UNIVERSITY OF PATRAS
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

DEPARTMENT of electronics and computers (PC)
INTERACTIVE TECHNOLOGIES LABORATORY

Study of a real time voice phishing detection system with machine learning technology

DIPLOMA THESIS

IOANNIS SKARPETIS

SUPERVISOR: FEIDAS CHRISTOS

PATRAS – FEBROUARY 2024

University of Patras, Department of Electrical and Computer Engineering.

Ioannis Skarpetis

CERTIFICATION

It is certified that the Diploma Thesis titled

Study of a real time voice phishing detection system with machine learning technology

of the Department of Electrical and Computer Engineering student

IOANNIS SKARPETIS

Registration Number: 1066539

was presented publicly at the Department of Electrical and Computer Engineering at

……../……../……

and was examined by the following examining committee:

Name Surname, Title, Affiliation (supervisor)

Name Surname, Title, Affiliation (committee member)

Name Surname, Title, Affiliation (committee member)

The Supervisor	The Director of the Division
Feidas Christos Substitute Professor	Name Surname Title

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΒΛΕΠΟΝΤΟΣ
ΕΡΓΑΣΤΗΡΙΟ ΕΠΙΒΛΕΠΟΝΤΟΣ

ΤΙΤΛΟΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Δ Ι Π Λ Ω Μ Α Τ Ι Κ Η Ε Ρ Γ Α Σ Ι Α

ΟΝΟΜΑΤΕΠΩΝΥΜΟ ΦΟΙΤΗΤΗ

ΕΠΙΒΛΕΠΩΝ: ΟΝΟΜΑΤΕΠΩΝΥΜΟ ΕΠΙΒΛΕΠΟΝΤΟΣ

ΠΑΤΡΑ - ΜΗΝΑΣ ΕΤΟΣ

Πανεπιστήμιο Πατρών, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών.

Ονομαπεπώνυμο φοιτητή/τριας

ΠΙΣΤΟΠΟΙΗΣΗ

Πιστοποιείται ότι η Διπλωματική Εργασία με τίτλο

ΤΙΤΛΟΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

του/της φοιτητή/τριας του Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

ΟΝΟΜΑ ΕΠΩΝΥΜΟ ΤΟΥ **ΠΑΤΡΩΝΥΜΟ**

Αριθμός Μητρώου: XXXXXXX

Παρουσιάστηκε δημόσια στο Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών στις

……../……../……

και εξετάστηκε από την ακόλουθη εξεταστική επιτροπή:

Όνομα Επώνυμο, Βαθμίδα, Τμήμα (επιβλέπων)

Όνομα Επώνυμο, Βαθμίδα, Τμήμα (μέλος επιτροπής)

Όνομα Επώνυμο, Βαθμίδα, Τμήμα (μέλος επιτροπής)

Ο/Η Επιβλέπων/ουσα	Ο/Η Διευθυντής/τρια του Τομέα
Ονοματεπώνυμο Βαθμίδα	Ονοματεπώνυμο Βαθμίδα

PREFACE

The Preface is optional. Here the author puts any information that is not directly related to the scientific content of the Diploma Thesis, such as acknowledgements, etc.

The structure, form and extent of the preface are at the student's discretion, unless otherwise specified by the supervisor.

ABSTRACT

DIPLOMA THESIS TITLE

STUDENT NAME, SURNAME: **SUPERVISOR NAME, SURNAME:**

The objectives, methods, procedures, experiments and results of the Diploma Thesis are briefly described.

The structure, format and scope of the abstract are at the student's discretion, unless otherwise specified by the supervisor.

ΕΚΤΕΤΑΜΕΝΗ ΕΛΛΗΝΙΚΗ ΠΕΡΙΛΗΨΗ

ΤΙΤΛΟΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

ΟΝΟΜΑΤΕΠΩΝΥΜΟ ΦΟΙΤΗΤΗ: **ΟΝΟΜΑΤΕΠΩΝΥΜΟ ΕΠΙΒΛΕΠΟΝΤΟΣ:**

Σύμφωνα με τον κανονισμό Διπλωματικών Εργασιών, η συγγραφή της Διπλωματικής Εργασίας στην Αγγλική γλώσσα θα συνοδεύεται απαραιτήτως από εκτενή περίληψη στα Ελληνικά, τύπου επιστημονικής εργασίας (paper).

Οι υπόλοιπες λεπτομέρειες σχετικά με τη δομή και τη μορφή της Εκτεταμένης Ελληνικής Περίληψης είναι στη διακριτική ευχέρεια του φοιτητή, εκτός αν προδιαγράψει διαφορετικά ο επιβλέπων.

Table Of Contents

Table Of Figures

1. Introduction

Since the dawn of the 21st century, humanity has embarked on a remarkable journey of technological progression. As our society becomes ever more reliant on technology for its daily functions, we've seen a corresponding escalation in the complexity of cybercriminal activities. The advancement of technology has not only streamlined our daily tasks but has simultaneously opened numerous vulnerabilities. These gaps are frequently exploited by cybercriminals, targeting those who struggle to stay abreast of the latest technological developments.

Voice phishing, or 'vishing', is a notable vulnerability within the cybersecurity landscape, manifesting as a sophisticated form of phishing conducted via phone calls. While phishing as a broader concept is not novel, having roots that extend well before the internet age, vishing represents its adaptation to the telecommunication realm. Phishing, at its core, is about manipulating individuals to reveal confidential information under misleading pretenses, a strategy that has seen various incarnations over many decades. The term "phishing" gained widespread recognition during the 1990s alongside the internet's growth. Originally, it denoted email-based frauds wherein culprits masqueraded as legitimate entities to coax out sensitive details like login credentials and financial data from unsuspecting victims.

Vishing exploits the same fundamental principle of deception but leverages the immediacy and personal touch of voice communication. Here, fraudsters make use of phone calls to impersonate legitimate organizations or authorities, aiming to instill a sense of urgency or fear in victims. This tactic often involves compelling narratives designed to prompt quick action, leading individuals to inadvertently disclose personal or financial information.

Efforts to bolster security against such attacks have been ongoing, yet the advent of caller ID spoofing technology has notably heightened the efficacy of vishing attacks. This technology enables attackers to disguise their actual identity, presenting a significant obstacle in

identifying and preventing these fraudulent calls. Consequently, the dynamic nature and sophistication of vishing have established it as a significant and continually evolving threat within the cybersecurity domain. This scenario highlights the critical necessity for advanced and adaptable security measures capable of effectively neutralizing the ever-changing tactics of cybercriminals. Consequently, the focus has shifted towards innovative methods for detecting vishing attacks, particularly by analyzing the conversation content between the two parties in real-time. This approach represents a significant step forward in proactive cybersecurity, aiming to identify and mitigate vishing threats as they occur.

2. Analysis of the problem under research

Before delving into the analysis of the research problem addressed in this dissertation, it is essential to first define the central focus, which is Phone Call Scam Detection. This foundational concept forms the basis of our investigation and discussion, setting the stage for a thorough exploration of the strategies and technologies employed in identifying and mitigating fraudulent phone communications.

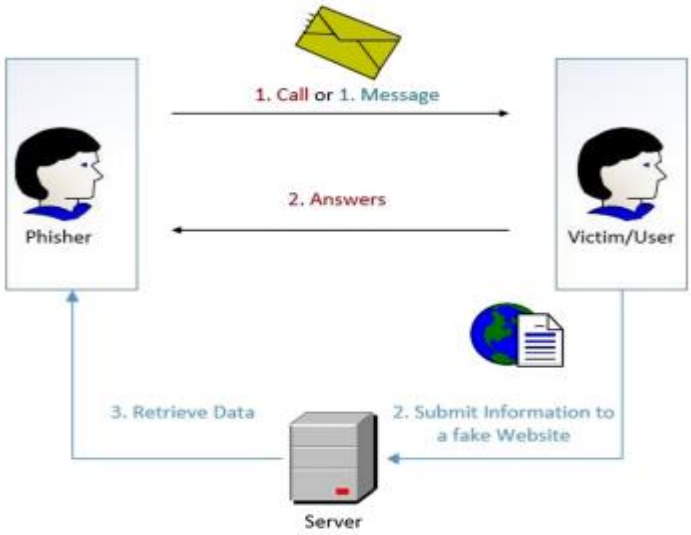


Fig 1: Mobile Phone Phishing Attack [..]

2.1 Vishing attacks and methods

To effectively develop security measures against vishing attacks, it is crucial to first comprehend the vast array of techniques employed by Vishers. Understanding their strategies and tactics provides the foundational knowledge necessary to design and implement robust countermeasures. The techniques implemented by Vishers can be broadly categorized into two main phases [4]:

Before the vishing attack:

- **Automated information gathering:** Vishers frequently use automated tools to 'scrape' large amounts of user data. This bulk collection can provide a broad base from which specific targets are identified and proceed to be evaluated as potential victims.

- **Victim Evaluation:** Vishing scams often begin with phishing attacks via automated emails, texts, or social media messages, posing as legitimate entities to gather contact information. Those who respond to these initial contacts are more likely to fall for subsequent vishing calls. Some Vishers skip initial contact and directly call numerous potential victims using automated services, aiming to reach susceptible individuals.

- **Data accumulation via research**: A key element of preparation for Vishers involves extensive research and collection of personal and private information about their potential victims. This phase often involves the use of social media platforms to gather

details such as phone numbers, employment data, and location information, which, while not overly sensitive, can lay the groundwork for a more targeted approach.

- **Direct Methods:** In some instances, Vishers may resort to direct, personal methods of information gathering, like sifting through an individual's trash also known as Dumpster Diving. This approach is more physical and hands-on where the perpetrators physically search through garbage to find discarded documents, letters, bills, and other materials that contain personal or sensitive information. Even seemingly harmless details such as phone lists, calendars, or organizational charts can be instrumental for attackers when planning strategy. [5]

- **Caller ID Spoofing:** An essential element in the success of vishing attacks is the ability to avoid detection by the victim. To achieve this, Vishers use a tactic known as "Caller ID Spoofing." This technique involves altering the phone number that appears on the recipient's caller ID. It proves highly effective because many individuals depend on the caller ID to verify the legitimacy of the caller. [6]

During the vishing attack:

- **Impersonation and Authority:** Vishers often pretend to be from reputable organizations, such as banks, government agencies, tech support, law enforcement agencies or members of a financial institution. They use authoritative tones and language to gain the victim's trust and compliance. [8]

- **Confirmation Bias:** They use known information about the victim (gathered during the preparation phase) to build trust and credibility. For example, referencing a recent transaction or a known issue to make the call seem legitimate.

- **Creating urgency:** Vishers frequently employ the tactic of instilling a sense of immediate urgency. They may assert that urgent action is required to address issues like a supposed security breach in the victim's bank account or a pressing financial matter. The aim of the attacker is to prompt the victim to act quickly and impulsively, bypassing the usual steps of call verification and thoughtful consideration.

- **Emotional Manipulation:** Vishers adeptly exploit the emotions of their victims, wielding tactics that evoke fear, sympathy, or excitement to steer their responses. They might threaten legal action or warn of imminent financial loss to instill fear and a sense of urgency. Conversely, they can appeal to the victim's emotions through sympathetic stories or scenarios, creating a false sense of connection or trust. By manipulating emotions in these ways, Vishers aim to cloud the victim's judgment, making it easier to extract sensitive information or coerce them into specific actions that further the scam. [7]

- **Information Overload:** When individuals are quickly presented with an excessive amount of information, it can lead to sensory overload, impairing their ability to logically process and evaluate the arguments being made. In such scenarios, people are more inclined to accept the statements presented to them, as their capacity for critical thinking is overwhelmed by the sheer volume of information. This tactic increases the likelihood of the person acquiescing to the Visher's demands or assertions. [8]

- **Diversion Tactics**: Victims are often redirected to alternate communication channels or actions. This could involve being instructed to call a different number, visit a fraudulent website, or mail sensitive documents to a specific address. These tactics serve to deepen the victim's involvement in the scam, making the deception appear more legitimate and complicating the victim's ability to discern the fraud.

- **Follow-Up and Persistence:** In vishing scams, Vishers often employ persistent follow-up calls, especially if the victim shows initial doubt or reluctance. This relentless approach aims to wear down the victim's defenses, increasing the likelihood of succumbing to the scam. Such persistence is a key tactic in gradually eroding skepticism and reinforcing the scam's credibility.

2.1.1 Typical Vishing Attack Examples

Vishers select from a diverse range of scenarios to deceive their victims. The subsequent examples will elaborate on the most frequently utilized themes in vishing scams. [2]

- **IRS and Immigration Scams:** Scammers impersonate IRS officials or immigration officers, threatening arrest or deportation if alleged debts aren't paid.

- **Romance Scams:** Fraudsters establish romantic connections, often through dating apps or phone calls, claiming to be past lovers in need of emergency funds.

- **Tech Support Scams**: Scammers pose as tech support, alleging urgent computer issues like viruses, gaining remote control of computers to access sensitive information or install malware.

