# CATHOLIC UNIVERSITY OF BUKAVU


B.P.285 Bukavu

Faculty of sciences

Department of Computer Science


---

# Development of an interface application for detection of spam on a mobile operator: Case study of Airtel, Vodacom and Orange.

---


Presented by : MURHULA BYABUSHI Christian


*Dissertation presented and defended in order to obtain the degree of Bachelor in Computer Science.*

Option: Network and Telecommunications
Degree: Final year of Bachelor


Supervised by: Hw. MUGISHO MUSHEGERHA Youen
Directed by : PhD. Elie ZIHINDULA


## Academic year: 2022-2023

# Epigraph

# In memoriam

# Dedicated

# Acknowledgments

# Contents

# List of Figures

# List of Tables

# Introduction

## 0.1 Context and generalities

With the increasing of use of mobile devices in mobile telecommunication, the number of text messages sent every day has grown exponentially. According to *Statista*, a company that provides market and consumer data on a wide range of topics, including digital media and technology; the number of mobile messages sent worldwide in 2020 reached 3.5 trillion [62].

In the same case, with the raise of web pages and social media messaging applications like Whatshap, Telgram, Snapchat Facebook, Instagram and many others, phone users can now send messages that are only based on text as in former time but also on video, audios which are more chestful comparatively [23].

For sure, for interacting with his partner more professionally, email message is the most mean used, but not in all cases, since in some countries a people commonly use *Sim card* delivered by a telecommunication provider to launch mobile bank services. Hence, it appears more valuable than only being a communication technology.

Along with various services and interests that mobiles messages encompasses, from mobile baking to social media communications, games entertainment, localization platforms and many others the number of messages sent have been increase. However these includes the genre that aim to deceive people into providing personal information, fakes news, sending unwillingly money, menacing to death or taking other actions that benefit the scammer.

To address this problem, the development of an interface application with spam detection capabilities appears crucial, since it allows on a side:The setting and filtering of messages in applications, and the delivering to the phone user an assistance helping him to take better decisions. On the other side, providing to networks operators a further solution worthy to be encompassed into the messages transmitters systems for more transactions monitoring.

## 0.2 Problematic

In telecommunication area, it is used mobile devices or phones for sharing either texts or emails and chats by using targeted applications. Among all, it is used specifically *SMS* for sharing personal and professional information [44]. The *SMS* stands for Short Message Service, which is a text messaging service for mobile phones. At the beginning it was allowing users to send messages least than 16O characters [45]. However in to-

day world, it is possible to send more than that even multimedia types. Their sources can be either human or other phones users, online applications or networks operators [32].

From time to time, a certain group of people benefit from this accessibility, liberality and send messages which are insane or completely unwilling by receivers. Their genre looks like this: "You won x amount of money send another amount to withdraw it", "Join me at x area to take your money but pay me the transport ...", "I'm *Sirene* Madam I have money for you", "I have a job for you" and many others fake messages. More of them are reported in this spams examples link.

Furthermore, other kinds of scammers even arrive at stage where they utilize the vulnerabilities which are present in messaging system, and inject, or even install mobile spyware users systems. For instance we have the *SimJacker* referenced by [15]. This shows that the threats can even come not only from messages content but from even its system.

Speaking about unwilling messages involves to investigate too, in the common organizations systems which afford their market plan or advertisement by paying networks operators. Therefore, scammers use that mean to appears too as authorized organization providing to users different services. Hence, tacking an official and nonofficial becomes challenging[14],[47].

Considering all issues and challenges mentioned above in which spams messages are implicated, it becomes urgent to find solution(s) helping or supporting all victims. But, how can these challenges including false market advertising messages to threats and unwilling messages, be tackled for facilitating a proper messaging communication ?

## 0.3 Hypotheses

According to the *Oxford* dictionary, hypotheses stands for a statement of the expected relationship between things being studied, which is intended to explain certain facts or observations [63]. An idea to be tested. Hence, using the content-based filtering techniques, which involves analyzing the content of messages and determining whether it is a spam or not would be considered as solution.

Actually, the solution at the mentioned issue involves the entire participation of victims and the systems, in other words human and human participation. Hence, it seems effective to use new technologies that can give capacity to systems to detect by themselves the threats and warns the users. This is possible by the modern techniques called : Machine learning algorithms.

For the benefit of this work, it is utilized several types of those new technologies encompassing algorithms which are: **Naive Bayes, Logistic Regression, and Supper vector Machines** helping to assistant the users. Since they have different performances, this work combines their capacities the other techniques called : Ensemble models allowing to get one result which can help the phone to interact with the user [69]. In their logic, they include functions allowing them to classify after being prepared and trained whether a message is legitimate or not [39].

Furthermore, the solution involves other techniques means helping the phone system to interact with the others online services providing to it a support internally so that it may know how to filter its inputs, whether they contain the threats or not. The third party contacted gives details about the message content and the **probability** to trust it.

Overall, tackling those issues leads to build a smart system which assists and provides the user with informations that can increase his capacity of judging the. That should start from the development of models which feeds data to its deployment in softwares ares allowing him to consume the model's services [28].

## 0.4 Delimitation and objectives

### 0.4.1 Delimitation

The present work aims to develop a messaging application for communication and detection of spam on a mobile network.

Geographically it focuses on all provinces of Democratic Republic Of Congo(DRC) where mobile phones are used and require techniques for implementation.

The solution provided is limited in terms of landscapes where it can be utilized since it's smartly function only on topics(data) on which it is concerned. As a result, it is challenging to claim its effectiveness only in languages like Swahili, French, and English.

Indeed, the current work in terms of planning and execution has spanned a duration of nine months : From January to November 2023.

### 0.4.2 Objectives

The current work compromises 2 types of objectives which are: functional and non-functional.

As functional objectives, it yields a interface application system that can interact with messaging applications and facilitates efficient communication between users; by implementing the machine learning models which having the capacity of classifying messages and preventing unwilling messages under a certain probability.

Concerning non-functional objectives, the current work allows users whose messaging applications complies with relevant privacy laws and regulation, to protect data information, reducing the attacks and frauds; increasing the trustfulness of other users. Consequently, it increases the credibility in networks operators companies services that gain another tool helpful in surveillance and monitoring.

## 0.5 Interest

Personally, the current work has allowed the authors to gain knowledge and more experience in the field of mobile networks and messaging applications trough the research

investigated in.

In the society, it is worthy to contribute to facilitating communication and reducing the impact of spam messages, which are somehow annoying and stressful for citizens.

Economically, it is appears as time saver in business employees by decreasing concerns based on threats and unwilling messages due to its capacity of providing filtering functions. For scientists, this work is a reference for those whose to delve in and gaining skills and techniques in mobile networks environment in machine learning operations systems.

## 0.6 Research Methodology

Throughout this paper, the research methodology is used to guide the study towards achieving its objectives. It adopts a descriptive research design to describe the development of a interface application for better communication and detection of spam on a mobile network. The study will focus on both qualitative and quantitative research methods [17]. The qualitative method involves a focus on literature review, interviews and analysis of collected messages, while the quantitative method focuses on development and testing of the interface messaging application.

The current research has been conducted in two phases. The first one involves data collection through a survey questionnaire form that is completed on website. The others involves the downloading of data from online working community. Once collected, they are analyzed using descriptive statistics [7] including the techniques of identifying the common types of messages and the frequency of occurrence and languages within it, etc.

The second phase involves the development of the pursuing of steps offered by the machine learning operating systems (MLOps) encompassing data prepocessing, wrangling, feature engineering, splitting for model training till it deployment for being integrated in a system. By following the workflows stages delivered by the MlOps for production models, it covers the architecture of consumming machine learning services trough a software system compromises the programming technologies like Python with its framework Django and HTML, Css, and JavaScript for interaction with it.
The evaluation of the messaging application interface is conducted using both quantitative and qualitative methods. In fact, the qualitative evaluation involves the measurement of the application's accuracy and efficiency for detection and filtering messages while the qualitative evaluates user stability and application's experience.

## 0.7 Work Plan (or Work Subdivision)

The work plan of this dissertation is divided into four parts. The first is the introduction, which provides a background information on the research problem.The second part consist of a situation analysis and assessment while the third part focuses on literature review and explanations on the methodology. Then the fourth part presents the practical result of this work. Finally the conclusion part summarizes the key findings and contributions of the study and presents limitations and provide recommendations for future research.

# Chapter 1

# Situation analysis and assessment on mobile phones

## 1.1 Introduction

In this chapter, we will focus on various aspects that enhance the comprehensiveness and practicality of this dissertation. It includes explanations of mobile messaging architecture, machine learning models, and spam messages in mobile world. Additionally, it provides an analysis of the architecture used by network operators, highlighting both positive and negative aspects of their approach to message handling.

## 1.2 Presentation of the working framework and definition of key concepts

### 1.2.1 Definition of key concepts

a) SMS(Short Message Service) :
The Short Message Service is a basic service allowing the exchange of short text messages between subscribers [46]. For supporting virtually all mobile devices, SMS is considered as a universal means of communication that enables users to communicate and function even though all users are not active simultaneously (asynchronous communication).

b) Enhanced Messaging Service (EMS) :
EMS has been created to allow the transmission of richer and more advanced messages. Unlike traditional SMS, EMS accepts not only text messages but also audios, melodies, and animations [45].

c) MMS (Multimedia Messaging Service):
MSS has been developed to facilitate the transmission of rich multimedia content in mobile messaging. Unlike SMS and EMS, MMS enables users to send not only text messages but also various types of multimedia files such as images, videos, audio recordings and even slideshows [45].

d) Spam message:
A spam message is understood as an unsolicited or undesired messages received on mobile phones which constitutes veritable nuisance to the mobile subscribers [72]. Clearly, this message can be sent with the intention of gaining financial benefits, collecting personal or organizational information such as security numbers, credit card details, or login credentials, and soliciting money by making false promises of future benefits or rewards that do not materialize.

e) Networks operators: The networks operators refers to companies or organizations that provide and manage telecommunication networks. These operators own and operate the infrastructure, such as mobile networks, fixed-line networks, or internet service provider (ISP) networks, that enable the transmission of user's information to another user of the network [26]

f) Artificial Intelligence (AI) : AI refers to the field of computer science that focuses on creating intelligent machines or systems that can perform tasks that would typically require human intelligence. For being practical, it encompasses algorithms, models and technologies that enable computers and machines to simulate human like cognitive processes such as learning, reasoning, problem-solving, perception and language understanding.

g) ML (Machine Learning) : Machine learning is a subfield of Artificial Intelligence that focus on the development of algorithms and models that enable computers to learn from data and make decisions or predictions without being explicitly programmed[77]. Clearly, when the data is labeled during the training, we refer to it as supervised model. If contrast, when the data is unlabeled and the model must discover patterns and relationships itself, it is an unsupervised model. Additionally, whenever it performs both the labeling and discovering patterns tasks, it refers to a semi-supervised model. Furthermore, there is the last type called reinforcement. This one, is used to teach a computer or an AI agent how to make series of decisions in an environment. Just like, we learn to play the game better by playing it over and over.

h) NLP (Natural Language Process): NLP is a subfield of Machine Learning that studies the human language and combing techniques from statistics, linguistics, life-hoods for making sentiment analysis, text classification, machine translation, question answering and text generation in a way that it can be understood computationally [11].

### 1.2.2 Presentation of the working framework

In the eastern party of DRC (South Kivu- and North Kivu) the usage of mobile phones has become more common, transforming communication and connectivity in the region. The DRC itself is a large country, covering over 2,345,000 square kilometers with the eastern provinces of North and South Kivu spanning approximately 59,483 and 65,070 square kilometers respectively [91]. According to recent statistics from *GlobalEdge* [1], an American company, around 95 million people were living in the DRC in 2022 [21], of

[1]GlobalEdge : Created in 1994 by the International Business Center and the Eli Broad College of Business at Michigan State University (IBC), globalEDGE™ is a knowledge web-portal that connects international business professionals worldwide to a wealth of information, insights, and learning resources on global business activities

which approximately 46.9% had active mobile phones based on *GSM* [2] research.

In this context, it is observed that more people in cities use mobile phones compared to those in villages, primarily due to limited accessibility. A research study conducted by *Target Canibet* [3]in 2015 focused on mobile connections in DRC cities including Bukavu, Goma, Kinshasa, Lubumbashi, and Matadi, found that among 1,000 people surveyed in each city, 9 out of 10 individuals were subscribed to a network operator. However, it was noted that approximately half of them subscribed to two operators, while a quarter subscribed to four operators, and 18% used the services of a single operator.

Furthermore, the recent statistics made by *DataReportal* [4] in DRC shows that the mobiles users continues to increase exponentially, merely because of new services provided by internet and Telecoms Operators, at the point that since 2021 to 2022, it is has been reported 3.6 million of new users between 2021 to 2022, a report that proves how much mobile phones is inevitable in this last decades.

### 1.2.3 Network coverage and infrastructure

In fact, two telecoms services exist in DRC such as : Fixed services (26%) and Mobile services (74%). The first one known as landline or wired services, involve the use of physical infrastructure; the second one which is popular is the mobiles services refer to telecommunications services provided through mobile networks. According to the Congolese Regulatory Agency (ARPTC), the DRC has four mobile operators - Vodacom RDC, Airtel Congo, Orange DRC and Africell DRC. Vodacom is the leader in the voice segment, with 35.2% of the market, followed by Orange (30%), Airtel (23.9%) and Africell (10.9%). In the mobile internet market, Vodacom has 37.44%, Airtel 31.25%, Orange 28.14% and Africell 3.17% [5].

Additionally, since the 190s, when the DRC witnessed the first installation of operator systems such as Celtel(now Airtel) and Vodacom, followed by Orange and Africell, the telecomunications sector has shown significant market growth, reaching 1 Billion in 2022$ and expanding at a rate of 21% per year according to *GlobalData* [5].

However, this growth necessitates the updating of the infrastructure, which includes various generations of technologies, namely the second generation, third, fourth, and fifth(under development).

In fact, the second generation have been deployed in various territories to enable more efficient **voice calls, data networks services, and introduce SMS for text messaging**. The infrastructure required for 2G networks includes the following equipments: 1)

[2]GSMA (Global System Communications Association) : An industry Organization which represents the interests of mobile network operators worldwide created in 1982 to ease cooperation between countries deploying *GSM* (Global System fo Mobile) technology.

[3]Target Canibet: Reseach & Consulting Group working in DRC. `https://www.target-sarl.cd/fr/content/etude-sur-la-telephonie-mobile-en-rdc`

[4]DataReportal: A online Company designed to help people and organizations all over the world to find the data, insights, and trends they need to make better informed decisions produced by Simon Kemp, `https://datareportal.com/reports/digital-2023-global-overview-report`

[5]GlobalData: Expert Company of Analysis, innovatove Solutions

BTS(Base Transceiver Station) : Transmit and receives signals between mobile devices. 2) MSC(Mobile Switch Controller) : serves as the switching entity that connects calls between mobile devices. 3) BSC (Base Station Controller) : manages multiples BTSs and controlling radio resources, managing handovers between cells and optimizing network performance, 4) AuC (Authentication Center) responsible for managing subscriber authentication and encryption keys to ensure communication between mobile devices and the network, 5) Home Location Register (HLR) the database that stores subscriber information such as phone numbers, authentication details, and service profiles, 6) Visitor Location Register (VLR) : The VlR is a temporary database that stores information about roaming subscribers within a specific area 7) MS (Mobile Station), including all the technologies used by the users's handset and has two parts : **Firstly, the mobile equipment which contains the radio equipment, the user interface, the processing capability and memory requirements for call signaling, encryption, SMS and the id of the mobile phone(equipment IMEI number). Secondly the Subscriber Identity module (SIM Card),** used in encryption of codes needed to identify the subscriber, storing subscriber's information, locate the user [18] as (+243 for each congolese number).
Indeed, all the 2G technologies covers a large distance varying between 1880MHz - 2700 MHz.

Besides, the third generation appears as revolution, **allowing multimedia messages, voice calls data, faster data speed**; however it requires a significant upgrade from the previous generation. Thus, the equipment involved in 3G technology includes : 1) BTS (Base station Transceiver) : Which plays the same role as for 2G; 2) Node B: Responsible for handling the radio interface and connecting mobile devices to the core network; 3) Radio Network Controller (RNC) : Controlling the Node B and managing the radio resources 4) Mobile Switching Center (MSC): The MSC is the central switching entity in the network that **connects calls between mobile devices**; 5) Serving GPRS Support Node (SGSN) : Responsible for managing packet-switched data services services and handling mobility for mobile internet access; 6) Gateway GPRS Support Node (GGSN): It serves as interface between the mobile network and external networks like internet; 7) The Home Location Register (HLR) and Authentication Center(AuC): plays the same role as in 2G; 8) Operations Support System (OSS) : It provides and functionalities for monitoring and managing the 3G network. Indeed, the 3G is appreciated for enabling higher- speed services and covers different frequency bands depending on countries, ranging between 850Mhz - 1700 Mhz [57].

Additionally, the fourth generation, commonly referred to as LTE(Long-Term Evolution) represents a significant advancement over previous generations in terms of infrastructure and services. This generation introduces higher data speeds, improved capacity, and better perfomance for mobile communication and data services. The upgrades in infrastructure include: 1) BTS and MSCs : These components remain unchanged from the previous generation 2) Evolved Packet Core (EPC) The EPC is a critical component of the 4G core network architecture which provides the packet-switched backbone that handles data traffic and ensures efficient data delivery between mobile devices and the internet or other networks; 3) Radio Access Network (RAN): The Ran is responsible for the radio interface between mobile devices and base stations; 4) LTE (Long-Term Evolution) : is the primary air interface enabling the high data speeds, low latency; 5) Back-haul Network

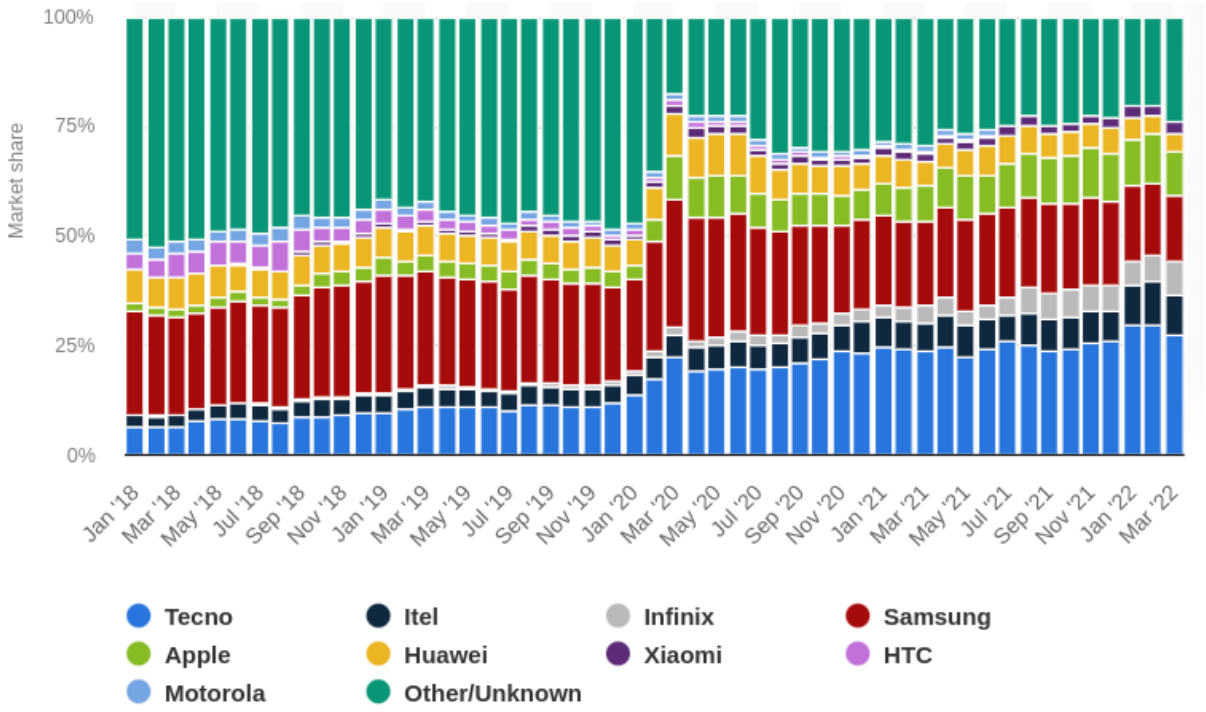



**Figure 1.1:** Market share of mobile device vendors in the Democratic Republic of the Congo from January 2018 to March 2022

: It connects base stations to the core network and internet infrastructure; 6) Spectrum Allocation: Hands over the mobile operator access to specific radio frequency bands; 7) Network Management System: These systems monitor and manage the 4G network, ensuring its smooth operation, performance optimization, and troubleshooting. However, it's spanning or coverage of 4G networks on frequency bands allowed in each country based on their preferences, ranging from 700MHz to 2600 Mhz. The higher frequency bands generally offer faster data speeds but may have a shorter range, while the lower frequency bands can provide broader coverage but with slightly lower data speeds [57].

### 1.2.4 Mobile Phone Models

Since the mobiles phones are essential tools for communication, there is a wide range of popular mobile widely used by citizens of the DRC. The popular models come from various brands and offer a range of features to cater to different user preferences and needs. Some of the popular mobile phone models in DRC include *Techno, Itel, Infinix, Samsung, Apple, Huwaei, Itel, HTC, Motorola*. As it can be seen on the figure 1.1, according to the recent statistics conducted by *Statistica*, Samsung was the market leader in terms of share from January 2018 to November 2020, but in 2022, Tecno has emerged as the market leader.

Furthermore, all these models provided above sell the telephone following different types which include mobile phones, offering the features such as touchscreen displays, cameras, internet, connectivity, and access to mobile apps; features phones, which are basic mobile phones used for calling and texting; smartphones used by the majority (around 35% in DRC ), providing access to mobile internet, mobile apps, multimedia messaging, and various productivity tools; Tablets, used for reading and for the same functionalities as

smartphones.

