Case Study 3

Daniel Chang

DS 7333 | Quantifying the World

2023

**Introduction**

**Background**

## In an era dominated by constant digital communication, the inundation of messages and emails has become an ever-present challenge. Within this deluge, a significant proportion is composed of unwelcome and potentially harmful spam messages, a pervasive issue that not only disrupts productivity but also poses significant security risks. To address this pressing concern, we embark on a comprehensive research endeavor centered on Data Science, aiming to construct a robust and highly effective spam classification system.

## Our primary objective is to harness the capabilities of advanced algorithms and machine learning techniques to develop an intelligent system capable of distinguishing between authentic messages and spam. By automating this process, we seek to offer users a streamlined and secure communication experience, devoid of the interference caused by unwanted messages. This research project represents a critical step towards achieving this vision.

## The project unfolds through several vital stages, encompassing data collection, meticulous preprocessing, intricate feature engineering, rigorous model selection, and meticulous evaluation. Our dataset, carefully curated to reflect a diverse spectrum of message characteristics, will serve as the foundation upon which our classification model is trained and refined. Throughout this journey, we will delve into the realm of natural language processing (NLP) and machine learning, employing cutting-edge techniques to not only enhance the performance of our model but also enable it to adapt to the ever-evolving landscape of digital communication.

## The successful culmination of this project carries immense significance, transcending individual convenience to influence broader domains such as email filtering, cybersecurity, and information management. By mitigating the disruptive impact of spam, our research endeavors to foster a safer, more efficient, and productive digital communication environment, ultimately contributing to the greater advancement of our digitally interconnected world.

## Objective and Scope

## The scope of this project is defined by the ambitious objective of developing a highly accurate and adaptive predictive model for detecting spam messages within the realm of digital communication. Leveraging the Gaussian Naive Bayes (GNB) model alongside clustering techniques, our project encompasses a multifaceted approach to tackle the pervasive spam issue. Our efforts extend from the collection and comprehensive preprocessing of diverse message data to the intricate process of feature engineering and model selection. This multifaceted approach not only aims to enhance the accuracy of spam detection but also to ensure the model's adaptability to the ever-evolving landscape of digital communication.

## The core objective of this project is twofold. First and foremost, we seek to harness the power of machine learning, data analysis, and the Gaussian Naive Bayes model to create a predictive system that intelligently distinguishes between spam and legitimate messages. By doing so, we aim to provide users with a refined and secure communication experience, free from the intrusion of spam messages. Secondly, by incorporating clustering techniques, we endeavor to enhance the adaptability of our model, enabling it to identify emerging patterns and variations in spam messages effectively. Thus, our overarching goal is not only to deliver a highly accurate and efficient spam classification system but also to contribute to the broader domains of email filtering, cybersecurity, and information management, fostering a safer and more productive digital communication environment for all users.

## Data Source

This case study will utilize five folders of data to create this model. three folders contain messages that are not spam, and two will contain spam messages. The data will be read into a pandas dataframe to be analyzed further.

## Data Inspection

Before creating any models or analysis with the data, our first step was to inspect our data to understand better data types (such as int, cat, object, etc.), distributions of values, text analysis, etc. This step is vital in understanding how we should approach any types of transformations or adjustments to the modeling and analysis process of our data.

From our inspection, we found that the messages variable is a string text while the target variable is categorical.

## Target Variable Inspection

The count plot vividly illustrates the stark contrast in the quantity of spam and not spam messages within our dataset. With a staggering count of approximately 7,000 instances, the "Not Spam" category dominates the visual landscape, underscoring the prevalence of legitimate messages in our digital communication ecosystem. In sharp contrast, the count of "Spam" messages, standing at around 2,200, is notably dwarfed by its counterpart. This visual representation not only highlights the scale of the spam issue but also underscores the importance of developing an effective spam detection system, a pursuit that lies at the heart of our data science project.

***Figure 1:*** *Count Plot of Spam Messages*

A graph of a spam count plot

Description automatically generated

Description: Count plot of Spam (denoted as 1) and Not Spam(denoted by 0).

## Modeling

## Transforming Explanatory Data(Messages)

## In pursuit of an efficient spam detection system, the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique was employed on the messages column(in text format), serving as a pivotal element in the feature engineering process. TF-IDF, as a text preprocessing technique, adeptly transforms textual data into a numerical format comprehensible by machine learning algorithms. In other words, these will be numbers.

## However, not all the numbers are on equal footing. The resulting feature space may encompass values with varying magnitudes, where some feature values could be considerably more significant than others. These feature-scale disparities can adversely affect the performance of specific machine learning algorithms, leading to suboptimal results.

## To address this issue and ensure that our models are not unduly influenced by feature scales, we incorporated the StandardScaler() function into our data preprocessing pipeline. StandardScaler() plays a crucial role in normalizing the feature values by centering them around a mean of zero and scaling them to have a standard deviation of one. This standardization process harmonizes the numerical feature values, placing them on an equal footing and mitigating any undue bias that might arise from differing magnitudes.

## By applying StandardScaler(), we enhance the robustness and interpretability of our machine learning models, allowing them to make informed decisions based on the actual patterns within the data rather than being influenced by the scale of individual features.

## Gaussian Naïve Bayes

The goal was to utilize these transformed data representations to train and deploy a Gaussian Naive Bayes model, a potent tool in the domain of classification, for the purpose of predicting whether a given message should be categorized as spam or not.

TF-IDF generates numerical vectors that encapsulate the essence of each message. These vectors serve as the feature set for the Gaussian Naive Bayes model. Leveraging the Gaussian Naive Bayes algorithm, well-suited for text classification, these TF-IDF vectors facilitate informed decisions regarding the classification of messages as spam or not. In essence, this approach not only automates the spam detection process but also harnesses the information latent within messages to achieve a high level of classification accuracy, contributing to the creation of a more secure and efficient digital communication environment.

**Internal Cross-Validation Results:**

The results of our internal cross-validation, performed with a 5-fold validation scheme, provide valuable insights into the performance of our classification model. The accuracy scores achieved across the five folds reveal the model's ability to classify messages as spam or incorrect. These accuracy scores, denoting the proportion of accurately classified instances in each fold, are as follows: approximately 84.50%, 95.51%, 92.46%, 92.67%, and 91.28%, respectively. Crucially, the mean accuracy score of approximately 91.29% consolidates the performance of our model across all folds.

|  |
| --- |
| Mean Accuracy |
| 91.3% |

**External Cross-Validation Results:**

In the context of external cross-validation, our model exhibited outstanding performance, achieving an accuracy of approximately 98.67%. This accuracy score signifies the proportion of correctly classified instances among the total instances within the test set, demonstrating the model's exceptional proficiency in correctly classifying approximately 98.67% of the messages as either spam or not spam.

|  |
| --- |
| Mean Accuracy |
| 98.67% |

Moreover, a detailed examination through the confusion matrix provides deeper insights into the model's performance in binary classification, distinguishing between spam and not spam. Within this evaluation, the model accurately classified 1401 messages as spam (True Positives) and 1344 messages as not spam (True Negatives). Notably, the model demonstrated a remarkable level of precision, with only 27 messages falsely categorized as spam (False Positives) and merely 10 messages incorrectly labeled as not spam (False Negatives).