- **Debt Relief and Credit Repair Scams:** Fraudsters offer debt relief or credit repair services for a fee, often worsening the victim's financial situation.

- **Business and Investment Scams:** Scammers pretend to be financial experts, persuading victims to invest money in fraudulent schemes.

- **Charity Scams:** Perpetrators pose as charity workers soliciting donations for fake causes, with funds going directly to the scammers.

- **Auto Warranty Scams:** Scammers make fake calls about car warranties, seeking personal information or payments for nonexistent warranty renewals.

- **Parcel Scams:** Aimed at immigrants, scammers claim a parcel is linked to a financial crime, posing as couriers and police to coerce money for a fake investigation.

- **Kidnapping Scams:** Scammers claim to have kidnapped a relative, using research or social engineering to extract ransom payments.

2.2 Current Approaches and Emerging Frontiers in Vishing Detection

It's evident that vishers possess an extensive arsenal of options, tactics, and scenarios, making the challenge of effectively detecting vishing attacks a complex one. Historically, methods like real-time detection of caller ID spoofing, with software like STIR/SHAKEN [2] and iVisher [6] and blacklisting phone numbers, represented a critical line of defense against such attacks.

However, the integration of machine learning into cybersecurity presents an evolving frontier. These new tools offer enhanced capabilities to security professionals, equipping them to more effectively counteract the sophisticated strategies employed by vishing scammers. This evolution in anti-vishing technology reflects the dynamic nature of the cybersecurity field, where constant innovation is essential to stay ahead of threats.

Efforts to integrate machine learning into vishing detection have primarily focused on analyzing metadata, both statistical and non-statistical, such as CALLING_NUMBER, CALLED_LOCATION, and CALL_DURATION [10]. However, the area of real-time detection that involves processing the actual speech content within a scam call, post-answer by the victim, remains a largely unexplored frontier in the field. This gap suggests potential for future advancements in vishing detection methodologies that could more directly address the nuances of in-call scam tactics. In this dissertation, we shift our focus to developing a system capable of real-time vishing detection. This system will leverage the actual conversational content of calls, utilizing machine learning algorithms. This approach represents a new direction in vishing detection, aiming to directly analyze the dynamics of scam calls as they occur.

2.3 Why Real-Time In-Call Detection Is Essential

Previous efforts to mitigate vishing attacks have predominantly focused on the pre-answer phase of the phone call. Pioneering research in this domain, such as the work of Xing [10], Song [6], and Manh-Hung Tran [12], has been directed towards proactive strategies against vishing. The concept of real-time detection during a call is relatively novel in this field. Lee and Park's [11] research was among the first to explore real-time vishing detection. Utilizing machine learning techniques, their study specifically addressed vishing attacks in the Korean language and yielded some promising results.

Why though, is real-time in-call detection essential? Real-time in-call vishing detection is mandatory for several reasons, which can be effectively communicated through bullet points:

- Limited Individual Defense: Once the phone call is answered, individuals primarily rely on their personal knowledge, emotional stability, and understanding to defend themselves against vishing attacks. This reliance is often insufficient as scammers are typically well-prepared, extensively researched, and skilled in manipulative tactics.

- Emotional Manipulation: As mentioned before, Vishers exploit emotional vulnerabilities, using tactics like fear, urgency, and sympathy, which can overwhelm the victim's rational decision-making. Real-time detection can provide a safeguard during these high-pressure situations, where individuals might not be able to think clearly.

- Information Discrepancy: Scammers often have more information about the victim than the victim has about the caller, creating an imbalance that can be exploited. Real-time detection systems can help level this playing field by providing additional context or warnings about the call.

- Sophistication of Scams: Vishing scams have evolved to be highly sophisticated, often bypassing traditional pre-answer detection methods. Real-time detection can adapt to these evolving tactics, offering a dynamic line of defense as the call progresses.

- Support for vulnerable populations: Certain groups, like the elderly or less tech-savvy individuals, may be more susceptible to vishing. Real-time detection can provide an additional layer of protection for these vulnerable populations.

Hence, it becomes clear that there is a pressing need for an enhanced defensive system. Such a system would bolster individual protection during the post-answer phase of phone calls, particularly in situations where proactive vishing detection systems fall short.

2.4 Addressing the Identified Problem

The objective of this dissertation is to explore the feasibility and effectiveness of a system that processes phone call conversations in real time for vishing detection. This proposed system will integrate technologies like speech-to-text conversion, machine learning algorithms, and streaming techniques to accomplish its goals. The research will focus on key questions:

- Is it possible to train a machine learning algorithm to detect vishing within a conversation as it happens?

- Can such an algorithm provide real-time predictions with a sufficiently low response time to effectively protect the victim?

This dissertation is structured to methodically address the research questions posed. Chapter 3 delves into the PySpark framework, which is pivotal for data preprocessing, model training, and enabling the streaming functionality of the system. Chapter 4 discusses the implementation of the Speech to Text module, with a particular focus on the integration of the Google Speech to Text API.

Chapters 5 and 6 are dedicated to dataset creation and preprocessing, detailing the methodologies and approaches used in preparing the data for analysis. Chapter 7 outlines the architecture of the selected machine learning algorithms, covering aspects of their training, hyperparameter tuning and useful knowledge on the machine learning sector.

Chapter 8 provides an overview of the system’s functionality as a cohesive unit. The testing procedures and their corresponding results are the focus of Chapter 9, offering insights into the system's performance and efficacy.

The dissertation concludes with Chapters 10, which reflect on the challenges encountered during system development and propose potential future enhancements. This final chapter also presents overall deductions and conclusions drawn from the research.

3. PySpark

PySpark, serving as the Python API for Apache Spark, allows us to utilize the capabilities of Spark in a Pythonic way. In this section, we will examine the components of Spark that contributed to the proposed solution. As the underlying framework upon which PySpark is built, the advanced data processing capabilities of Spark are pivotal to our research. Its potent architecture facilitates rapid and scalable handling of extensive datasets. This provides a

critical backdrop for appreciating how PySpark leverages these features, tailoring them to meet the specific requirements of our application.

3.1 Apache Spark Fundamentals

Apache Spark stands as a formidable open-source engine for analytics, renowned for its proficiency in handling extensive data processing tasks. This unified analytics engine is adept at managing both batch and real-time data processing, positioning it as an invaluable resource in data science and machine learning fields. Its versatility in processing massive datasets efficiently makes it a cornerstone technology for modern data-driven applications.

3.1.1 Spark Architecture

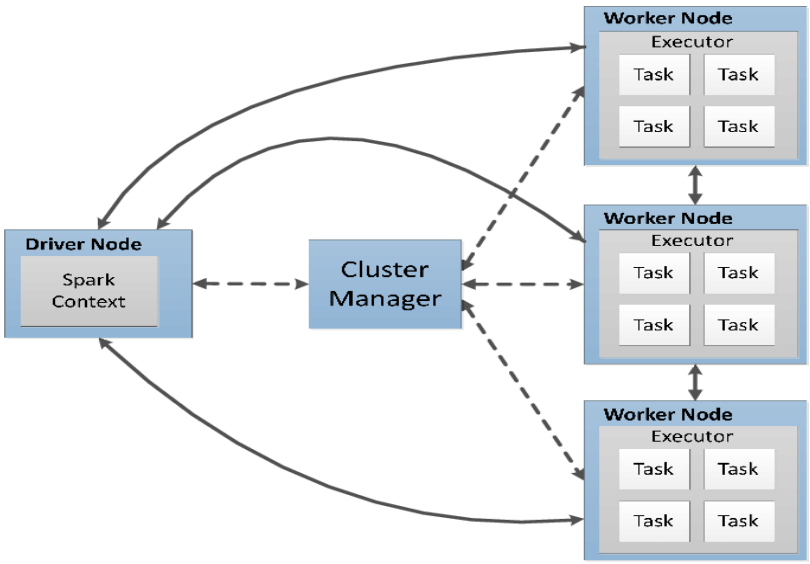


Fig 2: Spark Cluster Architecture [16]

Spark operates as a distributed computing system; in this section we explore its architecture. The architecture is depicted in Figure 3.1 as a **Spark Cluster**, showcasing its ability to coordinate complex processing tasks across multiple machines for efficient data management and computation. At the center of the architecture lies the **Driver Node**, in which the Spark Context is housed. This central coordinator initiates and manages the lifecycle of various Spark jobs. It converts the user application into tasks that are then distributed across the cluster for execution. The **Cluster Manager** is the resource negotiator responsible for allocating resources among the various applications running on the cluster. It ensures optimal utilization of resources and manages the distribution of tasks to the worker nodes. The **Worker Nodes** represent the distributed computational units of the cluster. Each worker node has Executors, which are processes responsible for executing tasks assigned to them by the driver node. Executors run the tasks in multiple threads, which allows Spark to execute tasks in parallel, significantly improving performance. **Tasks** are the smallest units of work in Spark, and they carry out computation and data processing. They are executed by the executors to process the data. The distribution of tasks across executors allows Spark to handle large datasets efficiently [13].

3.1.2 Spark Components

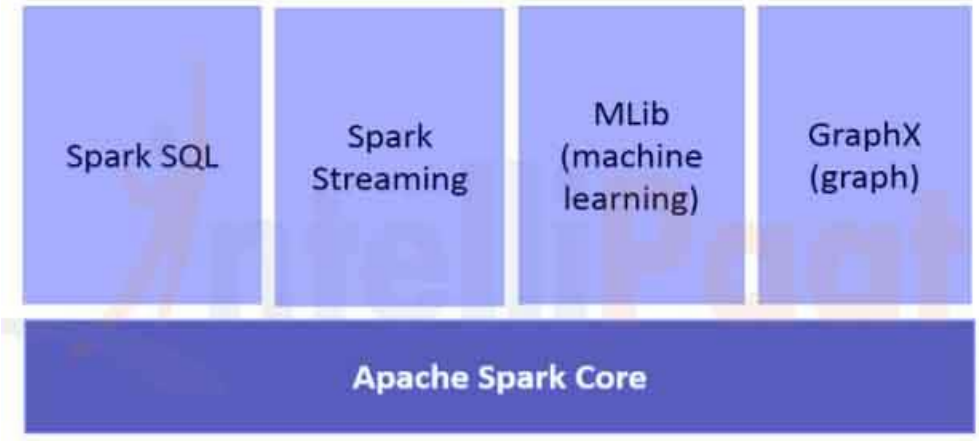


Fig 3: Spark Main Components [17]

The Apache Spark system is composed of several core components, which include the Spark Core and a range of high-level libraries. These libraries cater to specific needs, such as MLlib for machine learning applications, GraphX for graph analytics, Spark Streaming for real-time data processing, and Spark SQL for handling structured data.

3.1.3 Spark Core

Spark Core is the foundational part of Spark, upon which all other functionalities are built. Spark Core provides a variety of core features for distributed task dispatching, scheduling, and basic I/O functionalities, which form the heart of the Spark engine.

To further enhance our comprehension of Spark Core's functionality, and consequently the overall operation of Spark, our examination will pivot to the Resilient Distributed Dataset (RDD).

3.1.3.1 Resilient Distributed Dataset

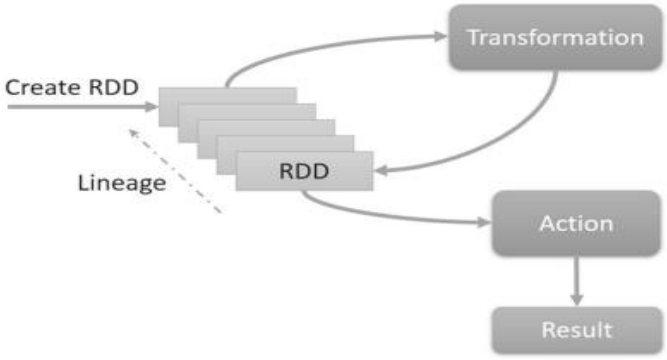


Fig 4: RDD lineage graph [13]

The Resilient Distributed Dataset (RDD) is the foundational element of Spark, providing the framework for fault-tolerant data distribution and parallel processing that underpins the entire system. RDDs can be generated from external data sources or transformed from

existing RDDs. As a key characteristic, RDDs maintain a lineage graph that allows for the efficient recompilation of data in case of node failures, as shown in Figure 4, rather than relying on data replication. This facilitates robust in-memory data processing and supports a variety of computational models. Moreover, RDDs enable complex operations through transformations, which are operations that create a new RDD from an existing one (e.g. map, filter), and actions, which are operations that trigger computation and produce non-RDD values (e.g. reduce, collect). Both transformations and actions are executed across the cluster only when necessary, optimizing resource utilization and performance [13].

