to the "Query-String", because it is the component after the "URL-Path". That means it is the part of the URL, after the question mark. (see section 2.1)

The character continuity rate is the longest sequence of alphabet, digits and special symbols in the URL. The equation is: character continuity rate = $(A + D + S)$/total length of URL. An example : abc567ti = $(3 + 3 + 1)/9 = 0.77$.

The filename component in the URL is the string that appears before the file extension of a file. In, for example, `http://www.example.com/dir/file.html` the filename is "file.html".

Table 3.3 provides a detailed description of the features and their corresponding types.

Word Vectorization / Word Embedding

Two vectorization methods are tested. **TF-IDF Vectorizer** [SJ72] in **feature_ex5** and **Count Vectorizer** in **feature_ex6**.

For TF-IDF, different parameters were used. We have not used any smoothing in the calculation, as this allows for a more strict calculation of IDF without any additive smoothing. We have also avoided normalization, especially since the length of the URLs does not differ significantly. To avoid any reduction in performance, we have capped the calculated features to 1000 features, as the number of features calculated for every instance can increase with the size of the dataset.

Similarly, for Count-Vectorizer, certain parameters need to be set. Since the URLs are primarily written in English, we have used a predefined set of English stop words. To ensure memory efficiency with larger datasets, we have used 32-bit float instead of 64-bit integer. Here as well, we have capped the features to 1000, due to the use of large datasets.

Feature	Description	Type
Entropy Domain	Calculates entropy (Detection of lookalike characters, e.g. CITI can be written as CIT1)	continuous value [0-1]
Arg Path Ratio	Length ratio of argument component (Query-string) to path component (URL-path) $\longrightarrow$ Division of length of argument and path component	continuous value [0-1]
Arg URL Ratio	Length ratio of argument component to URL string	continuous value [0-1]
Arg Hostname Ratio	Length ratio of argument component to hostname/host component	continuous value [0-1]
Path URL Ratio	Length ratio of path component to URL string	continuous value [0-1]
Char Cont Rate	Character continuity rate	continuous value [0-1]
Number Rate Filename	Proportion of digits in the filename component to the filename component length	continuous value [0-1]
Hostname URL Ratio	Length ratio of hostname component to URL string	continuous value [0-1]
Number Rate URL	Proportion of digits in URL string to length of URL string	continuous value [0-1]
Path Hostname Ratio	Length ratio of path component to hostname component	continuous value [0-1]
Number Rate Afterpath	Proportion of digits in afterpath to length of afterpath	continuous value [0-1]
Avg Path Token Length	Average length of tokens (maximal sequence of consecutive characters that are not delimiters) in path component	continuous value [0-1]

Table 3.3: Feature description and type for feature_ex2

Transformer

A transformer is used in this task only for tokenization and embedding creation. In the **feature_ex7** method, we are testing **Bidirectional Encoder Representations from Transformers (BERT)** [DCLT18b]. We will be using **bert-base-uncased** from Hugging Face [DCLT18a] for the required tasks. [JA23]

Dataset Splitting

The used datasets should be split in this task, as explained in subsection 2.6.1 into an initial training-set, test-set and unlabeled training-set. It is important to test whether a split of, for example, 80:10:10 or 10:80:10 is more useful. The ratio of the individual sets can be adjusted as needed. With less training-data, a model can be trained faster. The model is typically worse (produces low accuracy) but has more potential for improving with active learning. This strategy ensures that the initial trainings-data does not contain data irrelevant to the model. [Lon21]

In the Experimental Results chapter 4, we explain how the datasets are used. Further information about the actual splitting of the datasets is provided in the description of the individual experiments.

3.2.3 Machine Learning

We utilized two well-established algorithms, which have been shown to yield high accuracy results for machine learning problems: Random Forest (RF) [Ho95] and Stochastic Gradient Descent (SGD) [ZWSL10]. Both options are well-established in the literature and have been proven to produce high results on similar problems. The algorithms performed well and produced excellent results in testing on the used datasets. (reference for RF: [LBK21, ASBAK+23, MRL+16a]) (reference for SGD: [HY23, AAAS22])

Random Forest (RF) is a tree-based classifier and Stochastic Gradient Descent (SGD) can be used for various tasks depending on the type of loss function used. For the thesis, we utilize SGD with the “hinge” function, which allows us to use a linear SVM.

Additionally, both classifiers support incremental learning, which is essential for lifelong learning.

While it is possible to fine-tune the parameters of the machine learning classifier, our thesis's primary goal was to optimize the model through active learning rather than by adjusting the model parameters. This is something we plan to explore further in future work.

k-fold Cross-Validation:

After feature extraction, the k-fold Cross-Validation with $k = 5$ is applied. This is a common method in machine learning to verify a model using the training- and test-data. The created models can also be used for machine learning, but here, they are only used for validation. [BPM04], [NCB23]

3.2.4 Incremental Active Learning

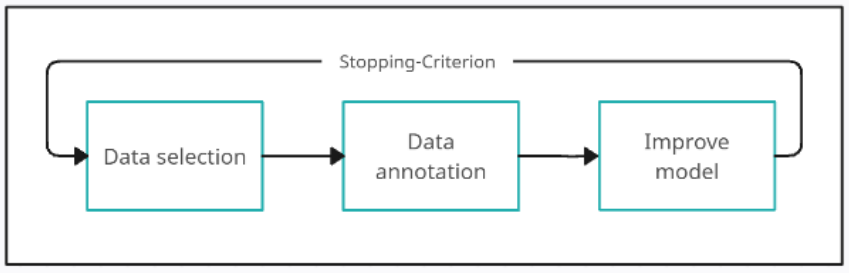

Figure 3.3: Incremental Active Learning Pipeline

The incremental active learning pipeline is shown in the Figure 3.3. The pipeline consists of a cycle, also known as an "Epoch", where data is selected using a query strategy, annotated, and used to improve the model. The active learning pipeline process is typically repeated over multiple epochs, until a stopping criterion is reached. In this case, the stopping criterion is the "Number of Epochs", which means a fixed number of epochs is performed. Once this fixed number of epochs is completed, the active learning process ends. [Set09]

Multiple query strategies are tested for the data selection on already trained models:

- Random-Sampling (Pool-based and Stream-based) [APH22, DW09]

- Uncertainty- (Least Confident), Margin- and Entropy-Sampling (Pool-based and Stream-based) [Set09, SACAF21]
- Query-by-Committee (QBC) → Max Disagreement-, Consensus Entropy- and Vote Entropy-Sampling [Set09, SACAF21]
- Clustering (Partitioning Around Medoids, Clustering Large Applications) [DW09, SR21]
- More strategies like Expected Model Output Change (EMOC) [FRD14]

EMOC was not further tested, because it is an unsuitable model for tasks with large datasets. It is built on Gaussian process models, and these have a time complexity of $O(n^3)$ and require $O(n^2)$ storage, where n represents the number of training examples. The primary issue with this model is the required memory. In active learning, the number of training examples and memory usage increases with every epoch. Although sub-sampling the data can be a potential solution, it remains a task for future work. [FRD14, Tit09]

This section does not include specific parameters for the query strategies and the active learning. They are specific to the experimental runs in the next section (chapter 4).

3.2.5 Lifelong Learning

To integrate the Lifelong learning technique into the thesis, a classifier needs to be capable of fitting new samples into a pre-trained model. In order to determine if this technique works, the Random Forest (RF) and Stochastic Gradient Descent (SGD) algorithms were utilized for testing.

RF has no particular function to do incremental learning, but with some parameter adjustments, new query samples can be fitted onto an existing model. On the other hand, SGD uses partial fitting, where the model learns new samples directly, as targeted by the Lifelong learning technique.

RF is a tree-based classifier. Therefore, it is important to test different models to compare various model types. Many classifiers from the literature are tree-based, which limits the variety of usable models. Some classifiers, like Logistic Regression

(LR) and Support Vector Machine (SVM), do not support incremental learning. Meanwhile, other classifiers can use partial fitting, but they cannot be used for probabilistic tasks (e.g., Model Perceptron, MultinominalNB). Therefore, the next part focuses only on RF and SGD.

Chapter 4

Experimental Results and Evaluation

In this chapter, we will provide an overview of the experimental settings and the datasets used in the thesis. Next, we will introduce two distinct approaches, along with their corresponding methods, parameters, and results. Finally, we will analyze the results in detail.

4.1 Settings

The experiments are performed in Python 3.9. The code is public available under [MM24]. The repository includes instructions on how to reproduce the experimental results using the code. Additionally, a **requirements.txt** file has been included, which lists all the packages and their versions necessary to run the code. It's important to note that some packages only work with specific versions of others, so using different versions or updating them can lead to errors.

The packages: **Scikit-Learn** provides the classifier, the methods for clustering, and some supporting functions. **modAL** provides functions for the remaining query strategies. **Matplotlib** is used for generating the plots, and some other packages provided offer additional supporting functions to aid in conducting the experiments.

For the machine learning and active learning tasks, a high-performance and high-memory cluster is used. The name of the cluster is "Kratos" and is a High-Performance Data Analysis Cluster (HDPA). The cluster is the main working tool used by the DLR in Jena. The main parts of the cluster are the 32 NVIDIA graphic processing units and up to 1.5 terabytes of RAM. [DLR20]

4.2 Datasets

Four URL datasets are utilized for the experiments, shown in the following table:

Dataset name	Class/Label	Count	Source	Reference
Urldata	All	450,176	[Kum19]	[RAA+22]
	Benign	345,738		
	Malicious	104,438		
Malicious_Phish	All	651,191	[Sid21]	[WWXZ22]
	Benign	428,103		
	Defacement	96,457		
	Malware	94,111		
	Phishing	32,520		
Phishing	All	549,346	[Tiw20]	[ZKV22]
	Benign	392,924		
	Malicious	156,422		
ISCX_2016	All	165,366	[MRL+16b]	[RSÁGVV23, MRL+16a]
	Benign	35,378		
	Defacement	96,457		
	Malware	11,566		
	Phishing	9,965		
	Spam	12,000		

Table 4.1: Initial datasets

4.3 Dataset Preprocessing

The datasets have been preprocessed to include only the URLs, labels (in string format), and results (as integer labels). This way, the experiment uses the complete URLs without relying on pre-calculated features or any other additional data. Additionally, all URLs are already labeled.

Before proceeding further, we need to refine the dataset as there is still an issue that needs to be addressed. When we tested the approach with single datasets,

the accuracy results of the pre-trained models were already very high, ranging from 90%, up to almost 100%. That's great for the machine learning models because they performed their job well. Testing on unknown URLs that had similar features and structure as the initial training data produced excellent results. However, the issue arose when we tested the models on novel datasets (datasets that were not involved in the initial training). Here, the achieved accuracies were lower than 50%. This problem called "Domain Shift" arises when the data structure of a domain changes, and the machine learning model may not be well-suited to handle it.

During the experiment, it was observed that training the model initially with more URLs resulted in reduced accuracy performance. However, this issue is not being further investigated at the moment. To prevent this problem, the training and testing process now involves the use of multiple datasets instead of just one.

The data splitting process is implemented as explained in subsection 2.6.1. One dataset is used for the initial model training; the remaining datasets are combined and used for the unlabeled training set (or unlabeled pool) and test set. This approach allows for a more realistic scenario to be set up: the model is trained on the initial training set, which means it is trained on the features and structure of the corresponding URLs. As new malicious URLs appear, their features and structure change. The remaining set mimics these new URLs, as they most likely have a different structure than the initial training set. As a result, the actual model is not well-fitted to the test set, which leads to low accuracy. The unlabeled pool can be used for active learning to identify the most uncertain URLs and refine the model to increase the accuracy of the test set.

The dataset refinement is shown in Table 4.2 and described in detail below:

In the following is spoken about two-class, four-class and five-class datasets. These classes relate to the labels that can be assigned to URLs in URL detection. You can see all possible classes in Table 4.1. A two-class dataset typically has the classes "Benign" and "Malicious". A four-class dataset includes "Benign", "Defacement", "Malware" and "Phishing" classes. The five-class dataset extends the four-class dataset by adding the "Spam" class.

We have decided to work with three datasets: a two-class, a four-class, and a five-class dataset. This will enable us to test if a higher class count makes it more challenging for a model to label URLs correctly. To create these datasets, we will need

to reduce the "Malicious_Phish" dataset to a two-class dataset and the "ISCX_2016" dataset to a four-class dataset.

As mentioned earlier, every named dataset will have its corresponding training set (i.e., a two-class training set, a four-class training set, or a five-class training set). Later, the remaining sets will be divided into an unlabeled pool and a test set.

Dataset	Data
two_class_training	Urldata
two_class_remaining	Malicious_Phish (reduced), Phishing, ISCX_2016 (reduced)
four_class_training	Malicious_Phish
four_class_remaining	ISCX_2016 (reduced)
five_class_training	ISCX_2016 (30%)
five_class_remaining	ISCX_2016 (70%)

Table 4.2: Refined datasets

Due to the limited amount of data available of five-class datasets, only one dataset has been split into a training set and a remaining set. This decision was made after the initial dataset selection. The four selected datasets should be used for the experiments, and no additional data should be used.