### 1.2.5 Mobile usage and prevalence

In fact, each phone has its own unique set of characteristics that define its capacity and performance compared to others. Some phones come with specific applications that can be used independently, even without being connected to an operator, such as a camera, calculator, games app, and many others. However, other phones may not have such features.

To access the services provided by the operator, the phone's sim-card must be functioning, recognized by the operator, and capable of sending and receiving communication signals. **Of course, all these services work only if the phone's battery has sufficient power**.

Moreover, the services that citizen's subscribers benefit from are as follows :
Firstly, the Internet Access: The Internet services are used to connect people from different nodes. In fact, In accordance with the *WorldBank* [6] been used by 23% of the DRC's population in 2020. Nonetheless, it requires payment which proportionally gives mobile data usually expressed in Megabytes.

Secondly, the Text Messaging (SMS): Even though the internet is the most used for texting, the SMS remains a widely used form of communication, especially in regions with **limited internet connectivity or among users who prefer simple text messaging or do not have the internet connection**.Furthermore, with the architecture of GSM(Global System for Mobile Communications) invented in the second generation, sending messages became possible. Nowadays, the web environment has developed application interfaces (API) that connect external systems to operators for sending messages [30]. One of the platforms that offer these services connects its SMS gateway to the GSM operator, as seen in the case of *Octopush* [7] architecture shown in Figure 1.2

Thirdly, the Mobile Banking and Payments : With this services subscriber can make **financial transactions**; paying bills, transferring money conveniently.

Fourthly, the Mobile Entertainment: Mobile phones offer a range of entertainment options, from streaming videos and music to mobile gaming.
And finally, the others Mobiles apps: This party includes the health services, education, social medias applications.

## 1.3 Purposes of spammers in mobile messages

Most of the time, spammers prefer to promise recipients prizes and then ask for money to claim the offer. They also attack SMS gateways with *DoS* (Denial of Service) messages [4] whose goal is to overwhelm the system with unnecessary messages. Spammers send advertising and promotional messages based on company objectives, as well as SMS containing fake links or impersonating organizations to deceive recipients into taking certain actions

[6]WorldBank : International Telecommunication Union ( ITU ) World Telecommunication/ICT Indicators Database https://data.worldbank.org/indicator

[7]Octopush : SMS platform for businesses connected with their audience

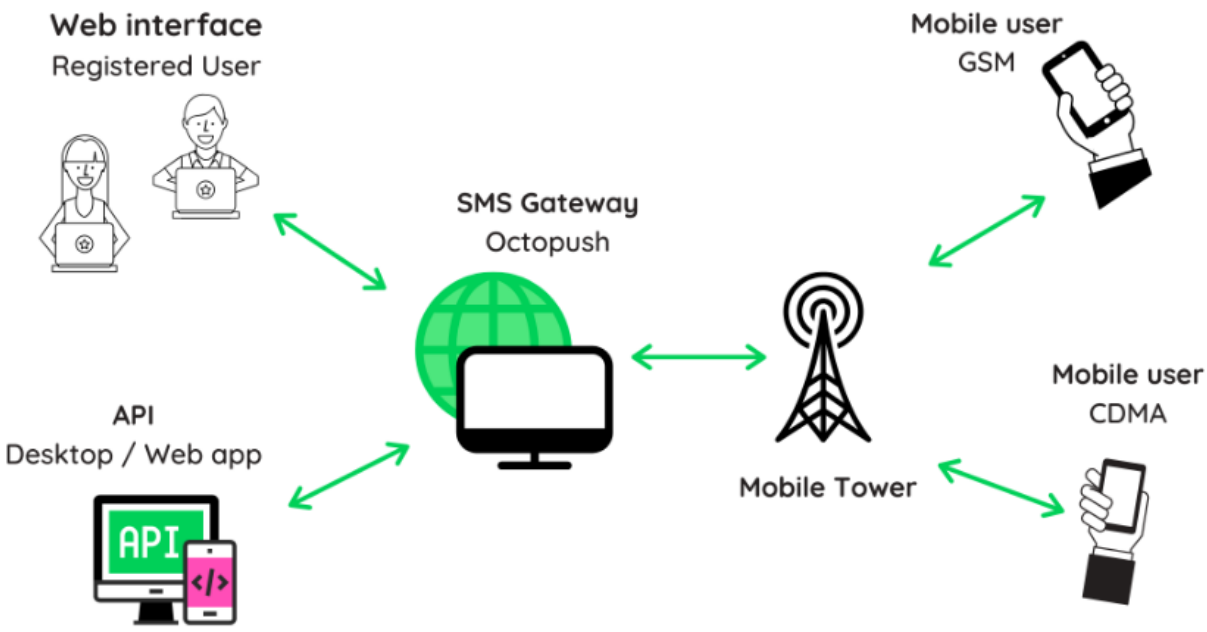



**Figure 1.2:** SMS Gateway Provider Architecture

or providing sensitive information [80]. Additionally, they may send SMS disguised as surveys to gather personal information for various fraudulent purposes.

## 1.4 Solutions

To address theses problems, it is necessary to involve various stakeholders, including network operators, app developers, regulatory bodies, and users.

Firstly, it is recommended to implement mechanisms at the network level [29] to filter messages and block users involved in spamming.

Secondly, users (subscribers) should be educated on how to analyze messages and report any one that is causing disturbances.

Thirdly, regulatory measures should be enforced to establish stringent regulations and penalties for spammers and those engaged in fraudulent mobile activities. Fourthly, the development of apps that enable filtering, classification, and reporting in the subscriber side would be beneficial.

Fifthly, a website can be set to collect messages whether spam or ham reported by users who have doubts about their legitimacy, and then Machine Learning models can be used for detection purposes. For this, supervised or unsupervised methods can be employed to classify and predicting whether a message is spam or ham.

## 1.5 Summary

Overall, with the growth of mobile technologies, subscribers benefit from diversified services including : text messaging, voice calls, mobile baking, entertainment apps, and many others. However, These advancements also bring new challenges, such as the development of spam messages that aim to disturb network subscribers with unwanted or threatening messages.

In DRC, especially in the eastern party, users face similar issues. This chapter emphasizes methods or techniques that can be used to address this problem and relatively reduce spam. One of the prominent techniques suggested is based on Artificial Intelligence, particularly Machine Learning algorithms.

# Chapter 2

# Review of the Literature and description of the approach

## 2.1 Introduction

This chapter delves into theory, methodologies, and machine learning techniques, including relevant algorithms and their deployment in the suggested solution. It also highlights the contributions of previous researchers in the field.

## 2.2 Revue of the Literature

Numerous researchers have extensively explored the subject of spam detection. Within this domain, some have directed their investigations towards the web environment, while others have delved into realm of mobile technologies.

Furthermore, these researchers have chosen to investigate the detection of spam across various communication channels, including emails and SMS, encompassing Multimedia SMS (MSS) as well. In the following sections, we comprehensively review the body of work that has been accomplished within this context as follows:

[40]. **Dr.V.M Veena K.Katankar** proposed a system that comprises an SMS gateway for transferring SMS messages after they have been stored and encrypted by the web server. This software operates through a web interface. Whenever a client sends a *POST* request, it is received by the web server, which is responsible for encryption or decryption if necessary. Subsequently, the gateway transfers the message as per its designated route. This solution proves to be particularly valuable in mobile banking and organizational marketing systems. Nevertheless, the author encourages other researchers to delve into channel services in communication and advanced encryption techniques.

[10] In their publication titled ***Short Message Service***, **Brown, Jeff and Shipman** members of IEEE, delve into several significant aspects. They start by exploring th growth of mobile phones and SMS services. They also examine the system architecture of SMS Centers and technologies used for message communication
Furthermore, they shine the spotlight on aggregators and services providers. These are the entities that enable users to send bulk messages, essentially sending messages with a

large amount of text to a group of recipients. This includes the interesting capability of converting email messages into SMS.
Moreover, the article highlights that some of these aggregators may choose to collaborate with cellular networks. In this collaborative role, they act as intermediaries, connecting third-party entities that don't have direct relationships with cellular service providers. To achieve this, they employ a *SMPP (Simple Messaging Peer to Peer)* protocols.

[56] **Researchers A. Medani and A. Gani**, affiliated with the University of Malaysia, have published a comprehensive review focusing on security concerns and techniques related to mobile Short Message Service (SMS).

In their paper, they illuminate the process by wich a subscriber sends a message to another party while adhering to specific principles of the Over-The-Air (OTA) structure. This process involves transmitting the message from the sender to the base station and then forwarding it to the intended recipient trough the SMS Center (SMSC).

Crucially, they emphasize the importance of securing every SMS using *Public Key Infrastructure (PKI)*, ensuring end-to-end transmission security and safeguarding the message from unauthorized modifications. However, it's worth noting that the use of PKI can potentially impact mobile device performance due to the significant power requirements for the encryption process, and it may not guarantee integrity across all standards.

To address these security concerns within GSM systems, the researchers propose the implementation of *XML Key Management Specification* as a middleware solution. This middle ware system servers an intermediary, facilitating secure communication between mobile devices and enhancing overall system security for the benefit of clients.

[16] **Nikhil Kumar**, a reseacher affiliated with the University of New Delphi in India, published an article focusing on the topic of Email Spam Detection Using Machine Learning. In his study, he placed particular emphasis on comparing various machine algorithms, including ***Naive Bayes, Support Vector Classifier, AdaBoosting, K-Nearest Neighbour, and Bagging Classifier***. The objective was to predict wether an email was categorized as spam or legitimate (ham). To demonstrate his approach, he utilized an existing dataset available in the Kaggle workspace.
Through experimentation and parameter tuning, Nikhil found that Naive Bayes delivered promising results in terms of accuracy. However, he also pointed out a limitation associated with the Naive Bayes algorithm. This limitation is tied to its assumption of class-conditional independence, which implies that each feature is considered independent of the presence of the other features. In cases where this assumption does not hold, it can lead to misclassification of data points.
To address this limitation and enhance the performance of spam detection, the author recommended the use of **ensemble methods**. These methods involve the use of multiple classifiers for making class predictions, allowing for more robust and accurate results.

[59] In 2018, researchers Pavas Navaney, Gaurav Dubey, and Ajax Rana, who are affiliated with the University of Southern California and Amity presented a conference paper titled "SMS Spam Filtering using Supervised Machine Learning Algorithms".
Their study concentrated on a dataset comprising 5574 records, of which 4827 messages

were categorized as "ham" (legimate messages), while 747 messages were classified as "spam" (unsollicited or unwanted messages).
The researchers applied three different machine learning methods to this dataset. Among these methods, it was observed that the *Support Vector Machine (SVM)* algorithm achieved the highest accuracy compared to the Naive Bayes and Maximum Entropy Classifier algorithms.

[73] **Houshmand Shirani-Mehr**, a researcher in Machine Learning, published an article in 2013 titled : "SMS Spam Detection using Machine Learning Approach". His purpose was to address the spam filtering problem by utilizing ML algorithms. Therefore, He utilized a dataset from the *UCI Machine Learning Repository* repository [1] , which contained real SMS messages. In development, he employed the algorithms to tackle that problem such as : Naive Bayes with Laplace smoothing, Support Vector Machine, and Ensemble methods (*AdaBoosting and Random Forest*). As an improvement, the author added meaningful features such as the length of messages in terms of the number of characters and certain thresholds.
The results obtained after applying these methods to the dataset indicate that the SVM algorithm achieved the highest accuracy score.

[27] The authors of the article titled "SMS Spam Detection Using Machine Learning", namely **Gupta, Suparna Das and Saha, Soumyabrata and Das, Suman Kumar**. Their focus was on reviewing various techniques employed by other researchers in the realm of machine algorithms for SMS spam detection.

In their research, they adopted a similar approach by incorporating the *TF-IDF (Term Frenquency-Inverse Document Frequency)* method. This technique assesses the frequency of a word within a document and evaluates its importance in that document. *TF-IDF* is a well-known method for measuring word relevance in a collection of texts.

To assess the effectiveness of these techniques, the authors applied them to a spam dataset obtained from *Kaggle* [2] . After conducting their experiments and evaluations, the authors arrived at a noteworthy conclusion. They found that among all the ML algorithms they employed, the Naive Bayes algorithm consistently achieved the highest level of accuracy in SMS spam detection.

As shown above, many researchers have investigated the same topic using different approaches. Some have focused on security within mobile architecture, including message transfer processes, while others have concentrated on using Machine Learning (ML) models to combat the issue of spam. In general, these approaches are valuable to this project and serve as its inspiration at the point that many techniques related to these approaches have been implemented in this project.

However, what sets this project apart is its pratical approach involving specific society. Rather than solely relying on existing datasets from platforms like *Kaggle and UC Ma-*

[1] **UCI ML** : The UCI Machine Learning Repository is a popular collection of datasets maintained by the University of California, Irvine (UCI). It serves as a valuable resource for researchers and practitioners in the field of machine learning and data mining. https://archive.ics.uci.edu/

[2] **Kaggle :** a platform for data science competitions and datasets. https://www.kaggle.com/

*chine Learning Repository*, this project has actively engaged with people to collect data. It has also integrated some data from these platform datasets to enhance the quality of information.

Furthermore, this project harnesses the latest advancements in machine learning. It utilizes Ensemble Methods to achieve high levels of accuracy. Addionnally, it employs technical parameter tuning, including *GridSearcher and VotingClassifier*. In fact, GridSearcher assists in identifying the most suitable parameters required for algorithm models.

Moreover, this project doesn't stop at model development, it extends to the deployment of the models generated through the processes. It provides backend *APIs* for certain platforms interested in learning from these results.

Additionally, it outlines the structure of GSM deployment, encompassing the SMSCenter's role in the mobile messaging system.

## 2.3 Tools and Techniques

In the mission of this project to create a messaging application that not only streamlines mobile communication but also tackles the pervasive issue of spam, this section serves as a technical guide. It will explore the tools and techniques at the core of the approach used.

Building an effective messaging application is like constructing a house- you need the right tools. Thus, this section discusses the software and technologies that form the foundation of our messaging app, including the programming languages, frameworks, and platforms utilized.

To combat spam effectively, this project is enlisting the help of machine learning. It delves into the world of data analysis and machine learning tools and frameworks : *numpy, pandas, matplotlib, scikit-learn,* that empower the authors to analyze, detect and prevent spam messages.

### 2.3.1 Messaging application development tools

In the realm of modern software development, the choices of development tools can significantly impact the efficiency and effectiveness of the project. When crafting application, the authors carefully considered the tools that will shape the foundation of the *backend* and *frontend* development. Actually, the *backend* references the environment where data of the app are stored, structured and more secured; while the *frontend* involves the space where techniques are developed to show to the user the interface comfortable for his understanding. Among all tools, some serve as programming languages and others as editors.

Hence, The choices made by the authors are *Python (with Django Framework)* as programming language and *SQL* for data structuring language in *backend* development and HTML, CSS and Javascript (with *Vue-Js Framework*) for *frontend* environment.

**Back-end Development with *Python* (*Django*) and *SQL* :**

 is a powerful language appeared in 1980 firstly implemented by Guido van Rossum at *Stichting Mathematisch Centrum* in the Netherlands as a successor of a language called *ABC* [83]. It newest version is *Python 3* [3] which is available for all most environments, either *Macos* or *Linux* and *Windows*.

In comparison with Java, C, C++, MATLAB, *Python* is more readable, since it requires few lines of code which are clean. Next, it is a good choice for those who want to start with programming [9]. Technically it is versatile, since it used for a wide range of applications, from web development to mobile apps (with Kivy [4] ), data analysis, machine learning, and scientific computing. Actually, the high reason that influenced the use of this programming language is its capacity to deal with data by proving scientists with many libraries.

Since the frameworks help to gain time in terms of development and structuring a project, *Python* provides many frameworks for web development. Notable these are Django, Flask, and more. Authors's choice, Django, is a high-level, open-source web framework for building robust and dynamic web applications. It's written in *Python*, which is one of the reasons for its popularity. It follows the ***batteries-included*** philosophy, providing a wealth of built-in features like authentification, URL routing, a powerful admin interface, and an ORM (Object-Relational Mapping) system, which simplifies database operations [1].

Actually, Django's ORM abstracts many *SQL* complexities, enabling developers to interact with the database using *Python* code, without needing to write raw *SQL* queries. This higher-level interaction simplifies database access and makes the development process more efficient. So, while *SQL* is at core of database interaction, Django's *ORM* providers a user-interface for developers, streamlining the development of data-driven web applications.

Suppose we want to retrieve all the employees in a database with salaries greater than $4000 using raw SQL. Here's what the SQL query would look like :

```
SELECT * FROM employees WHERE salary > 4000
```

While Django accomplishes the same task using the following code:

```
from myapp.models import Employee
  employees = Employee.objects.filters(salary__gt = 4000)
```

**How did we get there ?**

In fact, the project must be running for doing that. For installing *Python*, just go to the official website for the download, no matter the OS version, either *MACOS* , *Windows* or *Linux/UNIX* since *Python* is portable. Indeed, the last version at the writing of this project was Python 3.12.0. To test if the it is working, just enter the command python as in figure 2.1.

[3]Python : https://www.python.org/
[4]Kivy: https://realpython.com/mobile-app-kivy-python/

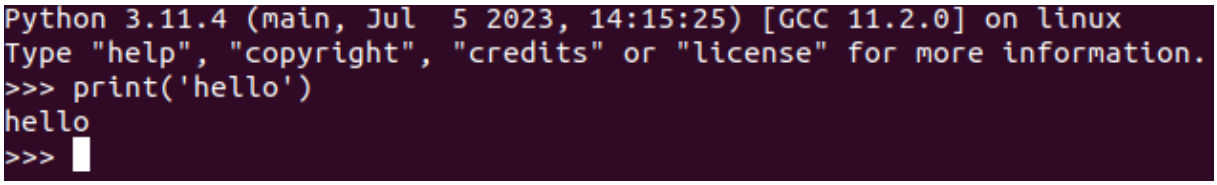



**Figure 2.1:** Testing if python is running on the OS (LINUX)

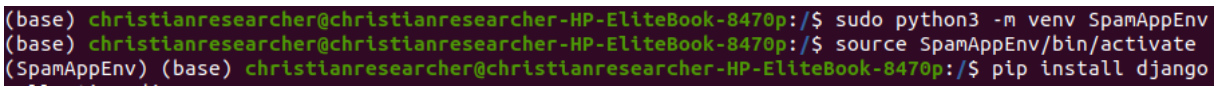



**Figure 2.2:** How to install django ? Here, the name of *VE* is SpamAppEnv

Next, for installing Django, the steps remain pretty the same, going on the official page and following the guides as resumed in the as follows: For the step 1, Installing the virtual environment ( *V.M*) [5] :

**In Windows** : `python -m venv myenv`; then active it by : `myenv\Scripts\activate`
The word *myenv* is just the name of virtual environment. But Why do we create it ? In fact, it is a best practice in Python development to manage dependencies, isolate projects, and maintain a clean and organized development environment.

In **MACOS/Linux** : Only the way of activating the *VM* changes. Then just by entering the following command:
`source myenv/bin/activate`; the virtual will be functioning.

**Finally**, as the *VM* is working Django, can be installed, by the command : `pip install Django`. At the writing of this project, the last version 4.2.6. The bit steps to follow for installing are demonstrated in the figure 2.2
Now, the project can be created, followed by the app inside, depending on interests. For that the commands : `django-admin startproject projectname` and : `django-admin startproject appname` can be utilized. The result is demonstrated in figure 2.3.

However, the configuration about the app created, ought to be made lest it should raise the error. Thus, in settings file, present in the project folder the properties called *INSTALLED_APPS*, as in the figure 2.4.
After, this we can just enter the commands :

```
python manage.py makemigrations
python manage.py migrate
```

For sure, the first command line says to Django the new changes, and the second applies or assesses them.