3.2 Spark SQL

Having established the foundational role of Resilient Distributed Datasets (RDDs) in Spark's architecture, we now transition to another cornerstone of Spark's data handling capabilities—Spark SQL. This component is integral to our system, particularly in the context of data preprocessing, a topic we will delve into in detail in Chapter 6. Spark SQL extends Spark's functionality to allow for structured data processing and SQL queries, **building upon the groundwork laid by RDDs.** This integration not only simplifies working with structured data but also optimizes query execution through advanced optimization techniques, bridging the gap between traditional database management systems and modern big data processing.

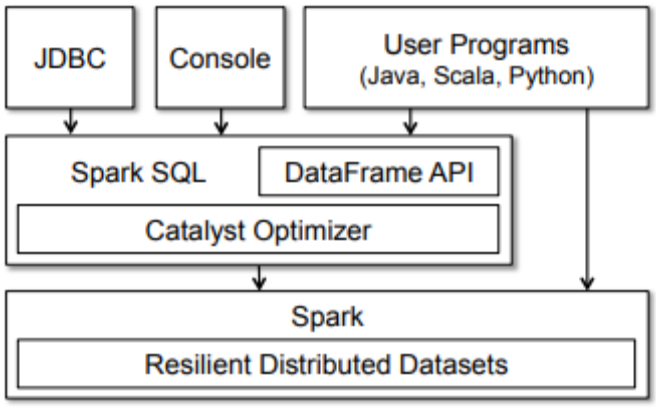


Fig 5: Spark SQL Interaction with Spark and RDDs [15]

The foundational concept in Spark SQL is the DataFrame, which is essentially a distributed set of rows that all conform to the same structure. Analogous to a table in a relational database, a DataFrame maintains awareness of its structure (schema), enabling it to support a variety of relational operations. These operations allow for more efficient query execution due to the inherent optimization in DataFrames, setting them apart from the more general RDDs. DataFrames in Spark SQL are adept at handling a variety of relational database operations, which include:

- **Projection**: Selecting specific columns from the DataFrame.
- **Filtering**: Applying conditions to the data, akin to the SQL 'WHERE' clause.
- **Join**: Combining two DataFrames based on a common key.
- **Grouping**: Aggregating data by specific categories, like the SQL 'GROUP BY' operation.

3.3 Spark's Machine Learning Package

In this section of the thesis, we focus on the Machine Learning package of Apache Spark, recognized for its scalable machine learning applications. Referencing the work of Assefi, Behravesh, and Liu (2017) [14], who tested Spark's machine learning capabilities on extensive datasets, it's evident that Apache Spark's machine learning package stands as a leading platform in machine learning for big data analysis. The Machine Learning package of Spark has been crucial in the development of our proposed system, augmenting Spark's foundational features with advanced machine learning functionalities. This package is an essential part of the Spark ecosystem, facilitating the effective implementation of machine learning algorithms on distributed data structures. The selection of Apache Spark's ML package for this project is primarily due to its proficiency in managing and processing large datasets, a requirement anticipated to be crucial for the solution of the problem addressed by our proposed system.

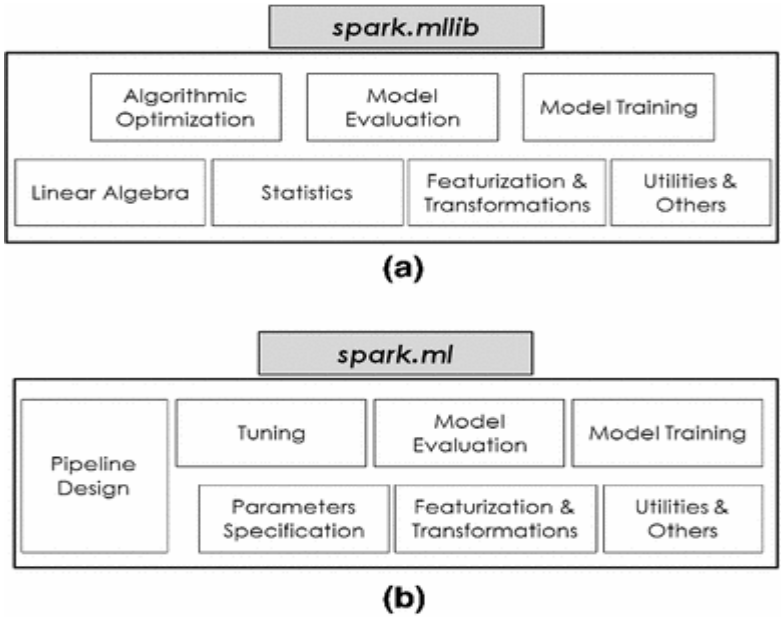


Fig 6: Key Features of Spark's Machine Learning package [18]

The Spark ML package consists of two libraries, MLlib and ML, which are going to be discussed in detail in the following sections.

3.3.1 Spark MLlib

Spark MLlib, the original machine learning library for Spark, is based on the Resilient Distributed Dataset (RDD) abstraction. Therefore, MLlib is suited for RDD-based applications and provides a wide range of machine learning algorithms that are scalable and optimized for big data processing. MLlib's capabilities, that differentiate it from Spark ML, include the following:

- **Linear Algebra**: Spark's MLlib offers both distributed and local abstractions for matrices and vectors, accommodating dense and sparse data representations. Sparse data structures are particularly valuable in big data analytics, as they are frequently encountered due to various factors such as high dimensionality, specific feature transformations, and the prevalence of missing values in large datasets [13].

- **Statistics**: This package features functions for summary statistics, correlation analysis, hypothesis testing, and sampling. Additionally, it provides kernel density estimation, streaming data testing, and random data generation from various distributions [13].

- **Algorithmic Optimization**: Spark MLlib supports two core optimization methods [20]:

    - **Gradient Descent Methods**: These methods involve iterative procedures to minimize the objective function, which in the context of machine learning is often the loss function representing the error between predictions and actual values. Key gradient descent methods included are:

        - **Gradient Descent**: This optimization method seeks a local minimum of the function. In each iteration, it updates the parameters in the direction of the negative gradient of the function at the current point, effectively moving towards the steepest descent to reduce the loss.

        - **Stochastic SubGradient descent (SGD)**: SGD is a variation of gradient descent where the gradient is estimated using a randomly selected subset of data rather than the full dataset. This estimation makes the method particularly suitable for large-scale data processing, as it significantly reduces computational cost while still converging towards the minimum.

    - **Limited-Memory BFGS (L-BFGS**): An optimization algorithm in the family of quasi-Newton methods. L-BFGS improves upon the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [21] by using a limited amount of computer memory. It's designed to optimize the performance of machine learning algorithms, especially when dealing with large datasets or in cases where the number of parameters is too high for the full BFGS method to handle efficiently.

3.3.2 Spark ML

In the evolving landscape of Apache Spark's machine learning offerings, Spark ML represents the next generation library, built upon the robust DataFrame API. Diverging from MLlib's reliance on the RDD framework, Spark ML caters to DataFrame-centric applications, offering a sophisticated, high-level API tailored for the construction of machine learning pipelines.

The inception of Spark ML marks a significant advancement over its predecessor, MLlib, by incorporating the Machine Learning Pipelines API, which will be discussed in the following sub-section. Additionally, Spark ML enhances model development through built-in hyperparameter tuning capabilities, facilitating the optimization of model parameters for improved performance.

Finally, Spark ML is integrated with the broader Spark ecosystem, most notably Spark SQL. This integration enables fluid data handling, allowing for efficient transitions from data preprocessing, performed with SQL-like commands, to subsequent phases of machine learning model development.

3.3.2.1 Machine Learning Pipelines API

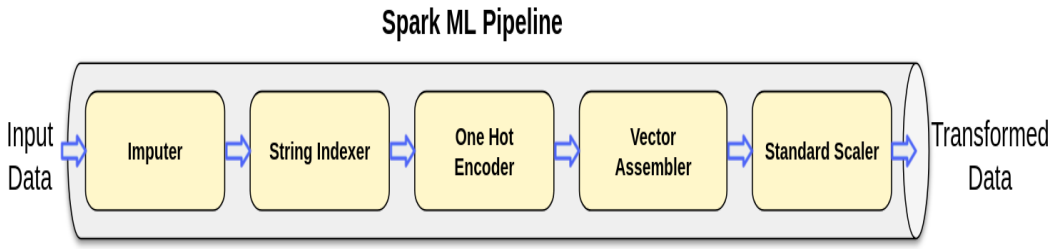


Fig 7: Spark Machine Learning Pipeline Example [23]

The Pipelines API introduced by Spark ML revolves around several core concepts:

- **DataFrame**: Within this ML API, the DataFrame, derived from Spark SQL, serves as the dataset for machine learning. It can accommodate diverse data types, for instance, columns that contain text, feature vectors, actual labels, and predicted values.

- **Transformer**: A Transformer is a type of algorithm that can convert one DataFrame into another. For example, a machine learning model can be viewed as a Transformer, taking a DataFrame with features and outputting a DataFrame that includes predictions.

- **Estimator**: Estimators are algorithms that, upon being "fitted" to a DataFrame, yield a Transformer. This fitting process typically involves training, as seen in learning algorithms that output a predictive model.

The Pipeline itself is a mechanism for stringing together multiple Transformers and Estimators to define a machine learning workflow. This sequence ensures that data transformation and algorithm application are carried out in an orderly and meaningful manner [22].

3.3.2.2 Spark ML Model Tuning

Spark ML provides built-in tools specifically designed for tuning hyperparameters, which are configurable parameters that are not directly learned from the data. Such tools are:

- **ParamGridBuilder**: This tool is used to construct a grid of parameters to search through during the tuning process. It systematically builds out a combination of parameters and their respective values that you want the tuning algorithm to test. This grid will be used in conjunction with a **model evaluation tool** to systematically work through multiple combinations, evaluating the performance of each to determine the best solution.

- **CrossValidator**: Cross-validation is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent dataset. The CrossValidator tool in Spark ML divides the dataset into a set number of folds, which are used as separate training and test datasets. For each parameter combination specified by ParamGridBuilder, CrossValidator trains the model on the training folds and evaluates the results on the test fold. It repeats this process for each combination of parameters in the grid, allowing it to identify the most effective parameters.

- **TrainValidationSplit**: This is another model-tuning tool provided by Spark ML. Unlike CrossValidator, which uses multiple folds, TrainValidationSplit only splits the data into

two sets: one for training and the other for validation. It is a simpler and faster approach, especially useful when you want a quick estimation of the best model parameters. The TrainValidationSplit method can be particularly useful when dealing with large datasets, where the computational cost of cross-validation may be too high.

The decision to select a particular library for our project depended on multiple factors, each playing a crucial role in determining the most suitable tool for our needs. For our specific solution, we opted for Spark ML, influenced primarily by its array of advanced tuning features and functionalities, primarily the access to the **pipelines API**, its ease of use and its scalability.

3.4 Spark Structured Streaming

Fig 8: Typical Streaming Process [24]

Structured Streaming extends Spark SQL's powerful data processing capabilities to handle real-time data. In essence, Spark Streaming can source data from a variety of origins, including Apache Kafka, Kinesis, or simple TCP sockets. This data is converted into streams and segments them into discrete batches, and is subsequently processed by the Spark engine, culminating in a continuous output of processed data, also delivered in batch form [24]. In Chapter 8, we will delve into how Spark Streaming significantly enhances our system by substantially reducing response times. Its role in the proposed system is crucial, contributing critically to overall efficiency and performance.

4. Speech-To-Text

4.1 Technology Analysis

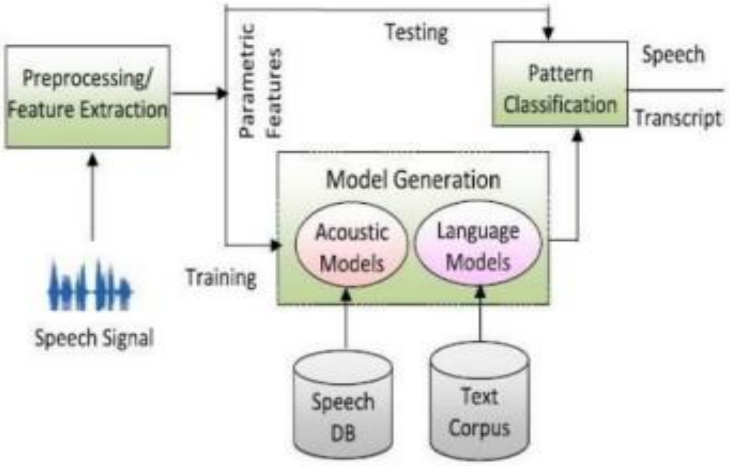


Fig 9: General Speech to Text Architecture [26]

This part focuses on the speech-to-text (STT) module integrated into the proposed system. STT technology, at its core, is a form of speech recognition software that leverages computational linguistics to convert spoken language into text. It is often referred to as speech recognition or computer speech recognition and it operates by capturing audio input and transforming it into an accurate, editable transcript. The functioning of a speech-to-text system hinges on advanced computational processes. It employs linguistic algorithms to distinguish and interpret auditory signals from spoken words, subsequently converting these signals into text represented by Unicode characters. This conversion process is underpinned by a machine learning model. The process of converting speech to text generally involves the following key steps [25]:

- Speech, which originates as a series of vibrations from vocal articulations, is captured by the speech-to-text technology. These sound vibrations are then translated into a digital format using an analog-to-digital converter, enabling the subsequent processing by the system's computational components.