After refining the datasets listed above, all of them need to be preprocessed as explained in subsection 3.2.1. During the experiment, the datasets are tested unprocessed and preprocessed. This means that every training set and remaining set will be preprocessed. The reason for this is to compare the results and see if preprocessing improves model training or active learning.

4.4 First Active Learning Experiment

We have decided to use the three refined datasets described in section 4.3 for this experiment. The datasets are preprocessed to obtain the original and preprocessed versions of the two-class, four-class, and five-class datasets. So, in total, 6 datasets are tested.

To enable machine learning, it is necessary to split the datasets, as explained in Table 3.2.2. A split of 80:10:10 is not possible with the refined datasets, but the distribution can be kept similar. The training set and unlabeled pool should be larger than the test set. While the training sets are already available through dataset refinement, the remaining sets need to be split into unlabeled pools and test sets. A test size of 30% is used, which means 70% of the remaining set is the unlabeled pool and 30% the test set. The URLs are randomly distributed into different sets.

In the next step, the seven feature extraction methods (subsection 3.2.2) are applied to the split datasets. The two selected classifiers, Random Forest (RF) and Stochastic Gradient Decent (SGD) are trained and evaluated (subsection 3.2.3).

The accuracy results of the pre-trained models are shown in the following tables: RF in Table 4.3 and SGD in Table 4.4. The best results per dataset are marked in green. Additionally, the feature extraction methods are abbreviated (e. g. feature_ex1 = fe1).

Dataset	fe1	fe2	fe3	fe4	fe5	fe6	fe7
two_class_dataset	0.255	0.419	0.275	0.371	0.701	0.715	0.590
two_class_dataset (prepr.)	0.607	0.627	0.615	0.269	0.337	0.350	0.611
four_class_dataset	0.307	0.249	0.289	0.276	0.300	0.269	0.294
four_class_dataset (prepr.)	0.326	0.366	0.349	0.416	0.364	0.363	0.234
five_class_dataset	0.900	0.944	0.951	0.966	0.972	0.969	0.479
five_class_dataset (prepr.)	0.834	0.918	0.931	0.958	0.966	0.969	0.509

Table 4.3: Accuracy results of pre-trained RF model after feature extraction

Dataset	fe1	fe2	fe3	fe4	fe5	fe6	fe7
two_class_dataset	0.618	0.641	0.539	0.367	0.407	0.372	0.627
two_class_dataset (prepr.)	0.637	0.628	0.624	0.258	0.421	0.295	0.627
four_class_dataset	0.787	0.204	0.206	0.222	0.309	0.309	0.147
four_class_dataset (prepr.)	0.282	0.237	0.308	0.395	0.357	0.351	0.147
five_class_dataset	0.626	0.747	0.746	0.747	0.959	0.956	0.581
five_class_dataset (prepr.)	0.645	0.637	0.701	0.777	0.951	0.950	0.581

Table 4.4: Accuracy results of pre-trained SGD model after feature extraction

To determine which feature extraction method is the best, two measurements are taken into account. The first measurement is the average of a method over six datasets. For example, the feature_ex1 method has an average accuracy value of $(0.255 + 0.607 + 0.307 + 0.326 + 0.900 + 0.834) / 6 = 0.538$ for the Random Forest algorithm. The second measurement is to determine how often an extraction method produces the highest result for a dataset (indicated by green text).

feature_ex5 got the highest average for RF, and feature_ex1 the highest for SGD. Each method also got the highest results for a dataset on three different datasets. When the average result values are summed up and averaged, feature_ex5 has an average value of 0.586, and feature_ex1 has an average value of 0.568. Therefore, it can be concluded that the TF-IDF Vectorization, which is feature_ex5, has the best overall results. This method will be used in the further process of feature extraction for model training and active learning.

4.4.1 Additional Information

As described in subsection 3.2.4, incremental active learning is tested with four sampling approaches. The first two are pool- and stream-based sampling with the strategies: Random-, Uncertainty-, Margin- and Entropy sampling. The next technique is Query by Committee (QBC) with the strategies: Max Disagreement-, Consensus Entropy- and Vote Entropy-Sampling. The last one is Clustering, where the CLARA algorithm is used. QBC and Clustering are also pool-based strategies, but for the experiments, they are divided into different approaches.

If random sampling is used, it is utilized as "baseline" in comparison to the other strategies. How QBC and Clustering are utilized, is further explained in the experiments itself.

Lifelong learning is used, as described in subsection 3.2.5, to further improve the pretrained models and, at the same time, maintain a high level of efficiency through the active learning process. It is important to mention that the used RF algorithm is not optimally implemented for lifelong learning. Random Forest is a tree-based model that consists of a number of trees containing nodes that are connected with each other. To fit new samples in every active learning epoch, the already created trees of the model should be extended. That implies the number of branches at the nodes or the level of the trees should be increased. With more branches, the trees get wider, and with more levels, the trees get deeper. In both ways, more complexity is added. The currently used algorithm fits the new data only on new trees. That indicates the number of trees or estimators needs to be increased per epoch. This way, some risks can appear, like higher computational cost or overfitting. The algorithm is still being used, because lifelong learning can be fulfilled nevertheless, to observe the quality of the results. After conducting tests, we observed that reducing the number of initial trees from 100 to 10 resulted in better performance improvements for the model. As a result, the initial tree size was adjusted accordingly.

Pool-based active learning requires a sampling size, that determines how many instances from the unlabeled pool are queried per epoch. In the case of the RF algorithm, it is essential for lifelong learning to have at least one sample of each class. Therefore, a sampling size is assigned for each experiment, except for stream-based sampling, which indicates the minimum number of samples that must be queried. If not at least one sample from each class is present in the newly queried samples per epoch, the sampling size is increased. This ensures that a different number of samples could possibly be queried every epoch. So, for each query strategy, the active learning could be calculated with a different sampling size.

The active learning process uses different sampling sizes based on the size of the unlabeled pool. The proportions queried per epoch are 0.01%, 0.1% and 1%. This is necessary because the active learning is performed on two-class, four-class and five-class datasets with varying unlabeled pool sizes. Three different sampling sizes were selected to compare their impact on the models. With 50 epochs, at least 0.5%,

5%, and 50% of the unlabeled pool is trained. However, due to the lifelong learning problem with RF, the proportion may change during the process.

In stream-based sampling, one sample at a time is examined to determine whether it should be queried. However, this approach does not work well with RF and lifelong learning because gathering enough samples for each class is impossible. To address this, stream-based sampling is tested only with incremental learning when using RF. On the other hand, SGD is tested with both incremental and lifelong learning.

In active learning, the aim is to select the most uncertain samples from a large unlabeled pool. However, not all the queried samples are labeled, even though a large number of new unlabeled samples are queried during the active learning process. For instance, if there are 900,000 instances in the unlabeled pool and 5% of the samples are queried, then 45,000 instances will be queried. The instances need to be labeled, which can be done by a human expert. However, labeling 45,000 URLs is very laborious. So, a limit of 2000 newly queried URLs is set for labeling. This limit is represented in the plot by a purple dotted line. The line can be used to compare different query strategies and determine which strategy yields the best results when it reaches the limit.

Before SGD could be used, with k-fold cross-validation (subsection 3.2.3), different usable loss functions were tested. Among these, "hinge" was found to be the most stable function, resulting in a linear SVM model. The results of SGD with this loss function for machine learning training and evaluation were quite satisfactory. The problem of "hinge" is that it does not support probabilistic classification [ZWSL10]. Therefore, the only query strategies that are compatible, are random sampling (pool-based and stream-based) and clustering. Nevertheless, RF was tested with every previously mentioned strategy.

In the comparison of the active learning results, four different measurements are considered to compare the results. These are the **maximum accuracy** achieved over the complete active learning process, also known as "Max Acc.". The **accuracy at the labeling limit** (AaLL) represents the accuracy result at the labeling limit of 2000 or just before the limit is reached (see subsection 4.4.1 for further explanation). The **area under curve** (AUC) represents the area under the active learning curve [LWLZ18] and this value is normalized as different sampling sizes are used. The

last measurement is the **total difference** (TD). This is the difference between the accuracy of a model at the start and end of the active learning process.

To simplify the presentation of the experimental results, we will reduce the number of used datasets or sampling sizes in the following shown results. The different datasets are tested to indicate if a higher class count makes it more challenging for a model to label URLs correctly. If that is not the case, further tests are only performed on the two-class dataset. This matches the real-world scenario of two classes ("Benign" and "Malicious"). Additionally, the sampling sizes are compared to discover differences and to find the overall best to improve a model.

4.4.2 Pool-based Sampling

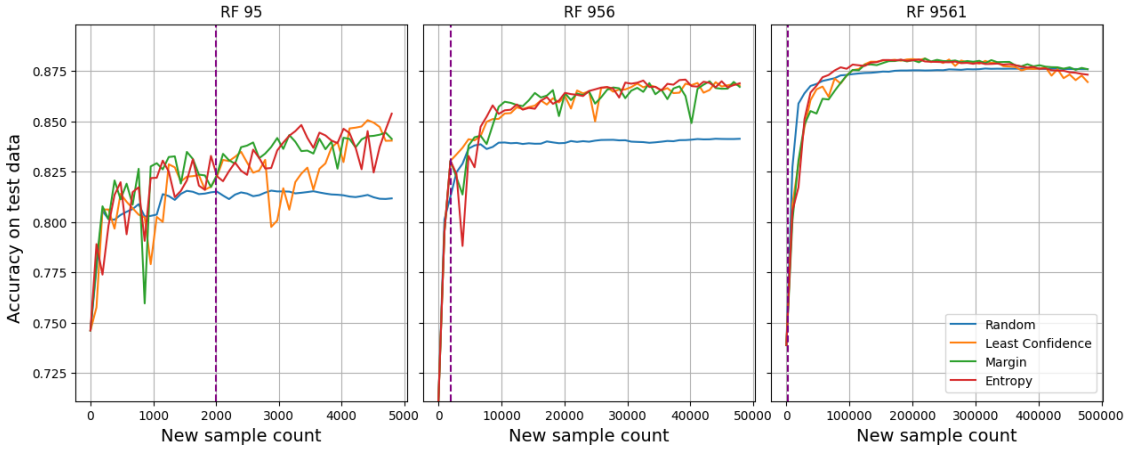

Figure 4.1: Pool-based Active learning with Random Forest, the two-class dataset and 50 epochs. Sampling size comparison: 95 vs. 956 vs. 9561

The accuracy results of active learning with pool-based sampling are presented in Figure 4.1. For the evaluation, the two-class dataset is used with Random Forest as the machine learning model. Four query strategies, namely Random-, Uncertainty/Least Confident-, Margin-, and Entropy-Sampling are compared.

In the three plots from left to right, the sampling sizes 95 (0.01%), 956 (0.1%), and 9561 (1%) are used. However, it is important to note that these are only the minimum values, and the actual sampling size per epoch may be higher. In the single plots, the query strategies have the same number of new queried samples. But the sizes per plot vary greatly. For the "RF 95" plot, it is 4800 samples, while

for the "RF 956" plot, it is 47,850 samples, and for the "RF 9561" plot, it is 478,100 samples in total.

It appears that the active learning process becomes more reliable as more samples are queried. However, as the number of newly queried samples increases, the steps become larger, which makes the fluctuations look smaller in the plots.

As mentioned before, the models that worked with random sampling are the baseline, and in all the cases, they produced a significant increase in the first epochs but quickly reached a plateau and did not increase much further.

	Max Acc.	AaLL	AUC	TD
RF 95 + Random	0.81	0.81	0.81	0.06
RF 95 + Entropy	0.85	0.83	0.82	0.10
RF 956 + Random	0.84	0.81	0.83	0.13
RF 956 + Entropy	0.87	0.83	0.85	0.15
RF 9561 + Random	0.87	0.73	0.87	0.13
RF 9561 + Entropy	0.88	0.73	0.87	0.13

Table 4.5: Accuracy measurements for Figure 4.1. Baseline with all sampling sizes compared.

In Table 4.5, the maximum accuracy (Max Acc.), accuracy at the labeling limit (AaLL), area under curve (AUC), and total difference from start to end (TD) are displayed. There is shown, that the AUC is similar to the maximum accuracy in all plots. As the sampling size increases, all measurements also increase. However, if the sampling size becomes too large, the accuracies first remain the same and then decrease. This implies that the models become worse. This problem is illustrated by "RF 9561", where the models produce a higher maximum accuracy than in the other plots, but the accuracy declines until the model is equal to or worse than the baseline model. The reason for that could be that the most uncertain instances are already trained, and at this point, new instances only make the model worse.

All the models start with the same accuracy value and tend to improve over time. With the exception of the baseline, all other methods produce similar results in the active learning process. However, the models continue to improve and are generally better than the baseline, except for "RF 9561" in the last epochs. The overall best method of these three is Entropy Sampling, which is also compared in Table 4.5

with the baseline. The measurements show that Entropy Sampling improves the model better than the baseline.