### How does Django deal with the database ?

Django deals with databases through *ORM* as mentioned above. Its management involves to follow different steps such as:

[5] How to create the VM ?https://realpython.com/python-virtual-environments-a-primer/

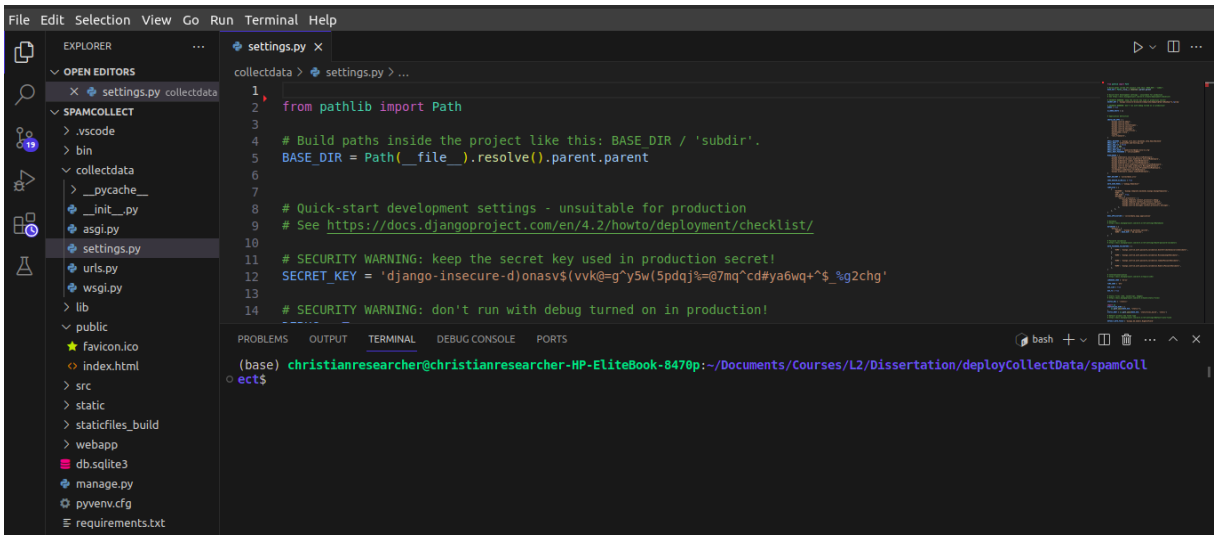



**Figure 2.3:** After the project and app are already created

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'appname',
]
```

**Figure 2.4:** Adding the appname in *INSTALLED_APPS* dependencies

**Database Configuration** : in the Django project's settings (usually in the *settings.py*, we have to specify the database we want to use. Django supports various databases types, including *PostgreSQL, Mysql, SQLite and Oracle* [6] . We have only define the database backend, connection details, and other options in the *DATABASES* setting. For instance the configuration of *MYSQL* will be like this :

```
DATABASES = {
  'default': {
  'ENGINE': 'django.db.backends.mysql',
  'NAME': 'the_db_name',
  'USER': 'the_db_user',
  'PASSWORD': 'the_db_password',
  'HOST': 'localhost',  # or the IP address of the MySQL server
  'PORT': '3306',  # MySQL default port
 }
}
```

**Model Definitions** : *Django* models are Python classes that define the structure of the database. Models are defined in the the ***models.py*** file. Each model class represents a database table, and each model field represents a table column. The syntax used for creating a database is as follows:

```
from django.db import models
class Employee(models.Model):
     name = models.CharField(max_length=100)
             salary = models.DecimalField(max_digits=10,
                decimal_places=2)
```

The corresponding *SQL* code is this:

```
        CREATE TABLE Employee (
         id INTEGER PRIMARY KEY,
         name VARCHAR(100),
         salary NUMERIC(10, 2)
        );
```

**Migrations** : Every time a model (database table) is created or modified, *Django* need to notified for assessing these changes. As mentioned earlier, the command'**python manage.py makemigrations**' is executed in the terminal for managing the changes, '**manage.py migrate**' command is entered to apply all of these changes.

**Database abstraction** : With this technique, we do'nt no longer need to write *SQL* code, since by *Python* it is possible to interact with the database, and making a *queryset* request(*crud* [7]).Let's see how it works once again:

```
new_employee = Employee(name ="Eistein", salary=450000)
new_employee.save()
```

instead of doing this in SQL :

```
INSERT INTO employees (name, salary) VALUES ('Einstein', 45000);
```

[6]Setting Databases in Django : https://docs.djangoproject.com/fr/4.2/ref/databases/#mysql-notes

[7]CRUD: Create Read Update Delete item

Indeed, we can see that even a non-professional in *SQL* can now deal with the database without any *SQL* code.
Furthermore, the use of abstraction techniques to interact with database, enhances its security of by guarding against *SQL injection*.

### How does Django do for interacting with APIs ?

*Django* presents a useful package for interacting with the *API* called *DRF* [8] It allows : authentication policies including optional packages for *OAuth1a* and *OAuth2* [9], web browsable API and serialization [10] and deserialization that supports both *ORM* And *NO-ORM* data sources [58].

## *Front-end* Development with *HTML*, *CSS* and *Js* :

The Front-end development is a crucial aspect of web development that focuses on creating the user interface and user experience of a website or web application. It involves using a combination of HTML,CSS, and Javascript to build the visible and interactive elements of a website.

The HTML (Hypertext Markup Language) is used as a maker of web pages by providing its structure and content. It uses a markup language with various tags to define headings, paragraphs, links, images, forms, and more [79]. For example a page with a heading and a paragraph should look like this in the content :

```
<div>
    <h1>Page Dev</h1>
    <p>this is our page</p>
</div}
```

Furthermore, the *CSS (Cascading Style Sheets)* is used for styling and layout. Thus, it controls the visual presentation of HTML elements. For example for our above code :

```
h1{
   color : blue; /* set the color to the element*/
   font-size: 24px; /* sets the font size */
   text-align: center; /* centers the element*/
}
```

Apart from that, the other powerful tool used in web development is *JS(JavaScript)*. It adds interactivity and dynamic behavior to web pages. It also leveraged for creating features like images sliders, form validation, animations and real-time updates without having to **reload the page**. The common frameworks used to extend its productivity are : React, Angular and Vue-js [88]. By updating the above *HTML* code, we can see the interactivity created by JS:

[8] DRF (Django Rest Framwork) : a powerful tools serving to interact with Apis. https://www.django-rest-framework.org/

[9] OAuth2 : (Open Authorization 2.0) : is a framework that allows third-party applications to access a user's data without exposing their credentials, such as passwords.OAuth 1.Oa is the old version of OAuth2

[10] Serialization : converting data into formats like *JSON*, *XML*, etc. The deserialization involves the reconstruction of the same operation

```
<!DOCTYPE html>
<html>
<head>
 <title>JavaScript Example</title>
</head>
<body>
 <div>
 <h1 id="pageTitle">Page Dev</h1>
 <p>this is our page</p>
 </div>

 <button id="changeTitleButton"  onclick="changeTitle()">Change
    Title</button>

 <script>
        // JavaScript function to  change the title
        function changeTitle() {
        // Get the h1 element by its ID
        var titleElement =  document.getElementById("pageTitle");

        // Check if the element exists
        if (titleElement) {
        // Change the text of the h1 element
        titleElement.innerText = "New Page Title";
          }
        }
 </script>
</body>
</html>
```

In this example, the button with the ID 'ChangeTitleButton' and the function called 'ChangeTitle()' are added. The click on the 'Change Title' button executes the changeTitle function. Then, the function retrieves the `<h1>` element by its ID ("pageTitle") and changes its text to "New page Title". Thus, by this bit snippet code, *JavaScript* really appears interactive.

### 2.3.2 Machine Learning tools and frameworks

Machine learning involves learning from data, visualizing, analyzing it, and making predictions. To do this, we need the right tools. *Python* is one of best and powerful tools used for these tasks. It has a large and active community that continuously contributes to its growth. *Python* offers a variety of packages that assist data scientists in their work.

In Python, common packages used for working with data include **numpy, pandas, matplotlib, seaborn, scikit-learn, tensorflow**, and more.
The Integrated Development Environment (*IDE*) used for working in Python, even for all backend-development is : Visual studio. It is portable and downloadable from the official page( https://code.visualstudio.com/). Additionally, we have another *IDE* called *Anaconda* including *Jupyter notebook* which is used for machine learning and data science projects. It's designed to simplify package management and deployment by using

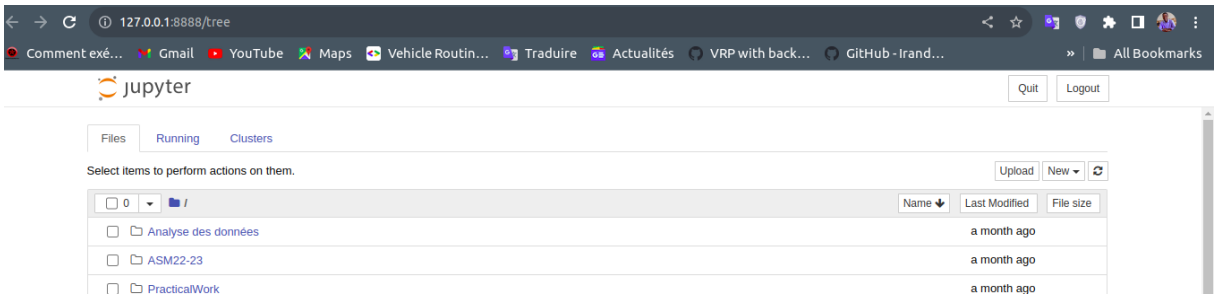



**Figure 2.5:** Start jupyter in conda kernel

a package manager called *Conda* which helps users to install, update, and manage packages, libraries and dependencies [81].

To start a new project we just execute in the terminal, the command : `jupyter notebook`, then a default browser configured just open directly the link. The page look like in the fig 2.5.

## Data manipulation with *Numpy*, *Pandas*

**Numpy** Library

Numpy is a fundamental library for scientific computing with Python. It provides support for large, mutli-dimensional arrays and matrices, along with an assortment of high-level mathematical functions to operate on these arrays.

Actually, the reason of thinking about a new tool of computing in Python is because, all data structures it provides: lists for enumerating a collection of objects, dictionaries to build hast tables, are not ideally suited to high-performance numerical computation [85]. Basically, the usage of numpy asks for importing its modules like this :

```
import numpy as np  #np is a common alias used
In [4]: np.__version__
Out [4] : '1.24.3'
```

**The structure and creation of a *NumPy* array**. The fundamental data structure provided by the Numpy library for representing arrays is **ndarray** it refers to the array which is n-dimensional or multi-dimensional [89]. Thus, several means can be used to create an array as follows :
- With 1D(dimension):

```
In[8] : np.array([1, 2, 3, 4, 5])
#Creates a NumPy array from a given list or iterable.
Out[8] : array([1, 2, 3, 4, 5])

#2D array
In[12] : np.array([[1, 2, 3], [4, 5, 6]])

In[12]: array([[1, 2, 3],
               [4, 5, 6]])
```

```
In[10] : np.arange(1, 10, 2)
# Creates a 1D array from 1 to 9 with a step of 2
Out[10] : array([1, 3, 5, 7, 9])
#2D array for arange function, requires manipulation
```

Many other functions can be used for creating arrays, like : empty, zeros, ones, eye, etc.

**Manipulation of arrays**. The manipulation operation includes : The splitting, slicing, indexing,reshaping, arithmetical operations, concatenations, comparisons and many others. Let's dive into some of that operations as follows:

```
#We generate the integers from zero to twelve and
# re pack them into a 4x3 array
In [21] : np.arange(12).reshape((4,3))
Out [21] : array([[ 0,  1,  2],
                  [ 3,  4,  5],
                  [ 6,  7,  8],
                  [ 9, 10, 11]])
#Suppose we want to multiply each vector element by 3
In [24] :  a = [3,7,4]
           [4*x for x in a]
Out[25] : [12, 28, 16]
#Concatenate arrays vertically
In [29] : np.vstack(([1, 2, 3],[1, 2, 3]))
Out [29] : array([[1, 2, 3],
                  [1, 2, 3]])
#Concatenate arrays horizontally
In [31] : np.hstack(([1, 2, 3],[1, 2, 3]))
Out [31] : array([1, 2, 3, 1, 2, 3])
```

Let's mention that ndarray object is homogeneous since all elements within must have the same data type. The types allowed are : Float, int, bytes, str, number and complex (for decimal complex numbers). To define, the type on the array on creation is made by **dtype** property like this:

```
In [55] : np.array([1, 2, 3, 4, 5], dtype='float')
Out [55] : array([1., 2., 3., 4., 5.])
```

Besides, dealing with slicing and splitting still depends on the dimension. Lets break in the code to see that :

```
In [56] : arr = np.array([[1, 2, 3],
[4, 5, 6],
[7, 8, 9]])

#split along rows (axis=0)
In [59] : split_rows = np.vsplit(arr, 3) # Split into threee 1-row
   arrays
Out[59] : [array([[1, 2, 3]]), array([[4, 5, 6]]), array([[7, 8,
   9]])]

#slicing
In [60] : arr[1:3, 1:3]  # Get a 2x2 subarray
```

```
Out [60] : array([[5, 6],
                  [8, 9]])
```

Some others functions are used as follows : all, any, cov-corrcoef, dot, where, random(), randint() and many others. The details of usage are given on the official of Numpy package [11].

## Pandas library :

*Pandas* [12] is an essential open-source Python library used for data manipulation and analysis. It provides functions for reading and writing data structures in various formats, including *CSV* and text files, Microsoft Excel, SQL databases, and the fast *HDF5* [13] format. Additionally, Pandas can **align data, handle missing data, allow reshaping and pivoting, perform data aggregation and transformation, and merge and join data sets** [54].
It offers many other capabilities for efficient data processing. To start with *pandas*, we have to create the pandas object like this:

```
import pandas as pd # pd is an alias name
```

**Data Structures**. Pandas has two main structures: Series and DataFrame. The first is one-dimensional array, while the second is a two-dimensional table that is similar to a spreadsheet or SQL. Let's break into an example:
- Series:

```
# Creating a Series from a list
In [46] : data = [1, 2, 3, 4, 5]
Out [46] : pd.Series(data)
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

- DataFrame:

```
In [48]: data = {'Name': ['Chris', 'Alice', 'Jojo'],
                 'Age': [25, 30, 22],
                 'City': ['Bukavu', 'Goma', 'Matadi']}
         pd.DataFrame(data)
```

Out[48]:

| | Name | Age | City |
|---|---|---|---|
| 0 | Chris | 25 | Bukavu |
| 1 | Alice | 30 | Goma |
| 2 | Jojo | 22 | Matadi |

[11] Numpy, official page : https://numpy.org/
[12] https://pandas.pydata.org/
[13] HDFR (Hierchical Data Format version 5) : It is a versatile and high-performance data storage format commonly used in scientific and data analysis fields.

**Reading and Writing data.** Pandas offers versatile functionality which deal with various sources and formats. The common sources include :
**- CSV files**: They are read and written like this :

```
#Reading from CSV
data = pd.read_csv('dafileName.csv')
#Writing to CSV
data.to_csv('new_data_file_name.csv', index=False)
```

**- Excel files** :

```
# Reading from Excel
data = pd.read_excel('dafileName.xlsx', sheet_name='Sheet1')

# Writing to Excel
data.to_excel('new_data_file_name.xlsx', sheet_name='Sheet1',
    index=False)
```

**- SQL Databases** :

```
# Reading from SQL database sqlite
connection = sqlite3.connect('my_database.db')
query = 'SELECT * FROM my_table'
data = pd.read_sql(query, connection)

# Writing to SQL database
data.to_sql('new_table', connection, if_exists='replace', index=
    False)
```

In the same way, functions **read_json, read_html** are also respectively used for dealing with the JSON and HTML data.

**- Data Cleaning.** Pandas uses several functions, among them we have drop_duplicates(), fillna(). Which are used like this:

```
# Removing duplicates
df = df.drop_duplicates()

# Handling missing values
df = df.fillna(0)
```

**- Data Exploration**. Pandas provides methods for exploring the dataset, such as **head(), tail(), describe(), and info()**. Thus, the application will look like :

```
data = {'A': [1, 2, 3, 4, 5],
'B': ['X', 'Y', 'Z', 'X', 'Y']}
df = pd.DataFrame(data)
# Display the summary statistics of numeric columns
print(df.describe())

              A
count  5.000000
mean   3.000000
std    1.581139
min    1.000000
```

```
25%     2.000000
50%     3.000000
75%     4.000000
max     5.000000
```

**Data Manipulation**. Pandas performs the manipulation of tables as Excel using the pivot_table() and merge() function perform combination operation for analysis. Additionally it allows indexing, slicing, filtering, modifying data. Let's break into the demonstration:

**- Pivot table** :

```
In [72] : df.pivot_table(index='B', values='A', aggfunc='mean')
print(pivot)
     A
B
X  2.5
Y  3.5
Z  3.0

#The operation computed is the average measurement
```

**- Merging** :

```
 #Create two DataFrames
 data1 = {'Key': [1, 2, 3, 4, 5],
'Value1': ['A', 'B', 'C', 'D', 'E']}
data2 = {'Key': [3, 4, 5, 6, 7],
'Value2': ['X', 'Y', 'Z', 'U', 'V']}
df1 = pd.DataFrame(data1)
df2 = pd.DataFrame(data2)

# Perform combination-like operation
result = pd.merge(df1, df2, on='Key', how='left')
print(result)

Key Value1 Value2
0    1      A    NaN
1    2      B    NaN
2    3      C      X
3    4      D      Y
4    5      E      Z
```

**- Slicing**:

```
#Consider the example on the top about
#name, age, city
# Slice specific rows and columns
  df.loc[1:3, 'Name':'Age']
#From the second column till the threeth line
#From the Name till the Age column
#The result gives this:

Name  Age
1      Bob   30
```

```
2  Charlie   35
```

Assume that we want to apply some characteristic on a column :

```
# Create a new column based on an existing column
df['days'] = df['Age'].apply(lambda x: x * 360)
print(df)
# The result is like this:
      Name   Age     City       days
0    Alice   25     New York    9000
1      Bob   30  Los Angeles   10800
2  Charlie   35      Chicago   12600
```

**- Modifying Data** :

```
df.loc[2, 'Name'] = "Jonathan"
print(df)
#The df becomes like this :
Name   Age          City    days
0      Alice    25      New York    9000
1        Bob    30   Los Angeles   10800
2   Jonathan    35       Chicago   12600
```

### Data Visualization With *Matplotlib* and *Seaborn*

#### *Matplotlib*

*Matplotlib* is a open-source, flexible and customizable Python package used for creating 2D and 3D [14] plotting having a high production-quality. In addition to this, it includes the capacity of saving images in different output formats (JPG, PNG, PS and others) [82].
To get started with it, we just go to the official page and download the dependencies [15]. However, for who are those using integrated development environment (*IDE*) like *Anaconda numpy*, *pandas*, and *matplotlib* are incorporated inside. When working with data visualization, the type of plot to choose should depend on the nature of the data. In the table 2.1 , the overview of types of plots and when to use them is given. However, some plots can be combined even though they are primarily made for whether categorical or numerical visualization.
The usage of *matplotlib* properties require a good manipulation of data, often done by the *numpy* and *pandas*. Let's break into examples of its usage:
First all, the import of *matplotlib* is required for any manipulation. It is done like this :

```
import matplotlib.pyplot as plt #plt is also an alias
```

```
import matplotlib.pyplot as plt
import numpy as np

# Data
x = np.arange(1, 11)
```

[14] mpl_toolkits.mplots3d : https://matplotlib.org/stable/tutorials/toolkits/mplot3d.html, tools used for generating 3D plots
[15] Matplotlib: https://matplotlib.org/

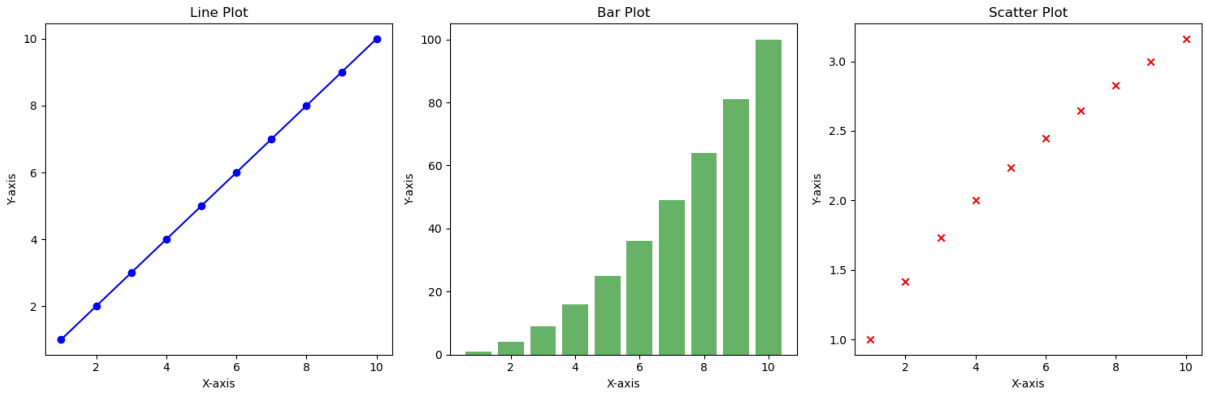



**Figure 2.6:** How to use *matplotlib* for visualization

```
y1 = x
y2 = x**2
y3 = np.sqrt(x)

# Create a figure with subplots (1row and 3 columns)
fig, axs = plt.subplots(1, 3, figsize=(15, 5))

# First Subplot: Line Plot
axs[0].plot(x, y1, color='b', marker='o') # 'o' is a clue

#define legends
axs[0].set_title('Line Plot')
axs[0].set_xlabel('X-axis')
axs[0].set_ylabel('Y-axis')

# Second Subplot: Bar Plot
axs[1].bar(x, y2, color='g', alpha=0.6)
axs[1].set_title('Bar Plot')
axs[1].set_xlabel('X-axis')
axs[1].set_ylabel('Y-axis')

# Third Subplot: Scatter Plot
axs[2].scatter(x, y3, color='r', marker='x')
axs[2].set_title('Scatter Plot')
axs[2].set_xlabel('X-axis')
axs[2].set_ylabel('Y-axis')

# Adjust subplot layout to prevent overlapping
plt.tight_layout()

# Display the figure
plt.show()
```

Additionally, *matplotlib* is used for creating 3D as follows :

```
        from mpl_toolkits.mplot3d import Axes3D
        import matplotlib.pyplot as plt
        import numpy as np
```

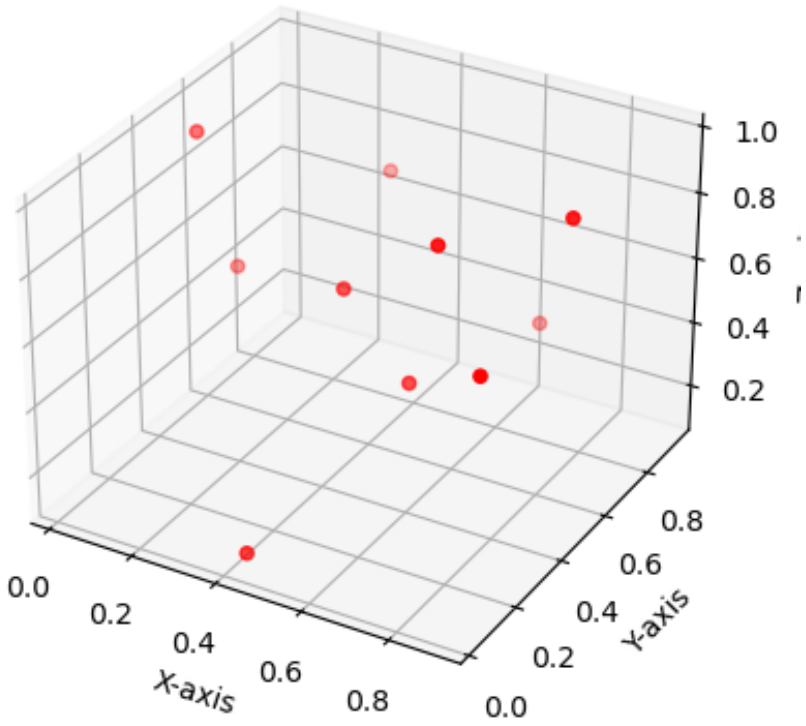



**Figure 2.7:** 3D plot with Matplotlib

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

x = np.random.rand(10)
y = np.random.rand(10)
z = np.random.rand(10)

ax.scatter(x, y, z, c='r', marker='o')

ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')

plt.show()
```

Notice that: *matplotlib*, uses *mpl_toolkits* for integrating with 3D, since at its release it was creating 2D plots only. Besides, the difference observed in the code at axis where the function *add_subplot* receives 111 value as the first argument. It just defines that the plot will be a single for all 3 dimensions as shown in the figure 2.7.

### *Seaborn*

*Seaborn* is another Python library for statistical data visualization, built on *matplotlib*. It just provides properties able to create more informative an visually appealing graphics [74].

| Type of data | Plot name | Utilities |
|---|---|---|
| Categorical Data | Bar Plot | It is deal for showing the frequency or count of categorical data.<br>Example: Comparing the number of apples, bananas, and oranges sold at a fruit stand |
| | Pie Chart | Suitable for displaying parts of a whole.<br>Example: Showing the composition of monthly expenses (eg. rent, groceries, utilities). |
| | Stacked Bar Chart | Useful for comparing categories while also showing their composition |
| Numerical Data | Histogram | Visualizes the distribution of a single variable or continuous data.<br>Example : Analyzing the distribution of ages in a population |
| | Box Plot | Displays the distribution and spread of numerical data |
| | Scatter Plot | Depicts relationships between two numerical variables<br>Example : Showing the correlation between the number of study hours and exam scores for multiple students. |
| | Line Plot | Shows changes in variable over a continuous range, often over time.<br>Example : Plotting stock prices over several months to visualize trends. |
| Mixed Data | Violion Plot | Combines a box plot with a kernel density estimation to visualize the distribution.<br>Example: Comparing the distribution of test scores across different schools. |
| | Heatmap | Useful for displaying relationship between multiple variables in matrix.<br>Example : Analyzing the correlation between various factors (eg., age, income, and education level) in a survey. |

**Table 2.1:** Choose the plot depending on data's nature

To get started with it, we just visit the official page [16] which presents all steps required for installing. Alternatively, we can use an the integrated development environment *IDE* like *Anaconda* once again, which already includes it.
The following example shows how to combine both *matplotlib* and *seaborn*:

```
import matplotlib.pyplot as plt
import seaborn as sns # Its how to start the

# Create a Seaborn plot
sns.set(style="whitegrid")
tips = sns.load_dataset("tips")
ax = sns.barplot(x="day", y="total_bill", data=tips)

# Customize the plot using Matplotlib
ax.set(xlabel="Day of the Week", ylabel="Total Bill Amount")
plt.title("Average Total Bill Amount by Day")

#Save the plot as an image
plt.savefig("seaborn_customized_plot.png")
plt.show() # Show the plot
```

The data used in the following code, named 'tips,' is provided for practice purposes. *Seaborn* is used to generate the plot, while *matplotlib* customizes it by adding a legend and saving the image generated in figure.

## Machine learning with scikit-learn

Since Python programming language is establishing itself as one the most popular languages for scientific computing. Many and various libraries are developed inside making it more and more appealing [65].

*Scikit-learn* is an open-source and commercially(usage - BSD license) machine learning library for the Python programming language. It was initially developed by David Counapeu in 2007 as part of the Google Summer of Code project. Since then, it has grown to become one the most popular and widely used Machine Learning libraries in Python ecosystem [64]. The techniques and properties used inside of it, allows it to predictive data analysis by performing the classification, regression, clustering, preprocessing and more other tasks.

To get started with it as usually, the official page present all the steps for downloading all the packages or using *IDE* like *Anaconda* which encompasses it.
Before using its functions we have to initialize the by import the class which includes the methods and properties achieving a given task. For example, suppose that we are about to process data a class called *StandardScaler* can be useful **when the dataset contains features with different scales** . Then, it standardizes the data, centering it around 0 and ensuring that it has a standard deviation close to 1.

```
In[2]:  from sklearn.preprocessing import StandardScaler
        import numpy as np
```

[16] Seaborn: https://seaborn.pydata.org/