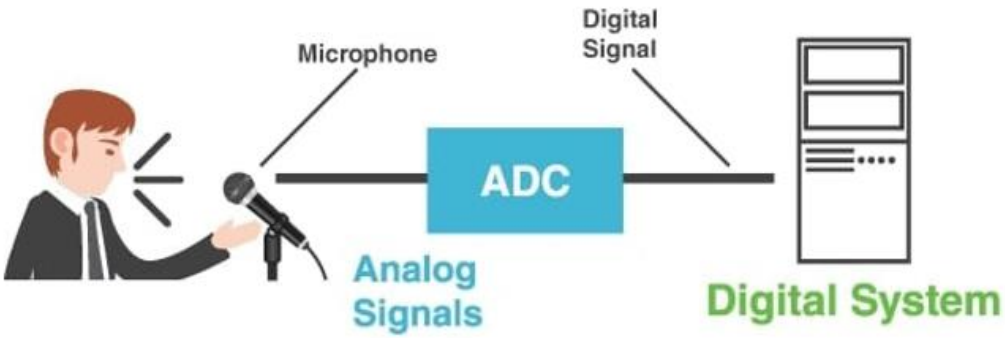


Fig 10: Speech to Text: Analog to Digital signal conversion

- The analog-to-digital converter takes sounds either from an audio file or from a real time sound stream such as a microphone, measures the waves in detail, and filters them to distinguish the relevant sounds.

- The sounds are then segmented into hundredths or thousandths of seconds and are then matched to phonemes which are the smallest distinctive sound units in a language that differentiate one word from another, such as the difference between the ‘c’ and ‘b’ sounds in ‘cat’ and ‘bat’.

- The identified phonemes are processed through a network using a mathematical model. This model compares the phonemes to a database of recognized sentences, words, and phrases for accurate interpretation.

- Finally, the speech-to-text system generates the output, presenting it either as written text or as a command for the computer, based on the most probable interpretation of the audio input.

4.2 Google Speech-To-Text API

In the proposed system, we have chosen the Google Speech-to-Text API for reasons that will be detailed subsequently. This API exemplifies major advancements in speech recognition technology. Developed by Google, it leverages sophisticated artificial intelligence and machine learning algorithms to accurately transcribe spoken language into text. As a part of Google's Cloud services, it provides developers with an efficient means to integrate speech recognition capabilities into diverse applications. The development of this API shows a significant advancement in digital communication technologies, effectively improving the interaction between human speech and computational interpretation.

The Google Speech-To-Text API offers an extensive set of features [27]:

- **Wide Language Support:** It supports over 125 languages and variants.

- **Real-Time recognition:** It provides immediate speech recognition results, processing audio input either streamed live from a microphone or from prerecorded files, whether inline or via Cloud Storage. This capability was a key factor in our decision to utilize Google Speech-to-Text in our system. We primarily leverage this feature for streaming both microphone and loopback audio data from the application to the API, subsequently receiving the transcribed text for analysis.

- **On-Premises Solution:** With Speech-to-Text On-Prem, users have the ability to leverage Google's technology within their own private data centers, offering full control over infrastructure and data privacy which is a key point when dealing with sensitive data.

- **Multichannel Recognition:** This feature makes the API able to identify and annotate distinct audio channels in complex audio environments, such as multichannel recordings or video conferences.

- **Noise Robustness:** Google Speech-to-Text efficiently processes audio with background noise, minimizing the need for additional noise reduction. This feature significantly influenced our choice to use this API in our system.

- **Domain-Specific Models:** Google offers specialized models for different applications like voice control, phone call transcription, and video transcription, each optimized for specific quality requirements.

- **Content Filtering:** An inbuilt profanity filter that helps in detecting and filtering inappropriate or unprofessional content in audio data.

- **Transcription Evaluation:** Users can evaluate the quality of transcriptions by uploading their voice data and iterating on the configuration, all without the need for coding.

- **Automatic Punctuation:** The API effectively punctuates transcriptions, accurately placing commas, question marks, periods, and other punctuation marks.

- **Speaker Diarization:** In the time of writing this dissertation this feature is still in its beta phase. It predicts the speaker for each part of the conversation, aiding in identifying who said what in multi-speaker scenarios.

The combined capabilities of the Google Speech-to-Text API align well with the requirements of our proposed system. Particularly noteworthy are its real-time prediction capabilities, enabling streaming speech-to-text conversion, and its inherent noise reduction proficiency as previously mentioned.

5. Dataset Creation

In the subsequent sections, we explore the dataset creation methodology of our research. Given the sensitivity of the data involved and the constraints in acquiring it, we opted to construct the dataset independently. This involved utilizing advanced AI tools, particularly the Chat GPT API, to generate a comprehensive dataset. To fully understand this process, we will first examine the workings of Generative Pretrained Transformers (GPT), setting the foundation for our discussion on the implementation of the Chat GPT API in developing our dataset.

5.1 Generative Pretrained Transformers

5.1.1 Transformers

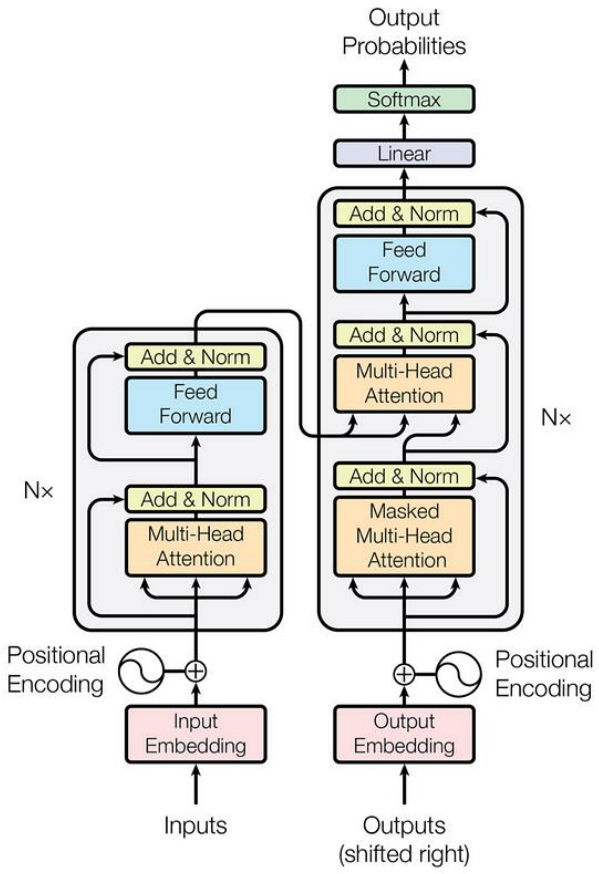


Fig 11: The Transformer – Model Architecture [30]

The transformer, a deep learning model employing the multi-head attention mechanism, distinguishes itself by the absence of recurrent units, unlike traditional sequence-to-sequence architectures like LSTM. This design feature allows for faster training compared to recurrent neural networks. Like LSTM in its objective to transform one sequence into another, the transformer is divided into two parts: the Encoder and the Decoder. This architecture has become prominent in training large language models on extensive datasets. The process involves splitting input text into n-grams and encoding these as tokens, with each token subsequently converted into a vector via word embeddings. Within the architecture, each

layer processes tokens contextually in relation to others within the context window, utilizing a parallel multi-head attention mechanism. This allows the model to amplify the importance of key tokens and diminish less significant ones, enhancing the model's ability to focus on relevant information [29][30].

5.1.2 Generative Pretrained Transformers

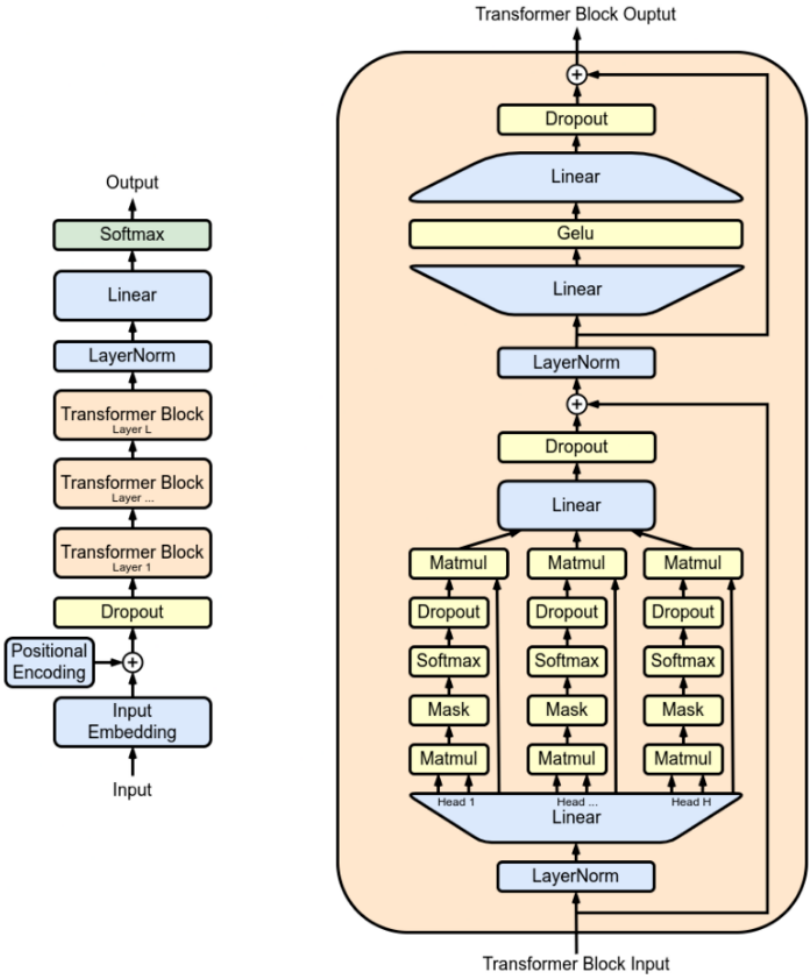


Fig 12: The Original Generative Pretrained Transformer Model [28]

Generative Pretrained Transformers (GPT) represent a class of Large Language Models (LLMs) and are a key component in the field of generative artificial intelligence. Utilized for a variety of natural language processing tasks, these neural networks are based on the transformer architecture. Pretrained on vast corpora of unlabeled text, GPTs exhibit the capability to generate text that mirrors human-like writing [28].

5.1.2.1 Utilizing GPT for Dataset Generation

Utilizing Generative Pretrained Transformers (GPT) for dataset generation capitalizes on their features, including rich language understanding and generation, the ability to produce customizable outputs, and generating data with diverse linguistic styles. Additionally, GPT models excel in simulating real-world applications and addressing data availability challenges, which was a primary factor in our decision to choose this method for dataset creation.

5.2 Data Generation with Chat GPT API

5.2.1 API Overview

Chat GPT is built upon the Generative Pretrained Transformers architecture and its fine tuned to handle conversational contexts. OpenAI, the developer of Chat GPT, provides the Chat GPT API which facilitates the creation of interactive and responsive chatbot systems capable of understanding and generating human-like text responses. This API can process and respond to a wide range of conversational inputs, maintain context over a series of interactions, and generate replies that are not only contextually relevant but also exhibit natural language coherence.

5.2.2 Prompt Engineering

In this research, we utilized the Chat GPT API, combined with Python, to generate our dataset. Prior to detailing the specific methods employed in data production, it's essential to understand the concept of prompts and the emerging field of prompt engineering. This field has gained significance with the advent of GPT systems in recent years and plays a crucial role in effectively interacting with the Chat GPT API [31].

Prompt engineering is the technique of crafting text inputs that are optimized for interpretation by generative AI models, such as Chat GPT. A prompt is essentially a piece of natural language text that outlines a task for the AI to execute. The art of prompt engineering might involve the formulation of a specific query, choosing a particular style, adding pertinent context, or assigning a role to the AI, e.g., "Act as a native French speaker" [31].

The ensuing pipeline provides a clear example of constructing a prompt designed to ensure that the AI algorithm fully understands the extent of the request.

- **Task:** A good prompt provides a clear task for the generative AI algorithm to pursuit.

- **Contextual Balance in Prompts:** In prompt creation, it's crucial to provide context, but with moderation. Striking a balance is key, as providing too much information can be as counterproductive as giving too little. The goal is to include just enough context to elicit an effective response.

- **Examples:** Providing an example of an ideal response to the algorithm can significantly enhance the quality of the resultant output, especially if the request is of complex nature.

- **Persona:** Incorporating a persona into the AI system can improve the probability of obtaining a response that closely mirrors real-life interactions. While not essential, this aspect of the prompt design is particularly beneficial for scenarios aiming to emulate specific individuals or target demographics.