The accuracy at the labeling limit indicates a problem with a large sample size. For "RF 95" and "RF 956", the accuracy is 81–83%, while for "RF 9561", the accuracy is 73%. The pre-trained models already have an accuracy of 73%. The sampling size is higher than the limit, so the model cannot be fitted before the limit is reached. In practical cases, it's best to use a sampling size that's under the limit to fit new instances and improve the model earlier.

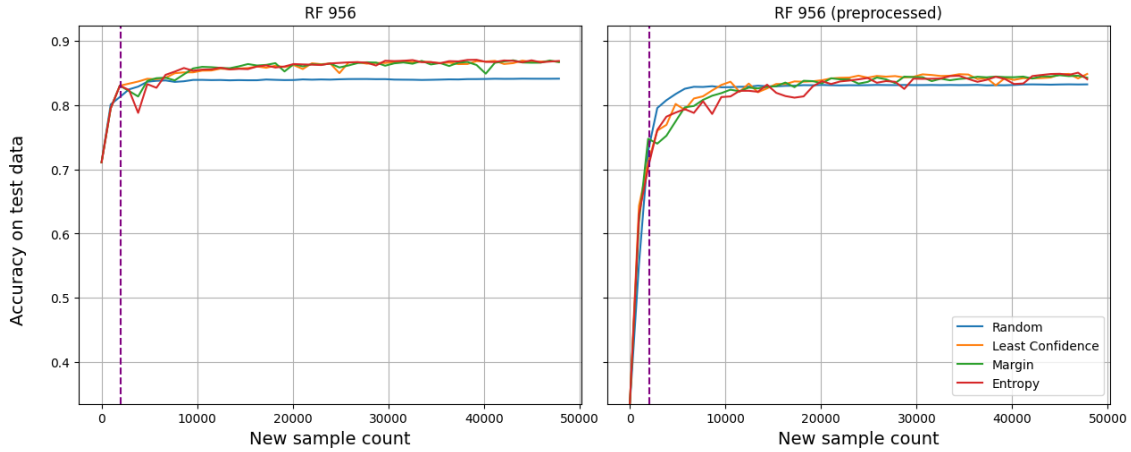

Figure 4.2: Pool-based Active learning with Random Forest, a sampling size of 956 and 50 epochs. Comparison between the use of two-class- and preprocessed two-class datasets

Figure 4.2 presents two plots. There on RF models, active learning is applied with a sampling size of 956 and 50 epochs. "RF 956" uses the two-class dataset and the other plot the preprocessed two-class dataset. The same number of newly queried samples, 47,850, is used in both cases.

	Max Acc.	AaLL	AUC	TD
RF 956 + Random	0.84	0.81	0.83	0.13
RF 956 (preprocessed) + Entropy	0.87	0.83	0.85	0.15
RF 956 + Random	0.83	0.72	0.81	0.49
RF 956 (preprocessed) + Entropy	0.85	0.70	0.81	0.50

Table 4.6: Accuracy measurements for Figure 4.2. Baseline and best method for both plots are compared.

The progression of the accuracy curve for the models in both plots is similar. However, the accuracy results for the models in "RF 956" are higher than in "RF 956 (preprocessed)". Moreover, the baseline as shown in Table 4.6 is also better. The model that uses Entropy Sampling in "RF 956" has the best performance, while in "RF 956 (preprocessed)" there is no clear best method. The model accuracies are less fluctuating than in the other plot. Furthermore, the baseline is nearly as effective as the other methods in improving the models.

For the other datasets and sampling sizes, the same effect appears; the models operating with the preprocessed datasets are worse.

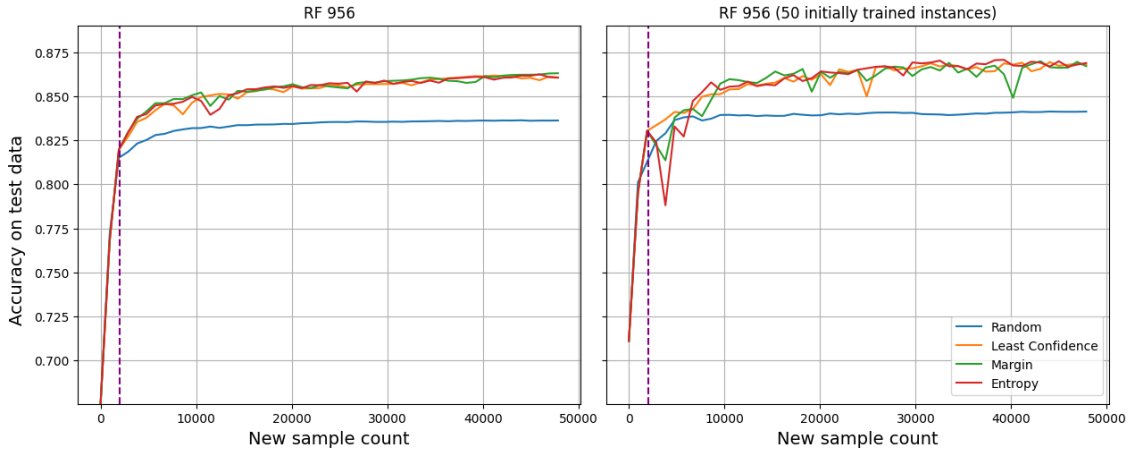

Figure 4.3: Pool-based Active learning with Random Forest, the two-class dataset and 50 epochs. Full-trained RF models vs. RF models trained with fewer data. Sampling size of 956.

The plots shown in Figure 4.3 illustrate the process of active learning using pool-based sampling. The experiments were conducted using a two-class dataset and 50 epochs. The Random Forest models shown in the figures were trained using a sampling size of 956, and the same models were also trained using the same sampling size, but only on 50 randomly selected instances instead of the full initial training set.

Both plots indicate similar accuracy results, and the curves progress in a similar way. The Max Acc., AaLL, and AUC are also similar. In some cases, the results for the models with less training data are even better. Additionally, the number of new queried samples is the same. Similar results were obtained for the other sampling sizes and datasets as well.

Changing the size of the training set would be, in this case, a great idea. With less training data, the computational resources get minimized, and the efficiency rises for the initial training and active learning. As previously mentioned, having more training data can result in reduced model performance, which is evident in Figure 4.3. Moreover, if newly queried samples are trained during active learning, they have a greater impact on the model. This can be observed in the right plot, where the accuracy curves have larger fluctuating changes.

The SGD model was also tested with active learning. As previously told, only random sampling can be used as a query strategy, which declines a proper comparison. Furthermore, the accuracy results fluctuate heavily. It is possible that the models require additional fine-tuning. In the comparison of a fully trained SGD model and an SGD model with less initial training data, the same effect appeared as for Random Forest.

The actual plan of the first approach is changed. the planned experiments for active learning are utilized with the same settings, but the initial training set is reduced to 50 random instances from the actual training set.

4.5 Second Active Learning Experiment

The second experimental approach is similar to the first one. All three refined datasets are used, but data preprocessing is not used further based on the results of experiment one. The results of the pre-trained models were lower than those of the models with the original datasets, and the active learning results were also worse.

In the first experiment, we trained the models using the entire training set, which consisted of over 100,000 labeled URLs. In a real-world situation, a big dataset of URLs can be given, but these URLs are typically unlabeled. Labeling these URLs requires the expertise of a human. Using around 50 to 100 URLs as an initial training set is more realistic. For this experiment, we used the same initial training set but trained the model using only a random sample of 50 URLs from the training set.

We have discovered that the initial training with a large number of URLs, such as 100,000, is not effective. This is because the model is already trained with a

significant amount of data, and adding more URLs, such as 1000 newly queried URLs, does not make a big difference in the model's continuous learning over the epochs. Furthermore, as explained in section 4.3, the more data is initially trained, the worse the model's accuracy. One reason for this could be that the model is trained with URLs that have no positive impact on improving the model for better URL labeling.

In this experiment, we have utilized seven different feature extraction methods along with two classifiers, RF and SGD.

For that, again, tables of accuracy results for both classifiers are calculated. RF is shown in Table 4.7 and SGD in Table 4.8. The best results per dataset are marked in green. Additionally, the feature extraction methods are abbreviated (e. g. feature_ex1 = fe1).

Dataset	fe1	fe2	fe3	fe4	fe5	fe6	fe7
two_class_dataset	0.624	0.514	0.650	0.364	0.391	0.361	0.589
four_class_dataset	0.232	0.215	0.452	0.325	0.160	0.151	0.186
five_class_dataset	0.599	0.676	0.723	0.773	0.673	0.676	0.518

Table 4.7: Accuracy results of pre-trained RF model after feature extraction

Dataset	fe1	fe2	fe3	fe4	fe5	fe6	fe7
two_class_dataset	0.628	0.608	0.601	0.338	0.746	0.727	0.408
four_class_dataset	0.771	0.440	0.388	0.298	0.167	0.171	0.166
five_class_dataset	0.657	0.722	0.704	0.724	0.628	0.696	0.560

Table 4.8: Accuracy results of pre-trained SGD model after feature extraction

To determine which feature extraction method is the best, the two measurements are again considered. The first measurement is the average of a method over six datasets. The second measurement determines how often an extraction method produces the highest result for a dataset (indicated by green marks).

In the evaluation of two different classifiers, feature_ex3 yielded the highest average result for RF, while feature_ex1 yielded the highest for SGD. Additionally, feature_ex3 had two instances of the highest result on a dataset, while feature_ex1

had only one. We will consider, in this case again, the average accuracy on both classifiers. For feature_ex3 an average result of 0.586 is reached and for feature_ex1 an average of 0.585. To conclude, both got similar results in the measurements, but feature_ex3 had a higher frequency of obtaining a higher result on a dataset. This feature extraction method combines the features of feature_ex1 and feature_ex2, shown in subsection 3.2.2. This method is further used to calculate the features used in the model training and active learning.

4.5.1 Additional Information

The second experiment used fewer data and skipped the data preprocessing step, but otherwise had the same additional information as the first experiment (see subsection 4.4.1).

4.5.2 Pool-based Sampling

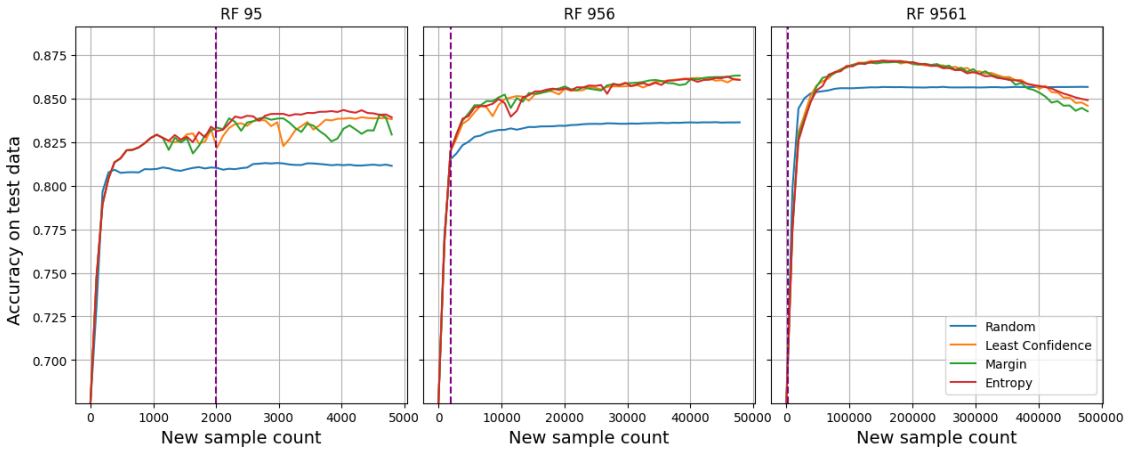

Figure 4.4: Pool-based Active learning with Random Forest, the two-class dataset and 50 epochs. Sampling size comparison: 95 vs. 956 vs. 9561

The accuracy results of active learning with pool-based sampling are presented in Figure 4.4. For the evaluation, the two-class dataset is used with Random Forest as the machine learning model. Four query strategies are utilized: Random-, Uncertainty/Least Confident-, Margin-, and Entropy-Sampling.

The experiment is equivalent to the one displayed in the first experiment section in Figure 4.3 where sampling sizes of 95, 956, and 9561 were compared. For the query strategies, the same sample count is used per epoch. For the "RF 95" plot, it is 4800 samples, while for the "RF 956" plot, it is 47,850 samples, and for the "RF 9561" plot, it is 478,100 samples in total.

	Max Acc.	AaLL	AUC	TD
RF 95 + Random	0.81	0.81	0.80	0.13
RF 95 + Entropy	0.84	0.83	0.82	0.16
RF 956 + Random	0.83	0.81	0.83	0.16
RF 956 + Entropy	0.86	0.81	0.85	0.18
RF 9561 + Random	0.85	0.67	0.85	0.18
RF 9561 + Entropy	0.87	0.67	0.85	0.17

Table 4.9: Accuracy measurements for Figure 4.4. Baseline and the best method with all sampling sizes are compared.