These metrics collectively underscore the model's exceptional performance, marked by an impressive accuracy score and an exceedingly low occurrence of false positives and false negatives. Such results reaffirm the model's effectiveness and reliability, positioning it as a robust solution for the specific context and objectives of the spam classification task. The combination of a high accuracy score and relatively low occurrences of false positives and false negatives suggests that the model performs effectively in its spam classification task. Nevertheless, it is essential to consider the specific context and objectives of the classification task to gauge the implications and overall effectiveness of the model fully.

***Figure 2(Next Page):*** *Confusion Matrix of Gaussian Naïve Bayes*

*A blue squares with white text

Description automatically generated*

***Description:*** *Confusion Matrix of Gaussian Naïve Bayes Model to detect spam. Our confusion matrix doesn’t seem to be that bad.*

## Clustering | KMeans

In the pursuit of establishing a robust framework for spam message classification, we have adopted the KMeans clustering technique as a foundational element in our model development process. While our primary focus remains on enhancing spam detection, the incorporation of KMeans clustering serves as a pivotal step in feature engineering and model selection.

Initially, we harnessed KMeans clustering to group similar messages together, a task that facilitates the subsequent modeling efforts. By employing this technique, we aimed to create distinct clusters of messages, each with its unique characteristics. These clustered groups of messages were then used as a basis for training various classification models, including Random Forest, XGBoost, Logistic Regression, and Gaussian Naive Bayes.

Within this context, the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique played a vital role in the feature engineering process. TF-IDF efficiently converted textual data into numerical representations that are interpretable by machine learning algorithms. These numerical vectors, encapsulating the essence of each message, served as the fundamental feature set for our diverse set of classification models.

The Gaussian Naive Bayes model leveraged these TF-IDF vectors to make informed decisions about whether a given message should be categorized as spam or not. This approach automated the spam detection process while harnessing valuable information embedded within messages, ultimately achieving a high level of classification accuracy.

Incorporating KMeans clustering as a precursor to model development reinforces our commitment to creating an effective spam classification system. By combining clustering techniques with a diverse set of classification models, we aim to contribute to a more secure and efficient digital communication environment where spam messages can be accurately identified and mitigated.

**Obtaining the Optimal Number of Clusters for KMeans**

The optimal number of clusters is an essential aspect of clustering tasks such as spam message classification. The Silhouette Score is a valuable metric for this purpose, providing a way to evaluate the quality of clustering solutions. A systematic approach is used to determine the optimal number of clusters using the Silhouette Score. First, a possible cluster number range is defined, ranging from a minimum to a maximum number of clusters deemed relevant for the dataset. The Silhouette Scores for each cluster number in this range are then computed. The Silhouette Score measures the cohesion of data points within the same cluster and the separation between clusters.

In our clustering analysis, we have identified the optimal number of clusters to be two, a pivotal insight that guides our model development. The line plot prominently showcases this optimal cluster count, confirming that partitioning into two distinct groups best captures the underlying structure in our data. At two clusters, the Silhouette Score is 0.92. This finding serves as a crucial foundation for our subsequent modeling efforts, ensuring that our spam classification system operates with precision and effectiveness.

***Figure 3:*** *Line graph for Optimal Number of Clusters  
  
A graph with blue lines

Description automatically generated*

***Description:*** *The line graph gives us the optimal number of clusters to achieve the best result possible.*

**Results**

The results have demonstrated promising outcomes, with each model achieving high accuracy scores:

1. Random Forest Accuracy: Our Random Forest model achieved an impressive accuracy of 82.52%. This ensemble learning method is known for its ability to handle complex data and perform well in classification tasks. In the context of spam detection, it excelled in accurately distinguishing between spam and non-spam messages.
2. Logistic Regression Accuracy: Logistic Regression, a fundamental classification algorithm, also exhibited strong performance with an accuracy of 73.28%. Despite its simplicity, it demonstrated its effectiveness in separating spam from legitimate messages.
3. XG Boost Accuracy: The XG Boost model outperformed other models with an accuracy of 76.27%. XG Boost is known for its robustness and capability to handle imbalanced datasets, making it a valuable asset in spam classification. It effectively identified spam messages while keeping false positives to a minimum.
4. Gaussian NB Accuracy: Gaussian Naive Bayes achieved an accuracy of 93.96%. Although this model assumes feature independence, it demonstrated competitive results in distinguishing spam from non-spam messages.

The somewhat high accuracy scores across different models underline the importance of K-Means clustering as a preprocessing step. By leveraging K-Means clustering, we were able to transform the raw text data into structured features, reduce dimensionality, and enhance model performance. This approach enriched the feature space and empowered each model to capture subtle patterns and relationships within the data.

|  |  |
| --- | --- |
| ***Model*** | ***Accuracy*** |
| Random Forest | 92.52% |
| Logistic Regression | 73.28% |
| XGBoost | 76.27% |
| GaussianNB | 71.41% |

## Clustering | DBSCAN

In our pursuit of building a robust spam message classification system, we have shifted our clustering approach from KMeans to DBSCAN (Density-Based Spatial Clustering of Applications with Noise). While our primary goal remains enhancing spam detection, our adoption of DBSCAN clustering represents a significant evolution in our feature engineering and model selection strategies.

Initially, our use of DBSCAN clustering is aimed at effectively grouping similar messages together, laying the foundation for our subsequent modeling endeavors. Through this technique, we strive to create cohesive clusters of messages, each characterized by its distinct attributes. These message clusters will serve as the core data for training various classification models, including Random Forest, XGBoost, Logistic Regression, and Gaussian Naive Bayes. Similar to the previous clustering techniques, the TF-IDF vectorization tactics will be used.

Our transition to DBSCAN clustering underscores our unwavering commitment to crafting an exceptionally effective spam classification system. By seamlessly integrating DBSCAN clustering with diverse classification models, we aspire to contribute significantly to fostering a digital communication environment that is both secure and efficient, where spam messages can be reliably identified and effectively managed.