```
3
4           # Example data
5           data = np.array([[1.0, 2.0, 3.0],
6           [4.0, 5.0, 6.0],
7           [7.0, 8.0, 9.0]])
8
9           # Create a StandardScaler instance
10          scaler = StandardScaler()
11
12          # Fit the scaler to the data and transform the data
13          data_standardized = scaler.fit_transform(data)
14
15          print("Original Data:")
16          print(data)
17          print("Standardized Data:")
18          print(data_standardized)
19
20  Out[2]:
21
22          Original Data:
23          [[1. 2. 3.]
24          [4. 5. 6.]
25          [7. 8. 9.]]
26          Standardized Data:
27          [[-1.22474487 -1.22474487 -1.22474487]
28          [ 0.          0.          0.        ]
29          [ 1.22474487  1.22474487  1.22474487]]
```

We just notice that, this time, all the standardized data have a mean close to 0 and a standard deviation close to 1. For that, the data become centered around zero and have now consistent units of measurement.

Additionally, the ***fitstransform()*** method on the object, defines both fitting and transformation process. Actually, the ***fit()*** method computes the mean and standard deviation of the trained data and ***transform()*** method scales the trained data based on the mean and standard deviation. Indeed, *fitstransform()* includes all both.
Actually, the explanations about training, fitting and others principles concerning a model are going to be tackled in the section 3 untitled :'Thinking in machine learning'.

### 2.3.3 Summary concerning the tools and frameworks

The completion of this project necessitated the utilization of various dependencies, tools, frameworks. These resources were instrumental in realizing the project's objectives. Notably, they were categorized into two main areas: those integral to the core functionality and others relevant to the user side, distinguishing the back-end from the front-end. Moreover, the project involved tools for data analysis and predictive modeling. The structure of these tools is presented in the table 2.2 for more clarity.

| | Tools | Roles |
|---|---|---|
| **Data analysis and Machine Learning libraries** | *Numpy* | Scientific computing library |
| | *Panda* | Data manipulation and analysis library |
| | *Matplotlib* | Python library used for 2D and 3D data visualization |
| | *Seaborn* | Another Python library for statistical data visualization, built on *matplotlib*. |
| | *Scickit-learn* | Machine learning machine learning library |
| **Programming Languages (*Front-end* and *Back-end*)** | *Django* Python | Python framework for developing web applications and *APIs* |
| | HTML | Maker of web pages by providing its structure and content |
| | CS | Styling the content |
| | JS | Rendering web pages interactive and dynamic |

**Table 2.2:** Structuring the tools

## 2.4 Description of the methodology and approach

Choosing a right methodology including approaches and methods to use, is an important task. However, the nature of environment defers and leads to challenging evaluation before determining the fit one. Thus, since this project investigates in Machine Learning project, the appealing methodology chosen for its achievement is inspired from MLOps(ML Operating systems).

### 2.4.1 ML operating systems

ML endeavors aim to deal with their projects from development till the production step. However, a large number of projects based on ML don't reach the final step according to their expectations. Actually, the paradigm of MLOps came to face this issue [42]. For that, it is a set of principles, best practices and concepts, and development culture that provide an end-to-end machine learning development process to design, build and manage reproducible, testable and evolvable ML-powered software.

Furthermore, MLops involves the union of many disciples, including Machine learning, software engineering (concerning DevOps too), data engineering as clarified in figure 2.8.

**T**he Machine learning part is dealt by **data scientists** and have the principle role of building the models that address the business question or needs. While **Data Engineering** is conducted by data engineers making sure that data pipelines which are the core of the ML model life cycle, are in turn and clean. Next, the **Software engineering**, encompasses **software engineers** not highly concerned by the machine learning model building since they bring ML problem into a well-engineered product (into applications). In the other hand, **DevOps** is directed by DevOps engineers who bridges the gap between

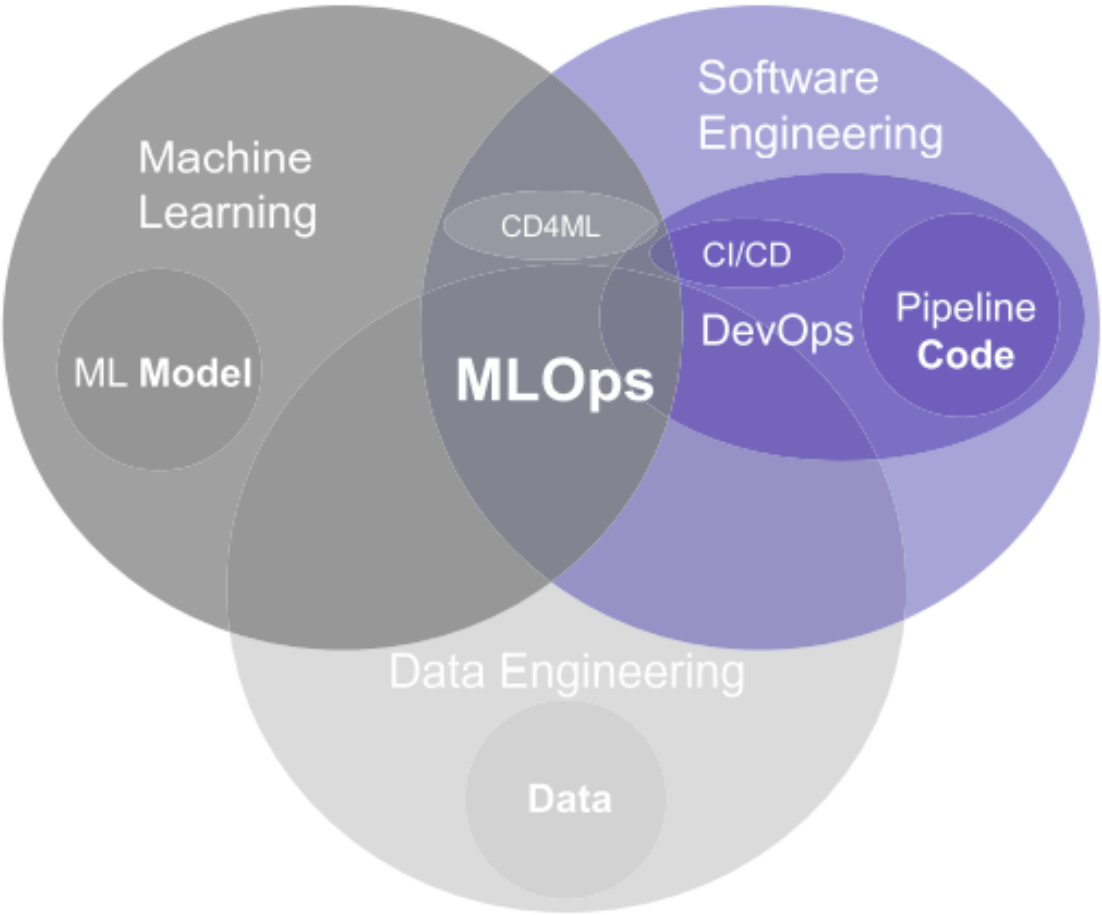



**Figure 2.8:** Intersection of disciplines in MLops

development and operations, granting that updates are continuously integrated and the development is still pursued (referring to *CI/CS*, Continuous Integration et Development ).

All these disciplines work together to follow the ML life cycle, demonstrating that ML endeavors are on ongoing process within the same project for ensuring that it is more impacting in terms of results. As shown in the figure 2.9 the ML life cycle follows several steps resumed in the following points :

- Definition of the object and specification
- Data Collection
- Preparation and exploratory Data Analysis
- Model training and selection
- Model evaluation
- Model Deployment
- Model Monitoring

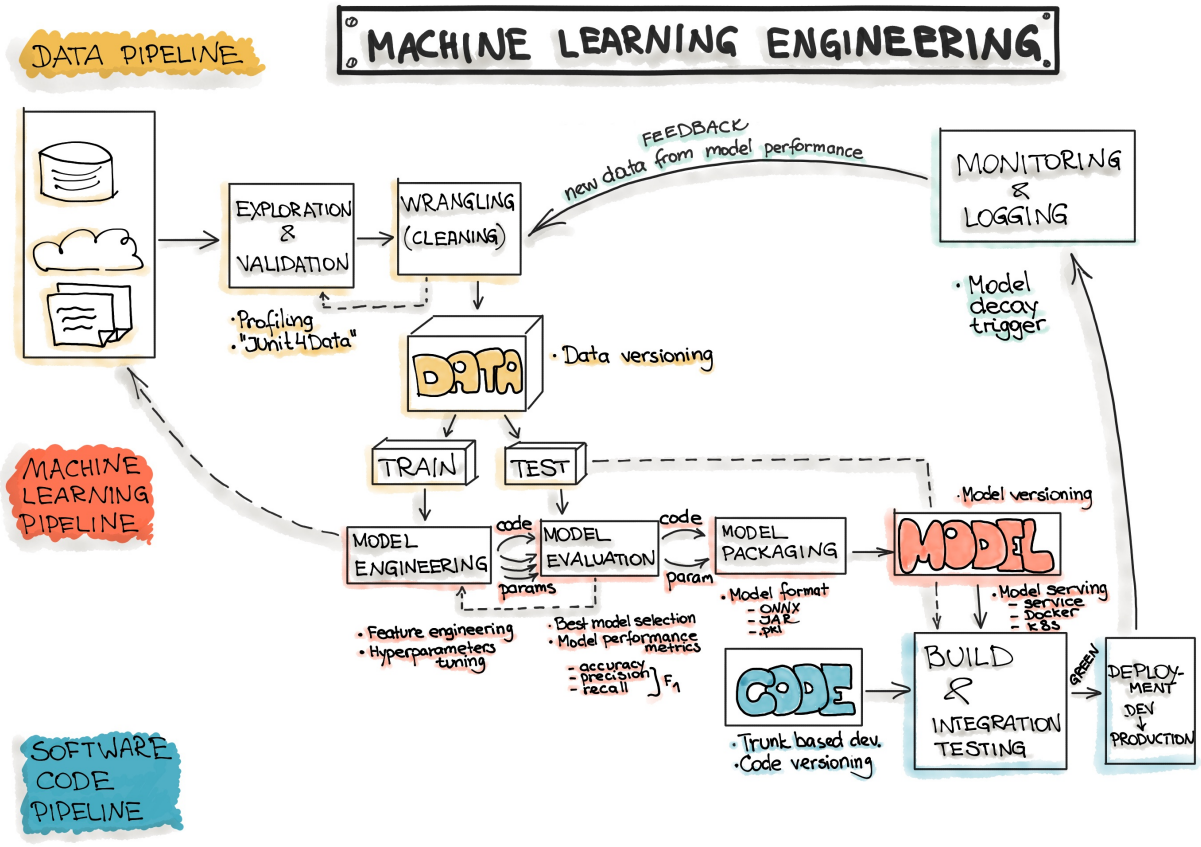



**Figure 2.9:** Machine Learning Workflow

### 2.4.2 Definition of the object and specification

The definition of the object and specification is start stage where details about the problem is given in the clear way for facilitating the ML development. This task is associated to Business stakeholders who define the goal pursued.

### 2.4.3 Gathering data

At this step, raw data are collected from various data pipeline(sources) depending on interests of project and environment. Thus, data's source can be :
(a) Databases, including data structured in relational databases like SQL and NoSQL databases. Actually, databases is mostly preferable source since they give structured data well-suited for analysis;
(b) Files (CSV, Text Files). This mean is highly used. It requires important analysis of the content for being aware of how and what part to extract (as spreadsheets for Excel), separated values (as for CSV files).
(c) APIs. They are also the common use for structured data( in JSON, XML format) , leveraging web services for providing data to third parties or internal parties of the system [92].
(d) Web Scrapping. This mean refers to techniques used for collecting data from web sites (sometimes illegally) and structure it into spreadsheets, CSV or others simple means [76]. It's more appreciated when data are not available from the APIs. That being, some

applications performs like
(e) Images and videos : Images, videos are generated from cameras, satellites and other imaging devices. Thus, they are useful in specific tasks of ML like classification of taxonomy videos [55]. Many others means exist like surveys and questionnaires, audio data, however the guidance is problem to address.
Once the data are collected, it stored in **dataset** where they are subjected to manipulation and analysis.

### 2.4.4 Data preparation and exploratory

The preparation refers to two main tasks : The data pre-processing and data exploration . The data pre-prepocessing involves the learning of nature of the data and its characteristics, types format and quality. It can result to the cleaning of data if it noticed that they are unnecessary or tidy for the analysis. In all cases based on preparing data, can be called **data wrangling** [25] process.
In the other hand, **the exploratory data analysis** process aims to receive collected and cleaned data and facilitates the visualization that can be helpful in detection of outliers, identification of clusters and trends [84].

### 2.4.5 Model training and selection

When this stage is reached, it's obviously understood that the previous, ie. gathering, preparation, exploratory steps are already fulfilled. However, it is possible to go back there once again according to the analysis requirements. What's happen during the training and testing stages ? Since all clean data are found in a dataset containing features in columns and data values in rows, the class based on choices made for the comfortable algorithm can be subsequently used to generate a model. This model is able to learn relationships and patterns within the data, is what we call **'training'**. Thus, the selection of suit ML algorithm is more competitive involving studies, testing. However it is more lead by the kind of problem we wish to solve, the number of features and its types, the kind of model that would suit the data more the best [86].
So, according to those principles, here are the types of ML algorithms used:

#### (a) Supervised models

The Supervise ML algorithms are one of algorithms often used in intelligent systems. Their manner of functioning is this: They get as inputs the data related to the features, then they map them with desired outputs (the output is input's entry).

Thus, before moving on the training step for finally getting the model, all the data are placed in a dataset where each entry represents **the feature vector** while the output defines **the target** or the answer gotten from the prediction. In addition, all the dataset entry values are considered as **feature matrix** and the columns are normally the **features**.

Therefore, for reaching a high score of precision, the prepared dataset can be split in tree parts : A part for training, used for the instructing the model, allowing it to learn patterns and relationships within the data; Second part for testing, used for testing if the model learns suitably the data provided to it, however data used for this stage are the

than contained in a dataset; then for making sure that the model is able to learn from unseen data, another part is generated from the same dataset by strictly considering the data as unknown.

A part from spiting the dataset, several tasks are used by the supervised ML models for solving problems, as follows:
**- Classification tasks**: These tasks involve categorizing input data into predefined classes or labels [61]. The model learns to assign the correct category to new data based on patterns learned during training. Thus, they can are used for: Spam detection for classifying whether a message is a spam or not; Sentiment analysis for determining the social media posts text which can be positive, negative or neutral; image classification for identifying objects or categories in images.
**- Regression tasks** : Regression tasks focus on prediction continuous numeric values based on input features. Besides, they are used to establish relationship among those features [53]. The algorithms based on these tasks are able to predict : The house prices, stock prices even the temperature forecasts.
**- Object Detection and Recognition** : The algorithms related to these tasks are used to develop which are able to locate and identify objects within images and videos. They are often used in autonomous vehicles and surveillance systems and crops detection diseases [6] .
**- Natural Language Processing (NLP)** : This task involves the algorithms generating models which are wonderful in process of translating text form one language to another and identifying entities (e.g, names, locations) in text [70] .
**- Time series Forecasting** : The model developed here, predict future values based on historical time-order data, with applications in in finance, weather forecasting (eg. prediction of wind speed [43]), and demand prediction.
**- Speech Recognition** : Here, the concerned models able to convert spoken language into text, for enabling voice assistants and transcription services [35].
**- Deep learning** : The uniqueness of models designed for this task lies in their capacity to delve deeply into extensive and intricate data, including unstructured forms such as images, text, and audio. These models possess the remarkable ability to autonomously learn complex hierarchies, enabling them to perform a wide range of tasks [33].

To gain deeper insight into the structure and impact of algorithms based on their tasks, the table 2.3 provides a more meaningful overview of some of it.

### (b) Unsupervised Models

Unlike supervised models which learns from the labels data, the unsupervised models are trained on unlabeled data. Their particularity lies in their ability to learn from complex and large amount of data. Their best goal is to find hidden patterns, structures or relationships within the data even though they are not proportional. Therefore, they are categorized as non-linear models [41].

However, for professionally dealing with complex dataset containing linear or non linear, supervised and unsupervised models can be combined. In this case, the unsupervised can be mostly used to reduce the dimensionality of data before applying and the supervised model for linear regression prediction for example. This approach let profit from all both

| Tasks | Algorithms | Genre and problem solving examples |
|---|---|---|
| Classification | Naive Bayes, Logistic Regression, Support Vector Machine (SM), Random Forest, Decision Trees, etc. | Categorize input data into predefined classes or labels.<br>E.g: Sentiment Analysis. |
| Regression | Linear Regression, Polynomial Regression, Lasso Regression, Ridge Regression, Suport Vector Regression (SVR), etc. | Predict continuous numerical output value.<br>E.g: House prices forecasting. |
| Object Detection | Convolutional Neural Networks (CNNs), etc. | YOLO (You Only Look Once).<br>E.g: Self-driving cars |
| Natural Language Processing (NLP) | RNNS(Reccurrent Neural Networks), LSTMs, etc. | Undestand human language.<br>E.g: Language translation. |
| Time series Analysis | ARIMA(Autoregressive Integral Moving average), Exponential Smoothing methods, Seasonal Decomposition of Time Series (STL), etc. | Predict future values in a time series.<br>E.g: Weather prediction. |
| Deep Learning | RNNs, NLP, GANs(Generative Adversarial Networks), etc. | Learn hierarchical representations from data, for deep prediction<br>E.g: Image recognition. |

**Table 2.3:** Which task for which Supervised Machine learning algorithm

method's advantages [50].

The common tasks concerned by this ML type include : clustering, dimensionality reduction, data compression, topic modeling and many others.

**Clustering task**, is the fundamental task of this type. It is helps to group data into sensible grouping objects according to similarities and characteristics. It obvious that all any class if pre-selected, or fore-grouped. Thus, the problem like : Grouping customers into clusters based on their purchasing behavior for market segmentation.

Subsequently, **the dimensionality reduction tasks** involves model's techniques simplifying the dataset to train while preserving its essential characteristics, that can be helpful to improve the performance of the model. The challenge of this method when projecting data [78], is to choose the right techniques, we mean parameters, features,... The related solutions include : Data processing for machine learning tasks, visualization of high-dimensional data in fields like biology and natural language processing.
In addition, **the data compression** task, is inspired from the recent advancements in the field of information technology resulted to the generation of huge amount of data at each and every second [34] . Therefore, the need of having a method able to eliminate data

redundancy and irrelevancy is observed. This need becomes fulfilled by the unsupervised models allowing further analysis and pre-processing.

Furthermore, **the topic modeling**, is due to the growth of texts in new environment like internet, where informations including news headlines, web pages, questions/answers are published [67]. Thus, analysis of text becomes essential. It is performed by unsupervised models focusing on language, frequent words, and many others parameters.

The table provides a comprehensive overview of these tasks and the associated algorithms employed to achieve them.

| Tasks | Algorithms | Genre and problem solving examples |
|---|---|---|
| Clustering | K-Means Clustering, Hierarchical Clustering, DBSCAN, etc. | Group data points into clusters based on similarity, without prior knowledge of group labels<br>E.g: Customer Segmentation . |
| Dimension Reduction | PCA(Principal Component Analysis), t-Distributed Stochastic Neighbor Embedding, etc. | Reduce the number of features(dimensions) in a dataset and retain essential information.<br>E.g: Image compression . |
| Topic Modeling | LDA(Latent Dirichlet Allocation), Non-Negative Matric Factorization(NMF), etc. | Discovering latent topics within a collection of documents<br>E.g: Content Recommendation |
| Data compression | PCA | Preprocess daat to make data more manageable.<br>E.g : Data storage. |

**Table 2.4:** Which task for which Unsupervised Machine learning algorithm

### 2.4.6 Model Evaluation

The evaluation of machine learning models is a critical aspect of the model development process. Its helps to assess the model's performance, its ability to make accurate predictions, and its generalizations to unseen data [68].

Therefore, various techniques are implemented to assess that the model is worthy to be used. However, it defers from problem or models types performing a certain tasks. Thus, two main techniques are used : **cross-validation and the measure of metrics**.

The cross-validation is used to measure the performance of the model. For performing this, the methods such as: ***K-Fold*** Cross-validation are used. This one; divides the dataset into **'k' subsets** of approximately equal size, then the model is trained and tested **'k' times** using each of the k subsets as the test set exactly once while the remaining **subsets are used for training**. The others cross-validation are : Stratified cross-validation and

Leave-One-Out Cross-validation (LOOCV).

A part from cross-validation, we have the measure of metrics as one of highly techniques used to estimate the model accuracy. However, the metrics depends on tasks performed by the model:

For classification tasks, it used what we call the confusion matrices to evaluate errors in classification problems. As shown in table 2.5, [68] , this method focuses on evaluating the ability of model to classify instances into different categories. Thus, there are **True Positives** (TP) which are instances that were correctly predicted as belonging to the positive class; **True Negatives** (TN), which are instances that were correctly predicted as belonging to the negative class. In the same way, **False Positives** (FP) are instances that were incorrectly predicted as belonging to the positive class. While the **False Negatives** (FN) are instances that were incorrectly predicted as belonging to the negative class [52].

| | **Observations** | | |
|---|---|---|---|
| **Predictions** | **Actual Negative** | **Actual Positive** | **Total** |
| Predicted Negative | True Negatives (TN) | False Negatives (FN) | Total of Negatives predicted (FN+TN) |
| Predicted Positive | False Positives (FP) | True Positives (TP) | Total of Positives predicted (FP+TP) |
| Total | Total of Negatives observed | Total of Positives Observed | Size of all total data |

**Table 2.5:** Confusion Matrix

Therefore, those parameters give sense to several metrics as following :
- **Accuracy**, which is the proportion of correct predictions over the total number of predictions (Eq.2.1) : ¨

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

- **The Precision**, is the fraction of true positive predictions(TP) out of all positive predictions (Eq.2.2).

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

- There's the **Recall** also called Sensitivity, True Positive Rate or TPR; measuring the fraction of true positive predictions out of all actual positive instances (Eq.2.3).

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

- There's too, the **F1-Score**, which is the harmonic mean of precision and recall since it balances precision and recall (Eq. 2.4).

$$F1Score = \frac{2 \cdot (Precision \cdot Recall)}{Precision + Recall} = \frac{2VP}{2VP + FP + FN} \tag{2.4}$$

For sure, all the metrics are convenient for a given situation or problem to solve and have to be high. Most of the time the percent of 80 % is high, while in others 90% with the possibility of taking care of false positives.

Besides, the **Regression Tasks** present too the a lot metrics among them there are [90] :
- **Mean Absolute Error(MAE)**, it measures the average absolute difference between the predicted values and the actual values.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.5}$$

- **Mean Squared Error (MSE)**, it measures the average of the squared differences between predicted and actual values. Every time a metric's value is lower, it indicates that the model's predictions are closer to the actual values.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.6}$$

- **R-squared ($\mathbf{R^2}$) or Coefficient of Determination**, it measures the proportion of the variance in the target variable that is explained by the model.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.7}$$

In terms of interpretation in a programming language , case of Python, these metrics are found in the packages : *sklearn.metrics*.

### 2.4.7 Model Deployment and monitoring

Machine Learning Model Deployment refers to the process of taking a trained machine learning model and making it available for use in real-world applications. Therefore, several means are used: Primarily, we have to choose the environment. Is it going to be integrated in a cloud platforms, or applications, edge devices ? Actually, it depends on interests. Some developers can decide to create an *API* that allows software components to interact with clean model built as Web Service [75]. For instance in python the *frameworks* like *Flask, Django* since they offer the *RestApi* services.

However, the model required to be integrated in a file for insuring the integration whether in the cloud or others services as emphasized above. Thus, the packages like : *joblib* [17] . Actually, *Joblib* is used for serialization(saving and loading) Python objects. It's convenient for task like **parallel computing and disk caching of functions** [24].

In part from, the model should be integrated in a Continuous integration and Continuous Deployment (CI/CD) (task of DevOps developer) to facilitate **monitoring** and updating, even the easy deployment.

[17]joblib: https://joblib.readthedocs.io/en/latest/index.html

Furthermore, the security of a model is so important too. Hence, the model's container should be protected in a safe area.

## 2.5 ML Algorithms and Scikit-learn

ML Algorithms are numerous, however some of them are known to be utilized in several tasks as shown above, both unsupervised and supervised algorithms are supported in Python Scikit-learn package, whereas the logic behind its implementation is more important since it's allows the understanding and optimization of solutions. Thus, the following lines summarize the common algorithms and how they can be implemented in Scikit-learn.

Since two main types of category are notable in analysis which are : categorical(qualitative) and quantitative, the presented algorithms are set in the same approach for more clarification.

### 2.5.1 Predict Qualitative feature

The prediction's result for this kind of data, involves an output which is commonly from binary, text, orders classes. Thus, we have :

#### Naive Bayes

The Naive Bayes is a classification algorithm based on Baye's theorem, which is a fundamental concept in probability theory [52]. Actually, the Naive Bayes formula is given like this (Eq.2.8) :

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{2.8}$$

In this equation 2.8, the event A represents the existence of **class** A; while the event B intervenes when the **input** B is given. Thus, we read the probability of having the class A when the input B is given.
By assuming that we have more than one input, the formula will look like this :

$$P(A|B_1, B_2, \ldots, B_n) = \frac{\mathbf{P(A)} \cdot P(B_1|A) \cdot P(B_2|A) \cdot \ldots \cdot P(B_n|A)}{P(B_1, B_2, \ldots, B_n)} \tag{2.9}$$

$$P(A|B_1, B_2, \ldots, B_n) \propto P(A) \cdot P(B_1|A) \cdot P(B_2|A) \cdot \ldots \cdot P(B_n|A) \tag{2.10}$$

Since in equation 2.9, the denominator is in the second member is independent from the class (A), we just substitute the comparison sign by : **proportional to** sign. Hence, the final formula used in the implementation is like this :

$$P(A|B_1, B_2, \ldots, B_n) \propto P(A) \prod_{i=1}^{n} P(B_i|A) \tag{2.11}$$

Great,the *Naive Baye* in sckit-learn will consequently look like this :
***sklearn.naive_bayes.GaussianNB()***. As it it is ***GaussianNB*** type, it means that the features's values affected to the model are continuous instead of being discrete, normally distributed. The model can be ***Multinomial*** too, when it is interested in discrete's values, it means the frequency of words.

### Logistic Regression

Logistic Regression is a binary and multi-class classification algorithm that models the relationship between inputs features and the probability of a particular event occurring [71]. For transforming a linear combination of features into probabilities between 0 and 1, it uses the *sigmoid* function as follows :

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.12}$$

The **z** variable is the linear combination of input features and coefficients.
A part from that, *LR* model uses techniques like regularization for preventing over-fitting, solver for optimizing problem, etc. To get started with it, with scickit-learn, we just use the class : ***sklearn.linear_ model.LogisticRegression()***

### Decision Trees

As all supervised learning algorithms presented above, Decision Trees(DTs) are a non-parametric method used for classification and regression. Its prediction's ability is based on the if-then-else learn's possibilities [18]. Actually, the tree is composed by the nodes and branches. Each node represents features in a category to be classified and each subset defines a value that can be taken by the node [12].

### Support vector machine (SVM)

SVMs are a class of powerful machine learning algorithms used for classification and regression. They aim to find the optimal hyperplane that maximally separates data points of different classes while maximizing a margin. It's a favorite solution for high-dimensional data[51]. Its class look like this : ***sklearn.svm.SVC()***

### *KNeighborsClassifier*

The ***KNeighborsClassifier*** is ML algorithm based on *k-Nearest Neighbors* (K-NN) principles. It classifies data points by finding the ***k*** nearest from the training dataset and assigning a class label based on majority voting. For measuring similarity between data points, it utilizes the distance metric [60]. To get started with it, we need to use the class ***sklearn.neighbors.KNeighborsClassifier()***.

### Random Forest

Random Forest is an updated algorithm version of decision trees. It combines trees to enhance predictive accuracy in classification and regression tasks. It employs *bagging* [19] to create diverse subsets of the training data and random feature selection for each tree[49]. Consequently, the overfitting is reduced and generalization is enhanced.

A part from optimizing prediction's result, it is reputed for being scalable and applicable to both small and large datasets.
To get started with it in Python, scikit-learn library we just the following class :
***klearn.ensemble.RandomForestClassifier()***.

[18] sckit-learn: https://scikit-learn.org/stable/modules/tree.html
[19] Bagging : Boostrap aggregating, improves the accuracy and robustness of models

#### Neural Networks (ANN)

The Artificial Neural Networks (ANNs) are ML models inspired by the human brain. They consist of interconnected nodes(neurons) [8] organized into layers: input, hidden, and output. They excel in learning complex patterns from data through a process called ***backpropagation*** **[48]**.
To get started with it, we can just use the simple variant's class :
***sklearn.neural_ network.MLPClassifier()***.

### 2.5.2 Predict quantitative feature

This section includes algorithms with the objective of prediction numerous values, which can be either discrete or continuous. Therefore, we have :

#### *ElasticNet*

ElasticNet is a crucial tool in building of robust and generalized models. It uses regularization techniques in the linear regression, allowing to address the limitations of L1(Lasso) and L2(Ridge) regularization methods [36]. To get started with it, we just implement the class : ***sklearn.linear_ model.ElasticNet()***

#### *Gradient Boosting Regressor (GBR)*

*GBR* is an improved version of *gradient descent*, making more accurate and good at minimizing a loss function, since it is an ensemble ML algorithms used for regression tasks. The other particularities of this model is the configuration's parameters supporting in ***boosting settings, model tuning***. To get started we wih *GBR*, we just use the following class : ***sklearn.ensemble.GradientBoostingRegressor()***

#### *Regression Lasso/Ridge*

All both (*Ridge/Lasso*) are updated version of *Linear Regression* algorithm, however they are used for avoiding overfitting (regularization) of a model. Therefore, *Lasso* uses *L1* regularization penalty added to the **loss function**. In the other hand, the Ridge type uses *L2* regularization integrating the loss function based on the square of the model's coefficients [31]. Actually the loss function is used during the model training stage for quantifying and minimizing the error between the model's predictions and the actual target values for a given data [87].

#### *KNeigboursRegressor*

The *K-Nearest Neighbors Regressor* often abbreviated as *KNeighborsRegressor* is a member of the *K-Nearest Neighbors(KNN)* family, including classification and regression variants. KNeighborsRegressor is particularly well-suited for solving problems where the target variable is continuous and requires predicting a numeric value, for instance the prediction of crime using KNN Regressor [3].
In *scikitlearn*, in order to deal with this model, it has to be instantiated from the following class :
***sklearn.neighbors.KNeighborsRegressor()***.

## 2.6 Techniques used for optimizing the model

Several techniques are employed to ensure that the implemented model achieves a high level of accuracy. While we've touched upon these concepts indirectly in preceding discussions, it's worth highlighting them for more clarity. Thus, we have:

### Regularization

The regularization is technique used upon a model to enhance the generalization ability of the models. The methods like Regression *Ridge*, *Lasso*, *Elastic* can be used for that concern [20]. For instance : *L1* regularization called *Lasso* can be used to drive all owner's name values feature to exactly zero, since it's **might have any impact on price**, concerning the house price prediction project.

### Ensemble models

The ensemble methods combine the predictions of multiple individual models(base models) to produce a more accurate and robust prediction [37]. Therefore, it raises new methods such as :

#### (a) *Bagging* (Boostrap Aggregating)

Actually, Bagging involves the process of training multiple instances of the same base model on different subsets of the training data and finally takes the mean of all the predictions. The case explaining this is how Random Forest works as ensemble method and the Decision Trees as its base models. The examples of the models's genre are : ***AdaBoost*** and ***Gradient Boosting***.

#### (b) *Boosting*

The boosting process focuses on training multiple base models sequentially. Each model in the ensemble corrects the errors of the previous one, with more emphasis on misclassified data points. Actually, the correction's goal is the achievement of accurate prediction rule [66].

#### (c) *Voting*

Voting ensembles is more special. It combines the predictions of multiple **base models by taking a majority vote**(for classification) or averaging(for regression) to make the final prediction [37]. However, if the model is not optimized, it is might be biased. The model like *VotingClassifier, VotingRegressor* are commonly used.

### *Dimensionality Reduction*

Dimension reduction techniques focus on reducing the number of features (dimensions) in a dataset while preserving or maximizing the relevant information. They are also used to address issues associated with high-dimensional data, such as the curse of dimensionality, overfitting, and computational complexity. Thus, we discuss **feature selection** when we aim to retain the relevant and informative features and **feature extraction** when data

is projected into a lower-dimensional space [19]. In practice, the original data is first transformed and then technically selected to create a new representative dataset.
The common models used for dimensionality reduction include techniques such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA).

### *Hyperparameter Tuning*

Hyperparamater tuning, often referred to as hyperparameter optimization, is the process of systematically searching for the best combination of hyperparameters for a machine learning model [13].

Actually, they are settings that are not learned from the data but are set prior to training and setting a model. Examples of hyperparameters include the learning rate in neural networks, the depth of a decision tree, the number of neighbors in k-nearest neighbors, or the regularization strength in linear models. In scikit learn model called ***GridSearchCv*** can be used to test the convenient parameters.

### *Normalization and standardization*

Both normalization and standardization are preprocessing techniques commonly used in machine learning and statistics to transform data before feeding it into a model. They serve different purposes but aim to make data more suitable for modeling and analysis.

The normalization is the process of rescaling data to a common scale, typically between 0 and 1. It's useful when features we have different ranges, and we want to bring them to a uniform scale [2], [22]. Common methods for normalization include Min-Max scaling.
The formula used for Min-Max scaling is :

$$X\_normalized = (X - X\_min)/(X\_max - X\_min) \tag{2.13}$$

Where $X$ is the original value, $X\_min$ is the minimum value in the dataset, and $X\_max$ is the maximum value.
The standardization, on the other hand, transforms data into a standard normal distribution with a mean of 0 and a standard deviation of 1 [2]. It's particularly useful when the features follow a normal or near-normal distribution, and the absolute values and relationships between them are important. Practically, it involves subtracting the mean from each value and dividing by the standard deviation.

$$XStardardized = (X - mean)/standardDeviation \tag{2.14}$$

In scikit learn the model like *PCA* can be used to compute this formula on a dataset.

## 2.7 Summary

In this chapter, the primary focus is on the theorical aspects of the current work. It starts by reviewing related works to identify potential enhancements offered by this work. Subsequently, it explores the tools employed for realizing the proposed solution and delved into the methodological approaches adopted.

Regarding tools and techniques, they classified into two main categories. The first category comprises the tools utilized for constructiong machine learning models, which include algorithms implemented in Python using various libraries. The second category involves application tools, such as web technologies like HTML, CSS, Javascript, and the Django Framework. These applications tools are instrumental in creating user interfaces for consuming a web service.

Furthermore, this chapter delves deeply into the *MLops*(Machine Learning Operations) methodology employed to make a machine learning model consumable and adaptable to evolving needs. During this phase, it explores the various steps necessary to develop a robust model with high precision and accuracy.

# Chapter 3

# Application of the methodology and results with analysis

## 3.1 Introduction

This chapter delves into the practical aspect of the present work. It demonstrates how the discussed methodology has been applied and how it culminates in proposing a solution for the identified problems. In addition, it addresses how individuals contribute to its completion, the constraints associated with its deployment and usage, as well as the encountered limitations. Furthermore, it explores potential perspectives for future research by other researchers.

## 3.2 Participants

### 3.2.1 Survey participants

This work couldn't have any sense without the collaboration, assistance of the volunteers who are almost students and friends at camp, members of social groups. They just provided to us the unwillingly messages as much as they were able.

### 3.2.2 Team work structure

The present project witnessed a dynamic partnership between the authors and an enthusiastic academic team, consisting of both the supervisor and director. They played pivotal roles in guiding the project, offering valuable insights into data collection methodologies, and suggesting techniques to achieve our goals effectively. Beyond the academic community, we also benefited from expertise of online scientists who provided valuable advise and coaching to ensure our solutions were relevant and impactfull.

## 3.3 Data collection strategy

Recognizing the pivotal role of data at the heart of our solution, an extensive investigation into data collection became imperative. Our fist approach to streamline this process was by harnessing the power of the online realm. Trough a dedicated website, we deployed a user-friendly survey form; which individuals could readily complete and submit. Visualize

this in figure 10 in the annexes. Therefore, all the data collected have been stored in a database. To really collect it for further investigations, it was just question of clicking on a download button in the website as shown in figure11 in the annexes.
Furthermore, in keeping with the common practice among data scientists, who often employ online datasets for their experiments and analysis, this work also delved into **a dataset readily available on** *Kaggle*, which allows this work to tackle broad languages even though they are not highly spoken in case study area. [1]

## 3.4 Application of the methodology

Pursuing the ML operating systems methodology: From the gathering step to model deployment, this section presents all the steps which lead to the model worthy to make predictions.

### 3.4.1 Gathering data

Collecting technically data required to know and its sources. Thus, pursuing that in *jupyter* notebook, we firstly import the essential dependencies like this :

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score,
   recall_score, f1_score
import matplotlib.pyplot as plt
import seaborn as sns
import time
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import VotingClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import roc_auc_score,accuracy_score,
   precision_score, recall_score, f1_score
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
import detectlanguage
from sklearn.feature_extraction.text import TfidfVectorizer
```

Hence, the pandas alias created allows to gather data, both based on two languages category, one highly concerns strange languages (French, English), and the other local (Swahili). Therefore, we used the following code :

```
# Multilinguages data
```

[1] https://www.kaggle.com/ : Actually, *Kaggle* is an advanced online community for networking works and storing data mostly used for analysis project.

```
messageInMultiLang = pd.read_csv('data-en-hi-de-fr.csv')

#local data
messageInSwahili = pd.read_csv('swahiliDataset.csv')
```

### 3.4.2 Data preparation and exploratory

During this step, we dropped null values, duplicates, reformatted data types and finally we created plot for visualizing data following the present steps and techniques:

- Check the columns inside of the data:

```
In [43]:# columns in multilingual dataset
         print(messageInMultiLang.columns)
         print(messageInSwahili.columns)

         Index(['labels', 'text', 'text_hi', 'text_de', 'text_fr'],
            dtype='object')
         Index(['labels', 'message'], dtype='object')
```

- Extract the valuable features:

```
         #extract english messages
         messageInEnglish = messageInMultiLang[['labels','text']]

         #extract french messages
         messageInFrench = messageInMultiLang[['labels','text_fr']]
```

- Visualize the sample data :

(a) English language data:

```
In [46]: messageInEnglish.head()
```

Out[46]:

| | labels | text |
|---|---|---|
| 0 | ham | Go until jurong point, crazy.. Available only ... |
| 1 | ham | Ok lar... Joking wif u oni... |
| 2 | spam | Free entry in 2 a wkly comp to win FA Cup fina... |
| 3 | ham | U dun say so early hor... U c already then say... |
| 4 | ham | Nah I don't think he goes to usf, he lives aro... |

(b) French language data:

```
In [47]: messageInFrench.head()
```

Out[47]:

| | labels | text_fr |
|---|---|---|
| 0 | ham | Allez jusqu'à Jurong point, fou.. Disponible s... |
| 1 | ham | J'ai fait une blague sur le wif u oni... |
| 2 | spam | Entrée libre dans 2 a wkly comp pour gagner FA... |
| 3 | ham | U dun dit si tôt hor... U c déjà dire alors... |
| 4 | ham | Non, je ne pense pas qu'il va à usf, il vit da... |

(c) Swahili Language data:

```
In [48]: messageInSwahili.head()
```

Out[48]:

| | labels | message |
|---|---|---|
| 0 | ham | Angalia filamu ya Telugu..? |
| 1 | ham | Anwani yangu ya ip ingejaribuje kuwa kwani kom... |
| 2 | ham | Asante sana kwa matakwa yako kwenye siku yangu... |
| 3 | ham | Bado siko huko usiku wa leo. |
| 4 | ham | Bado siko huko usiku wa leo. |

- Counting data:
(a) English Language data :

```
pd.crosstab(index=messageInSwahili['labels'], columns='count')
pd.crosstab(index=messageInFrench['labels'], columns='count')
pd.crosstab(index=messageInEnglish['labels'], columns='count')
```

The above code gives us results concerning raw data in our all these pictures and using, all both datasets contained the data as follows in the table 3.1 :

| Languages | Ham | Spam | Total |
|---|---|---|---|
| English | 4825 | 747 | 5572 |
| French | 4825 | 747 | 5572 |
| Swahili | 117 | 15 | 132 |

**Table 3.1:** Raw data counted before any wrangling

- Checking null values :

```
print(messageInEnglish.isna().sum())
print(messageInFrench.isna().sum())
print(messageInSwahili.isna().sum())
```

This snippet code prints that any entry is empty.