In Table 4.9, the maximum accuracy (Max Acc.), accuracy at the labeling limit (AaLL), area under curve (AUC), and total difference from start to end (TD) are displayed.

As stated in section 4.4, the curve progressions are comparable to those of the same experiment in the first experiment section (Figure 4.3). Only the accuracy results and measurements are different, but for further evaluation, please refer to that section.

In Figure 4.5 we observe again active learning with pool-based sampling. Random Forest and the same query strategies are used, but now with the four-class dataset.

The accuracy results for the different query strategies in this figure differ significantly from those of the two-class dataset. The curves here have a distinct progression. The curves of "RF 45" and "RF 455" are relatively similar. Throughout the process, Least Confidence and Entropy Sampling appear to be weaker strategies than Random Sampling, as they more unstable over the epochs. Conversely, Margin Sampling yielded better results than Random Sampling.

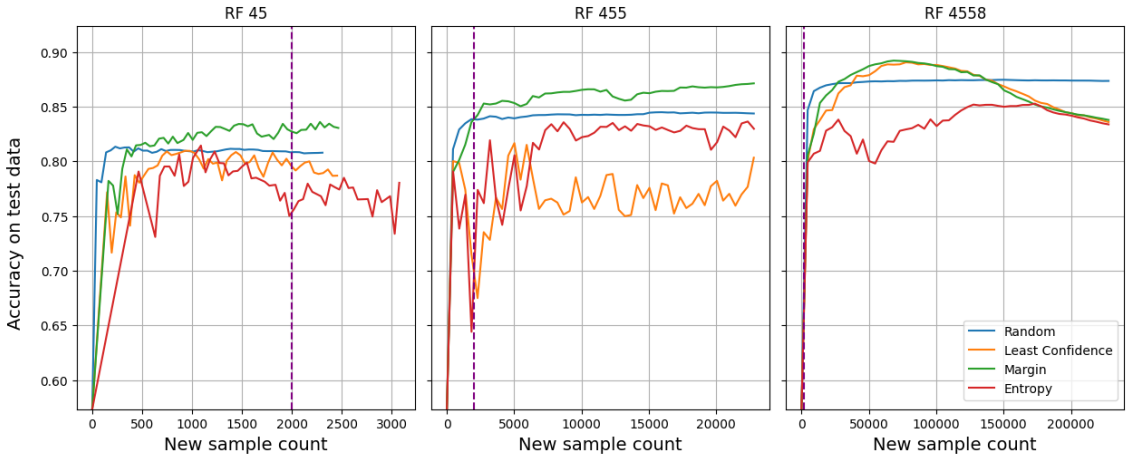

Figure 4.5: Pool-based Active learning with Random Forest, the four-class dataset and 50 epochs. Sampling size comparison: 45 vs 455 vs 4558

	Max Acc.	AaLL	AUC	TD
RF 45 + Random	0.81	0.80	0.80	0.23
RF 45 + Margin	0.83	0.82	0.81	0.25
RF 455 + Random	0.84	0.83	0.83	0.27
RF 455 + Margin	0.87	0.83	0.85	0.29
RF 4558 + Random	0.87	0.57	0.86	0.30
RF 4558 + Margin	0.89	0.57	0.86	0.26

Table 4.10: Accuracy measurements for figure Figure 4.5. Baseline and best method compared with sampling sizes 45, 455 and 4558.

Table 4.10 demonstrates that Margin Sampling performs at least as good as the baseline, but in most measurements, better. The total difference is also higher than that of the baseline and increases with the sampling size.

In terms of curve progression, "RF 4558" has the same pattern as "RF 9561", except that Entropy Sampling does not increase over 85% and decreases after some epochs. It performs worse than the baseline and obtains a similar curve to Least Confidence- and Margin Sampling in the last epochs. Least Confidence- and Margin Sampling reaches nearly 90% in the maximum value. However, the sampling size is again above the labeling limit, which means that it may not be practical to use in real-life scenarios.

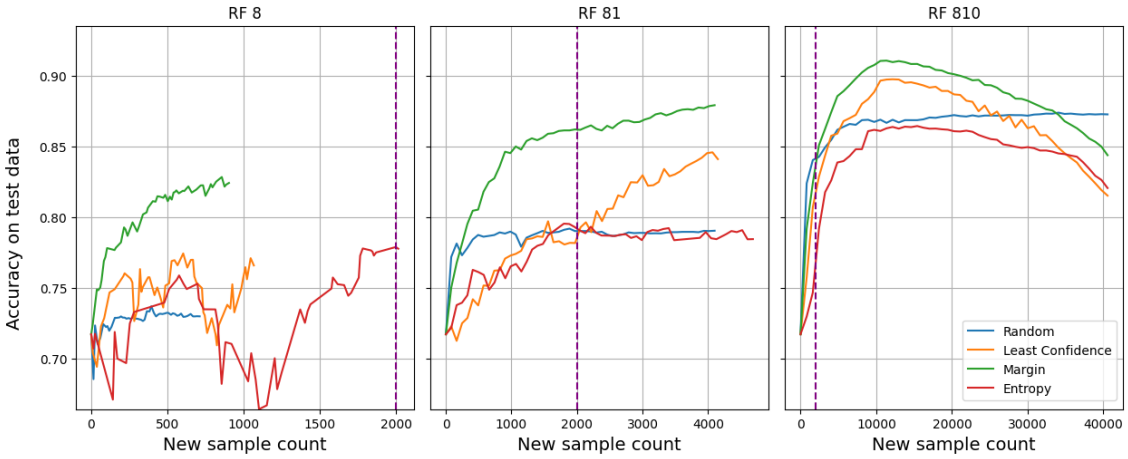

Figure 4.6: Pool-based Active learning with Random Forest, the five-class dataset and 50 epochs. Sampling size comparison: 8 vs 81 vs 810

Figure 4.6 also represents active learning with pool-based sampling, where Random Forest and the same query strategies are used, but with a five-class dataset. It's worth noting that the test set is taken from the same dataset as both training sets, which means that the test set has a similar structure and features to the instances the models are trained on before active learning.

For some models in the plots, active learning does not reach the labeling limit, but that is not problematic. "RF 810" has a smaller sampling size than the labeling limit, but has a similar curve progression as for the two-class- and four-class datasets. The main difference is that the models can be trained before the limit is reached, resulting in a higher accuracy.

In the plots "RF 8" and "RF 81", all models reach similar or higher results than the baseline model. The models with Margin Sampling got the best results. Especially in "RF 8", the results of Margin Sampling are higher, although it has reached almost half the count of new samples of the labeling limit.

Table 4.11 presents, that the models with Margin Sampling have higher accuracy for almost all four measurements than the baseline. Generally, the accuracy values increase as more samples are queried per epoch. However, with a sampling size of 455, Margin Sampling achieves the highest accuracy at the labeling limit, making it more practical for real-world use cases. It's worth noting that "RF 810" shows that a larger sampling size can increase accuracy values, but at the cost of sabotaging

	Max Acc.	AaLL	AUC	TD
RF 8 + Random	0.73	0.72	0.72	0.13
RF 8 + Margin	0.82	0.82	0.80	0.15
RF 81 + Random	0.79	0.79	0.78	0.16
RF 81 + Margin	0.87	0.86	0.85	0.18
RF 810 + Random	0.87	0.84	0.86	0.18
RF 810 + Margin	0.91	0.82	0.88	0.16

Table 4.11: Accuracy measurements for Figure 4.6. Baseline and best method compared with all sampling sizes.

the model's improvement over time. Therefore, the total difference of the Margin Sampling model is worse than the baseline.

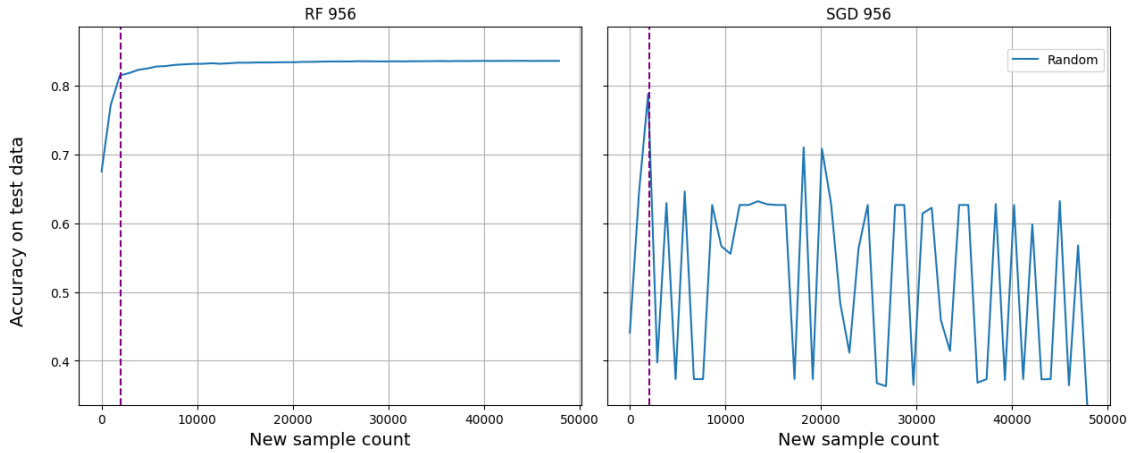

Figure 4.7: Pool-based Active learning with RF and SGD, the two-class dataset and 50 epochs. RF and SGD comparison with a sampling size of 956

In Figure 4.7, we compare the active learning process between Random Forest (RF) and Stochastic Gradient Descent (SGD) on a two-class dataset using pool-based sampling. As explained earlier, SGD cannot work with probabilistic query strategies, so only random sampling is utilized and compared for RF and SGD. During the active learning process, we observe that SGD shows inconsistency and randomness, which leads to fluctuating accuracy results. However, as we explained in the first experiment section, fine-tuning of the SGD models could be a solution to overcome this issue.

	Max Acc.	AaLL	AUC	TD
RF 95 + Random	0.83	0.81	0.83	0.16
SGD 95 + Random	0.78	0.78	0.52	-0.10

Table 4.12: Accuracy measurements for Figure 4.7. The baseline of RF and SGD compared with a sampling size of 95

Table 4.12 illustrates that, in general, RF achieved greater improvement, as evidenced by higher measurement values. SGD produced similar results on the other datasets. Because of the strong fluctuations in accuracy through active learning, the TD can be highly positive or highly negative. In general, the SGD model does not improve over time.

4.5.3 Stream-based Sampling

As previously explained, Random Forest cannot be utilized with stream-based sampling in combination with lifelong learning. Additionally, for SGD only random sampling can be used.

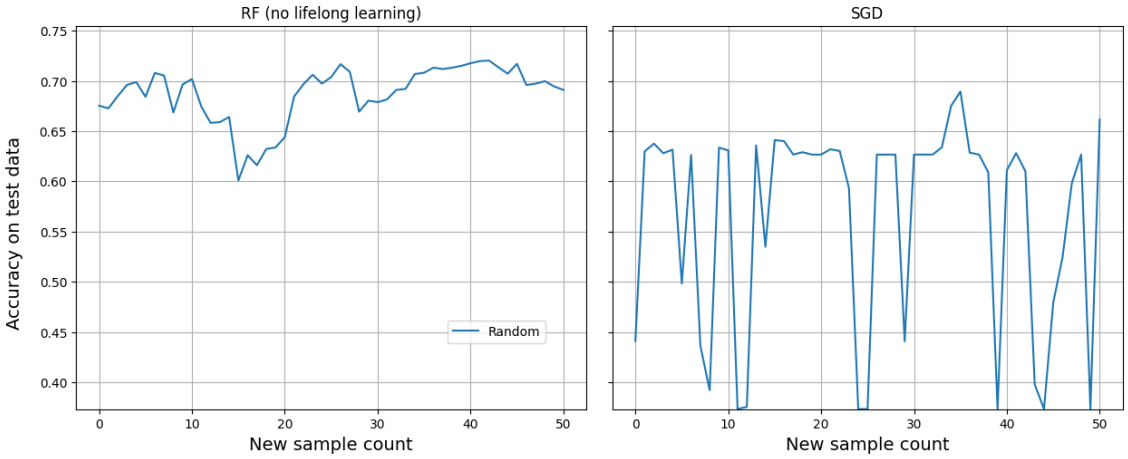

Figure 4.8: Stream-based Active learning with RF and SGD, the two-class dataset and 50 epochs. RF (no lifelong learning) and SGD comparison

Figure 4.8 displays stream-based active learning of RF compared to SGD. For RF, incremental learning is utilized with random sampling, and for SGD, lifelong learning with random sampling. Active learning is executed with the two-class dataset and 50 epochs.