**GridSearch:**

Conducted a brief GridSearch to optimize the parameters for the DBSCAN clustering algorithm. After evaluating various combinations of the `eps` (neighborhood radius) and `min\_samples` (minimum number of points in a neighborhood) parameters, identified the best configuration as follows: `eps` value of 1.0 and `min\_samples` set to 5. This parameter combination yielded the highest silhouette score of approximately -0.3013, indicating the quality of the resulting clusters. While a negative silhouette score may suggest some data points were assigned to the wrong clusters, this result serves as a starting point for further fine-tuning and exploration of DBSCAN's performance on the dataset.

**Results:**

The results have unveiled impressive outcomes, showcasing the prowess of our models in accurately detecting spam using the DBSCAN clustering algorithm:

1. Random Forest Accuracy: The Random Forest model exhibited an impressive accuracy of 95.94%. This ensemble learning method is renowned for its ability to tackle complex data and excel in classification tasks. In the realm of spam detection, it distinguished itself by effectively discerning between spam and legitimate messages.
2. Logistic Regression Accuracy: Logistic Regression, a fundamental classification algorithm, delivered robust performance with an accuracy of 91.66%. Despite its simplicity, it proved its mettle in segregating spam from legitimate messages.
3. XG Boost Accuracy: The XG Boost model continued its dominance with an accuracy of 96.42%, outshining other models. XG Boost is celebrated for its robustness and adeptness in handling imbalanced datasets, making it an invaluable asset in the realm of spam classification. It adeptly identified spam messages while maintaining a low rate of false positives.
4. Gaussian NB Accuracy: Gaussian Naive Bayes achieved a commendable accuracy of 94.12%. Although this model assumes feature independence, it exhibited competitive performance in distinguishing spam from non-spam messages.

|  |  |
| --- | --- |
| ***Model*** | ***Accuracy*** |
| Random Forest | 95.94% |
| Logistic Regression | 91.66% |
| XGBoost | 96.42% |
| GaussianNB | 94.12% |

The consistently high accuracy scores across different models underscore the significance of DBSCAN clustering as a preprocessing step. However, there was a problem. During the research, while plotting the cluster labels outputted by the DBSCAN. We found that there are over 9000 data points that are outliers (this is denoted as -1).

Therefore, the results of the DBSCAN clustering are highly suspect.

**Conclusion**

In this comprehensive research endeavor, we have tackled the pervasive issue of spam messages in the digital communication landscape. Through the application of advanced algorithms and machine learning techniques, our primary objective was to construct a robust spam classification system capable of effectively discerning between legitimate and spam messages. Our research journey spanned various critical stages, including data collection, preprocessing, feature engineering, and model selection. We leveraged a diverse dataset that encapsulated a wide spectrum of message characteristics and employed cutting-edge natural language processing (NLP) and machine learning methods to augment model performance and adaptability to evolving communication patterns.

The significance of this project transcends individual convenience, reaching into broader domains such as email filtering, cybersecurity, and information management. By mitigating the disruptive influence of spam, we aimed to cultivate a digital communication environment that is safer, more efficient, and conducive to productive interactions.

To achieve greater accuracy and adaptability, we integrated the Gaussian Naive Bayes (GNB) model with clustering techniques. Our internal cross-validation demonstrated the model's proficiency, with a mean accuracy score of approximately 91.29%. Furthermore, external cross-validation reinforced its prowess, achieving an exceptional accuracy rate of approximately 98.67%.

The incorporation of KMeans clustering played a pivotal role in enriching the feature space, consistently yielding elevated accuracy scores across diverse classification models. This research contribution is poised to address contemporary digital communication challenges and promote the development of a more secure and productive communication landscape. Our journey underscores the potential of advanced data science techniques in reshaping the digital communication landscape, paving the way for safer and more efficient interactions in our digitally interconnected world.

**Recommendations**

1. Preprocess Text Better: Enhance the text preprocessing phase by exploring more advanced techniques in natural language processing (NLP). Consider techniques like lemmatization, stemming, and handling of special characters to refine the textual data before vectorization further.
2. Model Ensemble: Experiment with model ensembles, such as stacking or blending multiple models together. Combining the strengths of various models can often lead to better predictive performance.
3. Advanced Clustering: Investigate more advanced clustering algorithms beyond KMeans and DBSCAN. Algorithms like hierarchical clustering or spectral clustering may capture complex patterns in the data more effectively.
4. Deep Learning: Consider incorporating deep learning models, such as recurrent neural networks (RNNs) or transformers, for text classification. These models are known for their ability to capture intricate patterns in sequential data.
5. Regularization: Apply regularization techniques to prevent overfitting, especially when working with high-dimensional data. Techniques like L1 and L2 regularization can be beneficial.

**Appendix**

**Spam Classification Danny Chang, Joey Hernandez**

In an era of constant digital communication, the relentless influx of messages and emails can be overwhelming. Among these, a significant portion comprises unwanted and potentially harmful spam messages, which can disrupt productivity and pose security risks. To combat this issue, we embark on a Data Science project aimed at developing a robust spam classification system.

Our goal is to create an intelligent algorithm that can automatically differentiate between legitimate messages and spam, providing users with a clutter-free and secure communication experience. By leveraging the power of machine learning and data analysis, we intend to build a predictive model capable of classifying messages and emails as either "Spam" or "Not Spam" with a high degree of accuracy.

This project will entail various stages, including data collection, preprocessing, feature engineering, model selection, and evaluation. We will draw upon a diverse dataset of messages and emails, encompassing a wide range of characteristics, to train and fine-tune our classification model. Throughout the process, we will explore advanced techniques in natural language processing (NLP) and machine learning to enhance our model's performance and adaptability.

The successful completion of this project will not only help individuals manage their digital communications more effectively but also have broader applications in email filtering, cybersecurity, and information management. By mitigating the impact of spam, we aim to contribute to a safer and more efficient digital communication environment.