- Dropping duplicated values :

```
messageInSwahili=messageInSwahili.drop_duplicates()
messageInFrench = messageInFrench.drop_duplicates()
messageInEnglish=messageInEnglish.drop_duplicates()
```

After dropping duplicated values, it is noticed that respectively 19, 438 and 415 Swahili, French and English entries were dropped. Thus, the illustration of clean data is in the table 3.2, also visualized in figure 3.1, 3.2, 3.3.

```
custom_colours = ['#ff7675', '#74b9ff']
labels = ['ham', 'spam']

def visualisationMsg(data, text):
sizes = data['labels'].value_counts().tolist()

plt.figure(figsize=(20, 8), dpi=227)
```

| Languages | Ham | Spam | Total |
|---|---|---|---|
| English | 4516 | 641 | 5135 |
| French | 4494 | 640 | 5134 |
| Swahili | 99 | 14 | 113 |

**Table 3.2:** Clean data counted after preprocessing

```
8
9  # Plot 1 - Pie Chart
10 plt.subplot(1, 2, 1)
11 plt.pie(sizes, labels=labels, textprops={'fontsize': 15},
       startangle=140,
12 autopct='%1.0f%%', colors=custom_colours, explode=[0, 0.05])
13 plt.title('Distribution of '+text)
14
15 # Plot 2 - Bar Plot
16 plt.subplot(1, 2, 2)
17 sns.barplot(x=data['labels'].unique(), y=data['labels'].
       value_counts(), palette='viridis')
18 plt.title('Bar Plot of Messages'+text)
19
20 plt.show()
```

```
1 visualisationMsg(messageInSwahili, " Swahili messages")
2 visualisationMsg(messageInFrench, ' French messages')
3 visualisationMsg(messageInEnglish, ' English messages')
```

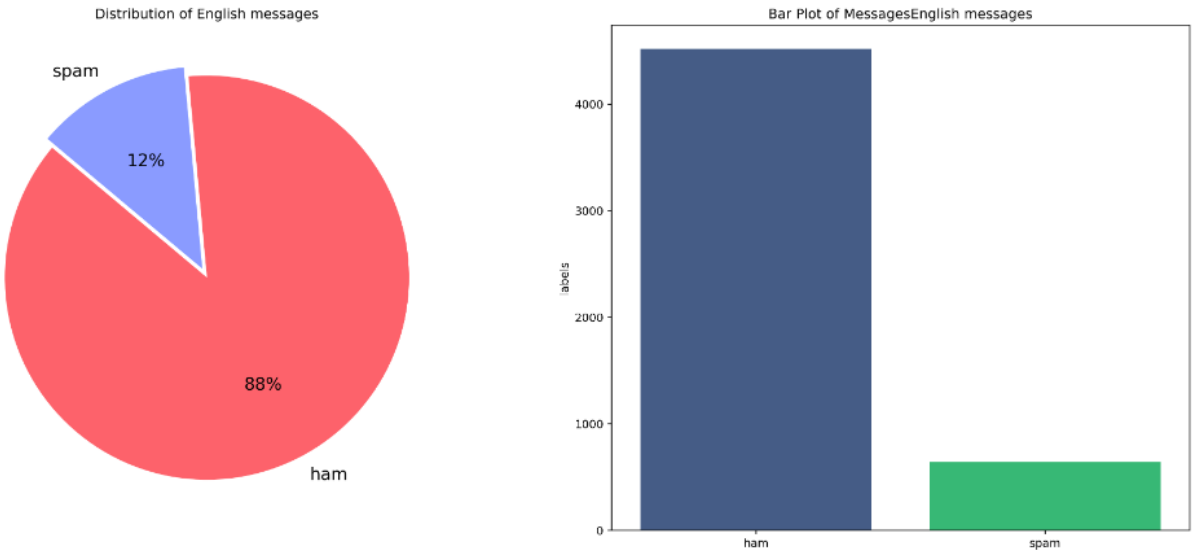



**Figure 3.1:** Clean English entries

- Feature engineering:
During this process, we select the target feature, and transform it in the boolean format in which it can be more understood by the model. For that, we *labelEncoder() model* as follows :

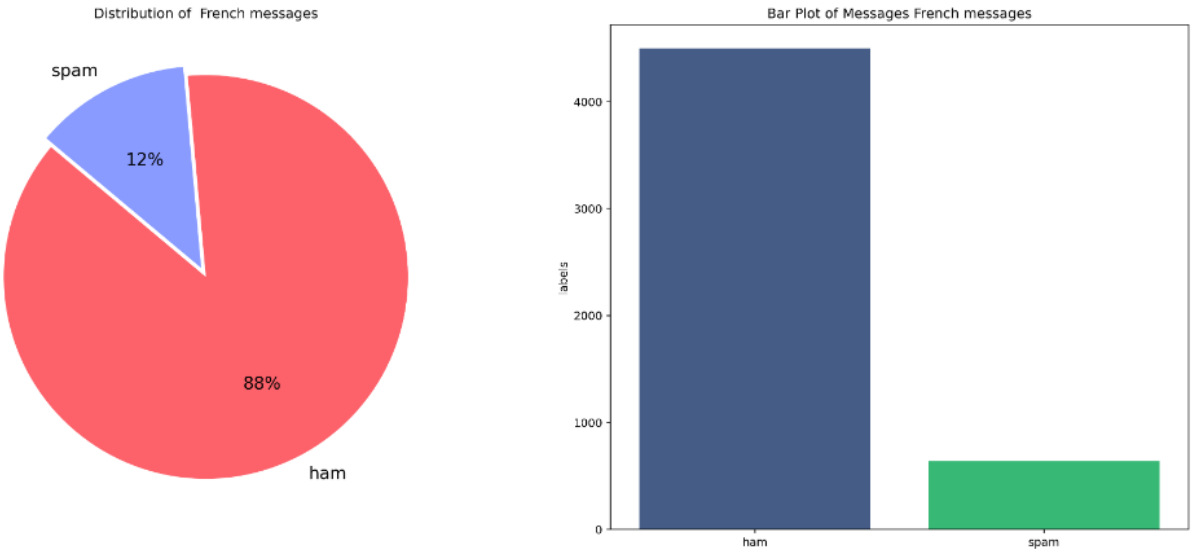



**Figure 3.2:** Clean French entries

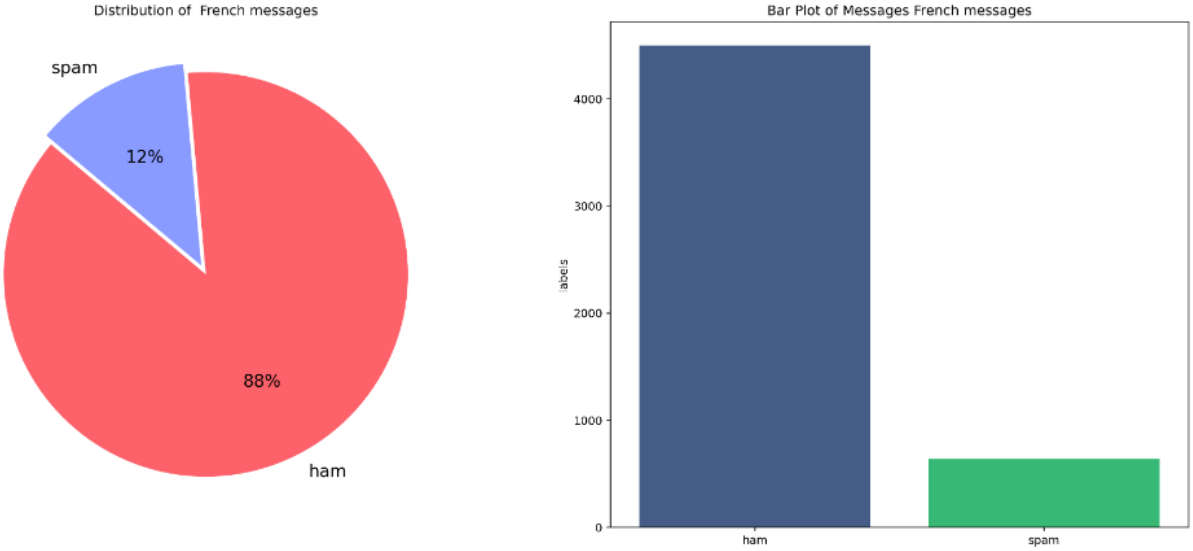



**Figure 3.3:** Clean Swahili entries

```
        #Create the instance
        modelLabelEncoder = LabelEncoder()

        #Create a helpul function
        def encondModel(dataMessage):
        #encod the target feature
          return modelLabelEncoder.fit_transform(dataMessage['
             labels'])

        #we create the y feature for each language
        y_en = encondModel(messageInEnglish)
        y_fr = encondModel(messageInFrench)
        y_sw = encondModel(messageInSwahili)
```

### 3.4.3 Model selection and prediction

The division of data in slices: Ones used for training and other for testing :

```
#English dataset
x_train_en, x_test_en, y_train_en,y_test_en = train_test_split(
    messageInEnglish['text'],\
y_en, test_size=0.2, stratify=messageInEnglish["labels"])

#French dataset
x_train_fr, x_test_fr, y_train_fr,y_test_fr = train_test_split(
    messageInFrench['text_fr'],\
y_fr, test_size=0.2, stratify=messageInFrench["labels"])
#Swahili dataset
x_train_sw, x_test_sw, y_train_sw,y_test_sw = train_test_split(
    messageInSwahili['message'],\
y_sw, test_size=0.2, stratify=messageInSwahili["labels"])
```

The words : *text, text_ft, message* are names of messages values. Besides, *stratify* parameter is optional, however used for ensuring that the distribution made remains consistent. - Feature extraction: The current extraction involved a use of model which is able to convert text into numeric values understandable by the model. Thus, *TfidfVectorizer* performed that tasks by even providing techniques that counts the frequency of word in a message. Hence, this model plays a pivotal role for each dataset:

```
#function to automate the function Tfidf
def featureExtractionModel(language):
return TfidfVectorizer(min_df=1, stop_words=language, lowercase =
    True)

#Convertion for englishs part
featureExtractionEn = featureExtractionModel('english')

#Prevent stops and configure it for french
from spacy.lang.fr.stop_words import STOP_WORDS
featureExtractionFr = featureExtractionModel(list(STOP_WORDS))

#For swahili
featureExtractionSw = featureExtractionModel(None)
```

After created the features extractions model for each language, now the transformation of test values were worthy to be done, as follows :

```
x_test_features_en = featureExtractionEn.transform(x_test_en)
x_test_features_fr = featureExtractionFr.transform(x_test_fr)
x_test_features_sw = featureExtractionSw.transform(x_test_sw)
```

- Initialize models: These are certain models used classify:

```
 #Naive Bayes models
 BayesModelEn=MultinomialNB()
 BayesModelFr=MultinomialNB()
 BayesModelSw=MultinomialNB()

```

```
 #SVM models
 SvmModelEn = SVC(probability=True)
 SvmModelFr = SVC(probability=True)
 SvmModelSw = SVC(probability=True)

 #Logistic regression models
 LogRegModelEn= LogisticRegression()
 LogRegModelFr= LogisticRegression()
 LogRegModelSw= LogisticRegression()

 #DecisionTree models
 DecisionTreeEn = DecisionTreeClassifier()
 DecisionTreeFr = DecisionTreeClassifier()
 DecisionTreeSw = DecisionTreeClassifier()
```

- Predictions based on each language model:

```
#Function to fit all the models
def modelReceiveFitting(model,x_train_features, y_train):
 fitmodel = model.fit(x_train_features, y_train)
 return fitmodel
```

Fitting each models as follows :

```
#For english languages
BayesModelEn=modelReceiveFitting(BayesModelEn,x_train_features_en,
   y_train_en)
SvmModelEn=modelReceiveFitting(SvmModelEn,x_train_features_en,
   y_train_en)
LogRegModelEn=modelReceiveFitting(LogRegModelEn,x_train_features_en
   ,y_train_en)
DecisionTreeEn=modelReceiveFitting(DecisionTreeEn,
   x_train_features_en,y_train_en)

#For french languages
BayesModelFr=modelReceiveFitting(BayesModelFr,x_train_features_fr,
   y_train_fr)
SvmModelFr=modelReceiveFitting(SvmModelFr,x_train_features_fr,
   y_train_fr)
LogRegModelFr=modelReceiveFitting(LogRegModelFr,x_train_features_fr
   ,y_train_fr)
DecisionTreeFr=modelReceiveFitting(DecisionTreeFr,
   x_train_features_fr,y_train_fr)

#For swhahili languages
BayesModelSw=modelReceiveFitting(BayesModelSw,x_train_features_sw,
   y_train_sw)
SvmModelSw=modelReceiveFitting(SvmModelSw,x_train_features_sw,
   y_train_sw)
LogRegModelSw=modelReceiveFitting(LogRegModelSw,x_train_features_sw
   ,y_train_sw)
DecisionTreSw=modelReceiveFitting(DecisionTreeSw,
   x_train_features_sw,y_train_sw)
```

### 3.4.4 Model Evaluation

Once the model was fitted, it was time for evaluating the precision those models by two indicators which are : accuracy and ROC metrics. Therefore, this function performs that task for every model leveraged, and it was technically made as follows:

```
def makeRocVisualize(bayes, svm, lg,decision,x_feature, y_test,
   language):
        models = [
        (bayes, "Bayes Model"),
        (svm, "SVM Model"),
        (lg, "Logistic Regression Model"),
        (decision, "Decision Tree Model"),
        ]

        plt.figure(figsize=(8, 6))

        for model, model_name in models:
        y_scores = model.predict_proba(x_feature)[:, 1]
        # Probability estimates for the positive class
        fpr, tpr, _ = roc_curve(y_test, y_scores)
        auc = roc_auc_score(y_test, y_scores)

        print(model_name + " Accuracy score is :", accuracy_score(
           y_test, model.predict(x_feature)))
    print(model_name + " Area under the curve for ROC is :", auc)


        plt.plot(fpr, tpr, label=f'{model_name} (AUC = {auc:.3f})')

        plt.plot([0, 1], [0, 1], 'k--', linewidth=2)  # Random
           classifier line
        plt.xlim([0.0, 1.0])
        plt.ylim([0.0, 1.05])
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.title('ROC Metrics'+language)
        plt.legend(loc='lower right')
        plt.show()
        }
```

Using the function for each dataset chosen we got the following result :
(a) English model's score:

```
In [343]: makeRocVisualize(BayesModelEn,SvmModelEn,LogRegModelEn,
   DecisionTreeEn,x_test_features_en,y_test_en, "(English Data)")

#results
Bayes Model Accuracy score is : 0.9641472868217055
Bayes Model Area under the curve for ROC is : 0.9893010232300885
SVM Model Accuracy score is : 0.9728682170542635
SVM Model Area under the curve for ROC is : 0.9972863661504424
Logistic Regression Model Accuracy score is : 0.9573643410852714
```

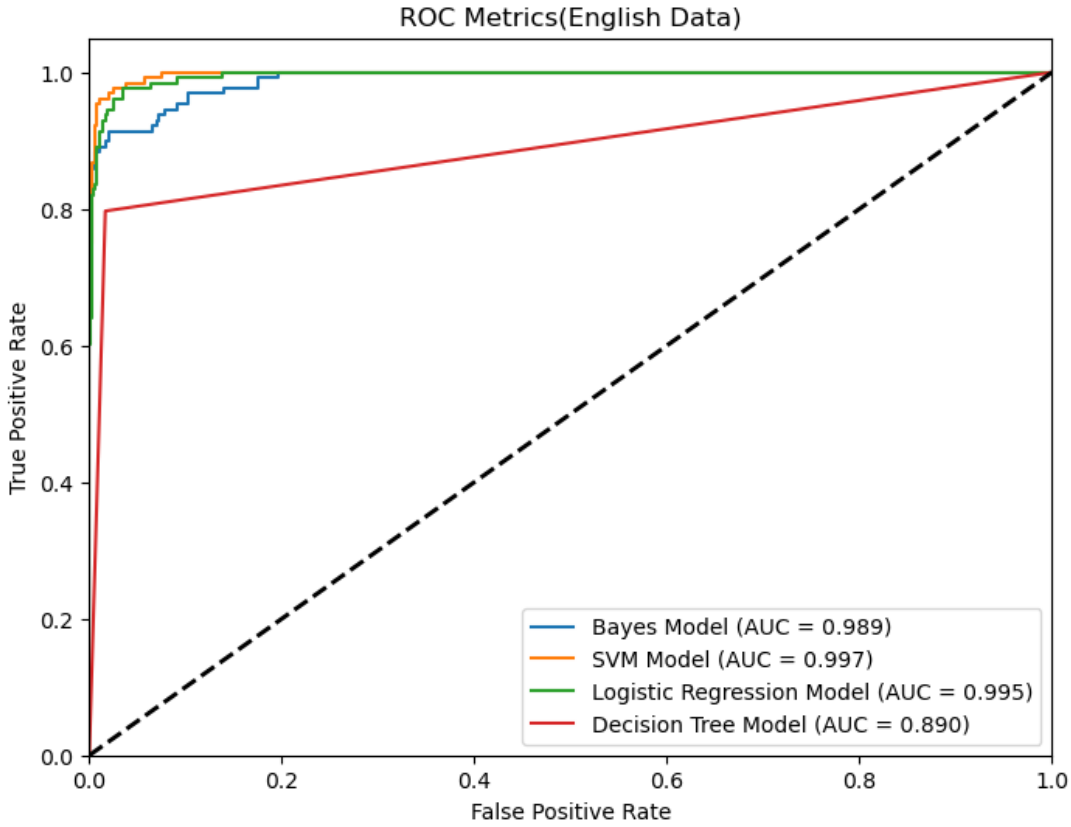



**Figure 3.4:** Comparing models's AUC score based on English messages

```
Logistic Regression Model Area under the curve for ROC is :
    0.9949875553097345
Decision Tree Model Accuracy score is : 0.9602713178294574
Decision Tree Model Area under the curve for ROC is :
    0.8901410398230089
```

Actually, SVM realizes the best score for accuracy metric (97.2%) and Bayes (98.9%). Thus, the positives are well-learned as ROC graphic shows it in figure3.4.

(b) Swahili model's score:

```
In [345]: makeRocVisualize(BayesModelSw,SvmModelSw,LogRegModelSw,
    DecisionTreeSw,x_test_features_sw,y_test_sw,"(Swahili Data)")

#results
Bayes Model Accuracy score is : 0.8695652173913043
Bayes Model Area under the curve for ROC is : 0.8999999999999999
SVM Model Accuracy score is : 0.8695652173913043
SVM Model Area under the curve for ROC is : 0.8666666666666667
Logistic Regression Model Accuracy score is : 0.8695652173913043
Logistic Regression Model Area under the curve for ROC is : 0.85
Decision Tree Model Accuracy score is : 0.9130434782608695
Decision Tree Model Area under the curve for ROC is :
    0.6666666666666666
```

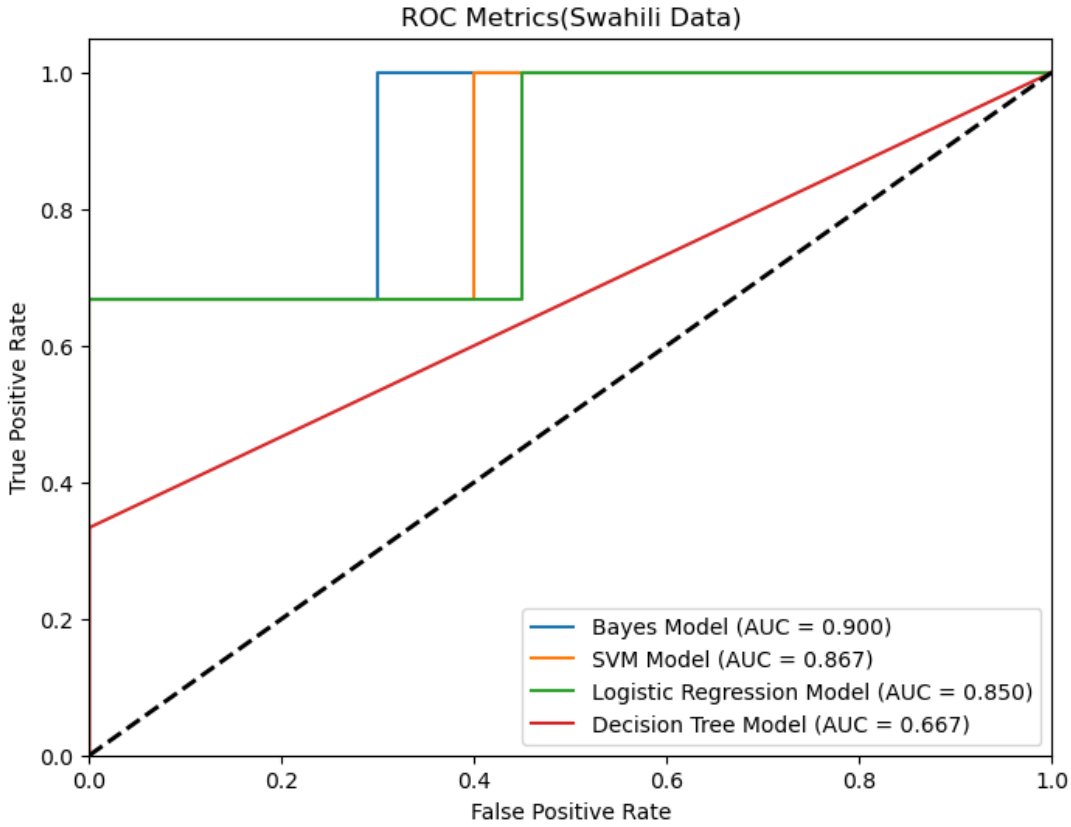



**Figure 3.5:** Comparing models's AUC score based on Swahili messages

For the case of Swahili messages, Decision Tree is the learning model with high score for accuracy (91.3%) and Bayes for AUC (90%) (as even shown in the figure 3.5).