	Max Acc.	AaLL	AUC	TD
RF + Random	0.72	0.69	0.68	0.01
SGD + Random	0.68	0.66	0.56	0.22

Table 4.13: Accuracy measurements for Figure 4.8. The baseline of RF and SGD compared with a sampling size of 95

Table 4.13 displays that the RF baseline delivered better results than the SGD baseline. In addition, the RF baseline had a small Total Deviation (TD) whereas the SGD baseline had a high TD. The accuracy of the SGD model varied a lot over the epochs, but there was no general positive increase displayed, which means that the TD could also be the same value in the negative area, similar to pool-based sampling.

The SGD models working with four-class and five-class datasets displayed a similar curve progression as the model with the two-class dataset has shown here. The accuracy results are only slightly different.

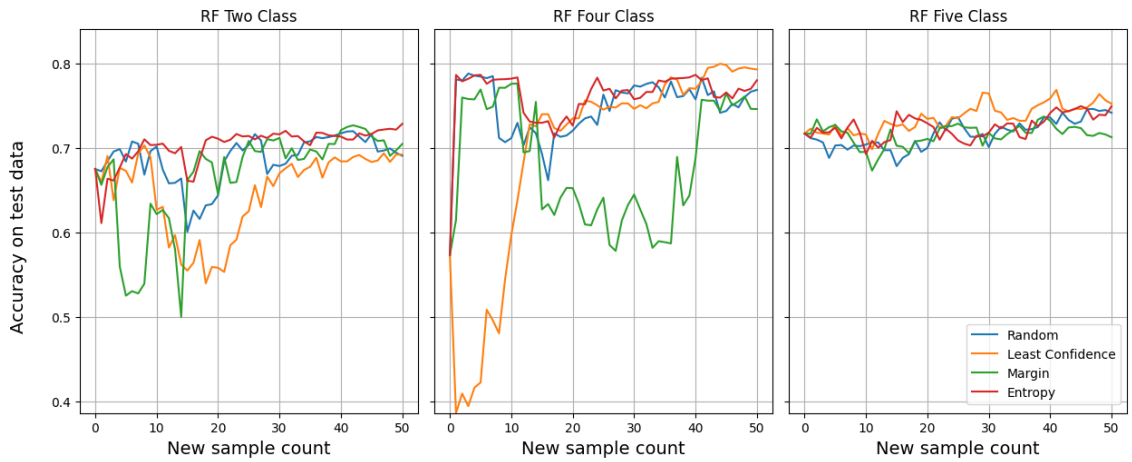

Figure 4.9: Stream-based Active learning with RF, the two-class, four-class, and five-class dataset and 50 epochs.

The graphs in Figure 4.9 showcase the results of stream-based active learning with RF on all three datasets. These results were obtained using 50 epochs and only incremental learning.

	Max Acc.	AaLL	AUC	TD
RF Two Class + Random	0.72	0.69	0.68	0.01
RF Two Class + Entropy	0.72	0.72	0.70	0.05
RF Four Class + Random	0.78	0.76	0.74	0.19
RF Four Class + Entropy	0.78	0.78	0.76	0.20
RF Five Class + Random	0.74	0.74	0.71	0.02
RF Five Class + Entropy	0.74	0.75	0.72	0.03

Table 4.14: Accuracy measurements for Figure 4.9. Baseline and best method compared.

It has been observed that models using Least Confidence- and Margin Sampling techniques display fluctuating accuracy changes during active learning for both two-class and four-class datasets. These models experienced a significant decrease in accuracy but eventually ended up with a value similar to the other models. Models that did not use the baseline method did not perform significantly better during active learning. Some strategies even resulted in worse outcomes than the baseline. The measurement results for the four-class dataset were higher than for the other datasets, with the models showing greater improvement, possibly due to the initial low accuracy.

Entropy Sampling was found to be the most effective method, consistently producing great improvements in stream-based sampling contexts. Generally, the models saw an improvement of no more than 5% with stream-based sampling, except for the four-class dataset, where the initial accuracy was very low in comparison.

4.5.4 Query-by-Committee

As mentioned earlier, the query strategies used for Query-by-Committee (QBC) cannot be employed with SGD. Therefore, only the RF model can be utilized. For QBC, multiple models are required to create a committee (see the definition of QBC in subsection 2.7.2), so two different RF models are created. Both use a tree size of ten in the initialization and for one model, the criterion is changed from “Gini impurity” to “Logarithmic loss”. The criterion is used to determine how to split the data at each step of building a tree. Both models are also provided with different

initial training data. From the initial training set, two sets of 50 instances are randomly selected.

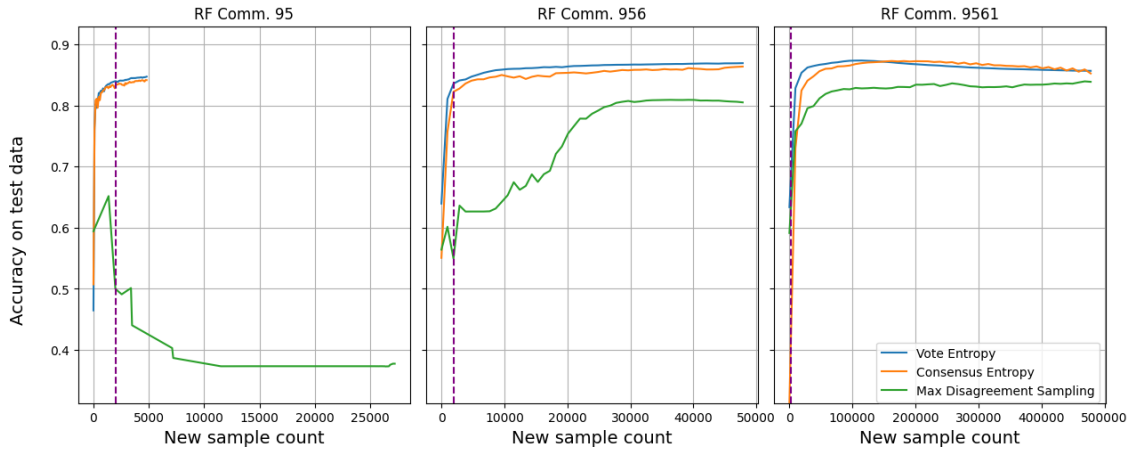

Figure 4.10: Active learning with Query-by-Committee, the two-class, four-class, and five-class dataset and 50 epochs. Sampling size comparison: 95 vs. 956 vs. 9561

In Figure 4.10 active learning with query-by-committee is displayed. There the explained committee is utilized with the two-class dataset and 50 epochs. The new queried samples per epoch are 27,194 samples for "RF Comm. 95", 47,850 for "RF Comm. 956" and for "RF Comm. 95" the sample count is 478,100. The same sample count for all models of one plot.

	Max Acc.	AaLL	AUC	TD
RF Comm. 95 + Vote Entropy	0.84	0.83	0.83	0.38
RF Comm. 956 + Vote Entropy	0.86	0.83	0.83	0.22
RF Comm. 9561 + Vote Entropy	0.87	0.63	0.86	0.22

Table 4.15: Accuracy measurements for Figure 4.10. Best method with sampling size 96, 956, and 9561

As in pool-based sampling already described, a sampling size above the labeling limit of 2000 is not recommended because the models cannot be improved before the labeling limit is reached. That renders them unusable for practical purposes, which also applies to "RF Comm. 9561". The curve progression for Vote Entropy- and Consensus Entropy Sampling are quite similar overall. With both query strategies, the accuracy rises fast in the first epoch and then nearly stagnates. In "RF Comm.

9561" for both strategies, model accuracies only decrease once they reach the highest accuracy value. Max Disagreement Sampling did not perform as well as the other two query strategies. In "RF Comm. 95", way more queries are sampled per epoch, and the accuracy decreases before the labeling limit and is worse than at the beginning. On the other hand, the Max Disagreement models show an increase in accuracy in the other two plots. However, Max Disagreement sampling is found to be less effective than other query strategies. The best method for all sampling sizes is Vote Entropy Sampling.

4.5.5 Clustering

As previously described, clustering is utilized with the CLARA algorithm. There are two types of approaches, that are tested. One is named "Training" and the second is named "Sampling".

In the "Training" approach the clustering is used to filter the entire initial training set. Fifty clusters are built, and the medoids of these clusters are selected as the initial training set for the model. Active learning is then utilized with Random-, Least Confidence-, Margin- and Entropy Sampling. However, it's important to note that SGD can only be used with random sampling.

The "Sampling" approach, on the other hand, uses clustering as a query strategy. This means that the unlabeled pool is split into 50 clusters each epoch, and the medoids are taken as the newly queried samples. To compare the performance of the two approaches, four models are created and processed differently. The first model is trained as usual, and then pool-based random sampling is applied. For the second model, the initial training set is filtered as in the "Training" approach and random sampling is utilized. The third model is trained as usual, and then clustering is used for sampling. Lastly, the fourth model uses cluster filtering for the initial training set and cluster sampling for the unlabeled pool.

The plots in Figure 4.11 illustrate the clustering approach "Training". The models are utilized with RF, all three datasets and a sampling size of 0.1% of the unlabeled pools of the different datasets. That implies a sampling size of 956 for the two-class dataset, 455 for the four-class, and 81 for the five-class dataset. The new queried samples per epoch per plot are 47,850 samples, 23,206 samples, and 4921 samples.

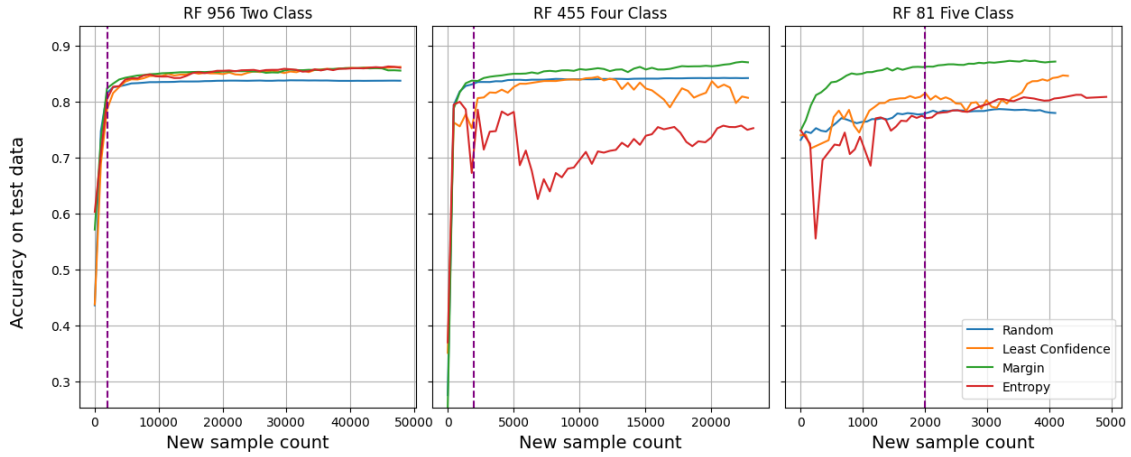

Figure 4.11: Cluster model training and pool-based Active learning with RF, the two-class, four-class, and five-class dataset. A sampling size of 0.1% of the unlabeled pool and 50 epochs used.

	Max Acc.	AaLL	AUC	TD
RF 956 Two Class + Random	0.83	0.81	0.82	0.40
RF 956 Two Class + Margin	0.86	0.82	0.84	0.28
RF 455 Four Class + Random	0.84	0.83	0.83	0.56
RF 455 Four Class + Margin	0.87	0.83	0.84	0.61
RF 81 Five Class + Random	0.78	0.77	0.77	0.04
RF 81 Five Class + Margin	0.87	0.86	0.85	0.12

Table 4.16: Accuracy measurements for Figure 4.11. Comparison of baseline and best method with a sampling size of 0.1%

Depending on the starting values, the models from "RF 956 Two Class" and "RF 455 Four Class" have a great improvement through active learning. The two-class models start from around 40-60% and the four-class models from around 20-40%. In "RF 956 Two Class" all models except the baseline have a similar curve progression and perform better than the baseline. In "RF 455 Four Class", the accuracy results were extremely diverse. All models initially improved greatly, but the Least Confidence and Entropy Sampling models performed worse over time than the baseline. The models of "RF 81 Five Class" presented significant differences. The models using Least Confidence and Entropy Sampling were unstable at first, but they improved over the epochs and achieved better results than the baseline model. On the other

hand, the Margin Sampling models were the best performing overall. They were consistent and benefited greatly from active learning.

As in the previous approaches, SGD did not perform well with active learning, and the accuracy results were again strongly fluctuating.