- **Format:** Utilizing a specific format can significantly enhance the response quality, especially if the output is intended for specific uses, such as dataset creation. For instance, structuring the response in JSON format, which can later be converted into a dataframe, is an effective approach to streamline data processing.

5.2.3 Data Collection Strategy

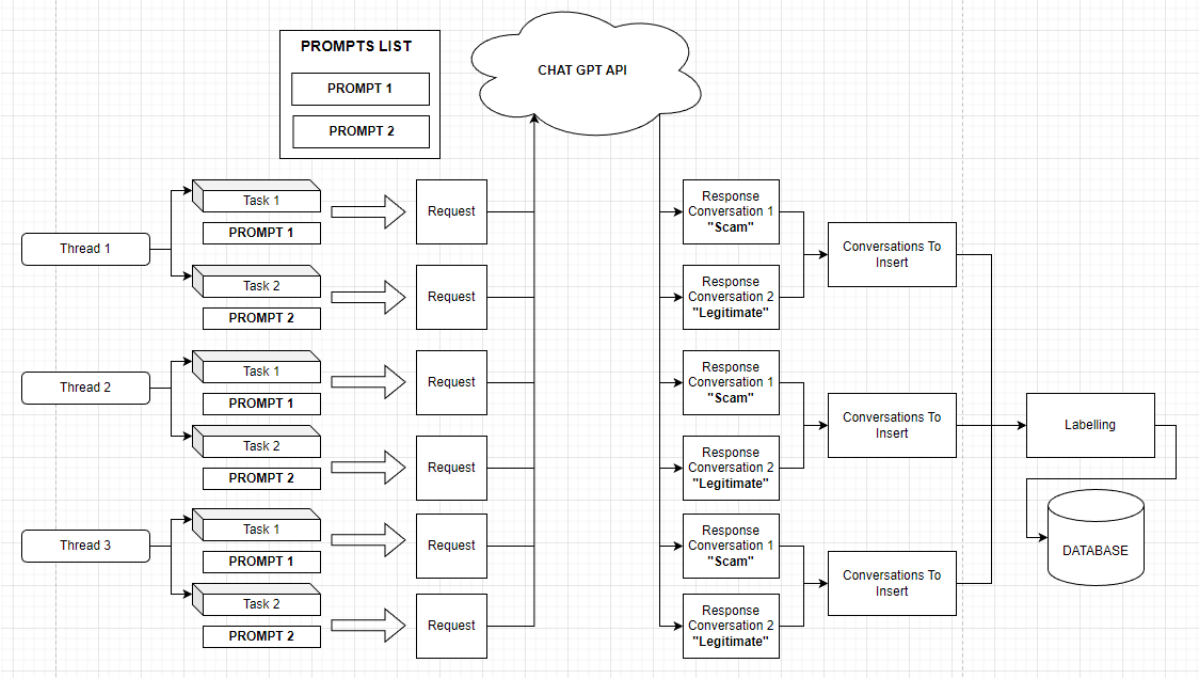


Fig 13: Dataset Creation Pipeline

To understand how the dataset was constructed the following points must be discussed:

- **Prompts**: Two prompts were created in order facilitate the construction of the dataset.

    - The first directs the AI algorithm to simulate a conversation between two individuals, with one person (the attacker) attempting to scam the other (the victim). The algorithm is guided to present the response in JSON format. Additionally, it is instructed to consider the characteristic attitudes and tones of scammers and their typical targets in phone scams, ensuring a realistic portrayal of such interactions.

    - The second prompt similarly guides the AI algorithm, but with a different context. Here, the algorithm is tasked with generating a routine conversation between a client and customer support. The objective is to create a dataset contrasting with the previous scam scenario, focusing instead on a typical, everyday interaction between a customer and support representative.

- **Multithreading:** To expedite dataset generation, a multithreading approach was employed. When launching the dataset creation program, multiple threads (denoted as 'N') are activated. Each thread is assigned two tasks: One to invoke the Chat GPT API using the first prompt (focused on simulating a scam interaction) and one to use the second prompt (geared towards creating a normal customer support dialogue).

- **Data Labeling Process:** Post-generation by the Chat GPT API, each conversation is appropriately labeled to facilitate subsequent analysis. Conversations derived from the scam-focused prompt are labeled as “1”, indicating a scam scenario, while those from the normal interaction prompt are marked as “0”, denoting a typical, non-scam

exchange. This labeling is crucial for distinguishing between the two types of interactions in the dataset.

- **Storing Conversations in CSV Format:** The concluding phase of the dataset creation involves saving the generated conversations into a CSV file, which serves as the dataset for our system. Each conversation is assigned a unique identifier upon entry into the CSV, ensuring easy differentiation and retrieval within the dataset. This step solidifies the structure and accessibility of the data for future use.

Conversation Id	Attacker_Helper	Victim
SsDyLEV3XJ4lc3	Hello, this is John from the ABC Bank. We noticed some suspicious activity on your account. Can I confirm if you made a purchase of $500 at an online store today?	Yes, I made that purchase. Thank you for checking before verifying!
SsDyLEV3XJ4lc3	Thank you for confirming. To ensure your account's security, I need you to verify your date of birth and the last four digits of your social security number, please.	I appreciate the caution, but I prefer not to share that sensitive information over the phone. Can we verify through another method?
SsDyLEV3XJ4lc3	Absolutely. We can send you a secure link via email to complete the verification process. Which email address should we send it to?	Please send it to Thank you for your assistance!
SsDyLEV3XJ4lc3	You're welcome! Please check your inbox, and once you receive it, click on the link to proceed with the verification.	Got it, I'll check my email and complete the verification promptly. Thanks again for your help!
SsDyLEV3XJ4lc3	My pleasure! Once you've completed the verification, your account will be secure. Should you have any further questions, feel free to reach out. Have a great day!	Thank you so much for your support. Have a wonderful day too!

Fig 14: Result of a normal (non-scam) conversation from Chat GPT API.

Conversation Id	Attacker_Helper	Victim
LPIvHieleyVcC1	Hello, this is Microsoft Support. Your computer has been infected with a virus.	Oh no, that's terrible! What should I do?
LPIvHieleyVcC1	Don't worry, we can fix it. But first, I need access to your computer.	I'm not sure if I should trust you. How can I be sure you're really from Microsoft?
LPIvHieleyVcC1	I understand your concern. You can verify my identity by checking the caller ID. It should display our official helpline number.	Okay, let me check. Oh, the number on my caller ID matches! That's reassuring.
LPIvHieleyVcC1	Great! Now, to fix the problem, I need you to download a remote access software.	Wait, why can't I just take my computer to a local technician?
LPIvHieleyVcC1	Taking your computer to a technician is expensive and time-consuming. We can save you both time and money by fixing it remotely.	I suppose that makes sense. Alright, what should I do next?
LPIvHieleyVcC1	Visit our website and download our remote access software. I'll guide you through the installation process.	Okay, I'm on the website now. Where should I click?
LPIvHieleyVcC1	Click on the 'Download Now' button and follow the instructions. Once installed, give me the access code displayed on your screen.	Alright, let me click on that. Done. The access code is 123456.
LPIvHieleyVcC1	Perfect! Now I have full control of your computer. I'll fix the virus issue for you.	Thank you so much for helping me. I really appreciate it.
LPIvHieleyVcC1	You're welcome. Just remember, never share your personal information or download any suspicious software in the future.	I'll keep that in mind. Thanks again for your assistance.
LPIvHieleyVcC1	No problem. Have a great day!	You too, goodbye!

Fig 15: Result of a scam conversation from Chat GPT API.

As illustrated in Figure 14 and Figure 15, the sample conversations generated by the Chat GPT API, while not flawless, effectively mirror a real-life scenario in its fundamental aspects.

5.2.4 Data Quality and Future Improvements

Although the dataset created via the outlined pipeline sufficiently serves this research's needs, enhancing its diversity is recommended for a more comprehensive representation. This enhancement could involve developing more sophisticated prompts or integrating real-

world data into the dataset. Additionally, since the conversations in the dataset adhere to a pattern as it would be expected from data generated by a machine, including a variety of more complex scam conversations in the dataset is advisable to accurately reflect the range of tactics employed by scammers and the intricacies of their fraudulent schemes.

6. Dataset Preprocessing

In any machine learning pipeline, data preprocessing is a vital step. It involves various processes designed to enhance data quality. Within this research, data preprocessing was conducted in two phases: the initial preprocessing to clean and prepare the data, and subsequent processing tailored for the data's compatibility with machine learning models.

6.1 Initial Preprocessing

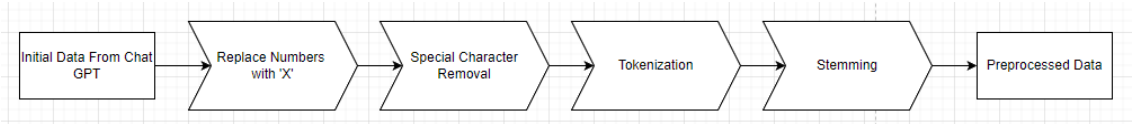


Fig 16: Initial Preprocessing Pipeline

In the initial preprocessing phase, the objective is to structure and standardize the data to ensure consistency for more intricate processing stages that follow. The initial preprocessing pipeline comprises three critical steps, as depicted in Figure 16. These steps are detailed in the ensuing sub-sections.

6.1.1 Numeric Character Replacement

During this step, numeric characters in each conversation are substituted with the letter 'X'. This approach serves several purposes:

- **Anonymization**: Either using the synthetically generated dataset or using a real one, the dataset may include data that resembles sensitive information, such as personal or financial details. Replacing numbers helps to anonymize this information, ensuring privacy and adherence to data protection standards like GDPR.

- **Standardization**: In the context of this study, the presence of a numeric value is more significant than the actual number itself. Substituting numbers with 'X' enhances data uniformity across the dataset.

- **Complexity Reduction**: Numeric values can add unnecessary complexity to machine learning models, particularly in natural language processing tasks.

- **Emphasis on Textual Data**: The objective is for the machine learning models to concentrate on textual and categorical data, rather than numeric values, to improve their performance on the relevant tasks.

6.1.2 Special Character Removal

In this step special characters are removed from each conversation; this aims to support the following key points:

- **Feature Simplification**: To improve the focus of our models, during the training process, on those features that provide important information about the conversation we have opted to remove special characters since those might introduce

unnecessary additional features that can complicate the model without adding informative value.

- **Tokenization Consistency**: During tokenization, special characters can lead to the creation of tokens with embedded special characters or tokens that consist only a single special character. Removing these characters helps achieve a more consistent and precise tokenization process.

- **Improving Model Focus**: As already mentioned before, the aim is to direct the machine learning model's attention toward text that expresses sentiment or factual content, rather than parsing potentially extraneous details signified by special characters.

6.1.3 Tokenization

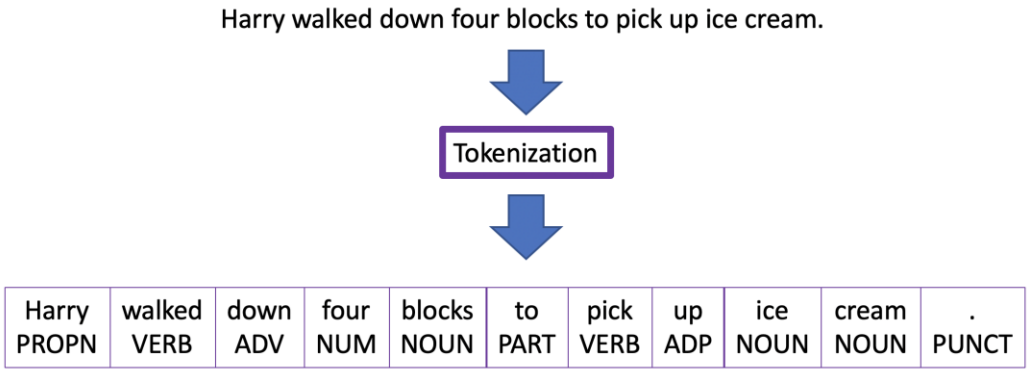


Fig 17: Word Tokenization Example [34]

In this stage, we explore tokenization, a fundamental step in natural language processing and machine learning. This process involves breaking down sentences, words, or larger bodies of text into smaller units called tokens. Tokenization serves the purpose of rendering human language into segments that are more manageable and interpretable for computational analysis. Tokenization can be categorized into three types: Word Tokenization, Character Tokenization, and Sub word Tokenization. For the purposes of this research, we focus on Word Tokenization, which segments text into distinct words [33].

In this dissertation, the focus is on utilizing the Natural Language Toolkit (NLTK) library in Python, with a specific emphasis on the word_tokenize function. This function operates using the PunktSentenceTokenizer, which is adept at identifying sentence boundaries through punctuation and character markers. Additionally, it incorporates the TreebankWordTokenizer, which effectively separates words within a sentence based on spaces and punctuation. This combination is highly effective for processing European languages, making it well-suited for this study where English, is the language of focus [35][36].