(c) French model's score :

```
In [355]: makeRocVisualize(BayesModelFr,SvmModelFr,LogRegModelFr,
   DecisionTreeFr,x_test_features_fr,y_test_fr,"(French Data)")

#results
Bayes Model Accuracy score is : 0.9629990262901655
Bayes Model Area under the curve for ROC is : 0.9922135706340378
SVM Model Accuracy score is : 0.9776046738072055
SVM Model Area under the curve for ROC is : 0.994794563403782
Logistic Regression Model Accuracy score is : 0.9659201557935735
Logistic Regression Model Area under the curve for ROC is :
   0.9910056312569522
Decision Tree Model Accuracy score is : 0.9591041869522883
Decision Tree Model Area under the curve for ROC is :
   0.9096391824249166
```

Models evaluated on French messages show that SVM is best predictor based on accuracy metric (97.2%). In terms of AUC all represented models give 99% except the Decision Tree Model as shown in ROC (fig. 3.6) graphic.

The all results models being combined show the better one according to the language

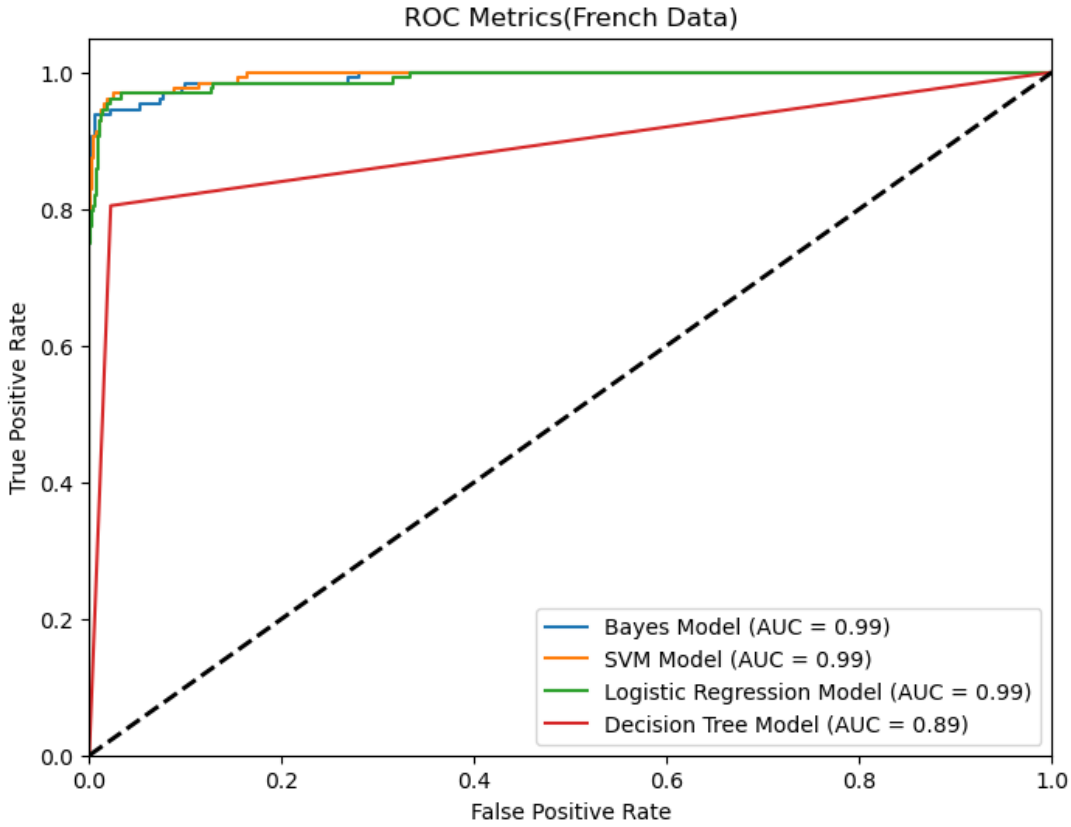



**Figure 3.6:** Comparing model's AUC score based on French messages

data. The prompt view is shown in the table and in the figure **??**

| Languages | Metrics | Naive | SVM | LR | DT |
|---|---|---|---|---|---|
| Swahili | Accuracy | 86.96% | 86.96% | 86.96% | 91.30% |
| | AUC | 90.00% | 86.67% | 85.00% | 66.67% |
| French | Accuracy | 96.30% | 97.76% | 96.59% | 90.96% |
| | AUC | 99.22% | 99.48% | 99.10% | 90.96% |
| English | Accuracy | 96.41% | 97.29% | 95.74% | 96.03% |
| | AUC | 98.93% | 99.73% | 99.50% | 89.01% |

**Table 3.3:** Models score (Accuracy vs AUC)

### 3.4.5 Optimization with Ensemble models

Actually, the graphic results (fig.3.7) shows that the every models have its characteristics allowing to be convenient to certain data, thus combining all for more optimization can be appealing by taking care of over-fitting. Therefore, two means are used :

(a) *VotingClassifier* :

- English messages :

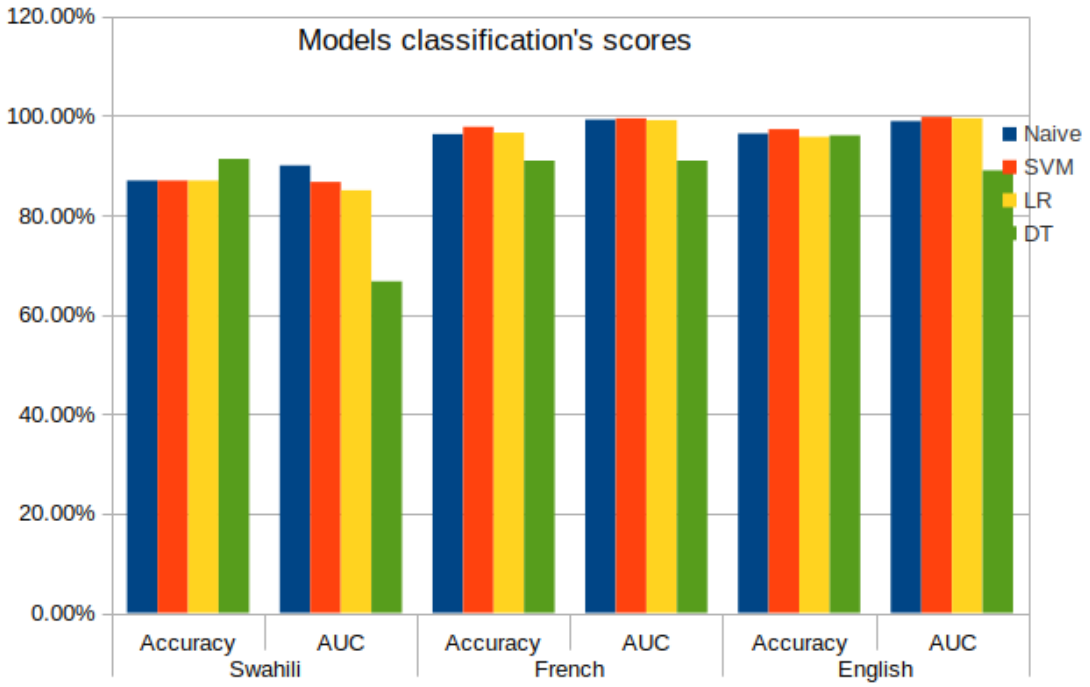



**Figure 3.7:** Models classification's scores

```
In [118]: EnsembleModelVotingEn=
   VotingClassifier(estimators=[('Bayes', BayesModelSw), ('SVM',
   SvmModelSw), ('LR', LogRegModelSw),('DecisionTreSw',
   DecisionTreSw)], voting='soft')

In [119]:EnsembleModelVotingEn.fit(x_train_features_en, y_train_en)
In [124]:print(accuracy_score(y_test_en, EnsembleModelVotingEn.
   predict(x_test_features_en)))
#Output:
0.9796511627906976
```

- French messages :

```
In [121]: EnsembleModelVotingFr = VotingClassifier(estimators=[('
   Bayes', BayesModelFr), ('SVM', SvmModelFr), ('LR', LogRegModelFr
   ),('DecisionTreSw',DecisionTreeFr)], voting='soft')

In [122]:EnsembleModelVotingFr.fit(x_train_features_fr, y_train_fr)
In [124]:print(accuracy_score(y_test_fr, EnsembleModelVotingFr.
   predict(x_test_features_fr)))
#Output:
0.9805258033106135
```

- Swahili messages :

```
In [124]: EnsembleModelVotingSw = VotingClassifier(estimators=[('
   Bayes', BayesModelSw), ('SVM', SvmModelSw), ('LR', LogRegModelSw
   ),('DecisionTreSw',DecisionTreeSw)], voting='soft')
In [125]:EnsembleModelVotingSw.fit(x_train_features_sw, y_train_sw)
```

```
In [126]:print(accuracy_score(y_test_sw, EnsembleModelVotingSw.
   predict(x_test_features_sw)))
#Output:
0.8695652173913043
```

(b) Using boost Method (GridSearch) Actually, by summarizing the scores metrics achieved are equals in terms of application, much more for the Swahili messages which lead to lower result than others. The main reason of that, is the a few number of messages comparing to others. Thus, by *gridsearch*, we tested if the accuracy can not once more increases.

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV

gb_classifier = GradientBoostingClassifier()

param_grid = {
        'n_estimators': [10, 50, 100],  # Number of boosting stages
            to be used
        'learning_rate': [0.01, 0.1, 0.2],  # Step size shrinking
           to prevent overfitting
        'max_depth': [3, 4, 5]  # Maximum depth of individual trees
        }

grid_search = GridSearchCV(gb_classifier, param_grid, cv=5, scoring
   ='accuracy')
```

```
In [131]: grid_search.fit(x_train_features_sw, y_train_sw)

#get the best parameters
In [132]: best_gb_classifier = grid_search.best_estimator_
best_gb_classifier
Out [132]:
        GradientBoostingClassifier(learning_rate=0.01)

##best gb model:
In [380]: best_gb_classifier = grid_search.best_estimator_
     y_test_en_pred = best_gb_classifier.predict(x_test_features_sw
        )
```

```
        y_test_en_pred = best_gb_classifier.predict(
           x_test_features_sw)

        accuracy = accuracy_score(y_test_sw, y_test_en_pred)
        print("Accuracy:", accuracy)

        #output :
        Accuracy: 0.8695652173913043
```

Great! The score is the same like the one yielded by the *votingClassifier*

### 3.4.6 Model Deployment

Concerning the current work, at this stage we converted the models into a files usable in production area. We considered all voting models are for each language because the scores they yielded, and dumped them in files.

#### Production With Joblib

```
import joblib
#Models for preprocessing
joblib.dump(featureExtractionEn, 'EnglishFeatureModel.pkl')
joblib.dump(featureExtractionFr, 'FrenchFeatureModel.pkl')
joblib.dump(featureExtractionSw, 'SwahiliFeatureModel.pkl')

#Models predictions
joblib.dump(EnsembleModelVotingEn, 'EnglishModelPrediction.pkl')
joblib.dump(EnsembleModelVotingFr, 'FrenchModelPrediction.pkl')
joblib.dump(EnsembleModelVotingSw, 'SwahiliModelPrediction.pkl')
```

We have to convert any input into numeric by The *featureExtraction* model from *TdifVectorizer* and then

## 3.5 Presentation and Discussion of Results

For sure, this step is the next party of the previous, we just show how the model generated during the deployment stage is integrated in a software for really performing it tasks. As basics, the following step delves into precepts of interaction.

### 3.5.1 Predicting system

Performing predictions which taking account of languages, required this project to use a Python detect_langs library. Hence, the following function allows us to judge if the input message should be considered as a French, English or Swahili text every time it doesn't match one those languages, it automatically considered as an Swahili text.

```
from langdetect import detect_langs
def detectLanguage(text) :
        result = detect_langs(text)
        for language in result:
        if language.lang == 'fr':
        return 'fr'
        elif language.lang == 'en':
        return 'en'
        else:
        return 'sw'
```

Next, the following function named *predictMessageState* performs the prediction task and returns the appealing answer including the numeric value and the probability behind that prediction. Indeed, for the numeric value understood as a boolean, every time it is 0, it means that the message is ham, when it is 1 it stands for that the message is a spam.

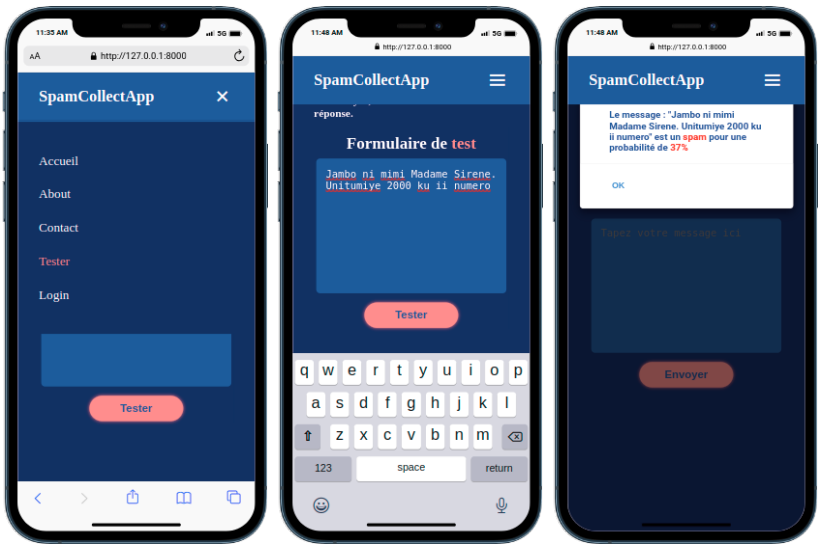
```
def predictMessageState(input_text):
        if detectLanguage(input_text)=='en':
        input_vector = featureExtractionEn.transform([input_text])
        prediction = EnsembleModelVotingEn.predict(input_vector)[0]
        predict_proba = EnsembleModelVotingEn.predict_proba(
           input_vector)[0]
        if detectLanguage(input_text)=='fr':
        input_vector = featureExtractionFr.transform([input_text])
        prediction = EnsembleModelVotingFr.predict(input_vector)[0]
        predict_proba = EnsembleModelVotingFr.predict_proba(
           input_vector)[0]
        else:
        input_vector = featureExtractionSw.transform([input_text])
        prediction = best_gb_classifier.predict(input_vector)[0]
        predict_proba = best_gb_classifier.predict_proba(
           input_vector)[0]

        return prediction, predict_proba
```

Then, the present snipped code, is the main or starter of the operations:

```
input_text = input("Enter your mail here: ")
predictMessageState(input_text)
##results
Enter your mail here: jambo kaka
(0, array([0.62955115, 0.37044885]))
```

### 3.5.2 Model interaction in a Platform for detection

Just as the user was completing the form for submitting messages that look like or are spams, at this stage. When he receives an message in his phones transmitted by a mobile network operator, he has to check its legitimacy by copping it and submitting through the following steps :




(a) (b) (c)

At the first stage (a), the user [38] has to click on the test link, then completing the form and sending it at (b), finally waiting for the response of the server.

### 3.5.3 Model's system with APIs

For network operators or other software developers who would like to integrate the model functionalities or networks operators, they have to follow the link delivered end-point end-point. The can consume the services as insomnia can do while testing: As the image

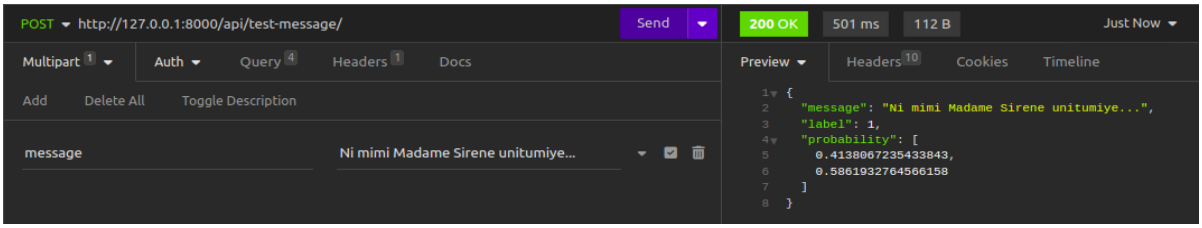



**Figure 3.9:** Testing the end-point for api Services

shows, the answer is 1 meaning that the message is spam based on the probability of 58%

## 3.6 Theoretical and Practical Contributions

Creating a model capable of handling situations involving panic, disturbance, and financial loss can be a formidable challenge. This work serves as an assistant for mobile phone users, helping them assess the legitimacy of received messages and make informed decisions. Consequently, individuals who have experienced significant losses through various means now have a valuable resource.

Moreover, network operators who record uncontrolled financial losses and customer defections can enhance their support systems by incorporating a feature that provides information on numbers involved in such activities. Alongside their existing techniques, this model can be integrated into their message transmission systems to filter processed messages for further investigation and enhance overall security.

Additionally, this solution offers a valuable addition for other developers utilizing the messaging system. By obtaining the necessary credentials, they can integrate this solution into their services, aiming to provide modern and high-quality service delivery.

## 3.7 Study Limitations and Future research paths

Although this work comes with several advantages, it also has its limitations. The solution it offers isn't universal, as it's primarily tailored to a specific region, especially the eastern part of the Democratic Republic of Congo.

Furthermore, a portion data, which is a fundamental component of any machine learning algorithms, was collected from an unofficial source, namely *Kaggle*. This data doesn't consider the local interests and language cultures, as it predominantly consists of content in French and English. Additionally, the other data sources were limited in quantity,

making it challenging to provide accurate estimations that represent the entire population.

Moreover, the solution is designed to work with only three languages. This means that messages containing a mixture of languages might pose challenges for the model in terms of accurate classification. There is a need for a more advanced multilingual model classifier to address this issue for instance by utilizing advanced techniques in artificial intelligence.

## 3.8 Summary

This chapter provides practical insights into the methodology. It focuses on the practical applications of this methodology within machine learning systems. It outlines the process of data collection, including the sources of data, the providers, and the methods used to acquire the data necessary to initiate the machine learning model.

Additionally, it offers various techniques and code snippets to aid in data analysis, ultimately leading to the development of a functional model that can be deployed effectively.

Furthermore, this chapter explores the potential impact of the project on individuals, software developers, and network operators, offering practical insights into how it can be integrated into their daily work and life.

In conclusion, it highlights the project's strengths and limitations and invites other researchers to further investigate and develop highly accurate spam prediction models.

# General Conclusion

This study primarily focused on the detection of spam messages within network operator environments. Throughout its course, it highlighted the numerous issues that mobile phones users encounter. Thus, they are occasionally vulnerable to threats, disturbances, and monetary losses, usually caused by scammers exploiting the messaging system or mobile devices themselves.

To address these problems, this project proposes potential solutions. It not only educates users on how to assess and identify malicious messages independently but also introduces a predictive model capable of assisting users in the fight against illegitimate messages. Obviously, this work emphasizes that users also have a crucial role to play in this endeavor.

Implementing these solutions requires the use of specific techniques and methods that align with professional standards. This involves using programming languages such as HTML, CSS, and JavaScript to develop user-friendly interfaces. For the development and management of machine learning models and message processing, Python and its libraries (including Pandas, NumPy, Matplotlib, and Scikit-Learn) were employed.

Data, often considered the heart of the model, received special attention. They underwent thorough static analysis, preparation, and visualization to ensure a clear understanding and prevent biased results. Once the data were ready, they were used to train machine learning models, employing various algorithms such as Naive Bayes, Logistic Regression, Support Vector Machine, Decision Trees, and ensemble models. These models were trained on datasets in Swahili, French, and English. After training, the models were integrated into a production environment following their deployment.

In terms of practical solution, the current work introduces a system that offers a server interaction system with APIs for software developers who wish to incorporate the model's capabilities into their applications. It also addresses the needs of network operators by enhancing monitoring and surveillance tasks.

However, the project has its limitations. The quantity of Swahili messages was limited, which could potentially lead to a biased model. Additionally, gaining access to network operator messages can be a challenging process due to legal procedures. Furthermore, the model only supports three languages, while the real-world scenario involves more diversity in language use. Messages often contain mixed languages, and some threats target system architecture rather than message content. These concerns, along with others, provide ample opportunities for future research.

# Bibliography


[1] Marty Alchin. *Pro Django*. 2013.

[2] Peshawa Jamal Muhammad Ali, Rezhna Hassan Faraj, Erbil Koya, Peshawa J Muhammad Ali, and Rezhna H Faraj. Data normalization and standardization: a technical report. *Mach Learn Tech Rep*, 1(1):1–6, 2014.

[3] Hamzah A Alsayadi, Nima Khodadadi, Sunil Kumar, et al. Improving the regression of communities and crime using ensemble of machine learning models. *Journal of Artificial Intelligence and Metaheuristics*, 1(1):27–7, 2022.

[4] Iosif Androulidakis, Vasileios Vlachos, and Alexandros Papanikolaou. Fimess: filtering mobile external sms spam. In *Proceedings of the 6th Balkan Conference in Informatics*, pages 221–227, 2013.

[5] Paul Kimumwe Lillian Nalwoga Juliet Nanfuka Edrine Wanyama Wairagala Wakabi PhD Ashnah Kalemera, Victor Kapiyo. State of internet freedom democratic republic of the congo 2019. *Mapping Trends in Government Internet Controls, 1999-2019*, 2020.

[6] Qiang Bai, Shaobo Li, Jing Yang, Qisong Song, Zhiang Li, and Xingxing Zhang. Object detection recognition and robot grasping based on machine learning: A survey. *IEEE access*, 8:181855–181879, 2020.

[7] Allan Bluman. *Elementary statistics: A step by step approach*. McGraw-Hill Education, 2017.

[8] Ernest Yeboah Boateng, Joseph Otoo, and Daniel A Abaye. Basic tenets of classification algorithms k-nearest-neighbor, support vector machine, random forest and neural network: a review. *Journal of Data Analysis and Information Processing*, 8(4):341–357, 2020.

[9] Andrey Bogdanchikov, Meirambek Zhaparov, and Rassim Suliyev. Python to learn programming. In *Journal of Physics: Conference Series*, volume 423, page 012027. IOP Publishing, 2013.

[10] Jeff Brown, Bill Shipman, and Ron Vetter. Sms: The short message service. *Computer*, 40(12):106–110, 2007.

[11] Erik Cambria and Bebo White. Jumping nlp curves: A review of natural language processing research. *IEEE Computational intelligence magazine*, 9(2):48–57, 2014.

[12] Bahzad Charbuty and Adnan Abdulazeez. Classification based on decision tree algorithm for machine learning. *Journal of Applied Science and Technology Trends*, 2(01):20–28, 2021.

[13] Anamika Chauhan et al. Detection of lung cancer using machine learning techniques based on routine blood indices. In *2020 IEEE international conference for innovation in technology (INOCON)*, pages 1–6. IEEE, 2020.

[14] Guangquan Chen, Weijun Wang, and Xuan Zhou. A survey on sms spam filtering techniques. *Journal of Network and Computer Applications*, 80:149–159, 2017.

[15] Catalin Cimpanu. Simjacker vulnerability exploited for surveillance by at least one nation-state. *ZDNet*, 2019.

[16] Michael Crawford, Taghi M Khoshgoftaar, Joseph D Prusa, Aaron N Richter, and Hamzah Al Najada. Survey of review spam detection using machine learning techniques. *Journal of Big Data*, 2(1):1–24, 2015.

[17] John W Creswell. *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, 2014.

[18] Prof. Dantu's CSE. Cellular network basics. In *ADV. REAL-WORLD NETWORKS*, volume 14-760. University of North Texas, 2018.

[19] Pádraig Cunningham. Dimension reduction. In *Machine learning techniques for multimedia: Case studies on organization and retrieval*, pages 91–112. Springer, 2008.

[20] Julian David. Designing machine learning systems with python. *Community Experience Distilled*, pages 381–386, April 2016.

[21] MONUSCO DRC. Protect, stabilize, consolidate peace, monusco's report. *United Nations Organisation Stabilization Mission in the Democratic Republic of Congo*, pages 1–4, 2015.

[22] Abdullah Elen and Emre Avuçlu. Standardized variable distances: A distance-based machine learning method. *Applied Soft Computing*, 98:106855, 2021.

[23] Cori Faklaris and Sara Anne Hook. Oh, snap! the state of electronic discovery amid the rise of snapchat, whatsapp, kik, and other mobile messaging apps. 2016.

[24] Johann Faouzi and Hicham Janati. pyts: A python package for time series classification. *The Journal of Machine Learning Research*, 21(1):1720–1725, 2020.

[25] Tim Furche, Georg Gottlob, Leonid Libkin, Giorgio Orsi, and Norman W Paton. Data wrangling for big data: Challenges and opportunities. In *EDBT*, volume 16, pages 473–478, 2016.

[26] Antonio Ghezzi, Marcelo Nogueira Cortimiglia, and Alejandro Germán Frank. Strategy and business model design in dynamic telecommunications industries: A study on italian mobile network operators. *Technological Forecasting and Social Change*, 90:346–354, 2015.

[27] Suparna Das Gupta, Soumyabrata Saha, and Suman Kumar Das. Sms spam detection using machine learning. In *Journal of Physics: Conference Series*, volume 1797, page 012017. IOP Publishing, 2021.

[28] Kennedy Ochilo Hadullo and DM Getuno. Machine learning software architecture and model workflow. a case of django rest framework. *American Journal of Applied Sciences*, 18(1):152–164, 2021.

[29] Shuang Hao, Nadeem Ahmed Syed, Nick Feamster, Alexander G Gray, and Sven Krasser. Detecting spammers with snare: Spatio-temporal network-level automatic reputation engine. In *USENIX security symposium*, volume 9, 2009.

[30] Marko Hassinen and Smile Markovski. Secure sms messaging using quasigroup encryption and java sms api. *SPLST*, 3:187, 2003.

[31] Elad Hazan and Tomer Koren. Optimal algorithms for ridge and lasso regression with partially observed attributes. *arXiv preprint arXiv:1108.4559*, 2011.

[32] Sunil Kumar Jangir, Manoj Kumar Sharma, and Pawan Kumar Gupta. Design and implementation of sms gateway api for mobile communication networks. *International Journal of Computer Applications*, 151(9):1–5, 2016.

[33] Christian Janiesch, Patrick Zschech, and Kai Heinrich. Machine learning and deep learning. *Electronic Markets*, 31(3):685–695, 2021.

[34] Uthayakumar Jayasankar, Vengattaraman Thirumal, and Dhavachelvan Ponnurangam. A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. *Journal of King Saud University-Computer and Information Sciences*, 33(2):119–140, 2021.

[35] Fengming Jiao, Jiao Song, Xin Zhao, Ping Zhao, and Ru Wang. A spoken english teaching system based on speech recognition and machine learning. *International Journal of Emerging Technologies in Learning (iJET)*, 16(14):68–82, 2021.

[36] Tim K Johnsen and Jerry Z Gao. Elastic net to forecast covid-19 cases. In *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*, pages 1–6. IEEE, 2020.

[37] Ledisi G Kabari and Ugochukwu C Onwuka. Comparison of bagging and voting ensemble machine learning algorithm as a classifier. *International Journals of Advanced Research in Computer Science and Software Engineering*, 9(3):19–23, 2019.

[38] Ledisi G Kabari and Ugochukwu C Onwuka. Comparison of bagging and voting ensemble machine learning algorithm as a classifier. *International Journals of Advanced Research in Computer Science and Software Engineering*, 9(3):19–23, 2019.

[39] Thomas R Karl, Claude N Williams Jr, Pamela J Young, and Wayne M Wendland. A model to estimate the time of observation bias associated with monthly mean maximum, minimum and mean temperatures for the united states. *Journal of Applied Meteorology and Climatology*, 25(2):145–160, 1986.

[40] Veena K Katankar and VM Thakare. Short message service using sms gateway. *International Journal on Computer Science and Engineering*, 2(04):1487–1491, 2010.

[41] Memoona Khanum, Tahira Mahboob, Warda Imtiaz, Humaraia Abdul Ghafoor, and Rabeea Sehar. A survey on unsupervised machine learning algorithms for automation, classification and maintenance. *International Journal of Computer Applications*, 119(13), 2015.

[42] Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. Machine learning operations (mlops): Overview, definition, and architecture. *IEEE Access*, 2023.

[43] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern recognition letters*, 42:11–24, 2014.

[44] M Lavanya and KR Aruna. Sms spam detection using deep learning. *Journal homepage: www. ijrpr. com ISSN*, 2582:7421.

[45] Gwenael Le Bodic. *Mobile messaging technologies and services: SMS, EMS and MMS*. John Wiley & Sons, 2005.

[46] Gwenaël Le Bodic and Hatim Zaghloul. *Mobile Messaging Technologies and Services: SMS, EMS, and MMS*. ABC Publishing, 2022.

[47] Matti Leppäniemi and Heikki Karjaluoto. Mobile marketing: From marketing strategy to mobile marketing campaign implementation. *International Journal of Mobile Marketing*, 3(1), 2008.

[48] Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

[49] Weiwei Lin, Ziming Wu, Longxin Lin, Angzhan Wen, and Jin Li. An ensemble random forest algorithm for insurance big data analysis. *Ieee access*, 5:16568–16575, 2017.

[50] Qiong Liu, Stephen Levinson, Ying Wu, and Thomas Huang. Interactive and incremental learning via a mixture of supervised and unsupervised learning strategies. In *Proceedings of the Fifth Joint Conference on Information Sciences*, volume 1, pages 555–558, 2000.

[51] FATIMA AIT MAHAMMED. Approches d'apprentissage automatique pour la dÉtection du spam web : Exploration de diverses caractÉristiques. *Computational Linguistics and Intelligent Systems*, pages 85–90, 2018.

[52] JR Maria Navin and R Pankaja. Performance analysis of text classification algorithms using confusion matrix. *International Journal of Engineering and Technical Research (IJETR)*, 6(4):75–8, 2016.

[53] Dastan Maulud and Adnan M Abdulazeez. A review on linear regression comprehensive in machine learning. *Journal of Applied Science and Technology Trends*, 1(4):140–147, 2020.

[54] Wes McKinney and PD Team. Pandas-powerful python data analysis toolkit. *Pandas—Powerful Python Data Analysis Toolkit*, 1625, 2015.

[55] Uttam Kumar Roy Jubayer AI Mahmud Md Shofiqul Islam, Shanjida Sultana. A review on video classification with methods, findings, performance, challenges, limitatios and future work. 2020.

[56] A Medani, Abdullah Gani, O Zakaria, AA Zaidan, and BB Zaidan. Review of mobile short message service security issues and techniques towards the solution. *Scientific Research and Essays*, 6(6):1147–1165, 2011.

[57] Ajay R Mishra. *Advanced cellular network planning and optimisation: 2G/2.5 G/3G... evolution to 4G*. John Wiley & Sons, 2007.

[58] Anuar Manuel Nader Meljem. Django rest framework (drf) secure code guidelines. 2023.

[59] Pavas Navaney, Gaurav Dubey, and Ajay Rana. Sms spam filtering using supervised machine learning algorithms. In *2018 8th international conference on cloud computing, data science & engineering (confluence)*, pages 43–48. IEEE, 2018.

[60] Swathi Nayak, Manisha Bhat, NV Subba Reddy, and B Ashwath Rao. Study of distance metrics on k-nearest neighbor algorithm for star categorization. In *Journal of Physics: Conference Series*, volume 2161, page 012004. IOP Publishing, 2022.

[61] FY Osisanwo, JET Akinsola, O Awodele, JO Hinmikaiye, O Olakanmi, J Akinjobi, et al. Supervised machine learning algorithms: classification and comparison. *International Journal of Computer Trends and Technology (IJCTT)*, 48(3):128–138, 2017.

[62] Statista's own team of researchers and analysts. *Number of mobile messages worldwide from 2019 to 2023 (in trillions)*. https://fr.statista.com/, 2020.

[63] Chris Park. *A Dictionary of Environment and Conservation*. Oxford University Press, 1 edition, 2012. Online Publication.

[64] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[65] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[66] M Praveena and V Jaiganesh. A literature review on supervised machine learning algorithms and boosting process. *International Journal of Computer Applications*, 169(8):32–35, 2017.

[67] Jipeng Qiang, Zhenyu Qian, Yun Li, Yunhao Yuan, and Xindong Wu. Short text topic modeling techniques, applications, and performance: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(3):1427–1445, 2020.

[68] Sebastian Raschka. Model evaluation, model selection, and algorithm selection in machine learning. *arXiv preprint arXiv:1811.12808*, 2018.

[69] Sebastian Raschka and Vahid Mirjalili. Python machine learning: Machine learning and deep learning with python. *Scikit-Learn, and TensorFlow. Second edition ed*, 3, 2017.

[70] Maria Razno. Machine learning text classification model with nlp approach. *Computational Linguistics and Intelligent Systems*, 2:71–73, 2019.

[71] Tomasz Rymarczyk, Edward Kozłowski, Grzegorz Kłosowski, and Konrad Niderla. Logistic regression for machine learning in process tomography. *Sensors*, 19(15):3400, 2019.

[72] Muhammad Abdulhamid Shafi'I, Muhammad Shafie Abd Latiff, Haruna Chiroma, Oluwafemi Osho, Gaddafi Abdul-Salaam, Adamu I Abubakar, and Tutut Herawan. A review on mobile sms spam filtering techniques. *IEEE Access*, 5:15650–15666, 2017.

[73] Houshmand Shirani-Mehr. Sms spam detection using machine learning approach. *unpublished) http://cs229. stanford. edu/proj2013/Shir aniMeh r-SMSSpamDetectionUsingMachineLearningApproach. pdf*, 2013.

[74] Ali Hassan Sial, Syed Yahya Shah Rashdi, and Abdul Hafeez Khan. Comparative analysis of data visualization libraries matplotlib and seaborn in python. *International Journal*, 10(1), 2021.

[75] Pramod Singh. Deploy machine learning models to production. *Cham, Switzerland: Springer*, 2021.

[76] De S Sirisuriya et al. A comparative study on web scraping. 2015.

[77] Alex Smola and S.V.N. Vishwanathan. *Introduction to Machine Learning*. Cambridge University Press, Cambridge, UK, 2010.

[78] Carlos Oscar Sánchez Sorzano, Javier Vargas, and A Pascual Montano. A survey of dimensionality reduction techniques. *arXiv preprint arXiv:1403.2877*, 2014.

[79] Jonathan Stark, Paco Nathan, John Papaconstantinou, Paco Lagerstrom, and Paco Hope. *Building Android apps with HTML, CSS, and JavaScript*. " O'Reilly Media, Inc.", 2010.

[80] Siyuan Tang, Xianghang Mi, Ying Li, XiaoFeng Wang, and Kai Chen. Clues in tweets: Twitter-guided discovery and analysis of sms spam. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2751–2764, 2022.

[81] Aria Teimourzadeh, Samaneh Kakavand, and Benjamin Kakavand. Application of python in marketing education: a big data analytics perspective. *Marketing Education Review*, pages 1–16, 2022.

[82] Sandro Tosi. *Matplotlib for Python developers*. Packt Publishing Ltd, 2009.

[83] LV Tulchak and AO Mapchuk. *History of python*. PhD thesis, 2016.

[84] Antony Unwin. Why is data visualization important? what is important in data visualization? *Harvard Data Science Review*, 2(1):1, 2020.

[85] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30, 2011.

[86] H Wang, ZeZXeZBePJ Lei, X Zhang, B Zhou, and J Peng. Machine learning basics. *Deep learning*, pages 98–164, 2016.

[87] Qi Wang, Yue Ma, Kun Zhao, and Yingjie Tian. A comprehensive survey of loss functions in machine learning. *Annals of Data Science*, pages 1–26, 2020.

[88] Eric Wohlgethan. *Supporting web development decisions by comparing three major javascript frameworks: Angular, react and vue. js*. PhD thesis, Hochschule für Angewandte Wissenschaften Hamburg, 2018.

[89] Phd. Elie Zihindula. Courses of multivariate statistical analysis. part2. python et les données : Numpy et pandas. (23):2–10, 2023.

[90] Phd. Elie Zihindula. Courses of multivariate statistical analysis. part8. elements de classification. (22):18–20, 2023.

[91] Pacifique Zikomangane. A free wireless network in the drc: An answer to internet shutdowns and exorbitant access costs. 2018.

[92] Michel Lutz Éric Biernat. *Data science : fondamentaux et études de cas. Machine learning avec Python et R*. Eyrolles, 61, bd Saint-Germain 75240 Paris Cedex 05., 2015.

# Annexes

## .1 Collect message spams

In the website running , using the current link [2] sent on the a form where its possible to complete both the number of the spam message's sender and the message.The form receiving the data and sent it to the management system where they are stored in the database if they are valid. This is th illustrative image:


**Figure 10:** Collect data by the website's form

[2] Spam Collect app : http://spam.kivu-cs.org/


**Figure 11:** Download collected data

## .2 Code for the api interaction

```
from django.shortcuts import render
from django.http import HttpResponse
from django.http import JsonResponse
from django.contrib.auth import authenticate, login
from django.shortcuts import render, redirect
from django.core.paginator import Paginator
import csv
from django.contrib.auth import logout
from .forms import SpamMessageForm, contactForm, TestMessageForm
from .models import spamsMessage
from django.shortcuts import render
from django.core.paginator import Paginator
from django.http import JsonResponse
import json
from django.conf import settings
from langdetect import detect_langs
import joblib
import numpy as np

from rest_framework.decorators import api_view
from rest_framework.response import Response
from rest_framework import status
from .models import TestMessage  # Replace with your actual model
from .serializers import TestMessageSerializer  # Create a
    serializer for your model

from  django.core.mail import EmailMessage, send_mail,
    get_connection

EnsembleModelVotingEn = joblib.load("/home/christianresearcher/
    Documents/Courses/L2/Dissertation/spamDetectionApp/project/
    static/models/EnglishModelPrediction.pkl")
EnsembleModelVotingFr = joblib.load('/home/christianresearcher/
    Documents/Courses/L2/Dissertation/spamDetectionApp/project/
```

```
    static/models/FrenchModelPrediction.pkl')
EnsembleModelVotingSw = joblib.load('/home/christianresearcher/
    Documents/Courses/L2/Dissertation/spamDetectionApp/project/
    static/models/SwahiliModelPrediction.pkl')

featureExtractionEn = joblib.load("/home/christianresearcher/
    Documents/Courses/L2/Dissertation/spamDetectionApp/project/
    static/models/EnglishFeatureModel.pkl")
featureExtractionFr = joblib.load("/home/christianresearcher/
    Documents/Courses/L2/Dissertation/spamDetectionApp/project/
    static/models/FrenchFeatureModel.pkl")
featureExtractionSw = joblib.load("/home/christianresearcher/
    Documents/Courses/L2/Dissertation/spamDetectionApp/project/
    static/models/SwahiliFeatureModel.pkl")


def index_view(request):
        return render(request, "indexu.html")


def download_csv(request):
        response = HttpResponse(content_type="text/csv")
        response["Content-Disposition"] = 'attachment; filename="
            messages.csv"'

        writer = csv.writer(response)
        writer.writerow(["id", "Contact", "Message"])
        messages = spamsMessage.objects.all().order_by("createat")

        for message in messages:
        writer.writerow(
        [
        message.createat,
        message.contact,
        message.message,
        ]
        )
        return response

def statistic_views(request):
        query_set = spamsMessage.objects.all()
        content_query = request.GET.get("message")
        contact_query = request.GET.get("contact")
        if contact_query:
        query_set = query_set.filter(contact__icontains=
            content_query)
        if content_query:
        query_set = query_set.filter(message__icontains=
            content_query)
        paginator = Paginator(query_set, 5)  # paginate by 10
            messages per page
```

```
68          page_number = request.GET.get("page")
69          page_obj = paginator.get_page(page_number)
70          context = {
71          "messages": page_obj,
72          "contact": contact_query,
73          "message": content_query,
74          }
75          return render(request, "statistics.html", context)
76
77
78  def login_view(request):
79          if request.method == "POST":
80          username = request.POST.get("username")
81          password = request.POST.get("password")
82          if username is not None and password is not None:
83          user = authenticate(request, username=username, password=
                password)
84          if user is not None:
85          login(request, user)
86          return redirect("/")
87          error = "Invalid username or password."
88          return render(request, "loginu.html", {"error": error})
89          else:
90          return render(request, "loginu.html")
91
92  def logout_view(request):
93          logout(request)
94          return redirect("login")
95
96
97  @api_view(['POST'])
98  def api_test_message_view(request):
99          if request.method == "POST":
100         serializer = TestMessageSerializer(data=request.data)
101         if serializer.is_valid():
102         serializer.save()
103         message = request.data.get('message')
104         response, proba = predictMessageState(message)
105         answer = {
106         'message': message,
107         'label': int(response),
108         'probability': proba.tolist()
109         }
110         return Response(answer, status=status.HTTP_200_OK)
111         else:
112         return Response(serializer.errors, status=status.
                HTTP_400_BAD_REQUEST)
113         else:
114         return Response({'error': 'Only POST requests are allowed.'
                }, status=status.HTTP_405_METHOD_NOT_ALLOWED)
115
```

```
116
117
118 def testMessageViewCollect(request):
119         if request.method == "POST":
120         form = TestMessageForm(request.POST)
121         if form.is_valid():
122         form.save()
123         message =  request.POST.get('message')
124         response, proba  = predictMessageState(message)
125         answer={}
126         answer['message']= message
127         answer['label']= int(response)
128         answer['probability']= proba.tolist()
129
130         return JsonResponse({"message": answer})
131         else:
132         return JsonResponse({"message": "error"})
133         else:
134         return JsonResponse({"error": "Only POST requests are
              allowed."})
135
136 def testMessageView(request):
137         return render(request, "test.html")
138
139 def predictMessageState(input_text):
140         if detectLanguage(input_text)=='en':
141         input_vector = featureExtractionEn.transform([input_text])
142         prediction = EnsembleModelVotingEn.predict(input_vector)[0]
143         predict_proba = EnsembleModelVotingEn.predict_proba(
              input_vector)[0]
144         if detectLanguage(input_text)=='fr':
145         input_vector = featureExtractionFr.transform([input_text])
146         prediction = EnsembleModelVotingFr.predict(input_vector)[0]
147         predict_proba = EnsembleModelVotingFr.predict_proba(
              input_vector)[0]
148         else:
149         input_vector = featureExtractionSw.transform([input_text])
150         prediction = EnsembleModelVotingSw.predict(input_vector)[0]
151         predict_proba = EnsembleModelVotingSw.predict_proba(
              input_vector)[0]
152
153         return prediction, predict_proba
154
155 def detectLanguage(text) :
156         result = detect_langs(text)
157         for language in result:
158         if language.lang == 'fr':
159         return 'fr'
160         elif language.lang == 'en':
161         return 'en'
162         else:
```

```
163         return 'sw'
```

The serializers

```
from rest_framework import serializers
from .models import TestMessage

class TestMessageSerializer(serializers.ModelSerializer):
class Meta:
model = TestMessage
fields = ('message', 'createat')
```

Models

```
from django.db import models
from django.core.files import File
from io import BytesIO

class TestMessage(models.Model):
message = models.TextField(null=False)
createat = models.DateField(auto_now_add=True)
```