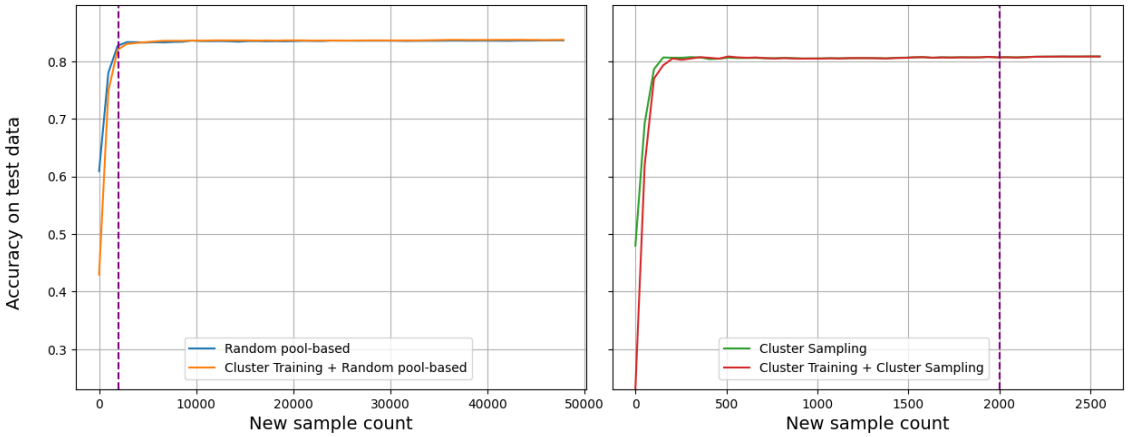

Figure 4.12: RF with the two-class dataset, a sampling size of 956 and 50 epochs. "Sampling" approaches compared.

Figure 4.12 displays the clustering approach "Sampling". The models are utilized with RF, the two-class dataset, 50 epochs, and a sampling size of 956. For the approaches random pool-based and cluster training, the queried sample count is 46,894. On the other hand, the queried sample count for the cluster sampling approach is the product of the number of epochs and the number of clusters, both of which are 50, resulting in a count of 2500.

	Max Acc.	AaLL	AUC	TD
Random pool-based	0.83	0.82	0.83	0.22
Cluster Training + Random pool-based	0.83	0.82	0.82	0.40
Cluster Sampling	0.80	0.80	0.80	0.32
Cluster Training + Cluster Sampling	0.80	0.80	0.79	0.57

Table 4.17: Accuracy measurements for Figure 4.12. Comparison of RF with sampling size 956 and the four approaches of "Sampling"

The plots and the results in Table 4.17 observed that the accuracy outcomes of the models using different learning approaches do not differ significantly. The models exhibit a similar pattern, with the accuracy rising rapidly at the beginning of the

active learning process and then plateauing. Therefore, none of the models seem to improve with more epochs. The standard-trained and clustering-trained models have slightly higher results in the measurements than the clustering sampling models. Although the cluster sampling models used fewer queried samples per epoch, their accuracy was slightly lower at the labeling limit. The TDs varied significantly, mainly due to the different starting values. The lower the starting value, the more significant the model improvement.

The SGD models are again strongly fluctuating, even with the "Sampling" approaches. This observation is consistent with other active learning approaches.

4.6 Discussion

Firstly, will summarize the results of the experiments.

The first experiment was short but already produced some helpful results. The initial model training demonstrated that preprocessing the datasets does not generally improve them. Some accuracy results were better, while others were worse than the original datasets. The feature extraction methods also produced feature datasets that caused very different accuracy results after the initial model training. From around 20% up to 97% is everything included. Random Forest and Stochastic Gradient Descent had similar results on most datasets. For the two-class dataset, up to 71.5% accuracy is reached. For the four-class dataset, the accuracy was around 40%, but SGD has got up to 78%. For the last, the five-class dataset, higher results were calculated up to 97%. The best feature extraction method for these settings was feature_ex5. The active learning is limited here to the pool-based sampling. The experiment was discontinued after two problems occurred. Firstly, in active learning, models with the original datasets produced higher accuracy results compared to models that used preprocessed datasets. Secondly, similar results were obtained in a comparison of RF and SGD with full initial dataset training and small initial dataset training (50 randomly selected instances). Therefore, the second experiment was initiated due to its lower computational resource usage, higher efficiency, and better initial performance.

In the second experiment, the same settings as in the first were used, but the data preprocessing step was skipped, and the initial training set was reduced to 50 instances. The accuracy results for both classifiers were similar once again. The accuracy for the two-class dataset was up to 74%, while for the four- and five-class datasets the accuracy was up to 77%. In general, SGD produced better results than RF. The best feature extraction method was feature_ex3.

For active learning, pool-based sampling, stream-based sampling, query-by-committee, and clustering are tested. In the pool-based sampling, the three different datasets (two-class-, four-class- and five-class dataset) are tested at 3 different sampling sizes (0.01%, 0.1%, and 1%). The first thing that was discovered is that if the sampling size exceeds the labeling limit of 2000, the models cannot be improved through active learning and produce better results after the labeling limit is already reached. First, the RF model results are presented.

	Max Acc.	AaLL	AUC	TD
Two-class dataset	0.87	0.83	0.85	0.18
Four-class dataset	0.89	0.83	0.85	0.30
Five-class dataset	0.91	0.86	0.88	0.18

Table 4.18: Highest accuracy measurements for pool-based sampling

Table 4.18 displays the highest results achieved for various measurements across different datasets. The results were obtained from different models in one comparison or plot. Nevertheless, the best query strategy overall for the two-class dataset was Entropy Sampling and the best model was "RF 956 + Entropy Sampling". For the four-class dataset, Margin Sampling performed the best, and "RF 455 + Margin Sampling" was the best model. For the five-class dataset, Margin Sampling performed the best, and "RF 81 + Margin Sampling" was the best performing model.

The SGD algorithm did not perform well when using the only viable query strategy, random sampling. The model's performance fluctuated significantly and did not show any improvement with more epochs. Although the two-class dataset showed an improvement of 18%, the other measurements were much worse than the comparable RF model used with random sampling. For the other datasets, the results were not different.

For the stream-based sampling, the three datasets are also tested with an exact sample count of 50. So, way fewer samples are queried through active learning than in pool-based sampling. A comparison was created between RF and SGD using the two-class dataset. The RF models did not show much improvement, while Entropy Sampling proved to be the best query strategy with a 5% improvement and the highest accuracy was 72%. In contrast, the SGD model showed an improvement of 22% with random sampling, but the maximum accuracy attained was only 68%. The SGD model also demonstrated significant fluctuations, which did not necessarily mean that the model improved with more epochs.

Another comparison was done using RF and all three datasets. All models started with high accuracies, with the two-class dataset at 67%, the four-class dataset at 57%, and the five-class dataset at 71%. For the two-class and five-class datasets, the improvement remained below 10% or sometimes even below 5%. On the other hand, the four-class dataset models showed an improvement of 17-22%. However, none of the models achieved an accuracy value above 80% with stream-based sampling.

Query-by-Committee was utilized with a committee of two RF models. Both models had slightly different parameters and different initial training data. The active learning process was conducted using a two-class dataset, and all three sampling sizes (95, 956, and 9561) were compared. Max Disagreement sampling turned out to be the worst method, as it decreased the accuracy of the model that used a sampling size of 95. Vote Entropy- and Consensus Entropy Sampling produced similar curve patterns. However, Vote Entropy was the most effective method overall, with the accuracy of the models reaching over 80% at the end. No Stochastic Gradient Descent (SGD) was used because the query strategies employ probability calculations.

In Cluster Sampling, two approaches are compared: “Training” and “Sampling”. In the “Training” approach, both RF and SGD are tested on all three datasets, using a sampling size of 0.1% (proportional to the unlabeled pools). The RF models showed a significant increase in accuracy, but the fluctuation of results increased when more classes were involved for some strategies. The Least Confidence and Entropy Sampling methods were very fluctuating with the four and five-class datasets. However, the models using Entropy Sampling showed the most improvement. They were consistently better than the baseline throughout the process and achieved the highest measurement results of around 80%. SGD did not perform well and showed fluctuating accuracy results like in other active learning approaches.

The results of the "Sampling" approach were not impressive for both RF and SGD. In this experiment, a two-class dataset was used with a sampling size of 956. The curve patterns of the different models during the active learning process were similar. Initially, the models improved rapidly to high values, but then the accuracy stagnated. Overall, the measurement values were above 80%. The cluster sampling models used fewer new queried samples per epoch, but at the labeling limit, they didn't deliver better results than the pool-based sampling baseline model and the cluster training model. The SGD models produced in this approach also fluctuating results.

The preprocessing steps from related work did not lead to any improvement in the datasets for better results in machine learning evaluation. This was observed both after the initial training and in the active learning phase. It is possible that the full URLs might provide further information or features that can be extracted and learned by the machine learning models. Another possibility is that the preprocessing steps made the features corresponding to the classes more ambiguous, which means that the model had difficulty in understanding which features mirror which class the most. This resulted in the test set being more difficult to label.

In the first experiment, it was observed that different feature extraction methods produced diverse results, with differences of up to 40-50%. The best results for Random Forest were achieved at 30-100%, while for Stochastic Gradient Descent, the range was 40-100%. The four-class dataset performed worse than the other datasets. If the same refinement were applied to the five-class dataset, it would also likely perform worse. Preprocessing the dataset did not improve the results, and both classifiers produced similar good results. The method feature_ex created the best features overall. Active learning tests demonstrated that models with preprocessed datasets improved less, and using a lower initial training set of 50 instances yielded similar improvements compared to using the full training set. This approach reduces the number of datasets that need to be tested, requires fewer computational resources, and improves the efficiency of the approaches.

In the second experiment, it was observed that the accuracy results on the pretrained models were similar to those of the first experiment. However, the best features were created by feature_ex3. The experiment also revealed that increasing the sampling size resulted in higher accuracies of the models. However, if the sampling sizes are too high, the accuracy at the labeling limit worsens and the model's

accuracy decreases after the highest value is reached. This is because more samples are added per epoch, which makes the model learn faster, and more diverse data is fitted to distinguish the feature structure for the different labeled URLs. On the other hand, a model can only be improved after the labeling limit is reached, so incremental learning cannot work properly. The decrease in accuracy after reaching high accuracy values can be explained by the imbalanced datasets. With more fitted data, the model is overfitting, which allows the model to perform better on the training set, but not on other datasets.

Different datasets with varying numbers of classes were used to test if models performed worse in labeling URLs as the number of classes increased. The curve patterns of the models with the same sampling size proportion were similar across different datasets, but the accuracy values and fluctuations differed between query strategies.

It's worth noting that during the labeling process, a model assigns a percentage to each class indicating how likely it is that the URL belongs to that class. For instance, a URL could be 70% malicious and 30% benign. The model can assign a total of 100% to all classes, but not more. If there are only two classes, the percentage change for each class is linear, making it easier for the model to assign percentages and resulting in similar active learning results for different query strategies. If more classes are available, the 100% needs to be distributed among more classes, leading to more variability in the results of the query strategies. This was observed in the pool-based sampling experiments. The curves for models working with two-class datasets were similar, despite using different query strategies. For models using four-class and five-class datasets, some strategies consistently produced great accurate results, while others were more inconsistent, and the results of the models varied more from each other. It is also possible that the different datasets used for refinement are a reason for the disparity in the results obtained. These datasets have varying numbers of URLs, and the diversity of the URLs can also differ significantly. For instance, the four-class dataset has fewer URLs in the training set and the remaining sets, in comparison to the two-class dataset. Additionally, the URLs in the training set of the four-class dataset may be less varied than those in the test set, making the test set more complex and harder to label. This could explain the lower accuracies observed after initial model training, as well as the higher number of classes. Active learning methods yield similar high results, around 80-90%, as with the two-class

dataset. On the other hand, the five-class dataset is not refined like the other two, so the domain shift problem is not solved. This would make URL labeling easier, as shown by the high results above 90% after initial model training. However, the dataset consists of much fewer data than in the other datasets, resulting in fewer different URLs available to train a model accordingly in the initial training and in the active learning. This could be a reason for the higher results on one side with some query strategies and lower results on the other.

The SGD classifier performed well during model evaluation after the initial training, but when it came to active learning, the results were inconsistent. It's possible that SGD or the specific loss function is not suitable for active learning, or fine-tuning of the models is needed. On the other hand, Random Forest worked great in active learning, producing the best results between 80-90%, even at the labeling limit. Stream-based sampling faced the issue of fewer queried samples, which could be used to improve the model. This method checks only one instance at a time to see whether it should be queried or not, making it very inefficient. Pool-based and stream-based sampling had similar runtimes with the used settings. However, the accuracy results of the models with stream-based sampling increased slightly for most models and never reached 80%. As a result, the measurement results were all worse than the model with pool-based sampling. The QBC models produced great results, improving accuracy up to over 80%. Cluster training involves training models with the most representative instances from the clusters, the medoids. Initially, the models are trained with the same number of instances. However, the results of the initial training were not as high compared to models that use randomly selected instances for training. The accuracy improvements through active learning were also not significantly higher for the cluster-trained models. Cluster sampling improved the models fairly well, but it did not achieve the accuracy levels of standard model training and pool-based sampling.