6.1.4 Stemming

	original_word	stemmed_words
0	connect	connect
1	connected	connect
2	connection	connect
3	connections	connect
4	connects	connect

	original_word	stemmed_word
0	trouble	troubl
1	troubled	troubl
2	troubles	troubl
3	troublemsome	troublemsom

Fig 18: Example of Stemming in the English language [38]

The last stage in our preliminary preprocessing sequence is stemming. As depicted in Figure 18, stemming involves condensing words with inflections down to their basic stem, which represents the core or fundamental form of the word. It's important to note that this stem may not always match the morphological root of the word [37][39].

A widely recognized and frequently utilized stemming algorithm is the Porter Stemmer. Originating in the 1980s, the Porter Stemmer stands as one of the foremost popular methods for stemming. Characterized by its simplicity and computational efficiency, it excels in handling English language's inflectional variations, like pluralization and tense changes. This ability significantly aids in generalizing patterns within text data. Tailored specifically for English, the algorithm demonstrates robust performance across a broad spectrum of English vocabulary, effectively accommodating even irregular word formations for example the word “running”. The irregularity in this instance is the double “n” before the addition of “-ing”. The algorithm would reduce “running” to its stem, “run” [37].

As a conclusion to the preprocessing pipeline, all conversations are saved in a new csv file with the format depicted in the subsequent image.

Conversation_ID	Attacker_Helper	Victim	Conversation_Type
GT1sURbxgG	[['hi', 'thi', 'is', 'john', 'from', 'the', 'ir', 'you', 'owe', 'us', 'xxxxx', 'in', 'back', 'tax'], ['ye', 'of',	[['im', 'sorri', 'i', 'dont', 'believ', 'you', 'can', 'you', 'provid', 'me', 'with', 'your', 'badg	1
TwaGOeC96w	[['hello', 'thi', 'is', 'xyz', 'bank', 'call', 'may', 'i', 'speak', 'with', 'victim', 'name', 'pleas'], ['goc	[['thi', 'is', 'victim', 'name', 'how', 'can', 'i', 'help', 'you'], ['im', 'sorri', 'i', 'dont', 'feel	0
V73ZDCviQL	[['hello', 'sir', 'thi', 'is', 'a', 'call', 'from', 'the', 'ir', 'we', 'have', 'found', 'discrep', 'in', 'your', '	[['ive', 'never', 'had', 'ani', 'issu', 'with', 'my', 'tax', 'befor', 'can', 'you', 'provid', 'me'	1
wNsWdbpeId	[['hello', 'thi', 'is', 'john', 'from', 'the', 'bank', 'i', 'am', 'call', 'to', 'confirm', 'a', 'recent', 'tran	[['sure', 'but', 'befor', 'that', 'can', 'you', 'pleas', 'confirm', 'which', 'bank', 'you', 'are	0
x0pSxAQx1K0abm	[['hello', 'thi', 'is', 'the', 'ir', 'call', 'our', 'record', 'show', 'that', 'you', 'owe', 'xxxx', 'in', 'tax',	[['oh', 'no', 'i', 'had', 'no', 'idea', 'can', 'you', 'give', 'me', 'some', 'more', 'inform', 'ab	1
I8QHkmsq5AdwAq	[['hello', 'thi', 'is', 'xyz', 'bank', 'can', 'i', 'speak', 'with', 'john', 'smith', 'pleas'], ['we', 'notic',	[['thi', 'is', 'john', 'smith', 'speak', 'may', 'i', 'know', 'whi', 'are', 'you', 'call'], ['oh', 'th	0
v9vN0BYTWa0e16	[['hello', 'thi', 'is', 'john', 'from', 'the', 'ir', 'your', 'tax', 'return', 'from', 'the', 'last', 'year', 'ha	[['im', 'sorri', 'i', 'dont', 'believ', 'you', 'can', 'you', 'give', 'me', 'your', 'employe', 'id'	1
pVeYjXj2DelffL	[['hello', 'thi', 'is', 'john', 'from', 'your', 'bank', 'may', 'i', 'speak', 'to', 'mr', 'smith'], ['hi', 'mr	[['speak', 'thi', 'is', 'mr', 'smith', 'what', 'can', 'i', 'help', 'you', 'with'], ['no', 'i', 'didnt	0
42e0AXIkESKc6y	[['hello', 'is', 'thi', 'xxxxxxxxxx'], ['i', 'am', 'call', 'from', 'the', 'ir', 'and', 'you', 'owe', 'us', 'mc	[['ye', 'who', 'is', 'thi'], ['i', 'dont', 'believ', 'you', 'can', 'you', 'prove', 'that', 'you', 'ar	1
vk2TK8MFd0hoZg	[['hello', 'thi', 'is', 'john', 'from', 'the', 'bank', 'im', 'call', 'in', 'regard', 'to', 'some', 'recent', 'i	[['oh', 'my', 'is', 'everyth', 'alright'], ['sure', 'my', 'name', 'is', 'sarah', 'johnson', 'and	0
YX24MIG7RJy5zD	[['hello', 'there', 'am', 'i', 'speak', 'with', 'mr', 'john', 'doe'], ['thi', 'is', 'toni', 'from', 'your', 't	[['ye', 'thi', 'is', 'john', 'who', 'is', 'thi'], ['no', 'i', 'did', 'not', 'make', 'ani', 'such', 'purc	1
VH7eX10IAQbZOV	[['hello', 'thi', 'is', 'the', 'ir', 'we', 'notic', 'that', 'you', 'owe', 'back', 'tax', 'and', 'are', 'at', 'risl	[['im', 'sorri', 'but', 'i', 'dont', 'think', 'that', 'true', 'i', 'alway', 'pay', 'my', 'tax', 'on', 't	1
VBwnvERVuU3jv9	[['hello', 'thi', 'is', 'john', 'from', 'the', 'bank', 'i', 'need', 'to', 'confirm', 'some', 'account', 'in	[['ok', 'sure', 'what', 'do', 'you', 'need'], ['i', 'dont', 'feel', 'comfort', 'give', 'that', 'ov	0
1gJHIiw7AGlHxH	[['hello', 'thi', 'is', 'the', 'ir', 'we', 'have', 'detect', 'fraudul', 'activ', 'on', 'your', 'account'], ['y	[['oh', 'realli', 'can', 'you', 'provid', 'me', 'with', 'some', 'more', 'inform'], ['im', 'sorri	1

Fig 19: Indicative Data Post Initial Preprocessing

Before Preprocessing	After Preprocessing
Hello, this is Microsoft Support. Your computer has been infected with a virus. **Don't worry, we can fix it. But first, I need access to your computer.**	[['hello', 'thi', 'is', 'microsoft', 'support', 'your', 'comput', 'ha', 'been', 'infect', 'with', 'a', 'viru'], ['dont', 'worri', 'we', 'can', 'fix', 'it', 'but', 'first', 'i', 'need', 'access', 'to', 'your', 'comput']]

Fig 20: Example Output from the Initial Preprocessing Pipeline

6.2 Preprocessing Pipeline for Model Training

For effective model training, it is important to convert the data into a format that can be comprehended and processed by machine learning algorithms. This involves transforming the raw data, from the initial preprocessing pipeline, into a numerical representation, as machine learning models inherently require data in a quantifiable format for analysis and pattern recognition.

To achieve the necessary numeric representation, the data will be subjected to a series of transformation steps, employing the PySpark pipelines API, as detailed in Section 3.3.2.1. This process involves the data passing through a sequence of transformers, collectively forming a pipeline, which is illustrated in the subsequent figure.

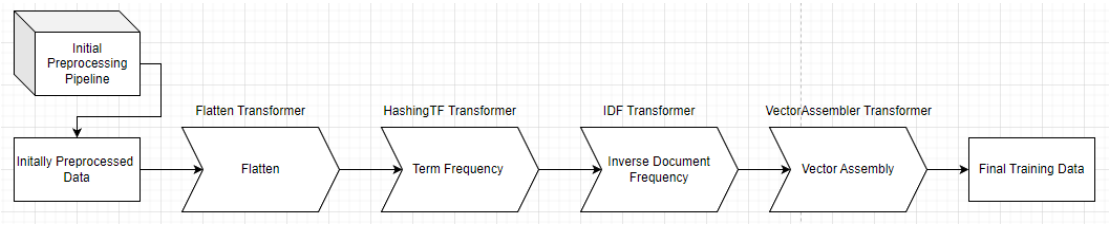


Fig 21: Preprocessing Pipeline for Model Training

6.2.1 Flatten Transformer

The first stage in the preprocessing pipeline for model training, as depicted in Figure 20, involves flattening each conversation. Prior to this stage, conversations are represented in two columns as per Figure 19, with each column containing the sentences spoken by an individual in the conversation. To facilitate more efficient preprocessing, it is essential to convert these lists of sentences into a single array for each individual. This transformation will result in an array that encompasses all the words uttered by each participant in the conversation.

To facilitate this transformation, the Flatten Transformer provided by the PySpark Machine Learning package is employed. This transformer is designed to merge an array of arrays into a single, unified array, deeming it suitable for our requirements [40].

6.2.2 TF-IDF

To transform the textual data into a numeric format, we utilize the TF-IDF (Term Frequency-Inverse Document Frequency) algorithm. This algorithm evaluates the significance of a word within a document and is extensively employed in text-based systems. The TF-IDF algorithm comprises two key components, Term Frequency (TF) and Inverse Document Frequency (IDF) [41].

- The term frequency is given conceptually as follows:

$$\boldsymbol{TF(t,d) = \frac{Number\ of\ times\ the\ term\ t\ appears\ in\ a\ document\ d}{Total\ Number\ of\ all\ terms\ in\ the\ document\ d}}$$

In a more analytical representation, the term frequency can be calculated as [41]:

$$\boldsymbol{TF(t,d) = \frac{ft,d}{\sum_{\acute{t}\epsilon d} f\acute{t},d}}$$

where ft, d signifies the raw count of a term in a document 'd'. In this research, the term 'document' refers to an individual row within the Attacker_Helper or Victim columns, as depicted in Figure 19.

- The inverse document frequency is given conceptually as follows:

$$\boldsymbol{IDF(t,D) = \log\left(\frac{Total\ number\ of\ documents\ N}{Number\ of\ documents\ where\ the\ term\ t\ appears}\right)}$$

In a more analytical representation, the inverse document frequency can be calculated as [41]:

$$\boldsymbol{IDF(t,D) = \log\left(\frac{N}{|\{d\epsilon D: t\epsilon d\}|}\right)}$$

Where 'N' represents the total number of documents in the corpus: In the context of this research, the corpus is defined as each individual row in the column to which the IDF is applied. Consequently, 'N' equates to the total number of rows in the dataset [41].

- Finally, the TF-IDF metric is given by the following equation:

$$\boldsymbol{TF_{IDF(t,d,D)} = TF(t,d) \times IDF(t,D)}$$

The TF-IDF metric proficiently signifies the relevance of a word within a document in a corpus. A higher TF-IDF score for a particular term suggests its rarity in both the document and the corpus, thereby denoting greater significance.

6.2.2.1 HashingTF Transformer

To incorporate the term frequency step into the pipeline, the HashingTF transformer provided by PySpark is utilized. This transformer transforms a document into a fixed-size vector, a requirement for machine learning models that necessitate a constant input vector size to function optimally.

To achieve this fixed vector sized the HashingTF transformer uses the “Hashing Trick”. The "Hashing Trick" is an efficient technique for encoding features, particularly useful in handling high-dimensional data like text. Using a hash function, it maps features into indices within a provided fixed-size vector. This approach is efficient as it does not require a vocabulary of all possible features, saving memory and reducing complexity. A notable limitation is the occurrence of collisions, where different features map to the same index, potentially leading to information loss, hence is it important to find the correct size for the vector in order to avoid such collisions [42].

Considering that phone conversations typically involve a limited number of unique words; we have selected 200 as the number of features for the HashingTF transformer. This decision is based on the expectation that no more than 100 unique words will be utilized in any given conversation.

6.2.2.2 IDF Transformer

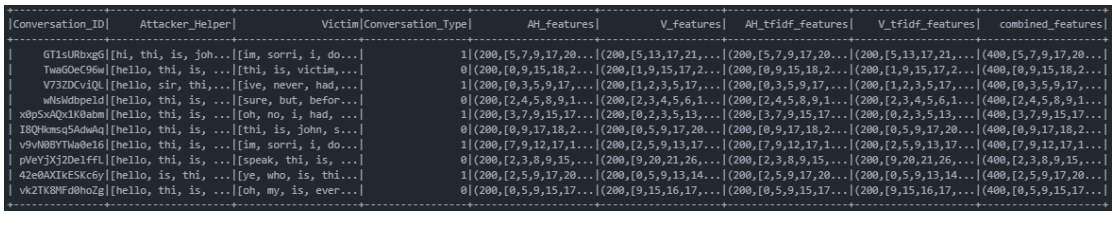
Once the data has been processed through the HashingTF transformer, it is converted into a collection of TF vectors, with each row or document in the 'Attacker_Helper' and 'Victim' columns represented by a sparse vector. To finalize the TF-IDF process, these TF vectors must then be passed through the IDF transformer. This transformer is specifically tailored to operate on the numerical vectors generated in the TF phase, adjusting the term frequencies within these vectors according to the terms' prevalence or rarity throughout the entire dataset.