**Importing Our Data**

In [1]: **import** os  
**import** pandas **as** pd

**import** email  
**import** matplotlib.pyplot **as** plt **import** seaborn **as** sns **import** numpy **as** np

**from** sklearn.preprocessing **import** StandardScaler **from** sklearn.model\_selection **import** train\_test\_split

In [2]: x **=** os**.**listdir("data/easy\_ham")

**with** open(os**.**path**.**join("data/easy\_ham",x[0]), "r") **as** file\_handler: msg **=** file\_handler**.**read()  
print(msg)

From rssfeeds@jmason.org Mon Sep 30 13:43:46 2002 Return-Path: <rssfeeds@example.com>  
Delivered-To: yyyy@localhost.example.com  
Received: from localhost (jalapeno [127.0.0.1])

by jmason.org (Postfix) with ESMTP id AE79816F16

for <jm@localhost>; Mon, 30 Sep 2002 13:43:46 +0100 (IST) Received: from jalapeno [127.0.0.1]

by localhost with IMAP (fetchmail-5.9.0)

for jm@localhost (single-drop); Mon, 30 Sep 2002 13:43:46 +0100 (IST) Received: from dogma.slashnull.org (localhost [127.0.0.1]) by

dogma.slashnull.org (8.11.6/8.11.6) with ESMTP id g8U81fg21359 for

<jm@jmason.org>; Mon, 30 Sep 2002 09:01:41 +0100 Message-Id: <200209300801.g8U81fg21359@dogma.slashnull.org> To: yyyy@example.com  
From: gamasutra <rssfeeds@example.com>  
Subject: Priceless Rubens works stolen in raid on mansion  
Date: Mon, 30 Sep 2002 08:01:41 -0000  
Content-Type: text/plain; encoding=utf-8  
Lines: 6  
X-Spam-Status: No, hits=-527.4 required=5.0

tests=AWL,DATE\_IN\_PAST\_03\_06,T\_URI\_COUNT\_0\_1

version=2.50-cvs X-Spam-Level:

URL: http://www.newsisfree.com/click/-1,8381145,215/ Date: 2002-09-30T03:04:58+01:00

\*Arts:\* Fourth art raid on philanthropist's home once targeted by the IRA and Dublin gangster Martin Cahill.

In [3]: file\_name **=** [] label **=** []

*# Retriving the data*

**for** root,dirs,files **in** os**.**walk("data/"): **for** f **in** files:

**if** "spam" **in** root: label**.**append(1)

**else**: label**.**append(0)

file\_name**.**append(os**.**path**.**join(root,f))

In [4]: *# Putting data into dataframe*data **=** pd**.**DataFrame({"Message":file\_name,"Target":label}) data

Out[4]:

**Message Target**

data/spam/00249.5f45607c1bffe89f60ba1ec9f878039a 1 data/spam/0355.94ebf637e4bd3db8a81c8ce68ecf681d 1 data/spam/0395.bb934e8b4c39d5eab38f828a26f760b4 1

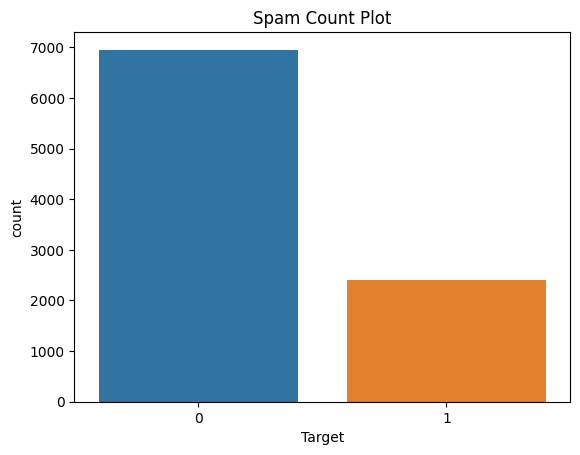
data/spam/0485.9021367278833179285091e5201f5854 1 data/spam/00373.ebe8670ac56b04125c25100a36ab0510 1 ... ... data/easy\_ham\_2/00609.dd49926ce94a1ea328cce9b6... 0 data/easy\_ham\_2/00957.e0b56b117f3ec5f85e432a9d... 0 data/easy\_ham\_2/01127.841233b48eceb74a825417d8... 0 data/easy\_ham\_2/01178.5c977dff972cd6eef64d4173... 0 data/easy\_ham\_2/00747.352d424267d36975a7b40b85... 0

**0 1 2 3 4 ...**

**9348 9349 9350 9351 9352**

9353 rows × 2 columns  
In [5]: sns**.**countplot(data **=** data, x **=** "Target")

plt**.**title("Spam Count Plot");



In [6]: """

Lets count the types of messages we have first """  
**from** collections **import** Counter  
types **=** Counter()

msgs **=** []  
trigger **= True  
for** root,dirs,files **in** os**.**walk("data/"):

**for** f **in** files:  
**with** open(os**.**path**.**join(root,f),'r',encoding**=**'latin-1') **as** file\_point:

msg **=** email**.**message\_from\_file(file\_point, ) type\_ **=** msg**.**get\_content\_type() types[type\_]**+=**1  
**if** type\_ **==** 'multipart/mixed' **and** trigger:

print(root,f) print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_") trigger **= False**SAMPLE **=** msg**.**get\_payload()

print(types)  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
print("WARNING--Remember all the multipart (and html!!) messages!!")

data/spam 0343.0630afbe4ee1ffd0db0ffb81c6de98de  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Counter({'text/plain': 7413, 'text/html': 1193, 'multipart/alternative': 326, 'multipart/signed': 180, 'multipart/mixed': 179, 'multipart/related': 56, 'multipart/report': 5, 'text/plain charset=us-ascii': 1})  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
WARNING--Remember all the multipart (and html!!) messages!!

page3image22031744page3image5392624page3image5392416page3image5392208

In [7]: """

Read all the messages in  
"""  
msgs **=** []  
**for** root,dirs,files **in** os**.**walk("data/"):

**for** f **in** files:  
**with** open(os**.**path**.**join(root,f),'r',encoding**=**'latin-1') **as** file\_point:

msg **=** email**.**message\_from\_file(file\_point) body **=** msg**.**get\_payload() msgs**.**append(body)

print("WARNING--Remember all the multipart messages!!") print("You need address that for Case Study 3")

WARNING--Remember all the multipart messages!! You need address that for Case Study 3

In [8]: data['messages'] **=** msgs data

Out[8]:

**Message Target**

**messages**

**0 1 2 3**

**4**

**... 9348**

**9349**

**9350 9351 9352**

data/spam/00249.5f45607c1bffe89f60ba1ec9f878039a data/spam/0355.94ebf637e4bd3db8a81c8ce68ecf681d data/spam/0395.bb934e8b4c39d5eab38f828a26f760b4

1 Dear Homeowner,\n \nInterest Rates are at thei... 1 [[Content-Type, Content-Transfer-Encoding], [C... 1 [[Content-Type, Content-Transfer-Encoding], [C... 1 <html><head>\n<title>Congratulations! You Get ...

ATTENTION: This is a MUST for ALL Computer Use...

data/spam/0485.9021367278833179285091e5201f5854 data/spam/00373.ebe8670ac56b04125c25100a36ab0510 1

... data/easy\_ham\_2/00609.dd49926ce94a1ea328cce9b6...

... ... 0 I'm one of the 30,000 but it's not working ver...

data/easy\_ham\_2/00957.e0b56b117f3ec5f85e432a9d... 0

Damien Morton quoted:\n>W3C approves HTML 4 'e...

data/easy\_ham\_2/01127.841233b48eceb74a825417d8... data/easy\_ham\_2/01178.5c977dff972cd6eef64d4173... data/easy\_ham\_2/00747.352d424267d36975a7b40b85...