Pool-based sampling and QBC produced the most favorable outcomes. The highest accuracy results are achieved with a sampling size proportion of 1%, but for practical purposes, 0.1% proved to be the most effective, particularly when reaching the labeling limit while aiming for high accuracy.

Chapter 5

Conclusions

The previous chapter summarizes and concludes the two experiments. In this chapter, we discuss the research questions and their answers.

RQ 1: Which preprocessing steps can be added to improve the Classification of malicious URLs?

In the experiments, different preprocessing steps are combined and tested. Furthermore, clustering was used to filter the initial training-set, which is also like a preprocessing step. Both approaches had no positive impact on the models' accuracy in the initial model training and evaluation, as well as in the active learning. The preprocessing steps probably reduced the information that could be extracted from the URLs or made the features more unclear, which means the models are less sure about which features represent which class.

RQ 2: Which feature extraction methods are most suitable for URL Labeling?

Heuristic approaches, as well as natural language processing feature extraction methods, are tested and compared with each other. There is no clear best method. It depends on the size, diversity, and number of classes in the datasets. Additionally, the accuracy results are linked to the used machine learning classifier. Even then,

the accuracy results sometimes are way different or pretty similar. For the full initial training set the best method was feature_ex5 (TF-IDF vectorization) and for the smaller set feature_ex3 (Heuristic approach and combination of feature_ex1 and feature_ex2).

In active learning, no difference between the first and second experiments was discovered. With different extraction methods, similar model improvement was reached.

RQ 3: Is Natural Language Processing (NLP) usable for feature extraction with URL data?

As already mentioned for the previous question, natural language processing worked well for feature extraction with URL data. Vectorization performed sufficiently and achieved high results in the initial training.

The used BERT transformer did not perform as expected. The created embeddings should contain more information about the URLs than the other extraction methods could extract, but the produced results on the pre-trained models were compared to the other methods as well as these or worse. The reason could be that the embeddings were not correctly calculated or transformed into feature vectors, or the features are too complex for the models.

RQ 4: Which machine learning classification models are appropriate for URL Classification?

The Random Forest- and Stochastic Gradient Descent Classifier were tested in the experiments. Both were consistent during the k-fold Cross Validation and produced bad to great results in the initial model training and evaluation. The results depend on the used dataset and feature extraction method.

For active learning, SGD produced fluctuating accuracy results in every approach. This classifier or the utilized loss function is not usable with active learning or the models were not properly fine-tuned. The RF models improved well through active learning, depending on the used approach, query strategy, dataset, and parameter (sampling size, epochs).

No special tests were done for the efficiency, but both classifiers were efficient in the initial model training.

RQ 5: Which query strategies perform the best for URL Labeling?

SGD did not work well with active learning, so the following results are only for Random Forest. Four approaches with different query strategies were tested. Overall, pool-based sampling and Query-By-Committee (QBC) improved the models the best. Over 80% accuracy was reached. Except QBC, Least Confidence-, Margin- and Entropy-Sampling were used in the other approaches. Both Margin- and Entropy Sampling produced the best results. They were more consistent and created the best improvement for different experimental settings. For QBC Vote Entropy- and Consensus Entropy produced pretty similar results, but overall Vote-Entropy improved the models better.

RQ 6: How can clustering be utilized for URL Labeling?

As in the second experiment explained, the clustering is tested for filtering the training data and as the query strategy. The models that were initially trained on the filtered data performed similarly to the standard trained models. Clustering for sampling the unlabeled pool improved the corresponding model significantly, but the results were worse than in the pool-based sampling. Therefore, clustering can be used as an additional preprocessing step before actual initial model training or as a query strategy in active learning. However, another clustering approach or different experimental settings may be needed to improve the results.

RQ 7: Should lifelong learning be integrated into the project?

Both experiments use incremental and lifelong learning. The classifiers were selected especially because they support lifelong learning. SGD can easily be used with lifelong learning because partial fitting can be used, but as previously mentioned, the accuracy results of the trained models strongly fluctuate. So only Random Forest can be evaluated.

The models improved pretty well during the incremental active learning process. If there is a labeling limit of new queried samples, the sampling size may not be above the limit, else the model cannot be improved before the limit is reached.

Lifelong learning worked for both models (even if SGD fluctuates). The actual benefit is that the efficiency of the models should be constant because the same number of new samples are trained per epoch, and not the whole training set needs to be trained again. So the difference between incremental- and lifelong learning should be, that the time consumed for calculations is lower over time for the lifelong learning, but the results for both are the same. The efficiency compared to incremental learning is not directly measured, so that cannot be answered. But previously, already two problems for RF models with lifelong learning were explained. First, the algorithm used needs, per epoch, at least one sample of each class, which increases the count of queried samples. Second, the algorithm did not extend the already created trees but added new trees every epoch to improve the model during the training. In incremental active learning, the queried sample count and the tree size do not need to be increased, so different accuracy results are produced if both approaches are compared. So first, another RF implementation is needed that solves the problems, or other classifiers are needed that support lifelong learning.

Chapter 6

Future work

This chapter lists tasks that need to be accomplished in the future. These tasks were determined during the testing phase of the thesis and the experiments. Some of these tasks were not considered before due to time constraints, while others emerged as a result of the experimental findings.

The tasks listed below are assigned to the main steps that were tested in this thesis.

6.1 Data Preprocessing

The implemented preprocessing steps did not yield positive results in the experiments. Therefore, it might be useful to test each step independently to determine if any of them can improve models when used alone. This would help identify if a specific method is responsible for the problem. Additionally, it may be beneficial to try out other preprocessing methods from related work.

The clustering used for filtering the initial training set did not work either. It may be worth trying a different clustering technique or alternative method to filter the initial training set. If none of these methods improve the accuracy of the model, then the training set might be too small to be meaningful, and certain URLs may have an unnecessary impact. In this case, it may not matter how diverse or uncertain the initial training data is.

6.2 Feature Extraction

In general, the results with the different feature extraction methods were great, there was no particular method, that was worse than the other.

However, it might be beneficial to search for more heuristic approaches from related work and implement them, in order to combine more features, similar to what was done in feature_ex3. Alternatively, it could be worthwhile to test other feature selection algorithms, or to improve the current one in feature_ex4 by fine-tuning it to select the best features for the given task.

Furthermore, the TF-IDF and Count-Vectorization algorithms could also be further optimized by fine-tuning them.

It might also be useful to search for and implement other transformers, specifically those designed for processing URL data or similar types of data. Additionally, a cased transformer model could be created for URL data. BERT can also be used for that task.

6.3 Machine Learning

The experiments have indicated that Random Forest (RF) and Stochastic Gradient Descent (SGD) algorithms have produced great models. However, the RF algorithm is not ideal for lifelong learning and the SGD algorithm cannot be used for active learning in its current configuration. Therefore, we should search for and implement an RF algorithm that overcomes the challenges mentioned in the experiments chapter. To address the issue with active learning, we can test hyper-tuning or modification of the loss function for SGD.

It would be beneficial to explore other machine learning classifiers, particularly those that are not tree-based or are more complex, such as deep learning models. However, we should be mindful that complex models may lead to overfitting. It would be great to find classifiers that support lifelong learning.

In general, tuning the parameters of the models can enhance their performance. For tree-based models, we could try different estimator sizes or adjust the estimator count added per epoch in active learning.

We could also consider using transformer as a machine learning model, like BERT.

6.4 Incremental Active- and Lifelong Learning

It is important to search and implement query strategies for incremental learning that work with non-probabilistic classifiers. Fine-tuning these strategies can also be helpful. More complex query strategies, such as self-implemented strategies from related work, could be implemented and tested. However, the challenge with these strategies is that they are often not fully explained, making it difficult to implement them correctly, and the implementation and testing process can be time-consuming.

For lifelong learning, it is important to search for classifiers that support this technique, such as classifiers that can perform partial fitting. Additionally, tests should be conducted to compare the efficiency of incremental active learning and lifelong learning over a longer active learning process. This will help determine if classifiers using lifelong learning are truly more efficient.

Bibliography

[AAA+22] Aljabri, Malak ; Altamimi, Hanan S. ; Albelali, Shahd A. ; Al-Harbi, Maimunah ; Alhuraib, Haya T. ; Alotaibi, Najd K. ; Alahmadi, Amal A. ; AlHaidari, Fahd ; Mohammad, Rami Mustafa A. ; Salah, Khaled: Detecting Malicious URLs Using Machine Learning Techniques: Review and Research Directions. 10 (2022), S. 121395–121417. http://dx.doi.org/10.1109/ACCESS.2022.3222307. – DOI 10.1109/ACCESS.2022.3222307

[AAAS22] Abdiyeva-Aliyeva, Gunay ; Aliyev, Jeyhun ; Sadigov, Ulfat: Application of classification algorithms of Machine learning in cybersecurity. In: *Procedia Computer Science* 215 (2022), S. 909–919

[APH22] Aggarwal, Umang ; Popescu, Adrian ; Hudelot, Céline: Optimizing active learning for low annotation budgets. In: *arXiv preprint arXiv:2201.07200* (2022)

[APK+11] Antoniades, Demetris ; Polakis, Iasonas ; Kontaxis, Georgios ; Athanasopoulos, Elias ; Ioannidis, Sotiris ; Markatos, Evangelos P. ; Karagiannis, Thomas: we. b: The web of short URLs. In: *Proceedings of the 20th international conference on World Wide Web*, 2011, S. 715–724

[ASBAK+23] Abdul Samad, Saleem R. ; Balasubaramanian, Sundarvadivazhagan ; Al-Kaabi, Amna S. ; Sharma, Bhisham ; Chowdhury, Subrata ; Mehbodniya, Abolfazl ; Webber, Julian L. ; Bostani, Ali: Analysis of the Performance Impact of Fine-Tuned Machine Learning Model for Phishing URL Detection. In: *Electronics* 12 (2023), Nr. 7, S. 1642

[AVW20] Albakry, Sara ; Vaniea, Kami ; Wolters, Maria K.: What is this URL's destination? empirical evaluation of users' URL reading. In: *Proceedings of the 2020 CHI conference on human factors in computing systems*, 2020, S. 1–12

[BAK+19] Bartusiak, Roman ; Augustyniak, Łukasz ; Kajdanowicz, Tomasz ; Kazienko, Przemysław ; Piasecki, Maciej: WordNet2Vec: Corpora agnostic word vectorization method. In: *Neurocomputing* 326 (2019), S. 141–150

[BBK+22] Bodesheim, Paul ; Blunk, Jan ; Körschens, Matthias ; Brust, Clemens-Alexander ; Käding, Christoph ; Denzler, Joachim: Pre-trained models are not enough: active and lifelong learning is important for long-term visual monitoring of mammals in biodiversity research—individual identification and attribute prediction with image features from deep neural networks and decoupled decision models applied to elephants and great apes. In: *Mammalian Biology* 102 (2022), Nr. 3, S. 875–897

[BCCR19] Bergadano, Francesco ; Carretto, Fabio ; Cogno, Fabio ; Ragno, Dario: Defacement detection with passive adversaries. In: *Algorithms* 12 (2019), Nr. 8, S. 150

[Ber19] Berrar, Daniel: Cross-Validation. Version: 2019. `http://dx.doi.org/10.1016/B978-0-12-809633-8.20349-X`. In: *Encyclopedia of Bioinformatics and Computational Biology - Volume 1.* Elsevier, 2019. – DOI 10.1016/B978–0–12–809633–8.20349–X, 542–545

[Bha14] Bhat, Aruna: K-medoids clustering using partitioning around medoids for performing face recognition. In: *International Journal of Soft Computing, Mathematics and Control* 3 (2014), Nr. 3, S. 1–12

[BPM04] Batista, Gustavo E. A. P. A. ; Prati, Ronaldo C. ; Monard, Maria C.: A study of the behavior of several methods for balancing machine learning training data. In: *SIGKDD Explor.* 6 (2004), Nr. 1, 20–29. `http://dx.doi.org/10.1145/1007730.1007735`. – DOI 10.1145/1007730.1007735

[Cha13] Chakraborty, Nilotpal: Intrusion detection system and intrusion prevention system: A comparative study. In: *International Journal of Computing and Business Research (IJCBR)* 4 (2013), Nr. 2, S. 1–8

[CYRR21] Cyprienna, Rakotoasimbahoaka A. ; Yannick, Raharijaona Zo L. ; Randria, Iadaloharivola ; Raft, Razafindrakoto N.: URL Classification based on Active Learning Approach. In: *2021 3rd International Cyber Resilience Conference (CRC)* IEEE, 2021, S. 1–6

[DCLT18a] Devlin, Jacob ; Chang, Ming-Wei ; Lee, Kenton ; Toutanova, Kristina: *BERT base model (uncased).* 2018. – `https://huggingface.co/bert-base-uncased` [Accessed: (20.01.2024)]

[DCLT18b] Devlin, Jacob ; Chang, Ming-Wei ; Lee, Kenton ; Toutanova, Kristina: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *CoRR* abs/1810.04805 (2018). `http://arxiv.org/abs/1810.04805`