The final output after applying IDF will be TF-IDF sparse vectors. These vectors represent both the term frequencies and the importance of the terms in the context of the entire dataset.

6.2.3 Vector Assembler Transformer

The concluding phase in the preprocessing pipeline for model training is Vector Assembly. During this stage, the feature vectors generated from the 'Attacker_Helper' and 'Victim' columns in previous steps are merged into a single, comprehensive vector. The resultant vector is effectively twice the size of either individual vector. This concatenation is achieved by inputting the data from both columns into the Vector Assembler transformer provided by PySpark.

An indicative result of the complete pipeline output can be seen in the subsequent figure.



Conversation_ID	Attacker_Helper	Victim	Conversation_Type	AH_features	V_features	AH_tfidf_features	V_tfidf_features	combined_features
GT1sURbxgG	[hi, thi, is, joh...	[im, sorri, i, do...	1	(200,[5,7,9,17,20...	(200,[5,13,17,21,...	(200,[5,7,9,17,20...	(200,[5,13,17,21,...	(400,[5,7,9,17,20...
TwaGOeC96w	[hello, thi, is, ...	[thi, is, victim,...	0	(200,[0,9,15,18,2...	(200,[1,9,15,17,2...	(200,[0,9,15,18,2...	(200,[1,9,15,17,2...	(400,[0,9,15,18,2...
V73ZDCviQL	[hello, sir, thi,...	[ive, never, had,...	1	(200,[0,3,5,9,17,...	(200,[1,2,3,5,17,...	(200,[0,3,5,9,17,...	(200,[1,2,3,5,17,...	(400,[0,3,5,9,17,...
wNsWdbpeld	[hello, thi, is, ...	[sure, but, befor...	0	(200,[2,4,5,8,9,1...	(200,[2,3,4,5,6,1...	(200,[2,4,5,8,9,1...	(200,[2,3,4,5,6,1...	(400,[2,4,5,8,9,1...
x0pSxAQx1K0abm	[hello, thi, is, ...	[oh, no, i, had, ...	1	(200,[3,7,9,15,17...	(200,[0,2,3,5,13,...	(200,[3,7,9,15,17...	(200,[0,2,3,5,13,...	(400,[3,7,9,15,17...
I8QHkmsq5AdwAq	[hello, thi, is, ...	[thi, is, john, s...	0	(200,[0,9,17,18,2...	(200,[0,5,9,17,20...	(200,[0,9,17,18,2...	(200,[0,5,9,17,20...	(400,[0,9,17,18,2...
v9vN0BYTWa0e16	[hello, thi, is, ...	[im, sorri, i, do...	1	(200,[7,9,12,17,1...	(200,[2,5,9,13,17...	(200,[7,9,12,17,1...	(200,[2,5,9,13,17...	(400,[7,9,12,17,1...
pVeYjXj2De1fFL	[hello, thi, is, ...	[speak, thi, is, ...	0	(200,[2,3,8,9,15,...	(200,[9,20,21,26,...	(200,[2,3,8,9,15,...	(200,[9,20,21,26,...	(400,[2,3,8,9,15,...
42e0AXIkESKc6y	[hello, is, thi, ...	[ye, who, is, thi...	1	(200,[2,5,9,17,20...	(200,[0,5,9,13,14...	(200,[2,5,9,17,20...	(200,[0,5,9,13,14...	(400,[2,5,9,17,20...
vk2TK8MFd0hoZg	[hello, thi, is, ...	[oh, my, is, ever...	0	(200,[0,5,9,15,17...	(200,[9,15,16,17,...	(200,[0,5,9,15,17...	(200,[9,15,16,17,...	(400,[0,5,9,15,17...

Fig 22: Example of Data Post-Processing Through the Model Training Preprocessing Pipeline

As illustrated in Figure 22, the "combined_features" sparse vector represents the final output that will be utilized for training the machine learning models.

6.3 Preprocessing Pipeline for Deployment

In addition to the pipeline developed for model training, a separate pipeline is established to manage data preprocessing in the system's actual deployment and usage. This deployment pipeline diverges from the training pipeline only in its initial step. In real-world scenarios, the data is already in a flattened format, rendering the flattening step redundant. Therefore, the data bypasses the flatten transformer. Subsequent steps of this pipeline remain consistent with those detailed in the preceding sections.

7. Model Architecture – Training – Tuning

Until now, this dissertation has detailed the generation and preprocessing of data, crucial steps to prepare them for utilization in machine learning models. We have meticulously navigated through the intricacies of data creation and the comprehensive preprocessing pipeline, ensuring that the data are optimally structured for both training and deployment phases. With these foundational aspects established, we now transition to a critical segment of this research: the exploration of the machine learning models themselves.

This section delves into the selection of models, dissecting their architectural frameworks and internal mechanisms. We will thoroughly examine the rationale behind choosing specific models, their design intricacies, and how they align with the unique challenges of our dataset. Additionally, this section will provide a detailed account of the training process, including strategies employed to optimize model performance.

Furthermore, we will explore the nuances of model tuning, discussing the techniques and methodologies implemented to fine-tune the models for enhanced performance and efficiency. The section will culminate in an evaluation of the models' performance during training, offering insights into their efficacy, strengths, and limitations. Through this comprehensive exploration, we aim to provide a clear understanding of the model lifecycle, from architectural design to performance evaluation, in the context of our specific machine learning endeavor.

7.1 Machine Learning Basics

Machine Learning is a transformative branch of artificial intelligence (AI) that focuses on the development and application of algorithms capable of learning from and making decisions or predictions based on data and patterns. Unlike traditional programming where rules and decision criteria are explicitly coded, machine learning algorithms build a model based on sample data, known as ‘training data’, to make predictions or decisions without being programmed to perform the relevant task. This ability to automatically learn and improve from experience makes machine learning a core technology in today’s data driven world [43].

There are 3 types of machine learning, each has its unique approach and application areas:

- **Supervised Learning:** In the realm of machine learning, Supervised Learning (SL) is a core approach. It involves training an algorithm using a dataset that comprises of both the input elements, often in the form of predictor variable vectors, and their corresponding desired outcomes, known as supervised labels. The training encompasses a set of observations $T = (xi, yi), i = 1, \dots, N$ where the inputs xi are processed by a learning

algorithm which subsequently produces outputs $f(xi)$. *T*he fundamental objective of supervised learning is to develop a function that can accurately predict expected output values for new, unseen data, a process requiring the model to extend its learned patterns from the training data to novel situations effectively. This extension, or generalization, of the model's learning from the training set to unfamiliar scenarios is a critical aspect. Throughout training, the supervised learning algorithm refines its input-output relationship, adapting in response to the error $yi - f(xi)$ between the actual and predicted outputs. This iterative adjustment, referred to as learning by example, aims to reduce the aforementioned error, thus honing the algorithm's predictive capabilities. Following the learning phase, the expectation is that the algorithm's output predictions $f(xi)$ will sufficiently match the true outputs for the entire range of potential input scenarios. This precision renders supervised learning particularly effective for tasks that require precise output prediction from given inputs, such as in classification and regression scenarios [44][45].

- **Unsupervised Learning:** Another core type is unsupervised learning; In this machine learning type, the learning phase involves the network attempting to replicate the input data it receives. This replication process is key to the network's self-improvement. The network uses the discrepancies between its generated outputs and the original input data to adjust itself, specifically by altering its weights and biases. This error, which drives the learning process, can manifest in various forms. In some instances, it's represented as a low probability, indicating the unlikelihood of the network's output being correct. Alternatively, this error might be depicted as a high-energy state within the network, signifying instability and deviation from the expected output [46]. In contrast to supervised learning there are no predefined labels or responses to guide the learning process, hence the network must rely on these internal error signals to refine its understanding and representation of the data. This approach underscores the exploratory and adaptive nature of unsupervised learning, where the network continuously evolves and adjusts its internal parameters in response to the intrinsic patterns and structures it discovers within the data.

- **Reinforcement Learning (RL):** Reinforcement learning, a distinct segment within machine learning, stands apart from the supervised and unsupervised learning approaches. This method involves an agent that learns by interacting with its environment: it makes decisions, performs actions, and receives feedback in terms of rewards or penalties. The central principle is the maximization of accumulated rewards over time. The agent must strike a balance between exploring new actions and exploiting known, rewarding ones. The learning cycle consists of the agent assessing the environment's state, acting accordingly, and then adapting based on the feedback and new state provided by the environment. This approach is aimed at continually enhancing the agent's strategy for optimal reward accumulation. Reinforcement learning, however, faces challenges like high computational demands and a significant need for data, particularly in scenarios where rewards are infrequent or delayed. This paradigm, centered around learning from trials, maximizing rewards, and managing the exploration-exploitation dichotomy, is fundamental to reinforcement learning [48].

In the context of our specific problem, supervised learning emerges as the optimal approach. Given that our task falls in the category of binary classification, a conversation is either a scam or it is not a scam, algorithms within the supervised learning paradigm are anticipated to be the most effective.

7.2 Pre-Modeling Considerations: Overfitting, Cross-Validation and Evaluation Metrics

Prior to delving into the specific models and algorithms chosen for this research, it is essential to understand several fundamental concepts.

7.2.1 Overfitting

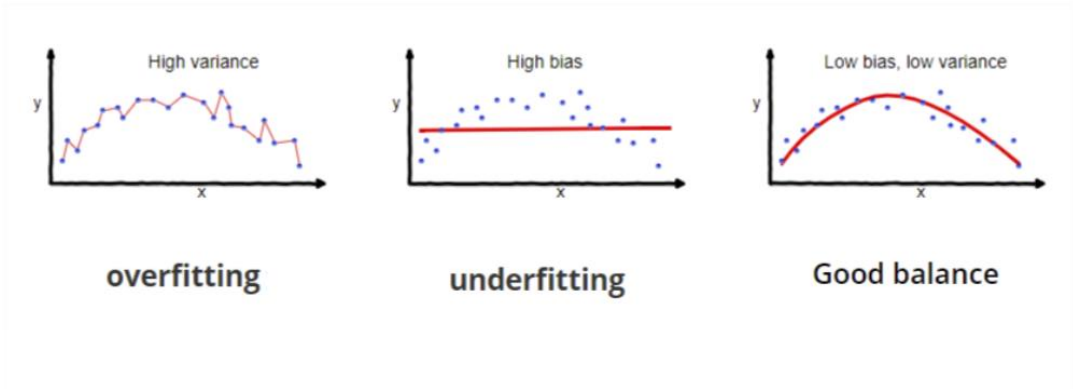


Fig 23: Example of Overfitting [53].

Overfitting predominantly occurs in supervised learning and is characterized by a model's inability to generalize effectively from the training data to new, unseen data. This phenomenon is typically indicated by perfect or near-perfect performance during training and validation, but a marked decrease in performance when the model encounters the test set. However, if the test set closely resembles the training data, for instance, if both are derived from the same dataset through a train-validation-test split, the model may still exhibit high performance on the test set. Overfitted models often memorize the data, including noise, instead of understanding and generalizing the underlying principles of the data [49].

To avoid overfitting in machine learning models, several strategies can be employed. Most notably:

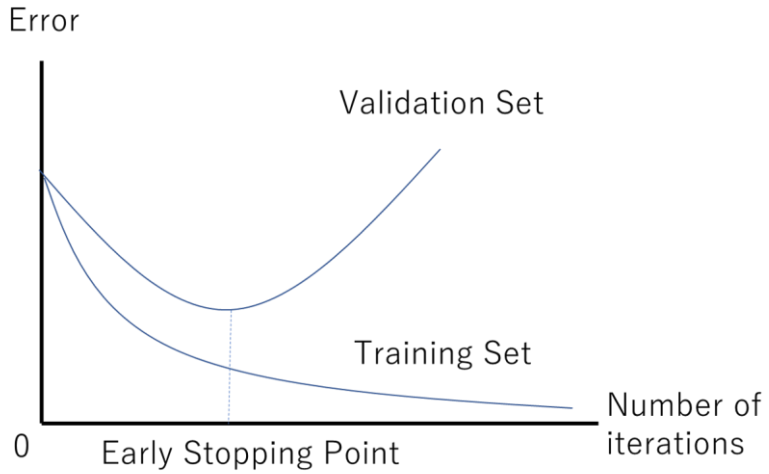


Fig 24: Example of Early Stopping Point [50]

- **Early Stopping:** This technique involves halting the training process when the model's performance on the validation set begins to decline. As illustrated in Figure 50, training ceases once the validation loss plateaus and then starts to increase. By implementing early stopping, we ensure that the model is retained in its optimal state of training. Continuing training beyond this point would likely lead to a deterioration in the model's ability to generalize to new data. Early Stopping is particularly useful in training neural networks where longer training times often lead to overfitting [49].

- **Data Augmentation:** This technique involves generating additional data by introducing variations to the existing dataset. It's achieved through modifications such as rotations, scaling, and other transformations in the case of image data, or synonym replacement and sentence restructuring for text data. Data augmentation not only helps in mitigating overfitting but is especially beneficial when dealing with small datasets that lack diversity. By expanding the dataset and introducing a wider range of examples, models can learn more robust and generalized patterns, enhancing their performance and reliability on unseen data [51][52].

- **Regularization:** This technique involves adding a penalty to the loss function to encourage simpler models. The three types of regularization are [53]:

    - **Lasso Regularization (L1)** plays a key role in inducing sparsity within a model by shrinking certain coefficients to zero, effectively aiding in feature selection. The mechanism of Lasso Regularization is analytically represented in the following equation:

$$\boldsymbol{Cost} = \frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}(\boldsymbol{yi} - \widehat{\boldsymbol{yi}})^{\mathbf{2}} + \boldsymbol{\lambda}\sum_{\boldsymbol{i=1}}^{\boldsymbol{m}}|\boldsymbol{wi}|$$

      Where the first term of the equation is the mean squared error (MSE) over n samples, and the second term is the regularization term, where λ is a non-negative hyperparameter that controls the strength of the regularization. The regularization term is the sum of the absolute values of the model coefficients wi, summed over m features.

    - **Ridge Regularization** (L2) minimizes coefficients but doesn't reduce them to zero, helping to manage multicollinearity and model complexity. The mechanism of Ridge Regularization is analytically represented in the following equation:

$$\boldsymbol{Cost} = \frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}(\boldsymbol{yi} - \widehat{\boldsymbol{yi}})^{\mathbf{2}} + \boldsymbol{\lambda}\sum_{\boldsymbol{i=1}}^{\boldsymbol{m}}\boldsymbol{wi}^{\mathbf{2}}$$

    - **Elastic Net** combines both the L1 and L2 approaches, offering a balanced solution for enhanced model generalization. The algorithm for Elastic Net is analytically represented in the following equation.

$$\boldsymbol{Cost} = \frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}(\boldsymbol{yi} - \widehat{\boldsymbol{yi}})^{\mathbf{2}} + \boldsymbol{\lambda}((\mathbf{1} - \boldsymbol{\alpha})\sum_{\boldsymbol{i=1}}^{\boldsymbol{m}}|\boldsymbol{wi}| + \sum_{\boldsymbol{i=1}}^{\boldsymbol{m}}\boldsymbol{wi}^{\mathbf{2}})$$

      In this regularization technique a new hyperparameter is utilized α which controls the ratio of the L1 and L2 regularization. When α approaches zero

then Elastic Net turns into L1 regularization and when α approaches one then it turns into L2 regularization.

- **Cross-Validation:** This method serves as an important instrument for evaluating the generalizability of statistical analysis results to independent datasets but also functions as a tool to avoid overfitting. The intricacies of this technique, given its critical relevance to our research, will be explored in the subsequent section.

7.2.2 Cross-Validation

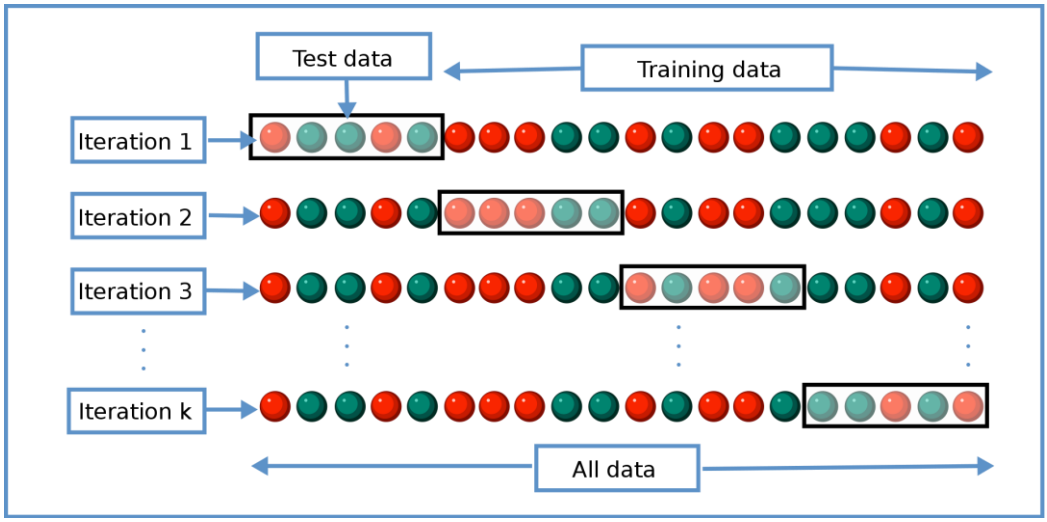


Fig 25: Diagram of K-fold cross-validation [54]

Cross-validation operates as a statistical method to estimate the proficiency of machine learning models, evaluating their capacity to generalize to independent datasets. Additionally, it plays a pivotal role in preventing overfitting, an aspect that becomes particularly essential when working with limited data. The prevalent technique of cross-validation is K-Fold Cross-Validation, which involves dividing the dataset into 'k' segments of equal size, known as 'folds.' The model undergoes training on 'k-1' of these folds and is evaluated on the remaining fold. This cycle is executed 'k' times, ensuring each fold is utilized once as the validation set, as illustrated in figure 25. The outcomes across all 'k' iterations are averaged to yield a comprehensive performance estimate. Typically, a 5-fold cross-validation is recommended for large datasets. However, for smaller datasets, such as in our study, increasing 'k' can yield a more reliable assessment. Consequently, we have selected 'k'=7 for our cross-validation process to align with the scale of our dataset [54].

7.2.3 Model Evaluation Metrics

As previously underscored, the evaluation of a model's performance is a critical element in the training process of machine learning models. Given that our task centers on binary classification, it is imperative to establish a foundational understanding related to the binary categorization of data, which will underpin all subsequent metrics.

- **True Positive Rate (TP):** The true positive rate is the count of data points correctly identified as belonging to the positive class.
- **True Negative Rate (TN):** The true negative rate is the count of data points correctly identified as belonging to the negative class.

- **False Positive Rate (FP):** The false positive rate is the count of data points incorrectly identified as belonging to the positive class when they actually belong to the negative class. In the context of this research this would mean a conversation is labelled as scam when in fact it is not.
- **False Negative Rate (FN):** The false negative rate is the count of data points incorrectly identified as belonging to the negative class when they actually belong to the positive class. This is of paramount importance to us since this would mean that a conversation is labelled as legitimate when in fact it is a scam.

The following section outlines the metrics that were employed to assess the performance of the models within the scope of this research.

Accuracy

Precision

Recall

F1score

7.3 Model Architecture

7.4 Model Training – Tuning

7.5 Model Validation Results

8. Overall System Functionality

9. Experimental Results

10. Deductions, Limitations and Future Research

Bibliography

[1] Inside the intricate world of voice phishing (joins.com)

[2] Voice phishing - Wikipedia

[3] What Is a Vishing Attack? | Fortinet

[4] What is Vishing (Voice Phishing)? Examples You Need to Know (softwarelab.org)

[5] What is dumpster diving? (techtarget.com)

[6] Song, J., Kim, H., & Gkelias, A. (2014). iVisher: Real-Time Detection of Caller ID Spoofing. *ETRI Journal*, *36*(5), 865-875.

[7] Norris, G., & Brookes, A. (2021). Personality, emotion and individual differences in response to online fraud. *Personality and Individual Differences*, *169*, 109847.

[8] Jones, K. S., Armstrong, M. E., Tornblad, M. K., & Siami Namin, A. (2021). How social engineers use persuasion principles during vishing attacks. *Information & Computer Security*, *29*(2), 314-331.

[9] What Is a Vishing Attack | Examples & Prevention | Imperva

[10] Xing, J., Yu, M., Wang, S., Zhang, Y., & Ding, Y. (2020). Automated fraudulent phone call recognition through deep learning. *Wireless Communications and Mobile Computing*, *2020*, 1-9.

[11] Lee, M., & Park, E. (2023). Real-time Korean voice phishing detection based on machine learning approaches. *Journal of Ambient Intelligence and Humanized Computing*, *14*(7), 8173-8184.

[12] Tran, M. H., Hoai, T. H. L., & Choo, H. (2020). A third-party intelligent system for preventing call phishing and message scams. In *Future Data and Security Engineering. Big Data, Security and Privacy, Smart City and Industry 4.0 Applications: 7th International Conference, FDSE 2020, Quy Nhon, Vietnam, November 25–27, 2020, Proceedings 7* (pp. 486-492). Springer Singapore.

[13] Salloum, S., Dautov, R., Chen, X., Peng, P. X., & Huang, J. Z. (2016). Big data analytics on Apache Spark. *International Journal of Data Science and Analytics*, *1*, 145-164.

[14] Assefi, M., Behravesh, E., Liu, G., & Tafti, A. P. (2017, December). Big data machine learning using apache spark MLlib. In *2017 ieee international conference on big data (big data)* (pp. 3492-3498). IEEE.

[15] Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K., ... & Zaharia, M. (2015, May). Spark sql: Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data* (pp. 1383-1394).

[16] The Spark cluster architecture for resource allocation and data transfer | Download Scientific Diagram (researchgate.net)

[17] Apache Spark Components (Updated) (intellipaat.com)

[18] Key features of Spark's MLlib: aspark.mllib is built on top of RDDs,... | Download Scientific Diagram (researchgate.net)

[19] Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., ... & Talwalkar, A. (2016). Mllib: Machine learning in apache spark. *The journal of machine learning research*, *17*(1), 1235-1241.

[20] Optimization - RDD-based API - Spark 3.5.0 Documentation (apache.org)

[21] Broyden–Fletcher–Goldfarb–Shanno algorithm - Wikipedia

[22] ML Pipelines - Spark 3.5.0 Documentation (apache.org)

[23] Spark ML pipeline for regression | gbhat.com

[24] Spark Streaming - Spark 3.5.0 Documentation (apache.org)

[25] What is Speech to Text? - Speech to Text Explained - AWS (amazon.com)

[26] Trivedi, A., Pant, N., Shah, P., Sonik, S., & Agrawal, S. (2018). Speech to text and text to speech recognition systems-Areview. *IOSR J. Comput. Eng*, *20*(2), 36-43.

[27] Speech-to-Text: Automatic Speech Recognition | Google Cloud

[28] Generative pre-trained transformer - Wikipedia

[29] Transformer (machine learning model) - Wikipedia

[30] What is a Transformer?. An Introduction to Transformers and… | by Maxime | Inside Machine learning | Medium

[31] Prompt engineering - Wikipedia

[32] Master the Perfect ChatGPT Prompt Formula (in just 8 minutes)! (youtube.com)

[33] What is Tokenization? Types, Use Cases, Implementation | DataCamp

[34] Tokenization in Natural Language Processing | by Mohammad Derakhshan | Medium

[35] Perkins, J. (2010). *Python text processing with NLTK 2.0 cookbook*. PACKT publishing.

[36] NLTK :: nltk.tokenize.word_tokenize

[37] Introduction to Stemming - GeeksforGeeks

[38] All you need to know about text preprocessing for NLP and Machine Learning - KDnuggets

[39] Stemming - Wikipedia

[40] pyspark.sql.functions.flatten — PySpark 3.5.0 documentation (apache.org)

[41] tf–idf - Wikipedia

[42] Feature hashing - Wikipedia

[43] Machine learning - Wikipedia

[44] Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Tibshirani, R., & Friedman, J. (2009). Overview of supervised learning. *The elements of statistical learning: Data mining, inference, and prediction*, 9-41.

[45] Supervised learning - Wikipedia

[46] Unsupervised learning - Wikipedia

[47] James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). Unsupervised learning. In *An Introduction to Statistical Learning: with Applications in Python* (pp. 503-556). Cham: Springer International Publishing.

[48] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *nature*, *518*(7540), 529-533.

[49] Ying, X. (2019, February). An overview of overfitting and its solutions. In *Journal of physics: Conference series* (Vol. 1168, p. 022022). IOP Publishing.

[50] overfitting - Search Images (bing.com)

[51] Shorten, C., Khoshgoftaar, T. M., & Furht, B. (2021). Text data augmentation for deep learning. *Journal of big Data*, *8*, 1-34.

[52] Data augmentation - Wikipedia

[53] Regularization in Machine Learning - GeeksforGeeks

[54] Cross-validation (statistics) - Wikipedia

[55] Binary classification - Wikipedia