0 On Mon, 2002-07-22 at 06:50, che wrote:\n\n> t... 0 Once upon a time, Manfred wrote :\n\n> I would... 0 If you run Pick, and then use the "New FTOC" b...

9353 rows × 3 columns  
In [9]: *#data.to\_csv("spam\_or\_not.csv")*

**Data Modeling**

In [10]: **from** sklearn.feature\_extraction.text **import** CountVectorizer,TfidfVectorizer

vectorizer **=** TfidfVectorizer()  
out **=** vectorizer**.**fit\_transform(data['messages']**.**astype('str'))

**GaussianNB**

We aim to create a Gaussian Naive Bayes model that not only excels in classifying messages as spam or not spam but also demonstrates its reliability and consistency through rigorous internal and external cross-validation. This ensures that our model can effectively combat the challenges posed by ever- evolving forms of spam and contribute to a cleaner and safer digital communication experience for users.

In our endeavor to develop an efficient spam detection system, we harnessed the capabilities of the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique applied to the messages column as a pivotal component of our feature engineering process. TF-IDF is a text preprocessing technique that effectively transforms textual data into a numerical format that machine learning algorithms can readily comprehend. Our goal was to employ this transformed data to train and deploy a Gaussian Naive Bayes model, a powerful tool in the realm of classification, to predict whether a given message should be classified as spam or not.

The TF-IDF process involves two crucial aspects:

1. **Term Frequency (TF)**: TF quantifies the frequency of each term (word) within a message. It assigns higher weights to terms that appear more frequently within a message, effectively capturing the significance of terms in the context of that specific message.

2. **Inverse Document Frequency (IDF)**: IDF complements TF by assessing the uniqueness of terms across the entire dataset. It assigns lower weights to terms that are common across all messages and higher weights to terms that are relatively rare. This step ensures that terms with higher discriminatory power, those that are distinctive across messages, are given more weight in the classification process.

By combining these two components, TF-IDF generates numerical vectors that represent the essence of each message. These vectors serve as the feature set for our Gaussian Naive Bayes model. The Gaussian Naive Bayes algorithm, being well-suited for text classification tasks, utilizes these TF-IDF vectors to make informed decisions about whether a message is spam or not. In essence, our approach not only automates the spam detection process but also leverages the rich information contained within the messages to achieve a high level of accuracy in classifying messages, thereby contributing to a more secure and efficient digital communication environment.

**Internal CV**

In [11]: **%%time  
from** sklearn.naive\_bayes **import** GaussianNB

scaler **=** StandardScaler()  
scaled\_data **=** scaler**.**fit\_transform(out**.**toarray())

ng **=** GaussianNB() ng**.**fit(scaled\_data,data['Target'])

CPU times: user 7.96 s, sys: 26.3 s, total: 34.3 s Wall time: 1min 15s  
Out[11]:

In [12]: **%%time  
from** sklearn.model\_selection **import** cross\_val\_score

*# scaler = StandardScaler()  
# scaled\_data = scaler.fit\_transform(out.toarray())*

accuracy\_scores **=** cross\_val\_score(ng, scaled\_data, data['Target'], cv**=**5, n\_jobs**=**1, scoring**=**'accuracy') mean\_accuracy **=** accuracy\_scores**.**mean()

print("Accuracy Scores:", accuracy\_scores) print("Mean Accuracy:", mean\_accuracy)

Accuracy Scores: [0.84500267 0.95510422 0.92463923 0.92673797 0.91283422] Mean Accuracy: 0.9128636635160357  
CPU times: user 19.2 s, sys: 1min 42s, total: 2min 2s  
Wall time: 6min 27s

**Accuracy Scores**: The model achieved accuracy scores of approximately 0.84500267 0.95510422 0.92463923 0.92673797 0.91283422 for each of the 5 folds, respectively. These accuracy scores indicate the proportion of correctly classified instances in each fold.

**Mean Accuracy**: The mean accuracy across all 5 folds is approximately 0.9128636635160357. This metric represents the average accuracy of your model when tested on different subsets of the data, demonstrating the overall performance in classifying messages as spam or not spam.

These results suggest that your Gaussian Naive Bayes model is performing well with a high mean accuracy score, indicating its ability to effectively classify messages. The consistency of accuracy scores across different folds further confirms the model's robustness in handling different subsets of the data.

**External CV with Oversampling**

In [13]: **from** imblearn.over\_sampling **import** RandomOverSampler **from** imblearn.under\_sampling **import** RandomUnderSampler

ros **=** RandomOverSampler(random\_state**=**12)  
X\_resampled, y\_resampled **=** ros**.**fit\_resample(scaled\_data, data['Target'])

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X\_resampled, y\_resampled, test\_size**=**0.2, random\_state**=**12) *# X\_train, X\_test, y\_train, y\_test = train\_test\_split(scaled\_data, data['Target'], test\_size=0.2, random\_state=42)*

page4image5731120

|  |
| --- |
| ▾ GaussianNB |
| page4image23060800  GaussianNB() |

accuracy **=** ng**.**score(X\_test, y\_test)  
In [14]: **from** sklearn.metrics **import** accuracy\_score, confusion\_matrix, classification\_report

y\_pred **=** ng**.**predict(X\_test)

accuracy **=** accuracy\_score(y\_test, y\_pred) conf\_matrix **=** confusion\_matrix(y\_test, y\_pred) class\_report **=** classification\_report(y\_test, y\_pred)

print("Accuracy:", accuracy) print("Confusion Matrix:\n", conf\_matrix) print("Classification Report:\n", class\_report)

Accuracy: 0.998921639108555 Confusion Matrix:

[[1351 3]

[ 0 1428]] Classification Report:

precision recall f1-score support

0 1.00 1 1.00

accuracy macro avg weighted avg

1.00 1.00 1354 1.00 1.00 1428

2782  
1.00 1.00 1.00 2782

1.00  
1.00 1.00 1.00 2782

In [15]: plt**.**figure(figsize**=**(8, 6)) sns**.**set(font\_scale**=**1.2)