[DGG23] Dorji, Ugyen ; Gyaltshen, Tenzin ; Gupta, Deepak: Big Data Security Analytics for Phishing URLs Detection. (2023). http://ir.juit.ac.in:8080/jspui/jspui/handle/123456789/9829

[DLR20] DLR: *HPDA system Kratos*. 2020. – https://www.dlr.de/dw/desktopdefault.aspx/tabid-13692/23853_read-55687/ [Accessed: (29.01.2024)]

[DW09] DeBarr, Dave ; Wechsler, Harry: Spam detection using clustering, random forests, and active learning. In: *Sixth conference on email and anti-spam. Mountain View, California* Citeseer, 2009, S. 1–6

[FRD14] Freytag, Alexander ; Rodner, Erik ; Denzler, Joachim: Selecting influential examples: Active learning with expected model output changes. In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV 13* Springer, 2014, S. 562–577

[Gau20] Gaur, Vishal: *Information extraction from patent office actions using NLP techniques*, Tampere University, Diplomarbeit, 2020

[GYR+21] Gupta, Brij B. ; Yadav, Krishna ; Razzak, Imran ; Psannis, Konstantinos E. ; Castiglione, Arcangelo ; Chang, Xiaojun: A novel approach for phishing URLs detection using lexical based machine learning in a real-time environment. In: *Comput. Commun.* 175 (2021), 47–57. http://dx.doi.org/10.1016/J.COMCOM.2021.04.023. – DOI 10.1016/J.COMCOM.2021.04.023

[Ho95] Ho, Tin K.: Random decision forests. In: *Third International Conference on Document Analysis and Recognition, ICDAR 1995, August 14 - 15, 1995, Montreal, Canada. Volume I*, IEEE Computer Society, 1995, 278–282

[HY23] Hu, Zeyuan ; Yuan, Ziang: A Review of Data-driven Approaches for Malicious Website Detection. In: *CoRR* abs/2305.09084 (2023). http://dx.doi.org/10.48550/ARXIV.2305.09084. – DOI 10.48550/ARXIV.2305.09084

[JA23] Jishnu, KS ; Arthi, B: Enhanced Phishing URL Detection Using Leveraging BERT with Additional URL Feature Extraction. In: *2023 5th International Conference on Inventive Research in Computing Applications (ICIRCA)* IEEE, 2023, S. 1745–1750

[KG20] Kumar, Punit ; Gupta, Atul: Active learning query strategies for classification, regression, and clustering: a survey. In: *Journal of Computer Science and Technology* 35 (2020), S. 913–945

[KRFD16] Käding, Christoph ; Rodner, Erik ; Freytag, Alexander ; Denzler, Joachim: Watch, Ask, Learn, and Improve: a lifelong learning cycle for visual recognition. In: *24th European Symposium on Artificial Neural Networks, ESANN 2016, Bruges, Belgium, April 27-29, 2016*, 2016, 381–386

[KRS21] Kalyan, Katikapalli S. ; Rajasekharan, Ajit ; Sangeetha, Sivanesan: Ammus: A survey of transformer-based pretrained models in natural language processing. In: *arXiv preprint arXiv:2108.05542* (2021)

[Kum19] Kumar, Siddarth: *Malicious And Benign URLs*. 2019. – `https://www.kaggle.com/datasets/siddharthkumar25/malicious-and-benign-urls` [Accessed: (05.10.2023)]

[LBK21] Lakshmanarao, A ; Babu, M R. ; Krishna, MM B.: Malicious URL Detection using NLP, Machine Learning and FLASK. In: *2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)* IEEE, 2021, S. 1–4

[Lid01] Liddy, Elizabeth D.: Natural language processing. (2001)

[Lon21] Lones, Michael A.: How to avoid machine learning pitfalls: a guide for academic researchers. In: *CoRR* abs/2108.02497 (2021). `https://arxiv.org/abs/2108.02497`

[LS95] Liu, Huan ; Setiono, Rudy: Chi2: Feature selection and discretization of numeric attributes. In: *Proceedings of 7th IEEE international conference on tools with artificial intelligence* Ieee, 1995, S. 388–391

[LS22] Lenssen, Lars ; Schubert, Erich: Clustering by Direct Optimization of the Medoid Silhouette. In: *International Conference on Similarity Search and Applications* Springer, 2022, S. 190–204

[LWLZ18] Liu, Chunlin ; Wang, Lidong ; Lang, Bo ; Zhou, Yuan: Finding effective classifier for malicious URL detection. In: *Proceedings of the 2018 2nd International Conference on Management Engineering, Software Engineering and Service Sciences*, 2018, S. 240–244

[MBA+21] Mourtaji, Youness ; Bouhorma, Mohammed ; Alghazzawi, Daniyal ; Aldabbagh, Ghadah ; Alghamdi, Abdullah: Hybrid rule-based solution for phishing URL detection using convolutional neural network. In: *Wireless Communications and Mobile Computing* 2021 (2021), S. 1–24

[MM24] Möbius Max, Bouhlal Badr-Eddine: *URL labeling with active learning.* `https://git.rz.uni-jena.de/pe37jeg/url-labeling-with-active-learning`, 2024

[MRL+16a] Mamun, Mohammad Saiful I. ; Rathore, Mohammad A. ; Lashkari, Arash H. ; Stakhanova, Natalia ; Ghorbani, Ali A.: Detecting malicious urls using lexical analysis. In: *Network and System Security: 10th International Conference, NSS 2016, Taipei, Taiwan, September 28-30, 2016, Proceedings 10* Springer, 2016, S. 467–482

[MRL+16b] Mamun, Mohammad Saiful I. ; Rathore, Mohammad A. ; Lashkari, Arash H. ; Stakhanova, Natalia ; Ghorbani, Ali A.: *URL dataset (ISCX-URL2016).* 2016. – `https://www.unb.ca/cic/datasets/url-2016.html` [Accessed: (05.10.2023)]

[NCB23] Naim, Or ; Cohen, Doron ; Ben-Gal, Irad: Malicious website identification using design attribute learning. In: *Int. J. Inf. Sec.* 22 (2023), Nr. 5, 1207–1217. `http://dx.doi.org/10.1007/S10207-023-00686-Y`. – DOI 10.1007/S10207–023–00686–Y

[PJ09] Park, Hae-Sang ; Jun, Chi-Hyuck: A simple and fast algorithm for K-medoids clustering. In: *Expert systems with applications* 36 (2009), Nr. 2, S. 3336–3341

[PKKG10] Prakash, Pawan ; Kumar, Manish ; Kompella, Ramana R. ; Gupta, Minaxi: Phishnet: predictive blacklisting to detect phishing attacks. In: *2010 Proceedings IEEE INFOCOM* IEEE, 2010, S. 1–5

[PLD03] Peng, Hanchuan ; Long, Fuhui ; Ding, C.: Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27 (2003), 1226-1238. `https://api.semanticscholar.org/CorpusID:206764015`

[PP22] Ponni, P. ; Prabha, D.: Randomized Active Learning to Identify Phishing URL. In: *Advanced Communication and Intelligent Systems - First International Conference, ICACIS 2022, Virtual Event, October 20-21, 2022, Revised Selected Papers* Bd. 1749, Springer, 2022 (Communications in Computer and Information Science), 533–539

[RAA+22] Roy, Sanjiban S. ; Awad, Ali I. ; Amare, Lamesgen A. ; Erkihun, Mabrie T. ; Anas, Mohd: Multimodel phishing url detection using lstm, bidirectional lstm, and gru models. In: *Future Internet* 14 (2022), Nr. 11, S. 340

[RPA22] Raja, A. S. ; Pradeepa, G. ; Arulkumar, N.: Mudhr: Malicious URL detection using heuristic rules based approach. In: *AIP Conference Proceedings* 2393 (2022), 05, Nr. 1, 020176. `http://dx.doi.org/10.1063/5.0074077`. – DOI 10.1063/5.0074077. – ISSN 0094–243X

[RSÁGVV23] Rendón-Segador, Fernando J. ; Álvarez-García, Juan A. ; Varela-Vaca, Angel J.: Paying Attention to cyber-attacks: A multi-layer perceptron with self-attention mechanism. In: *Computers & Security* (2023), S. 103318

[RVK21] Raja, A S. ; Vinodini, R ; Kavitha, A: Lexical features based malicious URL detection using machine learning techniques. In: *Materials Today: Proceedings* 47 (2021), S. 163–166

[SACAF21] Said, Naina ; Ahmad, Kashif ; Conci, Nicola ; Al-Fuqaha, Ala: Active learning for event detection in support of disaster analysis applications. In: *Signal, Image and Video Processing* (2021), S. 1–8

[Set09] Settles, Burr: Active learning literature survey. (2009)

[SGP+23] Shelar, Rohit ; Gujar, Yash ; Pawal, Niranjan ; Londhe, Pratiksha ; Rangdale, Sonali: NEWSIFY:-Article Summarization using Natural Language Processing and News Authentication using TF-IDF Vectorizer and Passive Aggressive Classifier. In: *2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS)* IEEE, 2023, S. 1654–1660

[Sid21] Siddhartha, Manu: *Malicious URLs dataset*. 2021. – `https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset` [Accessed: (05.10.2023)]

[SJ72] Sparck Jones, Karen: A statistical interpretation of term specificity and its application in retrieval. In: *Journal of documentation* 28 (1972), Nr. 1, S. 11–21

[SLH17] Sahoo, Doyen ; Liu, Chenghao ; Hoi, Steven C.: Malicious URL detection using machine learning: A survey. In: *arXiv preprint arXiv:1701.07179* (2017)

[SMSX19] Singh, Jasdeep ; McCann, Bryan ; Socher, Richard ; Xiong, Caiming: BERT is not an interlingua and the bias of tokenization. In: *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, 2019, S. 47–55

[SR21] Schubert, Erich ; Rousseeuw, Peter J.: Fast and eager k-medoids clustering: O (k) runtime improvement of the PAM, CLARA, and CLARANS algorithms. In: *Information Systems* 101 (2021), S. 101804

[SW+89] St, Lars ; Wold, Svante u. a.: Analysis of variance (ANOVA). In: *Chemometrics and intelligent laboratory systems* 6 (1989), Nr. 4, S. 259–272

[Tit09] Titsias, Michalis: Variational learning of inducing variables in sparse Gaussian processes. In: *Artificial intelligence and statistics* PMLR, 2009, S. 567–574

[Tiw20] Tiwari, Tarun: *Phishing Site URLs*. 2020. – `https://www.kaggle.com/datasets/taruntiwarihp/phishing-site-urls` [Accessed: (05.10.2023)]

[VBB23] Vörös, Tamás ; Bergeron, Sean P. ; Berlin, Konstantin: Web Content Filtering Through Knowledge Distillation of Large Language Models. (2023), 357–361. `http://dx.doi.org/10.1109/WI-IAT59888.2023.00058`. – DOI 10.1109/WI–IAT59888.2023.00058

[VV19] Vanitha, N ; Vinodhini, V: Malicious-URL detection using logistic regression technique. In: *International Journal of Engineering and Management Research* 9 (2019), Nr. 6, S. 108–113

[WWXZ22] Wu, Tiefeng ; Wang, Miao ; Xi, Yunfang ; Zhao, Zhichao: Malicious url detection model based on bidirectional gated recurrent unit and attention mechanism. In: *Applied Sciences* 12 (2022), Nr. 23, S. 12367

[XT15] Xu, Dongkuan ; Tian, Yingjie: A comprehensive survey of clustering algorithms. In: *Annals of Data Science* 2 (2015), S. 165–193

[ZKV22] Zongo, Wend-Benedo S. ; Kabore, Boukary ; Vaghela, Ravirajsinh S.: Phishing URLs Detection Using Machine Learning. In: *International Conference on Advancements in Smart Computing and Information Security* Springer, 2022, S. 159–167

[ZWSL10] Zinkevich, Martin ; Weimer, Markus ; Smola, Alexander J. ; Li, Lihong: Parallelized Stochastic Gradient Descent. (2010), 2595–2603. `https://proceedings.neurips.cc/paper/2010/hash/abea47ba24142ed16b7d8fbf2c740e0d-Abstract.html`

List of Figures

List of Tables

Erklärung

Ich versichere, dass ich die vorliegende Arbeit (bei Gruppenarbeiten die entsprechend gekennzeichneten Anteile) selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel und Quellen benutzt habe. Zitate und gedankliche Übernahmen aus fremden Quellen (einschließlich elektronischer Quellen) habe ich kenntlich gemacht. Die eingereichte Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und wurde auch nicht veröffentlicht. Mir ist bekannt, dass eine unwahre Erklärung rechtliche Folgen haben und insbesondere dazu führen kann, dass die Arbeit als nicht bestanden bewertet wird.

Seitens des Verfassers/der Verfasserin bestehen keine Einwände, die vorliegende Masterarbeit für die öffentliche Benutzung zur Verfügung zu stellen.

Jena, den 31.01.2024 Max Möbius