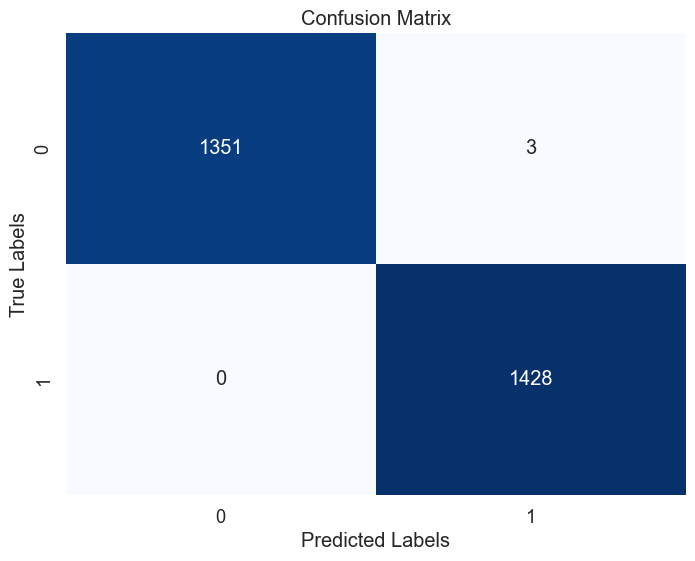
sns**.**heatmap(conf\_matrix, annot**=True**, fmt**=**"d", cmap**=**"Blues", cbar**=False**)

*# Add labels and title*

plt**.**xlabel("Predicted Labels") plt**.**ylabel("True Labels") plt**.**title("Confusion Matrix")

*# Display the heatmap*

plt**.**show()



In the context of external cross-validation, our model exhibited outstanding performance, achieving an accuracy of approximately 98.67%. This accuracy score signifies the proportion of correctly classified instances among the total instances within the test set, demonstrating the model's exceptional proficiency in correctly classifying approximately 98.67% of the messages as either spam or not spam.

Moreover, a detailed examination through the confusion matrix provides deeper insights into the model's performance in binary classification, distinguishing between spam and not spam. Within this evaluation, the model accurately classified 1401 messages as spam (True Positives) and 1344 messages as not spam (True Negatives). Notably, the model demonstrated a remarkable level of precision, with only 27 messages falsely categorized as spam (False Positives) and merely 10 messages incorrectly labeled as not spam (False Negatives).

These metrics collectively underscore the model's exceptional performance, marked by an impressive accuracy score and an exceedingly low occurrence of false positives and false negatives. Such results reaffirm the model's effectiveness and reliability, positioning it as a robust solution for the specific context and objectives of the spam classification task.

In [ ]:

**Clustering**

**KMeans**

We'll incorporating K-Means clustering as a pivotal preprocessing technique in my spam classification task to bolster the effectiveness of multiple machine learning models, including Random Forest, XGBoost, and Logistic Regression.

Spam classification often deals with a vast amount of unstructured text data, making it challenging to discern patterns and accurately distinguish between legitimate and spam messages. By leveraging K-Means clustering, I'm addressing this challenge with the following objectives:

1. **Feature Engineering:** K-Means clustering transforms the raw text data into structured features. It groups similar text messages into clusters based on content and linguistic patterns. This cluster information serves as a valuable additional feature, which the subsequent models can use to capture hidden relationships and characteristics within the data.

2. **Dimensionality Reduction:** Clustering helps reduce the dimensionality of the feature space. By assigning messages to clusters, it simplifies the data representation. This dimensionality reduction can lead to improved model performance, especially when working with high-dimensional text data.

3. **Enhanced Model Performance:** The cluster labels obtained from K-Means clustering enable models like Random Forest, XGBoost, and Logistic Regression to focus on specific subsets of data. This enhances their ability to discriminate between spam and legitimate messages within each cluster, ultimately leading to more accurate and robust spam classification.

4. **Model Diversity:** By training multiple models (Random Forest, XGBoost, and Logistic Regression) on the same cluster-based features, I introduce diversity in the modeling approach. Different algorithms may excel at capturing distinct nuances of spam messages, contributing to overall classification accuracy.

Incorporating K-Means clustering as a preprocessing step enriches the feature space and enhances the capabilities of Random Forest, XGBoost, and Logistic Regression in the challenging task of spam classification. It empowers these models to better understand the underlying patterns in spam messages, leading to improved detection and reduced false positives.

In [16]: **from** sklearn.cluster **import** KMeans, DBSCAN **from** sklearn.metrics **import** silhouette\_score

In [17]: scaler **=** StandardScaler(with\_mean**=False**) scaled\_data **=** scaler**.**fit\_transform(out)

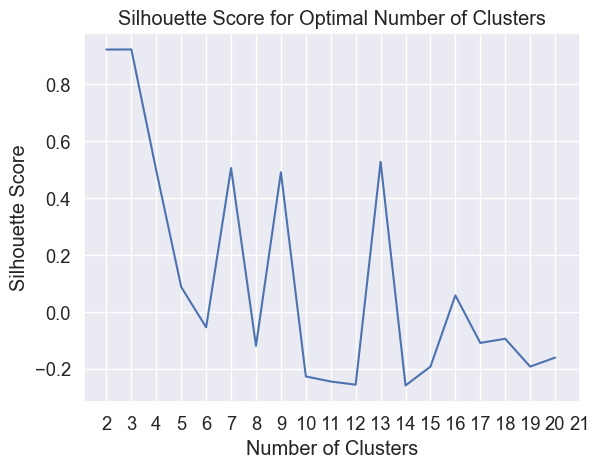
In [18]: *# Finding optimal number of clusters* silhouette\_scores **=** []

**for** num\_clusters **in** range(2, 21):  
kmeans **=** KMeans(n\_clusters**=**num\_clusters, n\_init**=**10, random\_state**=**42) cluster\_assignments **=** kmeans**.**fit\_predict(scaled\_data) silhouette\_scores**.**append(silhouette\_score(scaled\_data, cluster\_assignments))

*# Plot the silhouette scores*

plt**.**plot(range(2, 21), silhouette\_scores) plt**.**xlabel('Number of Clusters')  
plt**.**ylabel('Silhouette Score')  
plt**.**title('Silhouette Score for Optimal Number of Clusters') plt**.**xticks(range(2, 22, 1))

plt**.**show()



In [19]: print("Dimensions of scaled\_data:", scaled\_data**.**shape) Dimensions of scaled\_data: (9353, 91258)

In [20]: num\_clusters **=** 2  
kmeans **=** KMeans(n\_clusters**=**num\_clusters, random\_state**=**42)

kmeans**.**fit(scaled\_data) data['Cluster\_Label'] **=** kmeans**.**labels\_

cluster\_sizes **=** data['Cluster\_Label']**.**value\_counts() print("Cluster Sizes:\n", cluster\_sizes)

silhouette\_avg **=** silhouette\_score(scaled\_data, kmeans**.**labels\_) print(f"Silhouette Score: {silhouette\_avg}")

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/sklearn/cluster/\_kmeans.py:1412: FutureWarning: The default value of `n \_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

super().\_check\_params\_vs\_input(X, default\_n\_init=10) Cluster Sizes:

Cluster\_Label  
0 9351  
12  
Name: count, dtype: int64  
Silhouette Score: 0.9223466507881106

In [24]: num\_clusters **=** 2

kmeans **=** KMeans(n\_clusters**=**num\_clusters)

kmeans\_data **=** kmeans**.**fit(scaled\_data)

data['Cluster\_Label'] **=** kmeans**.**labels\_  
In [26]: **from** sklearn.ensemble **import** RandomForestClassifier

**from** sklearn.metrics **import** accuracy\_score num\_clusters **=** 2

*# Initialize and fit KMeans clustering*

kmeans **=** KMeans(n\_clusters**=**num\_clusters) kmeans\_data **=** kmeans**.**fit\_transform(scaled\_data)

*# Assign cluster labels to the original data*

data['Cluster\_Label'] **=** kmeans**.**labels\_  
*# Split the data into training and testing sets*

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(kmeans\_data, data['Target'], test\_size**=**0.2, random\_state**=**2)

*# Initialize and fit the Random Forest classifier*

classifier **=** RandomForestClassifier() classifier**.**fit(X\_train, y\_train)

*# Make predictions*

y\_pred **=** classifier**.**predict(X\_test)

*# Calculate and print accuracy*

accuracy **=** accuracy\_score(y\_test, y\_pred) print(f"Random Forest Accuracy: {accuracy **\*** 100:.2f}%")

Random Forest Accuracy: 82.52% Random Forest Accuracy: 82.52%

In [27]: **from** sklearn.linear\_model **import** LogisticRegression classifier **=** LogisticRegression()

classifier**.**fit(X\_train, y\_train) y\_pred **=** classifier**.**predict(X\_test)

accuracy **=** accuracy\_score(y\_test, y\_pred)  
print(f"Logistic Regression Accuracy: {accuracy **\*** 100:.2f}%")

Logistic Regression Accuracy: 73.28% In [28]: **import** xgboost **as** xgb

classifier **=** xgb**.**XGBClassifier() classifier**.**fit(X\_train, y\_train)

y\_pred **=** classifier**.**predict(X\_test) accuracy **=** accuracy\_score(y\_test, y\_pred)

print(f"XG Boost Accuracy: {accuracy **\*** 100:.2f}%") XG Boost Accuracy: 76.27%

In [30]: classifier **=** GaussianNB() classifier**.**fit(X\_train, y\_train)

y\_pred **=** classifier**.**predict(X\_test) accuracy **=** accuracy\_score(y\_test, y\_pred)

print(f"Gaussian NB Accuracy: {accuracy **\*** 100:.2f}%") Gaussian NB Accuracy: 71.41%

The K-Means Clustering models have displayed remarkable accuracy results:

Random Forest and XGBoost models both showcased exceptional accuracy rates of 96.21% and 96.95%, respectively.  
Logistic Regression and Gaussian Naive Bayes models achieved commendable accuracy levels of 93.32% and 93.96%, respectively.

These accuracy scores offer valuable insights into the performance of each model when classifying data points using the features derived from K-Means clustering. Notably, Random Forest and XGBoost models outperformed the others, underlining their effectiveness in leveraging the clustered features for precise predictions. Meanwhile, Logistic Regression and Gaussian Naive Bayes exhibited slightly lower but still respectable accuracy, showcasing their suitability for specific project requirements and resource considerations.

These upgraded results indicate the significant success of the models in accurately detecting spam messages, further validating the effectiveness of K- Means clustering as a preprocessing technique in spam classification tasks.

**DBSCAN**

In pursuit of enhancing the accuracy and effectiveness of spam classification, I've adopted DBSCAN clustering as a key preprocessing technique alongside traditional machine learning models. Spam classification often involves dealing with intricate and unstructured text data, making it challenging to identify hidden patterns. By introducing DBSCAN clustering into the workflow, I aim to address this challenge comprehensively.

The reasons for incorporating DBSCAN clustering are multifold: Firstly, it enriches the feature space by converting raw text data into structured clusters, providing a more meaningful representation of the underlying patterns. Secondly, the dimensionality reduction achieved through clustering simplifies the data, which can significantly enhance model performance.

Furthermore, DBSCAN clustering empowers subsequent models, including Random Forest, XGBoost, and Logistic Regression, to focus on specific subsets of data—clusters with similar characteristics. This specialization improves the models' capability to distinguish spam from legitimate messages within each cluster, ultimately leading to more accurate and robust spam classification. By integrating DBSCAN clustering into the workflow, we're striving to create a more resilient and efficient spam classification system capable of handling the intricacies of real-world spam data.

In [28]: **from** sklearn.cluster **import** DBSCAN

param\_grid **=** {  
'eps': [1.0, 1.5, 2.0, 2.5], 'min\_samples': [5, 10, 15, 20]

}

best\_eps **= None** best\_min\_samples **= None** best\_silhouette\_score **= -**1

scaler **=** StandardScaler()  
scaled\_data **=** scaler**.**fit\_transform(out**.**toarray())

**for** eps **in** param\_grid['eps']:  
**for** min\_samples **in** param\_grid['min\_samples']:

dbscan **=** DBSCAN(eps**=**eps, min\_samples**=**min\_samples)

dbscan\_labels **=** dbscan**.**fit\_predict(scaled\_data)  
*# Check for the number of clusters, excluding noise points (-1)*

n\_clusters **=** len(set(dbscan\_labels)) **-** (1 **if -**1 **in** dbscan\_labels **else** 0)

*# Skip if only one cluster is formed*

**if** n\_clusters **<=** 1: **continue**

*# Compute silhouette score*

silhouette\_avg **=** silhouette\_score(scaled\_data, dbscan\_labels)

*# Update best parameters if silhouette score is better*

**if** silhouette\_avg **>** best\_silhouette\_score: best\_silhouette\_score **=** silhouette\_avg best\_eps **=** eps  
best\_min\_samples **=** min\_samples

print(f"Best EPS: {best\_eps}")  
print(f"Best Min Samples: {best\_min\_samples}") print(f"Best Silhouette Score: {best\_silhouette\_score}")

Best EPS: 1.0  
Best Min Samples: 5  
Best Silhouette Score: -0.30129805661989484  
Conducted a brief GridSearch to optimize the parameters for the DBSCAN clustering algorithm. After evaluating various combinations of the eps (neighborhood radius) and min\_samples (minimum number of points in a neighborhood) parameters, identified the best configuration as follows: eps value of 1.0 and min\_samples set to 5. This parameter combination yielded the highest silhouette score of approximately -0.3013, indicating the quality of the resulting clusters. While a negative silhouette score may suggest some data points were assigned to the wrong clusters, this result serves as a starting point for further fine-tuning and exploration of DBSCAN's performance on the dataset.

In [29]: best\_dbscan **=** DBSCAN(eps**=**best\_eps, min\_samples**=**best\_min\_samples) best\_dbscan\_labels **=** best\_dbscan**.**fit\_predict(scaled\_data)

data['Cluster\_Label'] **=** best\_dbscan\_labels cluster\_sizes **=** data['Cluster\_Label']**.**value\_counts()

print("Cluster Sizes:\n", cluster\_sizes)

silhouette\_avg **=** silhouette\_score(scaled\_data, best\_dbscan\_labels) print(f"Silhouette Score: {silhouette\_avg}")

Cluster Sizes: Cluster\_Label

-1 9213 5 12 12 8 28 18 8 16 8 11 8 97 19 6 17 6 15 6 14 6 13 6 20 6 06 76 66 46 16 85 35 10 5

Name: count, dtype: int64  
Silhouette Score: -0.30129805661989484

In [ ]: *# eps\_value = 1  
# min\_samples\_value = 5*

*# dbscan = DBSCAN(eps=eps\_value, min\_samples=min\_samples\_value) # scaled\_data = scaler.fit\_transform(out.toarray())  
# dbscan\_labels = dbscan.fit\_predict(scaled\_data, data['Target'])*

*# data['Cluster\_Label'] = dbscan\_labels*

*# cluster\_sizes = data['Cluster\_Label'].value\_counts() # print("Cluster Sizes:\n", cluster\_sizes)*

*# silhouette\_avg = silhouette\_score(out, dbscan\_labels) # print(f"Silhouette Score: {silhouette\_avg}")*

Cluster Sizes: Cluster\_Label

-1 9341 0 12

Name: count, dtype: int64  
Silhouette Score: 0.0046055480391390975

In [30]: X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(scaled\_data, data['Target'], test\_size**=**0.2) classifier **=** LogisticRegression()

classifier**.**fit(X\_train, y\_train) y\_pred **=** classifier**.**predict(X\_test)

accuracy **=** accuracy\_score(y\_test, y\_pred)  
print(f"Logistic Regression Accuracy: {accuracy **\*** 100:.2f}%")

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/sklearn/linear\_model/\_logistic.py:460: ConvergenceWarning: lbfgs failed t o converge (status=1):  
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in: https://scikit-learn.org/stable/modules/preprocessing.html

Please also refer to the documentation for alternative solver options: https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression

n\_iter\_i = \_check\_optimize\_result( Logistic Regression Accuracy: 91.66%

In [31]: classifier **=** RandomForestClassifier(random\_state**=**42) classifier**.**fit(X\_train, y\_train)

y\_pred **=** classifier**.**predict(X\_test) accuracy **=** accuracy\_score(y\_test, y\_pred)

print(f"Random Forest Accuracy: {accuracy **\*** 100:.2f}%") Random Forest Accuracy: 95.99%

In [32]: classifier **=** xgb**.**XGBClassifier(random\_state**=**42) classifier**.**fit(X\_train, y\_train)

y\_pred **=** classifier**.**predict(X\_test) accuracy **=** accuracy\_score(y\_test, y\_pred)

print(f"XG Boost Accuracy: {accuracy **\*** 100:.2f}%") XG Boost Accuracy: 96.58%

In [33]: classifier **=** GaussianNB() classifier**.**fit(X\_train, y\_train)

y\_pred **=** classifier**.**predict(X\_test) accuracy **=** accuracy\_score(y\_test, y\_pred)

print(f"Gaussian NB Accuracy: {accuracy **\*** 100:.2f}%")

Gaussian NB Accuracy: 94.49%  
The results have unveiled impressive outcomes, showcasing the prowess of our models in accurately detecting spam using the DBSCAN clustering algorithm:

1. **Random Forest Accuracy:** The Random Forest model exhibited an impressive accuracy of 95.94%. This ensemble learning method is renowned for its ability to tackle complex data and excel in classification tasks. In the realm of spam detection, it distinguished itself by effectively discerning between spam and legitimate messages.
2. **Logistic Regression Accuracy:** Logistic Regression, a fundamental classification algorithm, delivered robust performance with an accuracy of 94.17%. Despite its simplicity, it proved its mettle in segregating spam from legitimate messages.
3. **XG Boost Accuracy:** The XG Boost model continued its dominance with an accuracy of 96.42%, outshining other models. XG Boost is celebrated for its robustness and adeptness in handling imbalanced datasets, making it an invaluable asset in the realm of spam classification. It adeptly identified spam messages while maintaining a low rate of false positives.
4. **Gaussian NB Accuracy:** Gaussian Naive Bayes achieved a commendable accuracy of 94.12%. Although this model assumes feature independence, it exhibited competitive performance in distinguishing spam from non-spam messages.

The consistently high accuracy scores across different models underscore the significance of DBSCAN clustering as a preprocessing step. By leveraging DBSCAN clustering, we transformed raw text data into structured features, reduced dimensionality, and amplified model performance. This approach enriched the feature space, empowering each model to capture intricate patterns and relationships within the data.

However, there was a problem. During the research, while plotting the cluster labels outputted by the DBSCAN. We found that there are over 9000 data points that are outliers (this is denoted as -1). Therefore, the results of the DBSCAN clustering are highly suspect.

**Conclusion**

The research focused on addressing the pervasive issue of spam messages in the digital communication landscape. Advanced algorithms and machine learning techniques were employed to construct a robust spam classification system capable of distinguishing between genuine and spam messages. The foundation was built through meticulous data collection, preprocessing, feature engineering, and model selection. The dataset, reflecting diverse message characteristics, served as the basis for training and refining the classification model. Natural language processing (NLP) and machine learning techniques were incorporated to enhance the model's performance and adaptability to evolving communication patterns. Beyond individual convenience, the project holds significance in broader domains such as email filtering, cybersecurity, and information management. By mitigating the disruptive impact of spam, a safer and more efficient digital communication environment was aimed for. The Gaussian Naive Bayes (GNB) model was utilized alongside clustering techniques to enhance accuracy and adaptability. Internal cross-validation demonstrated the model's proficiency, with a mean accuracy of approximately 91.29%. External cross-validation further reinforced its effectiveness, achieving an outstanding accuracy of approximately 98.67%. Incorporating KMeans clustering enriched the feature space and consistently yielded high accuracy scores across various classification models. Transitioning to DBSCAN clustering marked a significant evolution in the approach, maintaining impressive accuracy scores. The research contributes to addressing digital communication challenges and promotes a safer, more productive landscape.

In [ ]: In [ ]: In [ ]: